Departamento de Engenharia de Computação e Automação Industrial
Faculdade de Engenharia Elétrica e de Computação
Universidade Estadual de Campinas

IA376M/EA099M

Tópicos em Engenharia de Computação VII

Análise Visual (*Visual Analytics*) em Ciência de Dados

Profa. Wu, Shin - Ting

Março de 2025

Conteúdo

1 Introdução **1**

1.1 Fundamentos Teóricos . . . 2

1.1.1 Probabilidade e Estatística . . . 2

1.1.2 Aprendizado de Máquina . . . 3

1.2 Soluções Analíticas e Soluções Baseadas em Dados . . . 5

1.2.1 Um Exemplo . . . 5

1.2.2 Distorções e Limitações Estatísticas . . . 8

1.2.3 Mineração e Visualização de Dados . . . 9

1.3 Ciência de Dados . . . 10

1.4 Análise Visual de Dados . . . 11

1.5 Linguagens de Programação Python e R . . . 12

1.6 Considerações Finais . . . 14

1.7 Exercícios . . . 14

2 Sinergia entre Python e R **15**

2.1 Python . . . 16

2.1.1 Estrutura Básica . . . 18

2.1.2 Tipos de Dados . . . 18

2.1.3 Operações Lógico-Aritméticas e Relacionais . . . 20

2.1.4 Comandos Condicionais . . . 20

2.1.5 Comandos de Laços de Repetição . . . 20

2.1.6 Funções Incorporadas . . . 21

2.1.7 Funções Importadas . . . 21

2.1.8 Funções Personalizadas . . . 22

2.1.9 Conjuntos de Dados Incorporados . . . 23

2.2 R . . . 23

2.2.1 Estrutura Básica . . . 24

2.2.2 Tipos de Dados . . . 25

2.2.3 Operações Lógico-Aritméticas e Relacionais 27
2.2.4 Comandos Condicionais 28
2.2.5 Comandos de Laços de Repetição 28
2.2.6 Funções Incorporadas 29
2.2.7 Funções Importadas 30
2.2.8 Funções Personalizadas 31
2.2.9 Conjunto de Dados Incorporados 31
2.3 Documentação 31
2.3.1 R Markdown 32
2.3.2 Jupyter Notebooks 34
2.4 Sinergia entre R e Python 36
2.5 Considerações Finais 38
2.6 Exercícios 38

3 Interface para Análise Visual de Dados 41
3.1 Princípios de Schneiderman 42
3.2 Princípios de Ware 44
3.3 Princípios de Tufte 47
3.4 Tipos de Dados 52
3.4.1 Valores e Relações 52
3.4.2 Técnicas de Renderização 54
3.5 Gramática dos Gráficos 55
3.5.1 Gramática dos Gráficos (Estatísticos) 56
3.5.2 Gramática em Camada dos Gráficos 58
3.5.3 Exemplo 59
3.6 Considerações Finais 66
3.7 Exercícios 67

4 Percepção Visual e Cognição 69
4.1 Psicofísica Visual 70
4.2 Atributos Pré-atentivos 74
4.3 Percepção de Grupos e Correlações 76
4.4 Psicologia Cognitiva 79
4.5 Resolução de Problemas 80
4.6 Considerações Finais 81
4.7 Exercícios 82

5 Importação de Dados **83**

5.1 Conjunto de Dados Organizados (*Tidy Data*) 84

5.2 Fontes de Dados 87

5.2.1 Arquivos Estáticos de Dados Tabulares 87

5.2.2 Banco de Dados 92

5.2.3 APIs *Web* 93

5.2.4 *Web Scraping* 97

5.3 Conversão dos Dados 101

5.4 Validação dos Dados 104

5.5 Automação e Repetibilidade 106

5.6 Considerações Finais 107

5.7 Exercícios 109

6 Estatistica Descritiva **110**

6.1 Estatísticas Resumidas 111

6.1.1 Organização de Dados 111

6.1.2 Medidas de Posição 115

6.1.3 Medidas de Dispersão 116

6.1.4 Verificação da Normalidade 122

6.1.5 Programação 123

6.2 Medidas de Relação 125

6.3 Similaridade 128

6.4 Clusterização 132

6.4.1 K-means 135

6.4.2 Clusterização Hierárquica Aglomerativa 137

6.4.3 DBSCAN 139

6.4.4 CLARA 141

6.4.5 Agrupamentos por Percepçao Visual 142

6.5 Considerações Finais 144

6.6 Exercícios 145

7 Preparação de Dados **146**

7.1 Análise Exploratória de Dados 147

7.2 Organização e Reestruturação de Dados 149

7.2.1 Limpeza de Dados 150

7.2.2 Organização no Formato *tidy* 152

7.2.3 Eliminação de Redundâncias 157

7.2.4 Integração de Dados 158
7.3 Transformação de Dados 163
7.3.1 Filtragem 163
7.3.2 Reordenação de Dados 166
7.3.3 Normalização de Valores 167
7.3.4 Redução de Observações 170
7.3.5 Redução de Dimensionalidade 173
7.4 Considerações Finais 174
7.5 Exercícios 176

8 Probabilidade **177**
8.1 Conceitos Básicos 178
8.2 Operações Fundamentais de Probabilidade 179
8.3 Variáveis Aleatórias 182
8.3.1 Funções de Distribuição de Variáveis Aleatórias 183
8.3.2 Estatísticas Resumidas em Abordagem Probabilística 186
8.4 Teorema Central do Limite 190
8.5 Modelos de Probabilidade 191
8.5.1 Variáveis Discretas 191
8.5.2 Variáveis Contínuas 192
8.5.3 Gráficos de Distribuições em R e Python 195
8.6 Simulações de Monte Carlo 199
8.6.1 Exemplo 201
8.7 Considerações Finais 203
8.8 Exercícios 204

9 Estatística de Inferência: Estimativas Pontual e Intervalar **206**
9.1 Estimativas Pontuais 207
9.2 Distribuições Amostrais 211
9.2.1 Distribuições de Proporções Amostrais 215
9.2.2 Distribuições Teóricas 220
9.2.3 Normalização das Distribuições 222
9.2.4 Distribuições *Bootstrap* 223
9.3 Estimativa Intervalar 227
9.3.1 Distribuição Amostral Subjacente 229
9.3.2 Cálculo de Intervalos de Confiança 230
9.3.3 Interpretação de Intervalos de Confiança 235

9.4 Considerações Finais 235
9.5 Exercícios 236

10 Estatística de Inferência: Testes de Hipóteses 239
10.1 Hipóteses 240
10.2 Distribuições Amostrais sob Hipóteses 242
10.2.1 Distribuição da Estatística de Teste 242
10.2.2 Distribuições *Bootstrap* 246
10.3 Nível de Significância 247
10.4 Tipos de Testes de Hipóteses 248
10.5 Erros em Testes de Hipóteses 250
10.6 P-valor 253
10.7 Interpretação de Resultados dos Testes de Hipóteses 254
10.8 Procedimentos de Testes de Hipóteses 255
10.9 Considerações Finais 256
10.10Exercícios 257

11 Estatística de Inferência: Regressão 258
11.1 Fundamentos da Regressão 259
11.2 Pré-processamento 261
11.3 Modelagem 265
11.3.1 Regressão Linear Simples 266
11.3.2 Regressão Linear Multivariada 268
11.4 Métricas para Avaliação do Modelo 271
11.5 Considerações Finais 272
11.6 Exercícios 273

12 Modelagem Estatística: Regressão Preditiva 275
12.1 Modelos Estatísticos 277
12.2 Pressupostso para Validade de Regressão 283
12.3 Inferência Analítica 289
12.4 Inferência Computacional 291
12.5 Procedimento de Construção de um Modelo Estatístico de Regressão 294
12.6 Considerações Finais 295
12.7 Exercícios 296

Capítulo 1

Introdução

Existem problemas em diversos campos da ciência, engenharia, economia e outras áreas que não podem ser resolvidos de maneira exata ou analítica. Gödel, Turing e Church demonstraram nos anos 30 que existem problemas não computáveis, cujas soluções não podem ser descritas por meio de axiomas e derivações a partir deles [11]. Diante da impossibilidade de encontrar soluções exatas ou analíticas para esses e outros problemas, uma alternativa é adotar uma abordagem orientada a dados. Isso significa que, em vez de tentar encontrar uma solução analítica, a ênfase é colocada na coleta e análise de dados relevantes.

A abordagem orientada a dados reconhece que é possível alcançar soluções aproximadas com aplicabilidade prática, mesmo que a obtenção de soluções exatas seja muitas vezes desconhecida ou inviável. Essa perspectiva é de extrema relevância em campos como análise de dados, aprendizado de máquina e ciência de dados, onde a habilidade de lidar com volumes maciços de dados e encontrar soluções pragmáticas é essencial. Essa abordagem nos lança diante de um novo e desafiador cenário, que consiste em extrair informações eficazes de vastos conjuntos de dados, frequentemente heterogêneos devido à diversidade de suas origens. Além de mecanismos apropriados para armazená-los, é fundamental desenvolver ferramentas e estratégias adequadas para acessar, limpar, minerar e apresentar esses dados de maneira inteligível, de modo a efetivamente apoiar a solução de problemas complexos e as tomadas de decisão humanas.

É nesse contexto que surge um novo campo de pesquisa conhecido como Ciência de Dados. Este campo busca não apenas extrair significado de dados, mas também desenvolver técnicas avançadas para lidar com a complexidade e a escala dos dados modernos. A Ciência de Dados é interdisciplinar que continua a evoluir e desempenha um papel relevante na resolução de desafios do mundo real. Ela lida com a coleta, análise, interpretação e apresentação de dados para obter *insights*, tomar decisões informadas e resolver problemas complexos. Para isso, ela combina conceitos e técnicas de várias áreas, incluindo estatística, matemática, programação, aprendizado de máquina e conhecimento de domínio específico.

Para conduzir análises de dados com eficácia, os cientistas de dados geralmente possuem habilidades abrangentes em programação, estatística, aprendizado de máquina, visualização de dados e *expertise* em um domínio específico. O campo da Ciência de Dados está em constante evolução, adaptando-se continuamente às novas técnicas e ferramentas desenvolvidas para lidar com a crescente quantidade de dados disponíveis no mundo moderno. Neste capítulo, apresentamos alguns conceitos elementares na Ciência de Dados. A Seção 1.1 introduz os fundamentos teóricos aplicados na resolução de problemas baseados em dados. Em seguida, na Seção 1.2, exploramos comparativamente soluções analíticas e soluções baseadas em dados, destacando potenciais distorções introduzidas em medidas estatísticas e motivando os pesquisadores a recorrerem à mineração e visualização eficiente de dados. Somente após essa discussão, apresentamos uma visão introdutória da Ciência de Dados na Seção 1.3 e uma das suas subáreas, Análise Visual de Dados, na Seção 1.4. Finalmente, na Seção 1.5 são apresentadas duas linguagens de programação amplamente utilizadas na Ciência de Dados, R e Python. Ambas oferecem diversas bibliotecas e ferramentas para processamento e visualização de dados.

1.1 Fundamentos Teóricos

Nesta seção são apresentados brevemente os fundamentos teóricos na resolução de problemas baseados em dados, a probabilidade, a estatística e o aprendizado de máquina. Essas três disciplinas constituem uma base sólida para análises e interpretações de dados que possam ajudar na resolução de problemas sem formulação analítica.

1.1.1 Probabilidade e Estatística

A probabilidade e a estatística emergem como pilares essenciais na modelagem e análise de dados de problemas não solucionáveis analiticamente, lidando com a incerteza e a aleatoriedade em diversos contextos. Elas oferecem as ferramentas teóricas e práticas essenciais para resumir e simplificar dados, compreender sua variabilidade, realizar inferências estatísticas e extrair *insights* valiosos, como identificar relações entre variáveis. Além disso, possibilitam a previsão de cenários futuros com base em conjuntos de dados complexos e auxiliam na solução de problemas por meio da análise dos dados coletados.

A **probabilidade** constitui uma medida quantitativa da incerteza vinculada a eventos aleatórios. Essa abordagem busca caracterizar a chance ou frequência com que um evento específico pode se manifestar em um conjunto de resultados possíveis. Representada por números entre 0 e 1, a probabilidade se destaca por sua interpretação: 0 indica uma chance extremamente pequena de ocorrer, 1 confere certeza, enquanto valores entre esses extremos indicam nuances na probabilidade. A teoria da probabilidade, nesse contexto, oferece um robusto arcabouço matemático para modelar e compreender situações permeadas pela incerteza.

A **estatística** é uma disciplina abrangente que envolve a coleta, organização, análise, interpretação e apresentação de dados. Seu propósito primordial é extrair informações significativas de conjuntos de dados. Existem duas abordagens fundamentais para organizar e modelar dados: a primeira os considera como conjuntos de dados, enquanto a segunda adota uma perspectiva probabilística, associando probabilidade aos eventos. Essa última abordagem expandiu a capacidade de tomar decisões fundamentadas na estatística. Assim, a estatística se divide em dois ramos complementares, mas com focos e objetivos distintos: estatística descritiva e estatística de inferência. A **estatística descritiva** nos permite organizar, apresentar e interpretar os dados, resumindo-os por meio de medidas de tendência central, como a média e a mediana, e medidas de dispersão, como o desvio padrão. Isso nos proporcina uma visão abrangente das características dos dados. Já a **estatística de inferência**, com bas em abordagens probabilísticas, vai além ao nos permite extrapolar conclusões da amostra para a população, testar hipóteses sobre essa população e estabelecer intervalos de confiança para nossos resultados. Um ramo da estatística inferencial, conhecido por **estatística preditiva**, se dedica a construir modelos capazes de prever resultados futuros com base nos dados disponíveis. A regressão é uma das ferramentas mais utilizadas nesse contexto, permitindo modelar as relações entre variáveis e fazer previsões precisas. Juntas, essas abordagens nos permitem extrair conclusões sólidas a partir de amostras, gerenciar a incerteza e fundamentar decisões.

A interligação entre probabilidade e estatística é fundamental para a compreensão e análise de dados. A probabilidade estabelece a base teórica para compreender eventos aleatórios e quantificar a incerteza associada. Em contrapartida, a estatística utiliza esses princípios probabilísticos para realizar inferências e previsão a partir da distribuição de probabilidade das amostras observadas, avaliando a plausibilidade dessas conclusões. Técnicas estatísticas como amostragem, testes de hipóteses e intervalos de confiança são amplamente empregadas na análise de populações com base em amostras, todas ancoradas nos princípios da teoria de probabilidade.

1.1.2 Aprendizado de Máquina

O **aprendizado de máquina** (em inglês, *machine learning*) é uma subárea da inteligência artificial que se concentra no desenvolvimento de algoritmos capazes de aprender padrões e realizar tarefas sem serem explicitamente programados. Essa abordagem é especialmente poderosa em situações em que a formulação de regras específicas pode ser desafiadora ou impraticável. Em sua essência, o aprendizado de máquina utiliza modelos matemáticos e estatísticos para capacitar os sistemas a melhorar seu desempenho e construir modelos preditivos ou descritivos mais precisos ao longo do tempo, com base na "experiência" (memória) e no treinamento contínuo com dados atualizados. Uma característica fundamental do Aprendizado de Máquina é a capacidade de generalizar o aprendizado. Isso é possível através da construção de um modelo com base nos dados coletados, permitindo que os modelos façam previsões ou tomem decisões em novas situações.

A probabilidade e a estatística fornecem as bases para muitos dos métodos e técnicas de aprendizado de máquina. A probabilidade é usada para modelar a incerteza inerente aos dados, enquanto a estatística fornece ferramentas para inferência, avaliação de modelos e validação. Essa combinação permite que os algoritmos de aprendizado de máquina façam previsões e tomem decisões com base em dados, considerando a variabilidade e a incerteza inerentes.

Por exemplo, o teste de hipóteses, que é uma técnica estatística usada para tomar decisões com base em dados observados, envolve a formulação de duas hipóteses: a hipótese nula (H0) e a hipótese alternativa (Ha). A hipótese nula geralmente representa uma afirmação do cenário atul, enquanto a hipótese alternativa representa a afirmação que você deseja confirmar. A probabilidade p entra em cena ao calcular a chance de obter os resultados observados (ou resultados mais extremos) sob a suposição de que a hipótese nula seja verdadeira. Se p for muito baixa, pode-se rejeitar a hipótese nula em favor da hipótese alternativa. A probabilidade quantifica, portanto, a incerteza associada à decisão estatística de rejeitar ou não a hipótese nula.

Ao empregar algoritmos de aprendizado de máquina, os cientistas de dados conseguem automatizar a análise de grandes volumes de dados, identificar padrões complexos e desenvolver modelos preditivos. Isso contribui significativamente para a rápida tomada de decisões em diversas áreas, desde a medicina até o *marketing*, e portanto para a solução eficiente de problemas usando dados.

Um desafio central no campo do aprendizado de máquina de última geração é a escassa interpretabilidade dos modelos, frequentemente referida como a "caixa preta" desses sistemas. Modelos avançados, como redes neurais profundas (em inglês, *deep learning*), são muito complexos e, consequentemente, tornam-se extremamente desafiadores de entender e explicar. Essa falta de transparência suscita preocupações significativas, especialmente em setores nos quais a clareza e a interpretabilidade são essenciais, como saúde, automação, finanças e justiça. Profissionais nesses campos adotam uma postura cética em relação a esses sistemas devido ao receio de que a falta de transparência possa se tornar um problema crítico. Em contextos nos quais é crucial compreender como e por quê um determinado resultado foi gerado, principalmente quando tais decisões podem impactar diretamente a vida das pessoas, a opacidade dos modelos representa uma barreira substancial.

A exploração visual desempenha um papel fundamental na abordagem desses desafios. Acredita-se que ferramentas de visualização de dados podem aprimorar a compreensão do comportamento dos modelos, destacar padrões aprendidos e até mesmo fornecer *insights* sobre os processos de tomada de decisão dos modelos. Visualizações claras e informativas têm o potencial de aumentar a confiança nas previsões dos modelos, permitindo que os usuários compreendam e validem os resultados de maneira mais eficaz. Isso, por sua vez, contribui para uma aceitação e adoção mais amplas de sistemas de aprendizado de máquina. Além disso, a visualização desempenha um papel vital na monitoração contínua do desempenho do modelo, identificando potenciais problemas ou desvios. Essa área de pesquisa tende a crescer ainda mais à medida que a capacidade de aprendizado de máquina se aproxima

dos limites do estado-da-arte.

1.2 Soluções Analíticas e Soluções Baseadas em Dados

As soluções analíticas e as soluções baseadas em dados são dois métodos diferentes de abordar problemas complexos em matemática, ciência e engenharia. A principal diferença entre essas duas classes de soluções está na maneira como elas lidam com a complexidade dos problemas e na natureza das respostas que fornecem.

Soluções analíticas se referem a soluções encontradas através de métodos matemáticos exatos, muitas vezes envolvendo equações e fórmulas. Essas soluções são derivadas de maneira direta e precisa, sem a necessidade de coleta de dados observacionais, estimativas ou aproximações. A aplicação de soluções analíticas é destinada a problemas que podem ser formalmente descritos e resolvidos por meio de equações matemáticas. Esse enfoque é especialmente valioso em contextos de matemática pura e aplicada, física e engenharia, onde muitos problemas bem definidos podem ser tratados de maneira elegante e exata. A fórmula quadrática para determinar as raízes de uma equação quadrática é um exemplo clássico de solução analítica, assim como a resolução analítica da equação de difusão na física.

Soluções baseadas em dados emergem como uma alternativa relevante às soluções analíticas, especialmente diante de problemas complexos e dinâmicos que desafiam abordagens tradicionais. Enquanto as soluções analíticas dependem de métodos matemáticos exatos e equações, as soluções baseadas em dados exploram padrões e informações intrínsecas nos próprios dados, incorporando elementos fundamentais de **probabilidade e estatística**. Essa abordagem adaptativa é essencial quando os problemas não podem ser facilmente formulados por equações matemáticas ou quando a complexidade dos sistemas envolvidos torna a análise analítica impraticável. Além disso, nas situações em que a incerteza e a variabilidade são inerentes, as resoluções baseadas em dados oferecem flexibilidade, permitindo a adaptação contínua às mudanças e a incorporação de *insights* práticos no processo decisório. Respaldadas pela robustez estatística na interpretação dos resultados, tais resoluções suportam **análises descritivas** (síntese das características dos dados), **prescritivas** (prescrição de ações ou recomendaçãos com base nos dados disponíveis) e **preditivas** (previsões ou estimativas sobre eventos futuros com base em padrões observados nos dados históricos). Exemplos de resoluções baseadas em dados incluem a previsão do tempo com base em modelos meteorológicos, a estimativa da média de uma população com base em uma amostra de dados, a classificação de dados em aprendizado de máquina e a determinação de intervalos de confiança em estudos estatísticos.

1.2.1 Um Exemplo

Para ilustrar, são apresentados dois algoritmos para o cômputo da área de uma figura plana. O primeiro algoritmo é baseado em uma solução analítica, enquanto o segundo algoritmo utiliza uma

abordagem baseada em dados por meio da técnica de Monte Carlo.

Solução Analítica

A fórmula para determinar a área exata A de uma figura plana depende da geometria específica de figura. As fórmulas para as seguintes figuras regulares são

Retângulo : $A = \text{Base} \times \text{Altura}$.

Quadrado : $A = \text{Lado}^2$.

Triângulo : $A = \frac{\text{Base} \times \text{Altura}}{2}$.

Círculo : $A = \pi \times \text{Raio}^2$.

Paralelogramo : $A = \text{Base} \times \text{Altura}$.

Trapezóide : $A = \frac{\text{Base maior} + \text{Base menor}}{2} \times \text{Altura}$.

Quando se trata de um polígono arbitrário, é comum dividí-lo em triângulos e computar a soma das áreas desses triângulos. No entanto, quando os contornos das figuras planas não são mais segmentos de linhas simples e, portanto, quando não existe uma solução analítica fechada, a determinação da área pode se tornar mais complexa. Um exemplo clássico disso é o problema do cálculo da área de uma figura irregular, composta por uma combinação de curvas suaves, como arcos de círculos ou curvas de grau superior, e segmentos de linha. Esta figura não pode ser facilmente subdividida em triângulos ou formas regulares para calcular a área diretamente usando fórmulas simples. Para calcular a área de tal figura irregular, pode-se recorrer a uma solução baseada em dados.

Solução Baseada em Dados

A técnica de Monte Carlo é especialmente útil quando a área não pode ser calculada facilmente por métodos analíticos, mas uma boa aproximação pode ser obtida usando a simulação de amostragem aleatória. Essa técnica é amplamente aplicada em problemas de integração numérica, simulação estocástica e muitos outros campos da matemática e da ciência. O procedimento básico para calcular uma área usando a técnica de Monte Carlo envolve os seguintes passos:

Defina a região alvo : Comece por definir a região ou a forma da qual se deseja calcular a área. Isso pode ser uma figura geométrica simples, como um círculo ou um retângulo, ou uma região mais complexa e irregular.

Insira a região em um espaço conhecido : Coloque a região alvo em um espaço conhecido, como um quadrado ou retângulo, de tal forma que a região alvo esteja completamente contida nesse espaço. Isso é importante para que você saiba o limite dentro do qual as amostras aleatórias serão geradas.

Gere amostras aleatórias : Gere um grande número de pontos aleatórios dentro do espaço conhecido (o retângulo ou quadrado que contém a região alvo). Você pode fazer isso usando um gerador de números aleatórios.

Verifique a inclusão dos pontos : Para cada ponto gerado aleatoriamente, verifique se ele está dentro da região alvo. Isso pode ser feito comparando as coordenadas do ponto com as coordenadas da região alvo. Se o ponto estiver dentro da região alvo, conte-o como um ponto interno. Caso contrário, ele é um ponto externo.

Calcule a proporção de pontos internos : Após gerar um grande número de pontos, calcule a proporção de pontos internos em relação ao total de pontos gerados. Essa proporção pode ser vista como uma estimativa da razão entre a área da região alvo e a área do espaço conhecido.

Estime a área : Para calcular a área da região alvo, você multiplica a proporção de pontos internos pela área do espaço conhecido (o retângulo ou quadrado que contém a região alvo). A fórmula básica é:

$$\text{Área estimada} = \frac{\text{Pontos Internos}}{\text{Total de Pontos}} \times \text{Área do Espaço Conhecido.}$$

Quanto mais pontos aleatórios você gerar, mais precisa será a sua estimativa da área da região alvo.

Para determinar o intervalo de confiança da área estimada pela técnica de Monte Carlo, você pode usar métodos estatísticos que levam em consideração a variabilidade inerente à amostragem aleatória. Aqui estão os passos básicos para calcular o intervalo de confiança da área estimada:

Calcule a média e o desvio padrão das áreas estimadas : Após realizar várias simulações de Monte Carlo e obter várias estimativas da área da região alvo, calcule a média e o desvio padrão dessas estimativas.

Determine o nível de confiança desejado : Escolha o nível de confiança desejado. O nível de confiança é uma medida da probabilidade de que o intervalo de confiança contenha o valor real da área. O nível de confiança comum é 95%, o que significa que você está 95% confiante de que o intervalo de confiança conterá a verdadeira área.

Calcule o erro padrão da média : O erro padrão da média (também chamado de erro padrão da estimativa) é calculado dividindo o desvio padrão pelo quadrado da raiz do número de amostras. A fórmula é:

$$\text{Erro Padrão da Média} = (\text{Desvio Padrão})/\sqrt{(\text{Número de Amostras})}$$

Determine o valor crítico : Encontre o valor crítico associado ao seu nível de confiança e à distribuição da média das áreas estimadas. Para um nível de confiança de 95%, por exemplo, e uma distribuição normal padrão, que é uma distribuição normal com média 0 e desvio padrão 1, o valor crítico correspondente é geralmente em torno de ±1,96 (para um intervalo bilateral).

Calcule o intervalo de confiança : Com os valores obtidos nos passos anteriores, você pode calcular o intervalo de confiança da área estimada. A fórmula é:

$$\text{Intervalo de Confiança} = \text{Média da Área Estimada} \pm (\text{Valor Crítico} \times \text{Erro Padrão da Média})$$

O resultado desse cálculo será o intervalo de confiança que contém a área real com o nível de confiança especificado.

É importante observar que a validade desses cálculos pressupõe que a distribuição das áreas estimadas segue uma distribuição aproximadamente normal. No entanto, em alguns casos, especialmente quando o número de simulações é relativamente pequeno, a distribuição dos resultados das simulações pode não se assemelhar a uma distribuição normal. Nestes casos, se a distribuição for muito assimétrica ou desconhecida, você pode considerar métodos de *bootstrap* ou outros métodos de reamostragem para estimar o intervalo de confiança de forma mais robusta [12].

Lembre-se ainda de que a precisão do intervalo de confiança depende do tamanho da amostra (o número de pontos gerados na simulação). Quanto maior a amostra, menor será a amplitude do intervalo de confiança, o que significa uma estimativa mais precisa da área. Portanto, é importante equilibrar a quantidade de tempo computacional disponível com a precisão desejada ao realizar simulações de Monte Carlo. Vamos explorar esses aspectos em mais detalhes nos capítulos adiante.

1.2.2 Distorções e Limitações Estatísticas

Os estatísticos compartilham a preocupação de que as medidas estatísticas, incluindo aquelas aplicadas em aprendizado de máquina, podem, em alguns casos, distorcer a compreensão dos dados subjacentes e conduzir a soluções equivocadas. Um exemplo emblemático que ilustra essa inquietação são os quatro conjuntos de dados bidimensionais de Anscombe [17], que exibem estatísticas descritivas quase idênticas (média de x = 9, média de y = 7.50, variância de x = 11, variância de y = 4.125, correlação entre x e y = 0.816), mas têm distribuições muito diferentes e aparências distintas quando visualizados graficamente, como mostra a Figura 1.1. Esse cenário aponta para a complexidade inerente à interpretação de dados brutos e as implicações práticas na escolha entre abordagens analíticas e baseadas em dados. Recomenda-se que visualize e entenda atributos dos dados antes de passar para a engenharia de recursos e construir modelos estatísticos e de aprendizado de máquina em resolução de problemas baseados em dados.

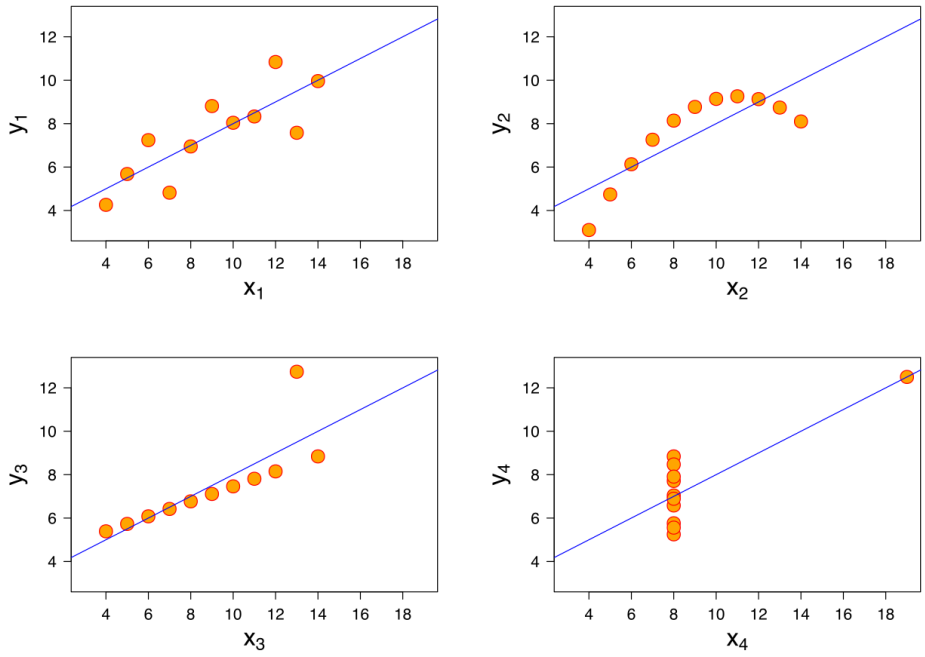


Figura 1.1: O quarteto de Ascombe. (Fonte: [17])

1.2.3 Mineração e Visualização de Dados

Diante de desafios nos quais a formulação analítica se mostra impraticável, os cientistas de dados introduziram a técnica de **mineração de dados** (em inglês, *data mining*). Pioneiros, como J. Han e M. Kamber na década de 1990, promoveram essa disciplina, que consiste na investigação meticulosa de grandes conjuntos de dados por meio da aplicação de diversas técnicas alternativas de manipulação. Nesse processo, os cientistas atuam como detetives, perscrutando informações fidedignas e padrões ocultos. A abordagem, como sugere seu nome, é uma forma de **minerar** dados em busca de tesouros de informação significativa, proporcionando uma compreensão mais profunda do fenômeno subjacente e contribuindo para a resolução de problemas. Hoje em dia, a mineração de dados se tornou uma técnica indispensável para extrair conhecimentos e padrões valiosos de grandes conjuntos de dados. Diversos esforços têm sido direcionados para sistematizar o procedimento, com o intuito de orientar os cientistas de dados na escolha de variáveis e algoritmos adequados ao tipo de problema e aos dados disponíveis [32].

O processo exploratório e analítico da mineração de dados, destinado a descobrir conhecimento útil em grandes volumes de dados, é ainda mais enriquecido pela **visualização** dos padrões complexos subjacentes. Dentre os cinco sentidos humanos fundamentais (tato, paladar, olfato, visão e audição), a visão é, em média, o sentido mais apurado e desenvolvido. Desde os tempos dos homens das cavernas, que registravam suas conquistas por meio de pinturas rupestres, até a primeira metade do século XIX, quando diferentes gráficos estatísticos já eram conhecidos (gráfico de barras, gráfico de pizza, gráfico de linhas, histograma, diagrama de extremos e quartis), a visualização tem sido uma ferramenta vas-

tamente explorada. Foi, no entanto, o estatístico americano John Wilder Tukey quem formalizou, em 1977, por meio de seu livro *Exploratory Data Analysis* [95], a interação visual com dados brutos através de gráficos, diagramas e cartas, permitindo uma análise sem condicionamento a um modelo específico. Pesquisadores, como Tufte [94] e Ware [98], dedicaram-se a estabelecer princípios na transformação de dados complexos em imagens compreensíveis intuitivamente por diversos públicos, possibilitando uma interpretação mais clara e uma comunicação mais eficaz de descobertas e *insights*. Essa abordagem promove uma compreensão mais holística e precisa da informação contida nos conjuntos de dados.

Vale ressaltar que mineração e visualização de dados representam domínios especializados distintos, com evoluções independentes. Enquanto a mineração de dados, uma parte da Ciência de Dados, se fundamenta em técnicas de classificação, agrupamento, inferência e previsão, a visualização está relacionada à computação gráfica e à cognição humana [98]. A ideia consiste essencialmente em utilizar essas técnicas para representar graficamente os dados, a fim de destacar padrões, tendências e anomalias, tornando-os mais acessíveis à cognição humana. Ao facilitar a compreensão e interpretação dos dados, a visualização orienta e enriquece o processo de mineração, auxiliando na aplicação de técnicas como classificação, agrupamento, sequenciamento ou sumarização.

1.3 Ciência de Dados

A Ciência de Dados é um campo interdisciplinar que aplica técnicas estatísticas, computacionais e de aprendizado de máquina para analisar e interpretar dados complexos, com o objetivo de extrair conhecimento e *insights* e fazer modelagem preditiva. Isso permite lidar com problemas sem formulação analítica direta em diversos setores, tais como finanças, saúde, *marketing*, ciências naturais, redes sociais, transporte, entre outros, tornando-se essencial diante da crescente complexidade e volume de dados.

A Ciência de Dados compreende integralmente o ciclo de vida dos dados, desde sua coleta até a interpretação e a tomada de decisões. Ela faz uso de ferramentas provenientes da probabilidade e estatística para modelar a incerteza inerente aos dados, realizando inferências e avaliando os modelos construídos. Ao empregar algoritmos de aprendizado de máquina, a Ciência de Dados capacita sistemas a aprender e executar tarefas específicas, eliminando a necessidade de programação explícita. A mineração de dados, parte integrante da Ciência de Dados, utiliza técnicas estatísticas e de aprendizado de máquina para descobrir padrões e *insights* relevantes em grandes conjuntos de dados e construir modelos preditivos. Além disso, a Ciência de Dados incorpora etapas de pré-processamento de dados, como limpeza e transformação, a fim de prepará-los para análises significativas. Essa abordagem abrangente visa extrair o máximo de valor e compreensão dos dados disponíveis.

Tipicamente, o ciclo de vida dos dados na Ciência de Dados compreende as seguintes etapas:

Coleta de Dados : Os cientistas de dados coletam dados de fontes diversas, como sen-

sores, bancos de dados, redes sociais, aplicativos da *web*, entre outros.

Limpeza e Preparação de Dados : Os cientistas de dados preparam os dados, que inclui a limpeza, transformação e formatação dos dados, para torná-los adequados à análise, uma vez que muitos dados coletados podem estar desorganizados, incompletos ou conter erros.

Análise Exploratória de Dados (em inglês, *Exploratory Data Analysis* - EDA): Os cientistas de dados exploram os dados para identificar padrões, tendências e características importantes.

Modelagem de Dados : Modelos estatísticos ou de aprendizado de máquina são desenvolvidos tanto por cientistas de dados, com base em seus conhecimentos e intuições, quanto por máquinas, que utilizam algoritmos e dados para construir modelos.

Avaliação de Modelos : Após a criação dos modelos, é crucial avaliar seu desempenho para assegurar a utilidade e a precisão. Isso geralmente envolve a utilização de técnicas estatísticas para verificar a robustez e a eficácia dos modelos.

Comunicação de Resultados : Os resultados da análise de dados são comunicados de forma clara e eficaz para as partes interessadas, muitas vezes por meio de relatórios, painéis interativos ou apresentações.

1.4 Análise Visual de Dados

A coleta e armazenamento de dados é uma prática essencial para a segurança nacional e a defesa dos Estados Unidos há muito tempo. Os ataques às Torres Gêmeas em 11 de setembro de 2001 destacaram a urgente necessidade de aprimorar a capacidade de análise de vastos volumes de dados complexos acumulados. Isso visava identificar ameaças e tomar decisões críticas de maneira mais eficaz. Em resposta a essa demanda, o Departamento de Segurança Interna dos EUA (em inglês, *Department of Homeland Security* - DHS) designou, em 2004, o Centro de Análise de Dados e Visualização Nacional (em inglês, *National Visualization and Analytics CenterTM* - NVACTM) para a tarefa de combater futuros ataques terroristas nos Estados Unidos e em todo o mundo. Uma agenda para programas de pesquisa e desenvolvimento de ferramentas que facilitem a obtenção de uma visão analítica avançada foi estabelecida em 2005, coordenada pelo Laboratório Nacional do Noroeste Pacífico (*Pacific Northwest National Laboratory*). Os estudos evidenciaram a relevância crítica da **visualização** de dados na interpretação e compreensão das informações subentendidas ao longo da exploração e mineração desses dados.

O termo **Análise Visual de Dados** (em inglês, *visual analytics*) foi introduzido em 2004 pelos editores Pak Chung Wong e Jim Thomas de uma edição especial da revista *IEEE Computer Graphics and Applications* para caracterizar uma coletânea de artigos selecionados que "*combine the art of*

human intuition and the science of mathematical deduction to directly perceive patterns and derive knowledge and insight from them" [111]. A característica comum desses artigos é a integração das técnicas de inferência e previsão de informações a partir de um volume grande de dados brutos e das técnicas de renderização e de interação para entender e analisar os dados brutos/processados volumosos com a finalidade de proporcionar um melhor suporte às tomadas de decisão em problemas complexos. Em 2010, no contexto do projeto VisMaster CA financiado pela União Européia foi publicado em [46] o estado-da-arte das pesquisas e desenvolvimento em análise de dados visual na Europa.

A Análise Visual de Dados é uma disciplina multidisciplinar inserida na Ciência de Dados, concentrando-se de forma específica na representação gráfica de dados para facilitar a compreensão e interpretação. Essa abordagem incorpora interfaces gráficas familiares e responsivas, criadas para interagir de maneira dinâmica com os dados visualizados. Utilizando uma variedade de técnicas visuais, como gráficos e mapas, a Análise Visual de Dados destaca padrões, tendências e anomalias nos dados, revelando nuances que podem não ser evidentes ao examinar apenas os valores numéricos brutos. Para a exploração eficaz dos dados, a Análise Visual de Dados emprega técnicas interativas, como seleção e manipulação direta. Essas interações capacitam os usuários a explorar dados de interesse, proporcionando *insights* importantes e subsidiando a construção de raciocínios complexos. Dessa forma, a tecnologia de Análise Visual de Dados não apenas transcende a simples visualização gráfica, mas também se torna uma ferramenta essencial para descoberta e compreensão em ambientes de Ciência de Dados.

1.5 Linguagens de Programação Python e R

A Ciência de Dados é frequentemente associada a linguagens de programação como R e Python, devido à ampla gama de bibliotecas e ferramentas especializadas que desempenham papéis significativos na coleta, análise, visualização e modelagem de dados.

Python[1] é conhecido por sua simplicidade e legibilidade, tornando-o uma escolha preferida para programadores de todos os níveis. Ele oferece uma vasta coleção de bibliotecas de Ciência de Dados e Visualização, como NumPy[2], pandas[3], scikit-learn[4] e Matplotlib[5], que permitem análise de dados, aprendizado de máquina e renderização eficientes. Python é também equipado de streamlit [6] para desenvolvimento de interfaces interativas. IPython[7] é um ambiente com interface em linha de comando para execução interativa de códigos Python, oferecendo recursos adicionais, como realce de sintaxe, histórico de comandos, preenchimento automático e suporte a várias linguagens de pro-

[1] `https://www.python.org/`
[2] `https://numpy.org/`
[3] `https://pandas.pydata.org/`
[4] `https://scikit-learn.org/stable/`
[5] `https://matplotlib.org/`
[6] https://streamlit.io/
[7] https://ipython.org/

gramação, enquanto Jupyter Lab ou Jupyter Notebook[8], que usa IPython como base para a execução de códigos Python, são frequentemente usados no Python para criar documentos interativos que combinam código, visualizações e explicações.

R[9], por outro lado, é altamente especializado em estatísticas e análise de dados. Ele oferece uma variedade de pacotes e funções específicos para análise estatística e gráficos, RcppArmadillo[10], dplyr[11], caret[12] e ggplot2[13]. R é equipado de shiny[14], similar a streamlit, para desenvolvimento de interfaces interativas baseada na tecnologia da *web*. O ambiente R é favorecido por estatísticos e pesquisadores, sendo RStudio[15] uma IDE (do inglês *Integrated Development Environment*) popular para desenvolvimento de códigos em R.

Uma característica distintiva tanto do R quanto do Python é a capacidade de criar documentos que integram código, texto formatado e visualizações. Essa capacidade é fundamental para a reprodutibilidade e a comunicação eficaz em análises de dados e pesquisa. A combinação de código, narração e gráficos em um único documento torna mais fácil compartilhar resultados, facilita a compreensão do processo de análise e permite que outros reproduzam e validem os resultados. Essas funcionalidades são especialmente valiosas na Ciência de Dados, onde **a transparência e a documentação adequada são essenciais**. Tanto o R Markdown[16] do RStudio, o Jupyter Notebook com nbconvert[17] em Python, quanto o Quarto[18], projetado para ser altamente integrável tanto no ecossistema RStudio quanto no ecossistema Jupyter Notebook, são ferramentas poderosas para criar documentos dinâmicos que simplificam a análise de dados, a criação de relatórios e a apresentação de resultados, tornando-os recursos fundamentais para profissionais em análise de dados e pesquisa.

Ambas as linguagens desfrutam de comunidades de código aberto ativas e oferecem recursos poderosos para análise de dados, tornando-se escolhas relevantes para cientistas de dados, analistas e pesquisadores. A combinação de Python e R com Notebooks, utilizando pacotes como Seaborn e plotnine em Python, e ggplot2 em R[19], facilita uma análise colaborativa e reproduzível de dados. Uma análise comparativa entre essas duas linguagens é apresentada por Luna no espaço Datacamp[20].

Outra característica interessante é que os dois ecossistemas RStudio e Jupyter são projetados para permitir o acesso a funções de outras linguagens, o que oferece uma grande vantagem em termos de interoperabilidade[21]. Isso significa que os usuários podem tirar proveito de recursos e bibliotecas

[8] https://jupyter.org/install
[9] `https://www.r-project.org/`
[10] `https://cran.r-project.org/web/packages/RcppArmadillo/index.html`
[11] `https://dplyr.tidyverse.org/`
[12] `https://topepo.github.io/caret/`
[13] `https://ggplot2.tidyverse.org/`
[14] https://www.rstudio.com/products/shiny/
[15] `https://posit.co/download/rstudio-desktop/`
[16] `https://rmarkdown.rstudio.com/`
[17] `https://nbconvert.readthedocs.io/en/latest/`
[18] `https://quarto.org/`
[19] `https://medium.com/mlearning-ai/an-alliance-python-and-r-seaborn-and-ggplot2-233864b77bc4`
[20] `https://www.datacamp.com/blog/python-vs-r-for-data-science-whats-the-difference`
[21] `https://www.datacamp.com/tutorial/using-both-python-r`

de ambas as linguagens em um único ambiente, dependendo de suas necessidades e preferências. O Jupyter Notebook suporta *kernels* R que permitem a execução de códigos estatísticos R diretamente em um *notebook* Jupyter. O ambiente RStudio também permite que os usuários executem código das bibliotecas de aprendizado de máquina populares em Python diretamente em seus *scripts* ou *notebooks*.

1.6 Considerações Finais

A explosão no volume de dados e o desenvolvimento de tecnologias que possibilitaram lidar com esses dados em larga escala, como o aumento da capacidade de armazenamento, o processamento em nuvem e algoritmos mais avançados, foram fatores que impulsionaram a ascensão da Ciência de Dados como uma disciplina distintiva. Dispondo dessa quantidade enorme de dados diversos, a probabilidade e a estatística fornecem fundamentos teóricos que permitem a geração de procedimentos capazes de resolver problemas por meio de inferências. No entanto, os métodos estatísticos subjacentes podem introduzir distorções nos resultados, levando à compreensão equivocada dos fenômenos físicos inerentes aos dados coletados.

Uma abordagem para mitigar essas distorções é permitir que especialistas em dados realizem a mineração dos dados, avaliem comparativamente diferentes configurações dos dados e cheguem a soluções mais plausíveis. Essa mineração de dados torna-se mais efetiva quando integrada às técnicas de interação e visualização da Computação Gráfica. Essa subárea específica da Ciência de Dados é conhecida como Análise Visual de Dados. Atualmente, é consenso que muitas áreas podem se beneficiar das tecnologias de Análise Visual de Dados em diferentes níveis de abstração, destacando-se Astrofísica, Meteorologia, Gerenciamento de Emergências, Medicina, Genética, Bioquímica e Finanças [20, 46]. É sobre essa parte da Ciência de Dados que esta apostila se concentra.

1.7 Exercícios

1. Instalação de um conjunto de ferramentas para atividades práticas: Python e Jupyter, ou R e RStudio.

Capítulo 2

Sinergia entre Python e R

Neste capítulo, é feita uma introdução ao universo dinâmico das linguagens de programação Python e R, duas ferramentas amplamente aplicadas na análise de dados. Inicialmente, é explorada a linguagem Python, reconhecida por sua sintaxe clara e flexibilidade. Utilizando bibliotecas como scikit-learn, TensorFlow e PyTorch, pode-se conduzir análise preditiva de dados, aplicando técnicas de aprendizado de máquina e aprendizado profundo na resolução de problemas complexos. Em seguida, é introduzida a linguagem R, também conhecida como a "linguagem estatística" devido à sua história e especialização em análises estatísticas, incluindo classificação, regressão, *clustering* e técnicas de redução de dimensionalidade.

Junto com o popular pacote gráfico ggplot2, a linguagem R oferece recursos visuais para a criação de gráficos estatísticos complexos. Por sua vez, Python possui uma ampla variedade de pacotes e bibliotecas para a construção de gráficos estatísticos, incluindo matplotlib, seaborn, pandas, altair e plotly. Vale destacar que os pacotes altair e plotly seguem o estilo do ggplot2, aderindo à gramática de gráficos apresentada na Seção 3.5. Adicionalmente, o pacote plotnine é uma implementação em Python que utiliza o matplotlib como base e segue a mesma filosofia do ggplot2, uma biblioteca de visualização de dados popular em R.

Tanto R quanto Python são linguagens de programação que seguem o paradigma de **programação orientada a objetos**. Em ambas as linguagens, tudo é considerado um objeto. Isso significa que os dados e as funcionalidades são encapsulados em objetos, que podem ser manipulados e interagir entre si. Em R, por exemplo, trabalha-se com objetos como vetores, matrizes, listas, quadros de dados, funções, entre outros. Cada um desses elementos é um objeto com propriedades e métodos específicos. Em Python, a orientação a objetos é ainda mais proeminente, e praticamente tudo em Python é um objeto, incluindo números, *strings*, listas, funções, módulos e até mesmo classes e instâncias de classes definidas. Essa abordagem orientada a objetos facilita a organização e a manipulação de dados, permitindo a criação de código mais modular e reutilizável. Além disso, ambas as linguagens oferecem suporte a conceitos avançados de programação orientada a objetos, como herança, polimorfismo e

encapsulamento.

O verdadeiro potencial emerge na intersecção dessas duas linguagens. R, conhecido por seu poder em análises estatísticas convencionais, se combina com Python, que tem ganhado popularidade graças à sua vasta coleção de bibliotecas voltadas para aprendizado de máquina. Essa fusão permite que analistas e cientistas de dados aproveitem as vantagens específicas de cada linguagem para enfrentar uma variedade de desafios em Ciência de Dados, análise estatística e aprendizado de máquina. Ao longo deste capítulo, exploramos não apenas as singularidades de cada linguagem, Python 2.1 e R 2.2, mas também as sinergias entre Python e R na abordagem dos desafios da análise de dados, conforme discutido na Seção 2.4.

A documentação e a comunicação dos resultados de uma análise de dados são tão importantes quanto a análise propriamente dita. Tanto o R Markdown [27] do RStudio, o Jupyter Notebook com nbconvert [43] em Python, quanto o Quarto [66] são ferramentas para criar documentos dinâmicos que simplificam a análise de dados, a criação de relatórios e a apresentação de resultados, tornando-os recursos fundamentais para profissionais em análise de dados e pesquisa. Como discutiremos na Seção 2.3, essas ferramentas oferecem uma interface fluida com o Markdown, facilitando a geração de documentações com diferentes níveis de detalhes nos ambientes de desenvolvimento integrados RStudio e Jupyter.

2.1 Python

Python é uma linguagem de programação interpretada e de tipagem dinâmica. Sendo interpretada, o código-fonte é lido e executado linha por linha pelo interpretador em tempo de execução, em linguagem nativa do sistema em que se encontra, passando por *bytecodes*, sem a necessidade de compilação prévia para código de máquina. Destacando sua natureza interpretada, os códigos em Python são chamados de ***scripts***. Além disso, é uma linguagem de tipagem dinâmica, pois permite que as variáveis sejam associadas a tipos de dados em tempo de execução, os quais podem ser alterados durante a execução do programa.

Sua origem remonta ao final dos anos 1980, quando Guido van Rossum concebeu a ideia de criar uma linguagem mais acessível, expressiva e produtiva do que C. A primeira implementação padrão do interpretador do Python, conhecida como CPython, foi desenvolvida em C por van Rossum e lançada em 1991. A escolha deliberada de C como linguagem subjacente confere ao Python um bom desempenho. Essa decisão estratégica também facilita a integração dos seus códigos com bibliotecas escritas em C, ampliando ainda mais as capacidades da linguagem. O CPython, desde então, permanece como a implementação de referência, mantendo sua posição de destaque como a mais amplamente utilizada entre as diversas implementações de Python que surgiram ao longo dos anos.

O Python é uma linguagem popular em Ciência de Dados, devido à sua simplicidade e legibilidade

de código, à presença de uma comunidade vasta e ativa, à oferta de bibliotecas poderosas para análise e visualização de dados, além de excelentes bibliotecas para aprendizado de máquina, Em termos de áreas de aplicação, cientistas de dados preferem Python para Análise de Dados, Visualização de Dados, Aprendizado de Máquina, Aprendizado Profundo, Processamento de Imagens, Visão Computacional e Processamento de Linguagem Natural (PLN).

Para começar a programar em Python *offline*, é necessário instalar um **interpretador Python**. A instalação é um processo simples e, geralmente, envolve baixar o instalador adequado para o seu sistema operacional (Windows, macOS ou Linux) e seguir as instruções. Para obter explicações detalhadas da instalação, pode-se consultar guias específicos para cada sistema operacional, como os encontrados em [68]. Durante a instalação, certifique-se de marcar a opção para adicionar o Python à variável ambiente PATH, facilitando a execução de *scripts* a partir do terminal ou prompt de comando. Em muitas distribuições modernas do Python, o comando pip (do inglês *Pip Installs Packages*), uma ferramenta utilizada no ecossistema Python para gerenciar pacotes de funções, é incluído por padrão e pronto para uso. Por exemplo, para instalar a versão 1.17.3 do pacote numpy, usa-se o comando

```
pip install numpy==1.17.3
```

Com o comando pip/pip3 pode-se instalar o ambiente de desenvolvimento interativo Jupyter com a linha de comando

```
pip3 install jupyter
```

O Jupyter utiliza o IPython (do inglês *Interactive Python*) como seu *kernel* padrão para Python, podendo ser considerado uma evolução do IPython. Após a instalação do ambiente Jupyter, pode-se verificar se IPython foi instalado corretamente digitando o seguinte comando no *prompt* de comando:

```
ipython
```

Isso abrirá o interpretador interativo do IPython, indicando que a instalação foi bem-sucedida. Caso não, entre a seguinte linha de comando no *prompt* de comando num terminal:

```
pip3 install ipython
```

Detalhes de instalações podem ser encontrados em [10].

Caso se opte por não instalar o interpretador Python localmente, uma excelente alternativa é o Google Colab [**?**]. Essa plataforma *online* gratuita, oferecida pelo Google, permite executar *notebooks* Jupyter diretamente no seu navegador, sem necessidade de instalação. O Google Colab oferece acesso a uma diversidade de recursos computacionais, incluindo GPUs, e se integra com o Google Drive [3], facilitando o armazenamento e o compartilhamento dos seus projetos.

2.1.1 Estrutura Básica

A estrutura básica de um *script* Python, tipicamente de extensão .py, segue uma abordagem simples e direta. Inicia-se com a declaração de módulos e bibliotecas necessárias, seguida pela definição de funções ou classes, quando aplicável. Em Python, a **formatação por indentação** substitui delimitadores explícitos, como chaves ou ponto e vírgula. O padrão recomendado é usar quatro espaços em branco para cada nível de identação. Esta simplicidade na estruturação, aliada à legibilidade do código, é uma característica distintiva que contribui para a popularidade e acessibilidade do Python. Por exemplo, o seguinte *script* hello.py mostra na tela Hello World!

```
#define as funcoes
def main:
    #comandos
    print ("Hello World!")


#inicia execucao
main()
```

2.1.2 Tipos de Dados

Variáveis são elementos essenciais para armazenar dados em Python. Diferentemente de algumas linguagens, Python possui tipagem dinâmica, o que significa que o tipo de uma variável é determinado em tempo de execução, e não durante a declaração. Ao atribuir um valor, Python automaticamente associa o tipo mais apropriado com base no conteúdo atribuído. A linguagem suporta uma ampla variedade de tipos de dados básicos, incluindo valores booleanos (bool), inteiros (int), pontos flutuantes (float), números complexos (complex) e *strings* (str).

Na manipulação, análise estatística e visualização de dados, a abstração em tipos de dados numéricos e categóricos permite que se aplicam técnicas apropriadas, otimizando o processamento, interpretação e modelagem dos dados. **Dados numéricos** representam valores quantitativos, podendo ser contínuos ou discretos. Eles são passíveis de operações matemáticas, estatísticas descritivas e técnicas de análise numérica. Por outro lado, **dados categóricos** representam categorias ou grupos, podendo ser nominais (sem ordem específica) ou ordinais (com ordem). Esses dados são submetidos a operações como contagem, cálculo de frequência e técnicas de visualização categórica.

Para simplificar a análise de dados em Python, a linguagem incorpora ainda cinco estruturas de dados básicas. Essas estruturas facilitam a organização e manipulação de informações, sendo frequentemente diferenciadas entre dados numéricos e categóricos. Cada uma delas oferece métodos específicos para a manipulação de seus elementos, conforme detalhado a seguir[100]:

Tuplas : São estruturas de dados **imutáveis** que podem armazenar elementos de dife-

rentes tipos ordenados. Os elementos são colocados entre parênteses , separados por vírgulas. Para os objetos da estrutura tupla, aplica-se os métodos count(element) e index(element).

Listas : São estruturas de dados **mutáveis** que podem armazenar elementos de diferentes tipos ordenados e podem ser acessados por meio de índices. Os elementos são colocados entre colchetes

, separados por vírgulas. Dos métodos associados aos objetos da estrutura lista destacam-se append (element), copy(), insert(position, element), pop(position), reverse(), e sort().

Dicionários : São estruturas de dados **mutáveis** que armazenam elementos como pares chave-valor, e cada elemento é associado a uma chave única. Os elementos são colocados entre chaves {}, com a chave e o valor separados por dois pontos ':'. Os métodos aplicáveis sobre objetos deste tipo de estrutura incluem fromkeys(keys, value), get(key, value), items(), keys(), values(), pop(key), popitem(), e setdefault(key,value). É possível converter uma variável do tipo dicionário para o formato tabular DataFrame, onde as chaves do dicionário se tornam os nomes das colunas e as listas (valores) associadas se tornam as colunas de dados, através da função DataFrame() da biblioteca pandas, uma das mais utilizadas em ciência de dados com Python (Seção 2.1.7.

Conjuntos : São coleções não ordenadas de elementos únicos, sobre os quais são aplicáveis as operações de conjunto,union(setB), intersection(ser1, set2, ..., setn), issubset(setB), symmetric_difference(setB), difference_update(setB), difference(setB). Incluem-se também os métodos de manipulação dos elementos de um conjunto, como add(element), clear(), discard(element) e remove(element. Esses elementos são colocados entre chaves {}, separados por vírgulas.

O seguinte trecho de códigos ilustra a definição de variáveis dos 4 tipos básicos de estruturas:

```
listaP = [0, 0, 1, 2, 3, 3]
tuplaP = ('Joana', 'Maria', 'Edna')
dicionarioP = {'a':1, 'b':2,'c':3}
conjuntoP = {0, 1, 2, 3}
```

A flexibilidade na manipulação de tipos de dados é uma característica distintiva do Python, permitindo que as variáveis mudem de tipo ao longo do programa conforme as necessidades evoluem.

2.1.3 Operações Lógico-Aritméticas e Relacionais

Python suporta operações aritméticas padrão, como adição (+), subtração (-), multiplicação (*), divisão (/), e exponenciação (**). Além disso, o operador de módulo (%) retorna o resto da divisão. Para avaliar condições lógicas e controlar fluxos de execução condicionais, Python dispõe de operações lógicas que incluem and, or e not. E para comparações entre valores numéricos (Seção 2.3 em [100]), existem operadores relacionais, como igual a (==), diferente de (!=), menor que (<), maior que (>), menor ou igual a (<=) e maior ou igual a (>=). Python também tem operadores lógicos (Seção 2.3 em [100]), *bit* a *bit*, como AND *bit* a *bit* (&), OR *bit* a *bit* (|), XOR *bit* a *bit* (^), NOT *bit* a *bit* (~), deslocamento para esquerda (<<), e deslocamento para direita (>>). Finalmente, para atribuir o resultado de uma expressão a uma variável, usa-se o operador =. A variável assume o tipo de dado do resultado recebido.

2.1.4 Comandos Condicionais

Em Python, os comandos condicionais são utilizados para controlar o fluxo do programa com base em condições lógicas. Eles são fundamentais para criar lógica de decisão em programas, permitindo que diferentes blocos de códigos sejam executados com base em condições específicas. Os principais comandos de condicionais incluem if, elif, e else:. Python ainda suporta uma forma compacta de expressão condicional conhecida como operador ternário:

```
variavel = <valor 1> if <condição 1> else <valor 2>
```

2.1.5 Comandos de Laços de Repetição

Em Python, os comandos de laços de repetição permitem executar um bloco de código várias vezes com base em uma condição específica. Os dois principais comandos de laços de repetição são while e for. Há ainda o comando break que interrompe a execução do laço de repetição mais próximo e o comando continue para desviar o restante do código do laço mais próximo e passar para a próxima iteração. Python permite ainda usar o comando else: para executar um bloco de instrução quando a condição do laço se torna falsa.

Além dos laços de repetição, Python oferece uma técnica versátil chamada fatiamento (em inglês *slicing*) para acessar subconjuntos de dados e realizar operações rápidas sem a necessidade de laços explícitos. O fatiamento é feito usando a notação de colchetes ([]) com a sintaxe start:stop:step. Na sintaxe de fatiamento, start é o índice inicial da fatia (inclusivo), stop é o índice final da fatia (exclusivo), e step é o intervalo entre os elementos. Se omitidos, o padrão é do início até o final da sequência com um intervalo igual a 1. O fatiamento permite extrair partes de uma sequência, como listas, tuplas e *strings*, de maneira eficiente e clara. Por exemplo, para obter uma sublista dos elementos dos índices

2 a 5, utiliza-se my_list[2:5]. Para extrair todos os elementos a partir do índice 2, usa-se my_list[2:], e para obter apenas os elementos em índices pares, usa-se my_list[::2].

2.1.6 Funções Incorporadas

As funções incorporadas (em inglês, built-in functions) são funções básicas da linguagem que já fazem parte do Python e estão sempre disponíveis para uso, sem a necessidade de importar módulos ou pacotes adicionais. Em Python, muitas dessas funções são implementações nativas escritas em C, o que garante o bom desempenho e eficiência na execução. Algumas das funções incorporadas mais comuns e amplamente utilizadas em Python incluem funções de entrada/saída, como print(object) e input(message), e funções de manipulação de dados, como type(object), round (number, digits), len(object), max(object), e min(object). A chamada de função em códigos segue geralmente o formato:

```
<nome_da_função> (arg1=val1, arg2=val2, ...)
```

onde nome_da_funcao é o nome da função que está sendo chamada, e arg1, arg2, etc., são os argumentos da função, aos quais são atribuídos valores específicos, como val1, val2, etc. A ordem dos argumentos é importante ao chamar uma função em Python. Se segui-la corretamente, pode omitir os nomes dos argumentos. Isso é conhecido como **chamada de função posicional**. Se a função tiver muitos argumentos ou se quiser fornecer apenas alguns dos argumentos e omitir outros, fornecer os nomes dos argumentos explicitamente pode tornar o código mais legível e menos propenso a erros.

2.1.7 Funções Importadas

Funções importadas em Python são aquelas que não fazem parte do conjunto padrão de funções incorporadas na linguagem, mas estão contidas em bibliotecas externas. De acordo com McKinney [100], as bibliotecas em Python essenciais para análise de dados são NumPy, pandas, matplotlib e SciPy. Para utilizar as funções dessas bibliotecas, é necessário importar, através do comando import, a biblioteca correspondente antes de seu uso no *script* ou programa.

A biblioteca NumPy facilita a criação e manipulação de arranjos (matrizes) de várias dimensões, o que é fundamental para diversas áreas da computação científica e análise de dados. Por exemplo, para criar um vetor de valores de 0 a 10 com um passo de 2, podemos importar os métodos da biblioteca numpy e utilizar a função arange implementada nele:

```
import numpy as np
arranjo = np.arange(0,10,2)
```

Note que, para facilitar o uso, foi renomeada no *script* a biblioteca numpy para np.

A biblioteca pandas é uma ferramenta essencial para manipulação e análise de dados em Python, oferecendo estruturas de dados flexíveis e eficientes. Uma das estruturas mais empregadas na análisa

de dados, DataFrames, é implementada em pandas. Diferente das sequências incorporadas conforme vimos na Seção 2.1.2, os DataFrames são estruturados em formato tabular, semelhante a uma planilha, com linhas e colunas, e equipados com uma ampla gama de métodos e funções. Adicionalmente, essa estrutura é altamente compatível com outras bibliotecas populares de análise de dados em Python, como NumPy, Matplotlib e Scikit-learn, permitindo uma análise abrangente dos dados.

Entre as numerosas funções integradas para manipular e analisar dados tabulares de forma eficiente, destacam-se a função filter, que permite a seleção de dados com base em condições específicas, simplificando a extração de informações relevantes. A função group_by facilita a segmentação de dados em grupos com base em variáveis definidas, enquanto a função summarize gera estatísticas resumidas para cada grupo segmentado. Além disso, o operador >> (*pipe*) proporciona uma abordagem encadeada, simplificando a aplicação sucessiva de operações e resultando em código mais conciso e legível.

Suponha que tenha um conjunto de dados (¡dados¿) do tipo DataFrame e deseje filtrá-lo sob uma ¡condicao¿ específica. Em seguida, queira agrupar ¡dados¿ por uma variável ¡variavel¿ e, finalmente, calcular a média cujo resultado é atribuído à variável ¡outra_variavel¿. Usando o operador "‘", pode-se encadear essas operações de maneira mais clara, como mostra a seguinte sequência de instruções cujo resultado sobre o conjunto de dados original ¡dados¿ é atribuído à variável ¡result¿:

```
import panda as pd
<result> = (<dados> \
.filter (<condicao>)  \           #selecionar colunas
.group_by ("<variavel>")  \       #agrupar os valores pelas variáveis especificadas
.summrize (media=("<outra_variaval>", "mean"))  #cômputo da média dos valores de <outra variav
```

Na área de visualização, além do matplotlib, que oferece funções básicas de renderização de gráficos 2D, destacam-se os pacotes Seaborn, Plotly e Plotnine, que é uma implementação em Python da gramática de gráficos do ggplot2, uma biblioteca de visualização de dados muito popular em R. A gramática de gráficos, que exploraremos no Capítulo 3, é uma abordagem que permite construir gráficos de forma modular, combinando diferentes camadas de elementos visuais. O Plotnine utiliza o Matplotlib como base para gerar os gráficos. Uma análise comparativa desses quatro pacotes é realizada por Dennis e discutida em [**?**].

2.1.8 Funções Personalizadas

Em Python, a capacidade de criar funções personalizadas flexibiliza a programação, permitindo o encapsulamento de lógica específica em unidades reutilizáveis de código. Essa característica facilita a modularização do código, tornando-o mais organizado, legível e fácil de dar manutenção.

A sintaxe básica para criar funções em Python é direta. Usamos a palavra-chave def seguida pelo nome da função e seus parâmetros entre parênteses. O bloco de código pertencente que define a função é indentado, e o uso da palavra-chave return é opcional para especificar o valor que a função deve retornar (Apêndice em [100]). Segue-se um exemplo:

```
def nome_da_funcao(argumento1, argumento2, ...):
    # corpo da função
    # pode incluir comandos, expressões, etc.
    resultado = argumento1 + argumento2
    return resultado
```

2.1.9 Conjuntos de Dados Incorporados

Para padronizar as análises estatísticas e permitir que os usuários se concentrem nos métodos estatísticos em vez de perder tempo buscando por dados adequados, a Seção 2.2.6 mostra que o R incorpora uma série de conjuntos de dados como conjuntos básicos incorporados (em inglês, *builtin*). Python, por sua vez, não inclui conjuntos de dados na instalação padrão do interpretador, seguindo a filosofia da linguagem que prioriza a simplicidade e a minimalidade na instalação.

O Python se destaca pela sua arquitetura modular, permitindo que desenvolvedores selecionem as bibliotecas e pacotes específicos para cada tarefa. Embora não ofereça conjuntoss de dados nativos, o Python se integra com bibliotecas como Scikit-learn, Seaborn, NLTK e outras, que disponibilizam conjuntos de dados prontos para uso. Após a instalação da biblioteca desejada, o acesso e a utilização desses conjuntos de dados são facilitados. Por exemplo, a biblioteca Scikit-learn oferece uma variedade de conjuntos de dados para tarefas de aprendizado de máquina, enquanto a Seaborn facilita a visualização dos dados disponibilizados por outras bibliotecas.

2.2 R

A linguagem R é destinada para estatística e análise de dados, oferecendo uma ampla gama de ferramentas e recursos incorporados especificamente para esse fim. Desenvolvida inicialmente por Ross Ihaka e Robert Gentleman na Universidade de Auckland, Nova Zelândia, durante a década de 1990, a linguagem R teve sua origem em projetos acadêmicos para oferecer uma plataforma robusta e flexível para análise estatística e visualização de dados.

Uma das principais peculiaridades da linguagem R é o seu foco explícito em estatística e gráficos. Ela possui uma vasta coleção de pacotes e bibliotecas dedicados à EDA (do inglês *Exploratory Data Analysis*), modelagem estatística, séries temporais, aprendizado de máquina e visualização gráfica, tornando-a uma escolha popular entre estatísticos, cientistas de dados e pesquisadores em diversas áreas.

O domínio de uso da linguagem R é amplo, sendo aplicável em diversas disciplinas, como bioinformática, finanças, epidemiologia, ciências sociais, entre outras. Sua comunidade ativa e engajada contribui constantemente com novos pacotes, ampliando ainda mais suas capacidades. Esse desenvolvimento contínuo garante que R permaneça na vanguarda das tecnologias de análise de dados.

Quanto à curva de aprendizado, a linguagem R pode apresentar desafios iniciais para aqueles que não têm experiência prévia em programação ou estatística. No entanto, a comunidade R fornece uma abundância de recursos, tutoriais e documentação para apoiar os iniciantes. Além disso, a linguagem R é reconhecida por sua sintaxe expressiva e rica em recursos, o que facilita a criação de análises estatísticas complexas. Kabacoff compartilha em [44] os seus conhecimentos sobre R de forma didática.

Para programar em R, você precisa instalar o interpretador R, que é o programa que executa o código R. Embora seja possível programar em R usando apenas o interpretador, o RStudio é um ambiente de desenvolvimento integrado (em inglês *Integrated Development Environment*) altamente recomendado, pois oferece recursos que facilitam muito a programação, como edição de código, visualização de dados e ferramentas de depuração. Uma referência rápida sobre RStudio se encontra em [78].

2.2.1 Estrutura Básica

A estrutura básica de um *script* R, tipicamente de extensão .R, segue uma abordagem simples e direta, tornando a linguagem acessível mesmo para iniciantes. Inicialmente, o *script* pode começar com comentários, marcados pelo símbolo #, que são úteis para documentar o código. Em seguida, são incluídas as importações de bibliotecas ou pacotes necessários para a análise, usando a função library().

O corpo principal do *script* contém as instruções de código. As operações estatísticas, manipulações de dados e visualizações gráficas são realizadas por meio de funções incorporadas específicas, como summary(), ggplot(), ou aquelas provenientes de pacotes especializados. O *script* pode incluir ainda a impressão ou exportação dos resultados finais, proporcionando uma visão clara dos resultados obtidos durante a execução das instruções.

Diferente do Python, o escopo de uma função em R é delimitado por chaves, seguindo uma sintaxe semelhante à de C, em vez de depender da indentação. Embora a indentação seja importante para a legibilidade do código em R, ela não desempenha um papel estrutural no determinismo do escopo. Abaixo segue-se um exemplo:

```
# Este é um comentário no script R
# Comentários são úteis para documentação

# Importar pacotes necessários
library(ggplot2)  # Pacote para gráficos
```

```
# Criar dados de exemplo
dados <- data.frame(
  X = c(1, 2, 3, 4, 5),
  Y = c(2, 4, 6, 8, 10)
)

# Realizar análise exploratória
resumo_dados <- summary(dados)
print(resumo_dados)

# Criar um gráfico de dispersão usando ggplot2
grafico <- ggplot(dados, aes(x = X, y = Y)) +
  geom_point() +
  labs(title = "Gráfico de Dispersão", x = "Eixo X", y = "Eixo Y")

# Exibir o gráfico
print(grafico)

# Salvar o gráfico em um arquivo
ggsave("grafico_dispersao.png", plot = grafico, width = 6, height = 4)
```

Uma visão básica sobre codificação de *scripts*/programas em R é apresentada em [42, 44]. Especificamente relativa à análise de dados em R pode ser encontrada em [102].

2.2.2 Tipos de Dados

A linguagem R oferece uma ampla variedade de tipos de dados, cada um projetado para atender a necessidades específicas. Entre os tipos básicos estão números inteiros (integer), pontos flutuantes (numeric), caracteres (character), lógicos (logical) e complexos (complex). Em linha com a abordagem dinâmica de Python (Seção 2.1.2), os tipos de dados em R são dinâmicos, o que significa que o próprio R automaticamente determina o tipo com base no conteúdo atribuído a uma variável ao criá-la. Além disso, uma variável pode mudar de tipo durante a execução do programa. Análoga a Python, a classificação dos dados em numéricos e categóricos é feita quanto à classe de operações aplicáveis sobre estes dados.

Além dos tipos básicos, o R oferece estruturas de dados mais complexas que facilitam a organização e manipulação de informações. Essas estruturas ampliam significativamente as capacidades da linguagem para lidar com uma ampla gama de dados, o que contribui para sua adaptabilidade e versatilidade em diversas aplicações de Ciência de Dados, desde a análise exploratória até a modelagem estatística

avançada. Para criar e manipular essas estruturas complexas, o R disponibiliza uma variedade de funções específicas para cada tipo de estrutura. Incluem-se entre as estruturas básicas [44]:

Vetores : São estruturas que armazenam elementos de mesmo tipo de dados. Podem ser criados com a função de concatenação básica c() que combina valores em vetores, e os valores dos elementos determinam automaticamente o tipo de dados do vetor resultante. A função c() permite ainda a combinação de vários vetores, expansão de vetores, concatenação de vetores com diferentes tipos de dados e remoção de valores não disponíveis. É importante notar que uma *string* em R é geralmente representada como um vetor de caracteres. Portanto, para criar uma variável contendo uma *string*, basta delimitar um vetor de caracteres entre aspas simples ou duplas em R.

Matrizes : São estruturas de dados bidimensional **imutáveis** que contém elementos dispostos em linhas e colunas. Esses elementos devem ser do mesmo tipo de dados e podem ser acessados usando índices numéricos para especificar a linha e a coluna. **A indexação em R começa em 1, não em 0,** como em algumas outras linguagens de programação. Podem ser criadas com a função básica matrix(¡dados¿, nrow = ¡número de linhas¿, ncol = ¡número de colunas¿, byrow = ¡preenchimento por linha/coluna¿, dimnames = ¡nomes opcionais para as dimensões da matriz¿). A função cbind() e rbind() permitem combinar vetores em matrizes por coluna ou linha, respectivamente. Os métodos comumente usados são acessos a elementos, transposição, concatenação de matrizes e operações matriciais aritméticas. Quando precisa organizar os dados em matrizes de mais de 2 dimensões, utiliza-se a função básica array.

Listas : São estruturas de dados que podem conter elementos de diferentes tipos, como números, *strings*, vetores, matrizes e até outras listas. Cada elemento pode ser acessado por um nome ou índice. Podem ser criadas com a função básica list().

Data Frames [1]: São estruturas tabulares bidimensionais que podem conter diferentes tipos de dados em colunas. Podem ser criadas com a função básica data.frame(). Essas estruturas são semelhantes a tabelas de banco de dados ou planilhas e são frequentemente utilizadas para armazenar tabelas de dados. Uma versão aprimorada e moderna do data.frame() é a estrutura tibble, fornecida pelo pacote tibble [57], integrado no pacote tidyverse. Essa nova versão oferece funcionalidades adicionais e uma interface mais amigável para manipulação de dados.

Factors : São estruturas de dados que representam variáveis categóricas. **Variáveis categóricas** são aquelas que possuem um número finito e predefinido de categorias ou

[1]Python não tem esse tipo de dado incorporado, mas é fornecido pelo pacote pandas como mostra a Seção 2.1.7). Esse pacote contém a função DataFrame capaz de transformar o tipo de dado Dicionário no tipo de dado Data Frame. Por exemplo, pode-se passar uma variável dados do tipo Dicionário para uma variável df do tipo Data Frame através da atribuição df = pandas.DataFrame (dados).

níveis, como cores, frutas, níveis de educação, entre outros. Os fatores podem ser ordenados ou não ordenados. Em fatores ordenados, os níveis têm uma ordem específica, enquanto em fatores não ordenados, os níveis não têm uma ordem específica. A função que cria fatores em R é a função factor(). Ela é utilizada para converter vetores de dados em fatores, permitindo especificar os níveis e a ordenação dos fatores, se necessário. Para mostrar a contagem de observações em cada categoria, pode-se usar a função nativa summary().

Sequências : São estruturas que contêm sequências de números com incrementos e quantidade de elementos específicos. Podem ser criadas com a função básica seq(). Quando se trata de uma sequência linear de números inteiros de um ponto de início para um ponto de fim, com um incremento padrão de 1, pode-se usar o operador de sequência :, como 1:10 que gera a sequência de 1 a 10 com incremento de 1.

O seguinte trecho de códigos ilustra a definição de variáveis dos tipos de estruturas suportadas em R:

```
vetorR <- c(0, 1, 1.5, 2,3)
listaR <- list(nome='Pedro', RA=12345, idade=23, notas=c(8.0, 7.8, 9.5))
stringR = "Olá, R!"
matrizR <- matrix(1:6, nrow = 3, ncol = 2, byrow = FALSE, dimnames=NULL)
dataFrameR <- data.frame(nome=c("Maria","Pedro","Clara"),idade=c(19, 20, 19))
factorR <- factor(c("insuficiente", "insuficiente", "suficiente", "suficiente"),
    levels=c("suficiente","insuficiente"))
```

2.2.3 Operações Lógico-Aritméticas e Relacionais

Assim como Python, R oferece suporte a uma variedade de operações lógico-aritméticas e relacionais. Entre as operações aritméticas, incluem-se adição (+), subtração (-), multiplicação (*), divisão (/), potenciação (^), resto da divisão (%%) e divisão inteira (%/%). Os operadores relacionais permitem comparações como igual a (==), diferente de (!=), menor que (<), maior que (>), menor ou igual a (<=) e maior ou igual a (>=) [44]. R dispõe operadores lógicos para avaliar elementos em dois vetores de mesma quantidade de valores numéricos ou lógicos, tais como AND (&), OR (|), NOT (!) e XOR ((xor())). Além disso, oferece avaliações condicionais otimizadas, interrompendo a avaliação assim que uma condição é avaliada como falsa, utilizando operadores AND (&&) e OR (||). R ainda suporta operações *bit* a *bit*, como AND (bitwAnd()), OR (bitwOr()) e XOR (bitwXor()). Para atribuir o resultado de uma expressão a uma variável em R, é utilizado o operador <-. Dessa forma, a variável assume automaticamente o tipo de dado do resultado recebido.

2.2.4 Comandos Condicionais

Em R, os comandos condicionais são usados para executar blocos de código com base em condições lógicas. Os principais comandos condicionais em R são:

if : esse comando é usado para executar um bloco de código se uma condição for verdadeira

else : é usado em conjunto com if como uma alternativa a ser executada se a condição do if for FALSA, como ilustra o seguinte trecho de instruções

```
if (condicao) {
  #codigo a ser executado se a condicao for verdadeira
} else {
  #codigo a ser executado se a condicao for falsa
}
```

else if : é usado em conjunto com if para verificar condições adicionais se as condições anteriores forem falsas.

```
if (condicao1) {
  #codigo a ser executado se a condicao1 for verdadeira
} else if (condicao2) {
  #codigo a ser executado se a condicao2 for verdadeira
} else {
  #codigo a ser executado se as condicoes anteriores fore falsas
}
```

Similar a Python, R também suporta o operador condicional ternário para expressar condicionais de forma mais compacta

```
variavel <- ifelse (condicao, valor_se_verdadeira, valor_se_falsa)
```

2.2.5 Comandos de Laços de Repetição

Os principais comandos de laços de repetição em R são:

for : é usado para iterar sobre uma sequência criada pela função seq().

```
for (variavel in sequencia) {
  # Codigo a ser repetido para cada elemento da sequência
}
```

while : executa um bloco de código repetidamente enquanto a condição especificada for verdadeira.

```
while (condicao) {
  # Codigo a ser repetido enquanto a condicao for verdadeira
}
```

repeat : repete a execução de um bloco de código infinitamente. Pode-se especificar uma condição, que ao ser satisfeita, cause a interrupção da repetição pelo comando break.

```
repeat {
  # Codigo a ser repetido
  if (condicao) {
    break
  }
}
```

2.2.6 Funções Incorporadas

R é equipado com uma variedade de funções incorporadas por padrão. Com ênfase na análise de dados, a biblioteca de estatísticas padrão, conhecida como base ou stats, é inclusa na instalação base do R e contém uma variedade de funções estatísticas e métodos amplamente empregados na análise de dados. A sintaxe básica de chamada de funções em R é semelhante à chamada em Python. Em R, a chamada de função é posicional e segue o formato:

```
<nome_da_função> (arg1 = val1, arg2 = val2, ...)
```

Assim como em Python, R permite a omissão dos nomes dos argumentos nas chamadas de funções, utilizando a posição dos argumentos para determinar seus valores.

Além das funções associadas a tipos de dados estruturais mais complexos, conforme discutido na Seção 2.2.2, estão entre as funções incorporadas as seguintes funções amplamente utilizadas:

c() (Concatenar): É usada para criar vetores, combinando elementos em uma única estrutura, como mostrada na Seção 2.2.2.

print() (Imprimir): mostra o conteúdo de variáveis ou resultados na saída como no Console.

length() (Comprimento): Retorna o número de elementos em um objeto, como um vetor.

sum() (Soma): Calcula a soma dos elementos de um vetor.

mean() (Média): Calcula a média dos elementos de um vetor.

sd() (Desvio padrão): Calcula o desvio padrão de um vetor.

max() e min() (Máximo e Mínimo): Retornam o valor máximo e mínimo de um vetor, respectivamente.

rep() (Repetição): Replica elementos para criar vetores com padrões repetidos.

unique() (Únicos): Retorna os valores únicos de um vetor, gerando um novo vetor sem repetições.

library() (Biblioteca): É usada para carregar um pacote específico no ambiente de trabalho R, expandindo as funcionalidades da linguagem.

require() (Requisição): É semelhante à função library(), mas com um comportamento adicional de instalação automática caso o pacote especificado não estiver instalado.

2.2.7 Funções Importadas

Em R, as funções importadas se referem à utilização de bibliotecas externas para expandir as capacidades da linguagem. A importação de bibliotecas é feita através da função library() ou require(). Uma vez importada uma biblioteca, pode-se usar as funções integradas nela.

Por exemplo, a biblioteca dplyr [107] é amplamente utilizada para manipulação de dados organizados em tabelas. Os data frames são um tipo comum de tabela em R, e o dplyr oferece funções otimizadas para trabalhar com eles (Seção 2.2.2). Para importá-lo numa sessão de trabalho, pode-se usar o seguinte comando

```
library(dplyr)
```

e utilizar as suas funções, como a de filtragem (filter()), a de agrupar os dados com base em uma ou mais colunas (group_by()), funções de estatística de resumo (summarize()), e seleção de colunas específicas (select()), sobre os dados. O pacote também dispõe o operador %>% (pipe) que encadeia operações em uma sequência lógica. O seguinte bloco de instruções tem o efeito equivalente ao efeito da sequência de instruções sobre o conjunto de dados ¡dados¿ original apresentado na Seção 2.1.7:

```
<result> <- <dados> %>%
filter (<condicao>) %>%
group_by (<variavel>) %>%
summarize (media = mean (<outra_variavel>))
```

A biblioteca amplamente utilizada para visualização de dados em R é o ggplot2 [105]. Conhecida por sua capacidade de criar gráficos estatísticos de maneira intuitiva a partir de dados organizados de forma "arrumada"(*tidy data*), o ggplot2 se destaca no cenário da visualização de dados. Sua principal função, ggplot(), introduzida na Seção 3.5, é fundamentada numa gramática dos gráficos, adotando uma abordagem declarativa na construção de gráficos. Através dessa abordagem, os usuários especificam apenas os elementos visuais desejados e as transformações nos dados, simplificando significativamente o processo de criação de gráficos estatísticos complexos.

O tidyverse [**?**] é um conjunto de pacotes do R que trabalham em conjunto para facilitar a manipulação, visualização e análise de dados. Sua proposta é oferecer um fluxo de trabalho coeso para dados organizados de forma tidy (Capítulo 7). Além do dplyr e do ggplot2, o tidyverse integra pacotes como o tidyr, para organização de dados, e o readr, para importação e exportação.

2.2.8 Funções Personalizadas

Análogo a Python, R também tem a capacidade de criar funções personalizadas. A sintaxe para a definição de uma nova função em R é, porém, diferente. Em R, usa-se a palavra-chave function para começar a definição da função, enquanto em Python, usa-se a palavra-chave def. Além disso, em Python, a indentação é fundamental para indicar o bloco de código que define a função, enquanto em R, utiliza-se chaves {} para delimitar o bloco de código de definição, como ilustra o seguinte exemplo de código [102]:

```
nome_da_funcao <- function(argumento1, argumento2, ...) {
  # corpo da função
  # pode incluir comandos, expressões, etc.
  resultado <- argumento1 + argumento2
  return(resultado)
}
```

2.2.9 Conjunto de Dados Incorporados

Ao contrário de Python, R incorpora alguns conjuntos de dados na instalação padrão do interpretador. Entre eles, destacam-se mtcars e iris (Figura 1.1). Isso está alinhado com os objetivos e o histórico de uso da linguagem, que visa facilitar o aprendizado, fornecer exemplos práticos, suportar tradições estatísticas e facilitar a exploração de dados para análises. Esses conjuntos de dados incorporados tornam a linguagem mais acessível e pronta para uso imediato em análises estatísticas e científicas. A referência [89] fornece uma guia rápida de uso desses conjuntos de dados incorporados.

2.3 Documentação

Tanto R quanto Python, junto com o Markdown [62], oferecem poderosas capacidades para a criação de documentos interativos que integram código executável, texto formatado e elementos visuais. Essa abordagem, que promove a combinação de análise de dados e comunicação de resultados de maneira integrada, é crucial para profissionais que buscam comunicar *insights* complexos de maneira acessível e eficaz, sem a preocupação com possíveis equívocos gerados no processo de "copiar e colar".

Markdown é uma linguagem de marcação leve e fácil de aprender, projetada para formatação simples de texto. Criada por John Gruber em 2004 com intuito de formatar texto para a *web*, a sintaxe

do Markdown é projetada para ser intuitiva e legível, permitindo que os usuários criem documentos formatados de forma rápida e eficiente, sem a necessidade de aprender códigos complicados ou extensos. Com o Markdown, pode-se adicionar formatação básica, como negrito, itálico, títulos, listas, *links* e imagens, usando apenas caracteres simples e intuitivos. Por exemplo, para adicionar um título, usa-se um ou mais símbolos "#"seguido pelo texto do título. Para criar uma lista, basta começar cada item com um asterisco ou um número seguido de um ponto. É até possível inserir fórmulas matemáticas usando a notação LaTeX, como mostra em [112].

A geração de documentação usando Markdown em R ou Python requer o uso do ambiente R Markdown e Jupyter Notebook, respectivamente, principalmente por causa de sua integração e facilidade de uso para diversos formatos, garantindo uma experiência interativa durante o desenvolvimento. Especificamente, documentos em formato HTML podem incorporar componentes interativos através do uso do pacote shiny [85]. R Markdown é especificamente voltado para a linguagem R, enquanto Jupyter Notebooks oferece suporte a várias linguagens, com ênfase especial em Python. Ambas as linguagens são amplamente adotadas em ambientes acadêmicos e industriais. A escolha entre as duas ferramentas frequentemente está associada à preferência do usuário e à linguagem de programação predominante em seu ambiente de trabalho. Independentemente da escolha feita, ambas proporcionam uma maneira robusta de integrar análise de dados, código e comunicação de resultados em um único ambiente colaborativo e interativo, como detalharemos nas seções subsequentes.

2.3.1 R Markdown

Em R, uma extensão de Markdown, conhecida por **R Markdown**, é amplamente usada para criar documentos dinâmicos, relatórios e apresentações. Com R Markdown, os usuários podem criar relatórios dinâmicos, apresentações, documentos e outros tipos de saída, combinando a simplicidade do Markdown com o poder estatístico do R. Os arquivos em R Markdown geralmente têm a extensão .Rmd ou .rmd.

A estrutura básica de um arquivo Rmd consiste de uma coleção de 3 tipos de células:

Célula de Metadados YAML : O início de um arquivo Rmd geralmente contém metadados YAML (do inglês *YAML Ain't Markup Language*) entre os dois "três hifens —". Esses metadados incluem informações sobre o documento, como título, autor, data e opções de renderização.

Células de código (em inglês, *chunk*): O corpo do documento consiste em *chunks* de código R delimitados por três crases (‘‘‘). Esses *chunks* de código podem conter código R que serão executados quando o documento for renderizado. Os resultados podem ser automaticamente inseridos no documento após a execução do *chunk* de código correspondente, facilitando a visualização dos resultados ao lado do código

que os gerou.

Célula de Markdown : O restante do documento consiste em texto formatado em Markdown. Isso inclui seções, cabeçalhos, listas, *links*, formatação de texto (negrito, itálico), entre outros recursos do Markdown.

A inclusão dos dados relativos a um *chunk* no documento final é controlada pelos 5 argumentos, echo (código em si), results (resultados da execução do código), error (erros), warning (avisos) e message (mensagens). O padrão das opções é

```
‘‘‘{r <nome_do_chunk>, echo=TRUE, results= markup , error=TRUE, warning=TRUE,
message=TRUE}
‘‘‘
```

Na prática, são usualmete omitidos o código, erros, avisos e mensagens, setando FALSE para essas quatro opções. Opcionalmente, pode-se setar globalmente as cinco opções para todos os *chunks* com um *chunk* no início do *script*, como

```
‘‘‘{r setup, include=FALSE}
knitr::opts_chunk$set(echo = FALSE, results='markup', error=FALSE, warning=FALSE,
message=FALSE)
‘‘‘
```

Três pacotes facilitam a geração de um documento elaborado em R Markdown: rmarkdown, knitr e yaml. O rmarkdown é essencial para compilar documentos, gerenciar configurações e executar código incorporado. Ele fornece funcionalidades básicas para a manipulação de documentos R Markdown. O knitr, por sua vez, é acionado durante o processo de "tricotar" (*knitting*), integrando o código e os resultados da execução no documento final, que pode ser gerado em vários formatos, como HTML, LaTeX e os compatíveis com o Word. O yaml, que lida com dados no formato YAML (do inglês *YAML Ain't Markup Language*), é utilizado para manipulação e processamento de parâmetros de configuração do documento R Markdown.

O ambiente de desenvolvimento integrado (IDE), ou o ecossistema, RStudio fornece suporte integrado para trabalhar com R Markdown, permitindo que os usuários criem e executem documentos R Markdown diretamente nele. Os pacotes rmarkdown e knitr geralmente vêm pré-instalados. O yaml também é comumente instalado com a maioria das versões do RStudio. No caso de ausência desses pacotes, é possível obter suas próprias cópias no CRAN (do inglês *Comprehensive R Archive Network*) usando os comandos

```
install.packages("rmarkdown")
install.packages("knitr")
install.packages("yaml")
```

Antes de usar as funcionalidades desses pacotes, é necessário carregá-las por meio das seguintes instruções no *script* de R Markdown:

```
library(rmarkdown)
library(knitr)
library(yaml)
```

Uma visão mais detalhada sobre R Markdown pode ser encontrada em [102] ou no material *online* em [27].

Para gerar documentos PDF a partir de arquivos R Markdown, são necessárias duas ferramentas adicionais, Pandoc [40] e LaTex [49], se quisermos que a conversão de Markdown para PDF seja via LaTeX. O Pandoc é um conversor de documentos universal que permite transformar arquivos R Markdown em diversos formatos, incluindo PDF – por padrão, via LaTex. Ele é essencial para a geração de PDFs e geralmente é instalado automaticamente com o RStudio. O LaTeX é um sistema de composição de textos de alta qualidade, amplamente utilizado para a criação de documentos PDF com formatação profissional. Para gerar PDFs a partir de arquivos R Markdown, o LaTeX precisa estar instalado no sistema.

Uma evolução de R Markdown é o Quarto [66], , com recursos aprimorados e maior flexibilidade. Uma das grandes vantagens do Quarto é a sua capacidade de gerar documentos PDF de alta qualidade de forma mais simples e intuitiva do que o R Markdown, pois ele já inclui o Pandoc e oferece integração facilitada com o LaTeX. Além disso, o Quarto oferece suporte a diversos formatos de saída, como HTML, Word e PowerPoint, tornando a criação de documentos mais versátil e acessível.

2.3.2 Jupyter Notebooks

Em Python, o Jupyter[2] Notebook é uma interface específica que permite a utilização de Markdown e suporta a execução de código em diversas linguagens, como Python, R, Julia, entre outras, oferecendo flexibilidade para trabalhar em ambientes multilínguas. Os documentos interativos, também conhecidos por *notebooks*, gerados pelo Jupyter Notebooks[3] são identificados pela extensão .ipynb. Uma das características distintivas do Jupyter Notebook é o suporte para células de código, que permitem a intercalação de texto formatado em Markdown com blocos de código executável.

Para adicionar um novo bloco de código, como um código em Python, no Jupyter Notebook, basta criar uma nova célula e selecionar o tipo de célula como "Code"[84]. Pode-se então digitar o código Python diretamente na célula, sem a necessidade de demarcações adicionais. O ambiente reconhecerá automaticamente que o código inserido nessa célula será interpretado como Python, facilitando o

[2]O Jupyter é o projeto que engloba várias interfaces, incluindo o Jupyter Notebook, JupyterLab e outros, mas é o Jupyter Notebook que é comumente usado para trabalhar com Markdown e código executável em células interativas. O nome "Jupyter" é uma combinação de três linguagens de programação principais suportadas: Julia, Python e R [23].

[3]Note que o termo "Jupyter Notebook" pode ser tanto o formato de arquivo usado para criar documentos interativos quanto o ecossistema que fornece a aplicação interativa para trabalhar com esses documentos.

processo de escrita e execução de código. Para sair da edição de uma célula e passar para uma outra célula, pode-se pressionar a tecla "Esc" no teclado.

Em contraste com o ambiente do R Markdown, onde o documento é tratado como uma entidade única, combinando tanto o código R quanto o texto em Markdown, nos Jupiter Notebooks as células de código e as células de Markdown são processadas separadamente. Isso ocorre porque o Jupyter Notebook suporta várias linguagens de programação. Dessa forma, o mecanismo computacional responsável por executar o código nas células de código de um *notebook* é denominado kernel. Cada *notebook* está associado a um kernel específico, que determina a linguagem de programação e o ambiente de execução.

Para habilitar essas funcionalidades em Jupyter Notebooks, é essencial garantir a presença de pacotes específicos. Os essenciais são o próprio jupyter, um aplicativo *web* interativa projetada para simplificar a criação e o compartilhamento de documentos que incorporam código executável, e a ferramenta nbconvert para converter *notebooks* em diferentes formatos de documento, incluindo HTML, PDF e LaTeX [43].

Ao instalar o Jupyter Notebook, geralmente são incluídos os pacotes jupyter e nbconvert, que fornecem as funcionalidades básicas necessárias para criar e converter *notebooks*. Como é um aplicativo baseado na *web*, o Jupyter Notebook pode ser acessado através de qualquer navegador da *web*, como Chrome, Firefox ou Safari. Quando se executa um código num Jupyter Notebook, ele é executado no contexto do kernel associado ao *notebook*. Esse kernel é essencialmente o mecanismo computacional que executa o código e já possui acesso a todas as funcionalidades fornecidas pelo Jupyter. Uma visão introdutória a Jupyter Notebook pode ser encontrada em [67, 84].

A utilização de pacotes, como numpy para operações numéricas [21], pandas para manipulação de dados em formato tabular [97], e matplotlib para a criação de gráficos [93], é comum na edição de um *notebook* relacionado com a análise de dados. A instalação desses pacotes é realizada por meio do gerenciador de pacotes do Python, pip/pip3 (Seção 2.1):

```
pip3 install pandas
pip3 install numpy
pip3 install matplotlib
```

É recomendável realizar essa instalação antes de abrir o Jupyter Notebook. Certifique-se de ter uma versão do Python e o respectivo pip instalados antes de executar esses comandos. Dentro de um *notebook*, é necessário carregar os pacotes antes de usar suas funções, como demonstram as seguintes instruções. Além disso, é comum renomear os nomes dos pacotes para abreviações amplamente adotadas usando o operador as:

```
import pandas as pd
import numpy as np
```

```
import matplotlib as mpl
```

Note que a biblioteca matplotlib é um conjunto abrangente de pacotes gráficos que inclui funcionalidades essenciais para criação e personalização de visualizações. Além do pyplot, que oferece uma interface semelhante ao MATLAB para criar gráficos, outros subpacotes importantes são axes, colors, patches e muitos outros. Ao trabalhar em projetos específicos, é possível importar apenas o subpacote necessário em vez do pacote inteiro, o que otimiza o uso de recursos e simplifica a organização do código. Por exemplo, para importar somente as funções gráficas, podemos usar o seguinte comando de importação:

```
import matplotlib.pyplot as plt
```

Ao gerar um documento a partir de um *notebook* (arquivo de extensão .ipynb), o Jupyter Notebook oferece diversas opções para personalizar a renderização das células. Por padrão, todas as células Markdown são renderizadas, incluindo texto formatado, equações matemáticas e elementos multimídia. As células de código geralmente são executadas e seus resultados são incluídos no documento, mas é possível modificar essa configuração por metadados, extensões ou configurações de exportação. Os metadados são

É importante ressaltar que existe uma alternativa *online* que elimina todas essas etapas: o Google Colab [1]. O Google Colab é um serviço gratuito do Google que oferece notebooks Jupyter *online*, ou seja, é possível criar e executar os *notebooks* de extensão .ipynb diretamente no navegador, sem precisar instalar nada no seu computador.

No entanto, no ambiente Jupyter Notebook, a geração de documentos PDF é possível após a instalação do Pandoc [40] e de um sistema LaTeX (como TeX Live ou MiKTeX) no computador. Com essas ferramentas instaladas, pode-se exportar notebook para PDF diretamente do Jupyter Notebook. O Google Colab, por sua vez, processa nativamente apenas arquivos .ipynb. Para gerar documentos PDF, é necessário baixar o *notebook* do Colab e usar um conversor externo, como o nbconvert, que faz parte do pacote jupyter. Esse conversor utiliza o Pandoc e o LaTeX (ou equivalente) instalados no computador para realizar a conversão.

Uma forma de contornar essa limitação do Google Colab e simplificar o processo de geração de documentos é utilizar o Quarto [66]. O Quarto pode ser integrado tanto ao Jupyter Notebook quanto ao Google Colab, permitindo que as funcionalidades do Quarto sejam usadas nos dois ecossistemas para criar e renderizar documentos diretamente nesses ambientes, sem nos preocupar com as instalações de Pandoc e LaTeX.

2.4 Sinergia entre R e Python

A integração eficaz entre R e Python representa uma estratégia robusta para análise de dados e desenvolvimento estatístico, consolidando as vantagens distintas de ambas as linguagens em um ambiente

unificado. Essa sinergia possibilita a combinação das bibliotecas especializadas em estatísticas do R com as poderosas ferramentas de aprendizado de máquina em Python. Além disso, ela facilita a execução de análises estatísticas em R enquanto implementa modelos de aprendizado de máquina em Python, tudo dentro de um único contexto. Essa abordagem requer a instalação dos dois interpretadores R e Python, mas amplia significativamente as capacidades analíticas, concedendo aos cientistas de dados a flexibilidade de escolher a ferramenta mais adequada para cada tarefa, maximizando assim a eficiência e a precisão nas análises.

Para incorporar R em um ambiente Python, pode-se usar a biblioteca rpy2. Para isso, é necessário instalar o pacote rpy2:

```
pip3 install rpy2
```

importar a biblioteca rpy2, ou um dos seus módulos, no *script* de Python como

```
import rpy2.robjects as robjects
```

e executar comandos R dentro de blocos específicos, como

```
robjects.r ('vetorR <- c(0, 1, 1.5, 2,3)')
```

A compatibilidade do rpy2 com diferentes versões de Python e R pode variar de acordo com a versão específica do rpy2. Uma forma de verificar qual versão o python acessa em tempo de execução no seu sistema é usar o comando

```
 python -m rpy2.situation
```

Para incorporar Python em um ambiente R, a biblioteca reticulate é frequentemente utilizada. Para isso, é necessário instalar o pacote reticulate

```
install.packages ("reticulate")
```

importar a biblioteca

```
library(reticulate)
```

e executar qualquer função Python no R usando a função py_run_string

```
py_result <- py_run_string ("3+4")
```

O reticulate é uma biblioteca em R que permite a integração com Python. Em teoria, ele pode ser utilizado com qualquer versão do R que suporte pacotes e é compatível com Python 2 e Python 3. Para verificar a disponibilidade das funções Python e acessá-las a partir do R, utilize o seguinte comando:

```
reticulate::py_available()
```

2.5 Considerações Finais

Neste capítulo, exploramos as duas linguagens de programação amplamente utilizadas para análise de dados: Python e R. Desde a estrutura básica até os tipos de dados, operações lógico-aritméticas e relacionais, comandos fundamentais e o ecossistema de cada linguagem, com destaque especial para RStudio, Jupyter Notebook e Google CoLab, buscamos proporcionar uma visão abrangente de suas características. Enfatizamos a sintaxe amigável de ambas as linguagens para representação de dados em estruturas complexas. Com uma diversidade de estruturas disponíveis, ambas as linguagens oferecem uma variedade de bibliotecas e pacotes específicos para manipulação, visualização e análise estatística dessas estruturas de dados. Adotamos uma abordagem que integra conceitos e prática, introduzindo as funções que implementam as técnicas à medida que são apresentadas ao longo do texto. Para reforçar a equivalência das funções disponíveis nas duas linguagens, procuramos sempre ilustrar os exemplos tanto em R quanto em Python.

Apresentamos o Markdown como uma ferramenta versátil para documentação, capacitando a criação de documentos dinâmicos e interativos. Abordamos os aplicativos R Markdown e Jupyter Notebooks, que promovem a integração fluida de código, texto e visualizações, simplificando a comunicação eficaz de resultados complexos. Embora a criação de um primeiro arquivo em linguagem Markdown, incorporando código em R ou Python, num novo ecossistema possa parecer desafiadora devido à variedade de informações a serem assimiladas, especialmente ao usar diversas funções importadas, é crucial que os iniciantes pratiquem extensivamente. Recomendamos aplicar as novas funções aprendidas em diferentes conjuntos de dados, comparando os resultados para compreender as nuances de cada parâmetro das funções. Esta prática diligente facilitará a familiarização com a linguagem e aprofundará o entendimento das funcionalidades oferecidas.

Destacamos a sinergia entre Python e R, apresentando uma abordagem que visa unir bibliotecas estatísticas especializadas em R com as robustas ferramentas de aprendizado de máquina em Python. Essa integração oferece aos profissionais de dados a vantagem de explorar o melhor de ambos os mundos, ampliando suas capacidades analíticas e permitindo a escolha da ferramenta mais adequada para cada tarefa. É importante ressaltar que a abordagem híbrida não será discutida no escopo deste volume. No entanto, incentivamos fortemente aqueles interessados em aproveitar ao máximo as duas linguagens a explorar e praticar análises mais profundas e abrangentes, utilizando os recursos proporcionados por ambas as linguagens.

2.6 Exercícios

1. De acordo com a linguagem optada, faça os exercícios de um dos itens, R ou Python.

    - R: Os capítulos 1, 2 e 3 em [71] dão uma boa introdução à linguagem R, com muitos

exemplos. Faça o que se pede em cada item

(a) Resolva os exercícios 1 e 2 da Seção 2.5 em [102].

(b) Confira as dicas sobre o uso do RStudio em `https://www.dataquest.io/blog/rstudio-tips-tri` e compartile uma que tenha chamado sua atenção.

(c) Sintetize, com base em `https://statsandr.com/blog/top-10-errors-in-r/` e na a seção 1.7 em [102], os erros comuns em R.

(d) Sintetize com base em `https://support.posit.co/hc/en-us/articles/205753617-Code-Diagn` as dicas que você achou mais úteis para diagnosticar erros em RStudio.

- Python: Os capítulos 2 e 3 em [100] dão uma boa introdução à linguagem Python, com muitos exemplos. Faça o que se pede em cada item.

  (a) Ajuste os comandos em Python do seguinte código para que ele seja executável:

```
import matplotlib.pyplot
import seaborn as sea
mpg = sea.load_dataset('mpg')
diamonds = sea.load_dataset ('diamond')
pyplot.scatter(x=mpg['displ'], y=mpg['hwy'])
pyplot.xlabel('displ')
pyplot.ylabel('hwy')
pyplot.show()
mpg_filtered = mpg[mpg['cyl'] = 8]
diamonds_filtered = diamonds[diamonds['carat'] > 3]
```

  (b) Confira as dicas sobre o uso do Jupyter Notebook em `https://www.makeuseof.com/jupyter-notebook-tips-tricks/` e compartilhe uma que tenha chamado sua atenção.

  (c) Sintetize com base em `https://betterstack.com/community/guides/scaling-python/python-errors/` os erros comuns em Python.

  (d) Sintetize com base em `https://www.linkedin.com/advice/3/what-best-ways-troubleshoot-e` as dicas que você achou mais úteis para diagnosticar erros em Jupyter Notebook.

2. Implemente em Python/R o **Jogo de Adivinhação de Números** em que o jogador deve adivinhar um número aleatório gerado pelo computador. As regras do jogo são:

   (a) O computador gera um número aleatório entre 1 e 100.

   (b) O jogador tem um número limitado de tentativas para adivinhar o número.

   (c) Após cada tentativa do jogador, o computador fornece dicas se o número fornecido pelo jogador é maior ou menor que o número gerado aleatoriamente.

(d) O jogo continua até que o jogador adivinhe corretamente o número ou até que ele esgote todas as tentativas.

Interface do jogador:

(a) O programa deve fornecer *feedback* após cada tentativa do jogador, indicando se o número é maior, menor ou igual ao número gerado aleatoriamente.

(b) O programa deve acompanhar o número de tentativas do jogador e informar quando ele acertar o número ou quando suas tentativas se esgotarem.

(c) Após o término do jogo, o resultado é salvo e o programa deve perguntar se o jogador deseja jogar novamente. Se a opção for não, são listadas todas as tentativas para cada número aleatório gerado e termina o jogo. Caso contrário, é iniciada uma nova rodada do jogo.

Capítulo 3

Interface para Análise Visual de Dados

A interface gráfica homem-máquina é fundamental para a interação eficiente entre o usuário e os dados, oferecendo uma plataforma para análise visual eficaz. Yi et al. [114] destacam que fatores como codificação visual inadequada, baixa usabilidade e desorganização da interface podem prejudicar a percepção de detalhes, dificultando a resolução de problemas. Além de apresentar dados de forma clara, um sistema de análise visual deve ser ágil, respondendo rapidamente às interações do usuário e garantindo uma experiência fluida. Nielsen [38] sugere que respostas em até 1 segundo mantêm o fluxo de pensamento do usuário sem distrações. Ao projetar uma interface para análise visual, é essencial considerar não apenas a qualidade dos algoritmos de renderização, mas também o tempo de processamento dos dados para que a resposta seja alinhada à intenção do usuário. É importante também que a interface permita explorar dados em diferentes níveis de abstração, promovendo uma compreensão mais profunda através da visualização coordenada desses níveis [77]. Isso, no entanto, exige soluções para desafios relacionados ao espaço na tela, desempenho computacional, capacidade de assimilação dos usuários e coordenação entre diferentes níveis de abstração.

Em geral, a interface gráfica é dividida em duas áreas principais, como ilustrado na Figura 3.1: a área de elementos de interação e a área de visualização. A área de elementos de interação contém botões, menus e campos de entrada, permitindo que o usuário realize ações e interaja com o sistema. Já a área de visualização exibe os dados ou informações relevantes, como gráficos e texto, e fornece *feedback* sobre as interações. Na abordagem de interações diretas, o usuário interage diretamente com os elementos visuais dentro da área de visualização.

Para garantir uma interface intuitiva e sem distrações, vários pesquisadores propuseram princípios de *design* para os elementos de interação e visualização. Ben Shneiderman, Colin Ware, Edward R. Tufte e Leland Wilkinson influenciaram significativamente essas áreas. Shneiderman, por exemplo, focou em criar interfaces que permitissem interação direta, sem a necessidade de teclado e *mouse*, desenvolvendo princípios que garantem uma interação eficaz na exploração de dados. Sua abordagem é descrita na Seção 3.1, onde é evidenciado como seus princípios promovem interfaces claras e previsíveis.

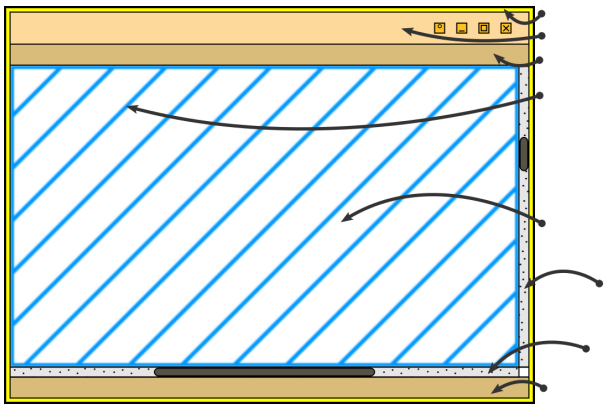


Figura 3.1: Uma janela numa interface gráfica com uma área de visualização (representada pela área hachurada) e uma área de interação (contendo elementos ao redor da área hachurada).

Ware, interessado em como a percepção visual influencia a criação de imagens informativas, e Tufte, focado na apresentação de dados complexos, especialmente em análises estatísticas, abordaram principalmente a área de visualização. Apesar de seus enfoques distintos, ambos desenvolveram princípios para criar representações visuais eficazes. Na Seção 3.2, discute-se como Ware enfatiza a importância de representações alinhadas à percepção humana, enquanto na Seção 3.3, Tufte defende a clareza e a minimização de elementos não essenciais na apresentação de dados.

Em 1999, Leland Wilkinson formalizou a Gramática dos Gráficos 2D para padronizar o *design* de visualizações estatísticas [108]. Essa abordagem visa facilitar a comunicação entre *designers* e fornecer uma base teórica para a criação de gráficos claros. A Seção 3.5 detalha as regras que orientam a composição visual dos gráficos. Mais tarde, em 2005, Hadley Wickham e sua equipe criaram a Gramática dos Gráficos para a linguagem R, resultando no pacote ggplot2, que rapidamente se tornou popular na comunidade de ciência de dados. Em Python, o pacote plotnine segue princípios semelhantes, oferecendo uma abordagem estruturada para a criação de visualizações. Embora os princípios da Gramática dos Gráficos sejam aplicáveis a gráficos 3D, representações tridimensionais apresentam desafios adicionais relacionados à percepção espacial.

3.1 Princípios de Schneiderman

A forma como os usuários raciocinam influencia diretamente a abordagem para explorar um conjunto de dados. As estratégias de apresentação e interação com os dados devem ser personalizadas. Ben Schneiderman, pioneiro em pesquisas de interface homem-máquina, propôs a Mantra de Busca de Informações Visuais (em inglês, *Visual Information-Seeking Mantra*) em [86] para orientar a busca de informações visuais. Essa mantra, conhecida por **Mantra de Schneiderman** (*Overview first, zoom and filter, then details on demand*), envolve 7 tarefas sobre 7 tipos de dados (1D, 2D, 3D, temporal, multi-dimensional, em árvore e em rede):

Visão Geral (*Overview*): Obter uma visão geral do conjunto de dados para entender a

sua estrutura e distribuição.

Ampliação (*Zoom*): Ampliar para explorar áreas específicas do conjunto de dados que requerem uma análise mais detalhada.

Filtragem (*Filter*): Aplicar filtros para reduzir o conjunto de dados a uma área específica relevante para a análise.

Detalhamento sob Demanda (*Details on Demand*): Obter informações detalhadas sobre elementos específicos do conjunto de dados, geralmente através de interações diretas com dados renderizados.

Relações (*Relate*): Identificar e compreender relações entre diferentes partes do conjunto de dados, destacando conexões e correlações.

Histórico (*history*): Manter um histórico das ações realizadas, permitindo a revisão e a reversão de alterações.

Extração (*extracts*): Identificar e utilizar informações específicas do conjunto de dados para análise e resolução de problemas.

Essas 7 tarefas promovem uma exploração de dados mais eficaz. Com base em suas vastas experiências em *design* de interfaces de usuário, Schneiderman também propôs as "Oito Regras de Ouro" em [79] que visam aprimorar a usabilidade e a experiência do usuário. Essas regras, também reconhecidas como "Oito Princípios para a Interface de Usuário", são:

Consistência e Padronização (*strive for consistency*): Manter uniformidade na apresentação e posicionamento de elementos ao longo das diferentes versões de um produto e, se possível, entre diferentes produtos. Isso reduz a confusão e facilita a curva de aprendizado para os usuários, proporcionando uma experiência mais fluida e intuitiva.

Flexibilidade e Eficiência de Uso (*seek universal usability*): Oferecer opções avançadas, como atalhos, para usuários experientes, ao mesmo tempo em que proporciona opções mais simples e com explicações detalhadas para usuários iniciantes. Isso permite contemplar diferentes níveis de conhecimento prévio.

Realimentação (*offer informative feedback*): Prover realimentação apropriada para informar os usuários sobre a interpretação do sistema em relação às suas ações. Isso contribui para uma compreensão clara e imediata das consequências das interações.

Diálogos conclusivos (*design dialogs to yield closure*): Organizar sequências de ações em grupos com um início, meio e fim. Junto com realimentações informativas, isso proporciona aos usuários a satisfação de conquista e o preparo para o próximo grupo de ações.

Prevenção de Erros (*prevent errors*): Projetar a interface de forma a reduzir a ocorrência de erros. Quando inevitáveis, apresentar mensagens de erro de forma amigável, minimizando impactos negativos e facilitando a recuperação.

Reversão das ações (*permit easy reversal of actions*): Permitir que os usuários desfaçam ações já realizadas (*undo*). Isso proporciona um senso de segurança e liberdade ao usuário.

Controlabilidade (*keep users in control*): Empoderar os usuários oferecendo a sensação de que eles estão no comando da interface e que ela responde às suas ações dentro das expectativas. Isso proporciona um senso de controle e intimidade ao usuário.

Redução da carga de memória de curto prazo (*reduce short-term memory load*): Evitar que os usuários lembrem informações da tela anterior ao interagir com a tela atual. Isso reduz a carga cognitiva, contribuindo para uma experiência mais fluente e intuitiva.

Em [110], Wong ressalta que a renomada empresa de tecnologia Apple Inc. alcançou notável sucesso em toda a sua gama de produtos, desde o Macintosh até os dispositivos móveis, graças aos seus *designs* de interface consistentes, intuitivos e atraentes. Em 2014, a empresa revelou ter incorporado os princípios de *design* propostos por Ben Shneiderman nas Diretrizes de Interface Humana do iOS da Apple. Essa aplicação consistente dos princípios de Shneiderman se destaca como um fator contribuinte para a reputação da Apple na criação de experiências de usuário positivas e eficientes ao longo dos anos.

3.2 Princípios de Ware

Com especial atenção à aplicação da percepção e cognição visual na visualização de dados, Colin Ware oferece uma perspectiva distintiva que se destaca pela ênfase no processo perceptual humano e nas convenções sociais presentes em nosso meio. Ao contrário da abordagem abrangente de Schneiderman, que incorpora vários aspectos da interação humano-computador, Ware direciona seu foco de maneira intensiva à conexão entre a representação visual e a interpretação cognitiva. Suas técnicas visam minimizar a carga cognitiva, buscando representações visuais que estejam alinhadas com as capacidades naturais de processamento de informações pelo cérebro humano.

Em [98], Ware introduz um modelo de processamento visual humano dividido em três estágios, conforme ilustrado na Figura 3.2, a ser detalhado na Seção 4.1. Este modelo serve como base para uma série de diretrizes detalhadas para a representação visual de dados complexos, visando facilitar a compreensão, análise e tomada de decisões. Destacam-se aqui algumas dessas diretrizes, as quais estão intrinsecamente relacionadas à percepção e cognição humanas, como discutido no Capítulo 4:

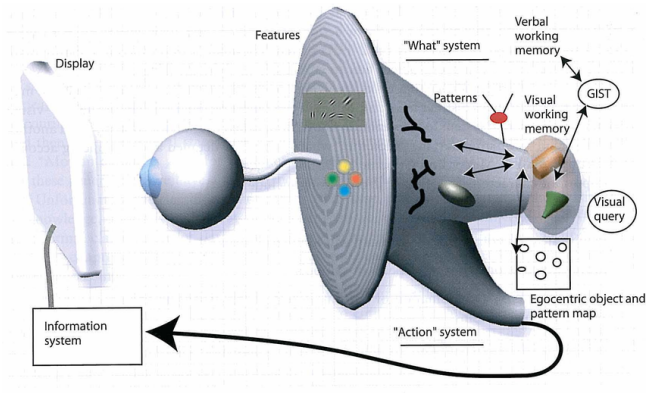


Figura 3.2: Um modelo de três estágios para processamento visual humano proposto pelo Ware: pré-atentivo (*features*), percepção de padrão (*patterns*) e processamento orientado à busca visual (*visual query*). (Fonte: `https://www.researchgate.net/figure/A-three-stage-model-of-human-visual-information-processing_fig6_224285723`)

- **Resolução de Detalhes:** Simplificar a visualização, omitindo detalhes sutis que podem passar despercebidos pelos usuários devido às limitações naturais da percepção humana. Isso reduz a sobrecarga visual e otimiza o uso de recursos computacionais, sem comprometer a qualidade.
- **Suavização de Variações de Níveis de Cinza:** Garantir transições suaves entre tons de cinza para uma percepção precisa da luminosidade na visualização, exceto quando transições abruptas forem necessárias para realçar contornos, como no efeito Cornsweet (Figura 3.3)

Figura 3.3: Efeito Cornsweet: realce do contraste na borda de transição entre dois retângulos.

- **Representação de Cores em Cromas:** Alinhar as visualizações com a forma como nosso sistema visual processa as cores para melhorar a interpretação e a fidelidade dos dados visualizados. A percepção humana é mais sensível às variações de croma (pureza da cor) do que à luminosidade (intensidade da luz), o que permite um *design* mais eficiente e intuitivo. No entanto, os canais cromáticos transmitem cerca de um terço da quantidade de detalhes finos em comparação ao canal de luminância, o que

exige um equilíbrio no contraste de luminância para garantir uma visualização eficaz.

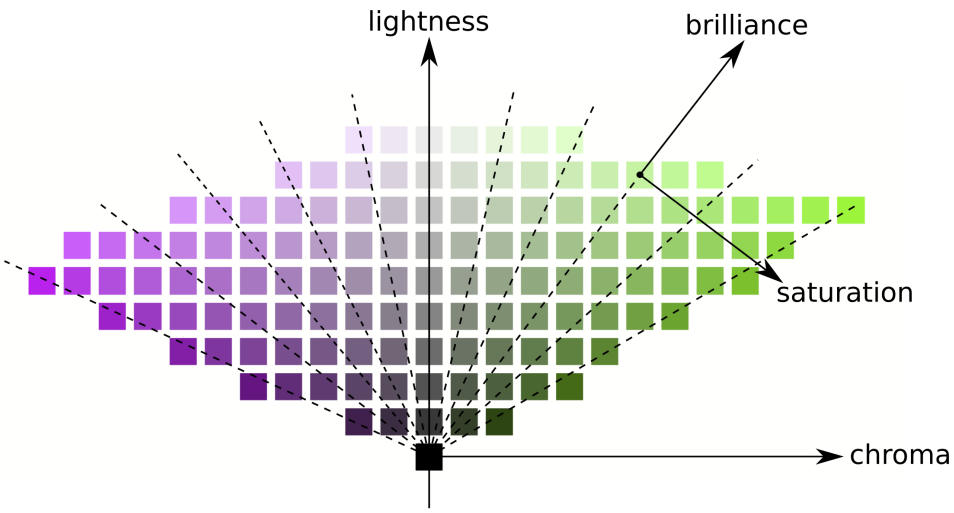


Figura 3.4: Croma e luminosidade num espaço de cores.

- **Processamento Pré-atentivo**: Utilizar propriedades visuais, como forma, cor, textura e movimento, que são prontamente percebidas pelos distintos canais perceptuais paralelos no primeiro estágio do sistema visual humano. Essas propriedades permitem destacar informações relevantes de forma eficiente e intuitiva, propiciando criação de visualizações eficazes.
- **Organização Perceptual**: Mapeiar os atributos dos dados em propriedades visuais como posição, forma, cor e tamanho, de maneira estruturada, facilitando a percepção de padrões e a interpretação dos dados no segundo estágio do sistema visual humano. Animações podem ser usadas para adicionar uma dimensão temporal e aprofundar a compreensão, mas use-as com moderação para evitar distrações.
- **Reconhecimento de um Objeto**: Usar diagramas que ilustrem as interconexões entre os componentes de um objeto. A contextualização favorece a organização dos elementos visuais de maneira a refletir a estrutura dos objetos no mundo real, facilitando assim o reconhecimento e a interpretação das informações de forma mais rápida e intuitiva no terceiro estágio do sistema visual humano. A inclusão de informações contextuais adicionais também pode auxiliar na compreensão.
- **Reconhecimento ao Invés de Lembrança**: Minimizar o esforço mental, utilizando elementos visuais que sejam facilmente reconhecíveis, sem exigir que os usuários precisem memorizar informações.
- **Visualização 3D orientada a Tarefa**: Escolher pistas de profundidade (oclusão, perspectiva, sombreamento, tamanho relativo, etc) em visualizações 3D considerando

as interações do usuário com os dados e não apenas a aparência. Por exemplo, a pista de oclusão, que se refere à capacidade de um objeto próximo bloquear a visão de um objeto mais distante, pode não ser a mais adequada para a tarefa de seleção de um objeto em um cenário 3D, especialmente se o objeto de interesse estiver obstruído.

- **Limitações da Memória de Trabalho Visual**: Fornecer informações de forma rápida e concisa, respeitando os limites da memória visual e verbal para evitar sobrecarga cognitiva e manter o foco do usuário.

Embora essas diretrizes de *design* sejam eficazes em situações simples, Ware destaca que, em cenários complexos, o desenvolvedor precisa ter critério na escolha e uso dos recursos gráficos, utilizando-os de maneira perspicaz.

3.3 Princípios de Tufte

Os estatísticos John W. Turkey e Edward Tufte são reconhecidos por suas contribuições ao campo da visualização de dados estatísticos, destacando-se por suas abordagens inovadoras na representação gráfica desses dados. Turkey é amplamente conhecido por suas contribuições à análise e à construção de gráficos que facilitam a compreensão da variabilidade dos dados, como o gráfico de caixa (em inglês, *boxplot*). Por sua vez, os trabalhos de Tufte não apenas resgatam métodos eficazes de representação gráfica, mas também reinterpretam e aprimoram esses métodos, enfatizando a continuidade e a evolução dos princípios de visualização ao longo do tempo.

Diferentemente de Colin Ware, que explorou a percepção visual e cognição, e de Ben Schneiderman, cujo foco residiu nos princípios de *design* de comunicação homem-máquina, Tufte concentrou sua atenção na apresentação eficaz de informações quantitativas. Embora tenha se dedicado principalmente a gráficos bidimensionais, os princípios de Tufte transcendem categorias específicas de dados, aplicando-se a uma variedade de dados, desde dados estatísticos até representações gráficas de fenômenos científicos. Contudo, sua ênfase na simplicidade e clareza, recusando excessos ornamentais e destacando a necessidade de gráficos que permitam aos usuários extrair informações de maneira rápida e precisa, compartilha afinidades com os princípios de Schneiderman, que valoriza a clareza na apresentação de dados abstratos e complexos.

Em [94], Tufte define a excelência em gráficos com base na clareza, precisão e eficiência na comunicação de ideias. Para ele, com base no princípio de **Excelência em um Gráfico**, um gráfico deve revelar dados de maneira transparente, priorizando a apresentação das informações em vez das computações estatísticas subjacentes. O objetivo principal de um gráfico é servir como um veículo eficaz para transmitir mensagens e *insights* contidos nos dados, sendo compreendido de forma rápida e precisa. Sua excelência reside na capacidade de comunicar visualmente, indo além da mera apresentação de números para contar uma história clara e impactante.

O segundo princípio proposto pelo Tufte é a **Integridade de Gráficos**. Esse princípio destaca a importância de manter a fidelidade dos dados apresentados. Para Tufte, um gráfico deve ser uma representação honesta e precisa, evitando distorções, omissões ou manipulações que possam levar a interpretações equivocadas. A integridade refere-se à responsabilidade do *designer* em garantir que a representação visual seja fiel à realidade dos dados, permitindo que os leitores façam conclusões válidas e informadas. Esse princípio reforça a ética na construção de gráficos estatísticos, enfatizando a transparência e a precisão como elementos essenciais na comunicação efetiva de informações quantitativas.

De acordo com Tufte, a excelência em *design* gráfico requer a combinação de três conjuntos distintos de habilidades: habilidades substantivas, estatísticas e artísticas. Tufte enfatiza a importância de uma compreensão sólida do assunto em questão, habilidades estatísticas para resumir e inferir dados relevantes e representá-los com precisão, e habilidades artísticas para criar uma apresentação visualmente atraente. No entanto, Tufte observa que, na prática, grande parte do trabalho gráfico, especialmente em publicações de notícias, muitas vezes é supervisionada apenas por especialistas em habilidades artísticas. Isso evidencia uma lacuna na abordagem, destacando a importância de integrar conhecimentos substantivos e estatísticos ao processo de *design* gráfico para garantir a integridade gráfica e aprimorar a clareza e a qualidade informativa. Para evitar distrações ou elementos visuais que não contribuem para a compreensão do conteúdo, priorizando uma representação clara e direta, Tufte propus o terceiro princípio, **Maximização da Taxa de Tinta de Dados** em gráficos desenhados sobre folhas de papel

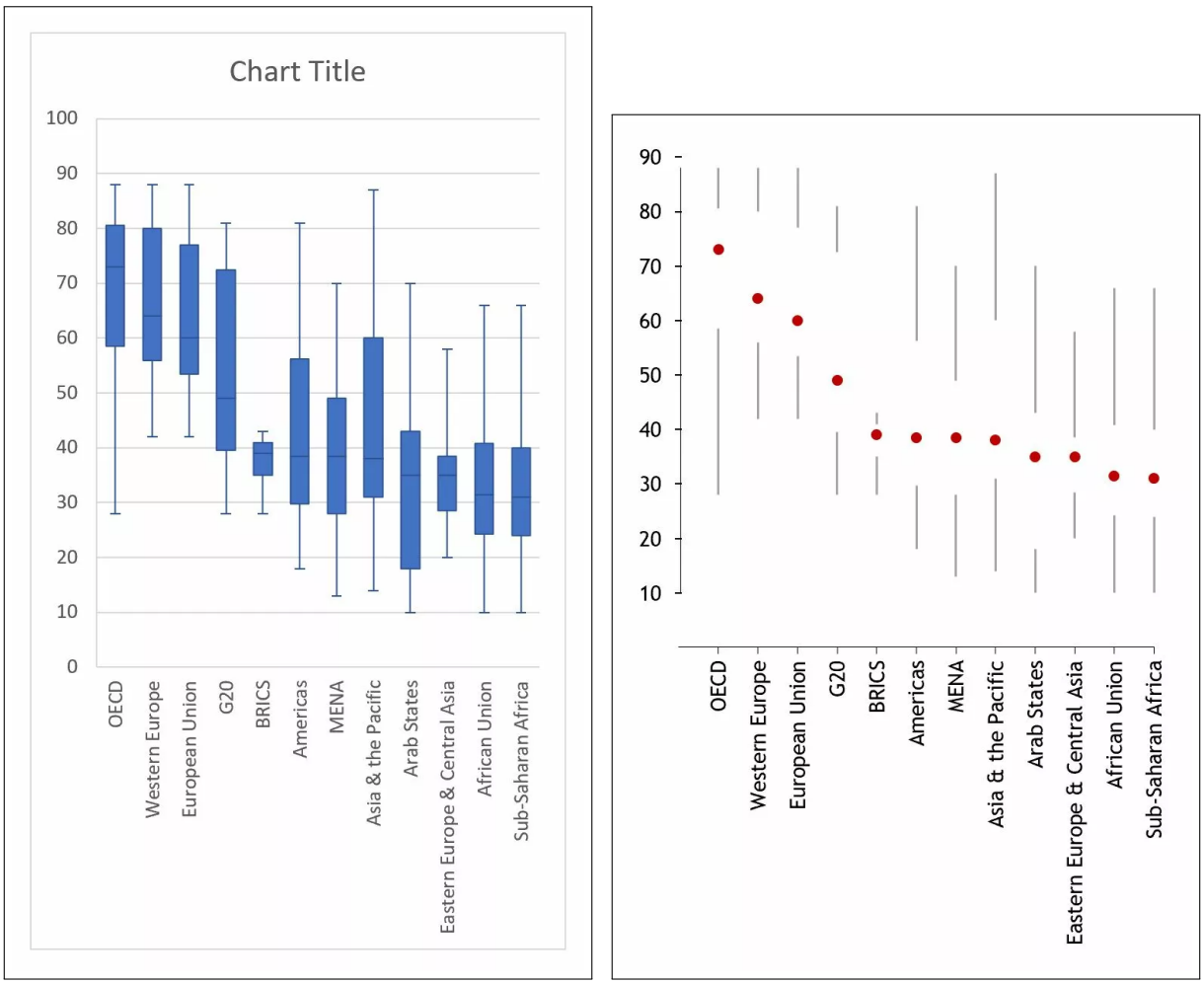
$$\text{Taxa de tinta de dados} = \frac{\text{Tinta de dados}}{\text{Quantidade total de tinta usada na impressão do gráfico}},$$

onde tinta de dados é a tinta usada para "desenhar" os elementos visuais correspondentes às informações quantitativas essenciais[1].

Tufte ilustra suas ideias apresentando versões aprimoradas, em termos de taxa de tinta de dados, de quatro gráficos estatísticos: gráficos de caixa (introduzidos por John W. Tukey em 1969), gráficos de barras (popularizados por William Playfair), histogramas (criados por Karl Pearson no final do século XIX) e gráficos de dispersão (concebidos por John F. W. Herschel no final do século XIX e ganhando importância com a análise de correlações no trabalho de Francis Galton no mesmo período). A Figura 3.5 exemplifica a proposta de Tufte para melhorar a utilização da tinta de dados em um gráfico de caixa. Em vez de retângulos sólidos, ele utiliza segmentos de reta "vazados" para representar os quartis. Os estudos [51] e [7] exploram as funções disponíveis em R e Python, respectivamente, para criar gráficos estatísticos seguindo o paradigma de Tufte.

Com os avanços da tecnologia de computação gráfica, Tufte expressa preocupação com a proli-

[1] Podemos associar tinta de dados aos *pixels* quando plotamos gráficos em monitores matricias (*raster*).



(a) Versão original (b) Versão de Tufte

Figura 3.5: Proposta de Tufte: maximização da taxa de tinta de dados em gráficos de caixa. (Fonte: `https://simplexct.com/tufte-in-excel-the-box-plot`)

feração de objetos gráficos decorativos, conhecidos como *chartjunk*. Esses elementos consomem tinta ao serem plotados, mas não agregam informações relevantes, podendo, muitas vezes, obscurecer os dados essenciais, com o uso excessivo das grades, e introduzir efeitos indesejáveis, como as vibrações de moiré mostradas na Figura 3.6. Ele ressalta que, em meio à variedade de recursos gráficos disponíveis, alguns *designers* parecem buscar reconhecimento simplesmente por explorar novas tecnologias, em vez de empregá-las para criar *designs* mais eficazes. Tufte destaca que computadores e suas ferramentas relacionadas têm capacidades gráficas poderosas, especialmente na produção de uma grande quantidade de gráficos necessários para uma boa análise de dados. No entanto, aponta que alguns gráficos gerados por computador podem receber uma resposta mais relacionada à admiração pela capacidade da máquina em desenhar, em vez de despertar o interesse pelos dados apresentados.

Para utilizar eficientemente a tinta (ou elementos gráficos) em um gráfico e representar informações relevantes, Tufte propôs o princípio do **Elemento Gráfico Multifuncional**. Este princípio visa maximizar a eficácia dos elementos gráficos ao permitir que um único recurso sirva a múltiplos propósitos, combinando a representação de dados com funções de *design* que normalmente seriam atribuídas a

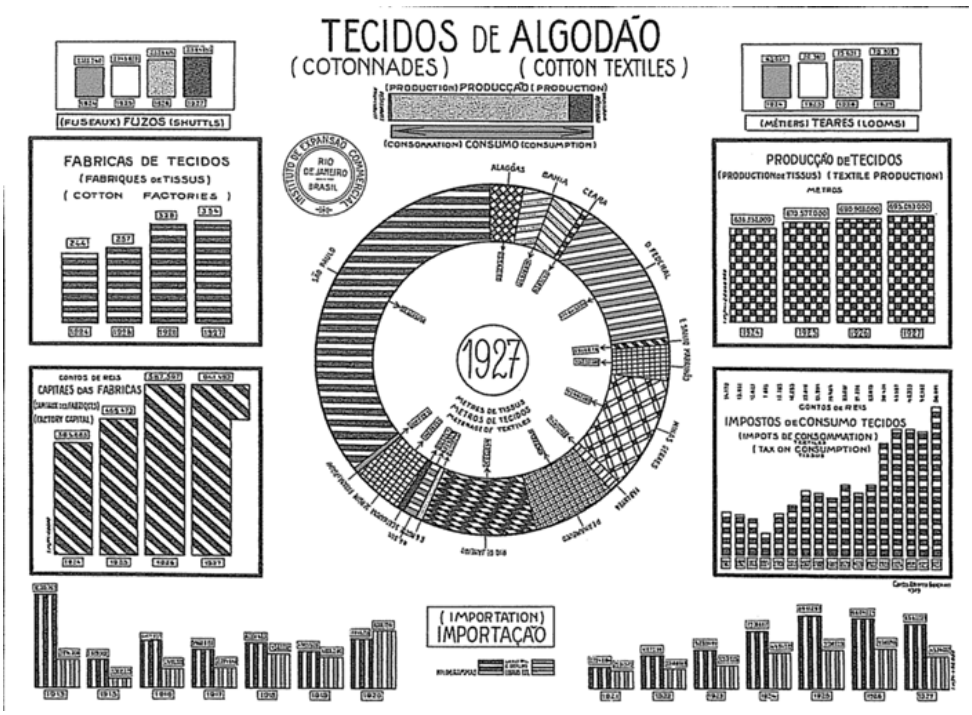


Figura 3.6: *Chartjunk*, como o efeito de moiré para sugerir vibrações ou movimentos, não contribui para a eficácia comunicativa dos gráficos de dados estatísticos; ao contrário, pode ser distrativo e até mesmo visualmente irritante. (Fonte: `https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/datapresentation/DataPresentation4.html`)

elementos não relacionados a dados. Isso possibilita uma representação mais compacta e eficiente de dados complexos e multivariados, como ilustrado no gráfico da Figura 3.7, que mostra a distribuição das divisões americanas na França entre junho de 1917 e outubro de 1918 durante a Primeira Guerra Mundial [50]. No entanto, Tufte adverte que a implementação de elementos gráficos multifuncionais deve ser feita com cuidado e sutileza. Embora, quando bem projetados, esses elementos possam revelar várias facetas dos dados de forma simplificada, há o risco de criar gráficos que se tornam quebra-cabeças para o público, onde as codificações são compreendidas apenas pelo criador, dificultando a interpretação geral.

Na busca pela maximização da eficácia na comunicação visual de dados, Tufte introduziu e desenvolveu o conceito de densidade de dados (em inglês, *data density*) em um gráfico, uma métrica que quantifica o volume de informações significativas contidas em uma determinada área do gráfico. A fórmula para a densidade de dados é dada por

$$\text{densidade de dados} = \frac{\text{número de dados}}{\text{área do gráfico}}.$$

Gráficos com alta densidade de dados conseguem transmitir uma grande quantidade de informações de maneira eficiente. Além disso, Tufte demonstrou que a acuidade visual humana é suficiente para distinguir variações em um conjunto de dados através de uma série de pequenos múltiplos (*small multiples* em inglês) de gráficos, cada um focando em uma parte específica dos dados. Ao seguir o

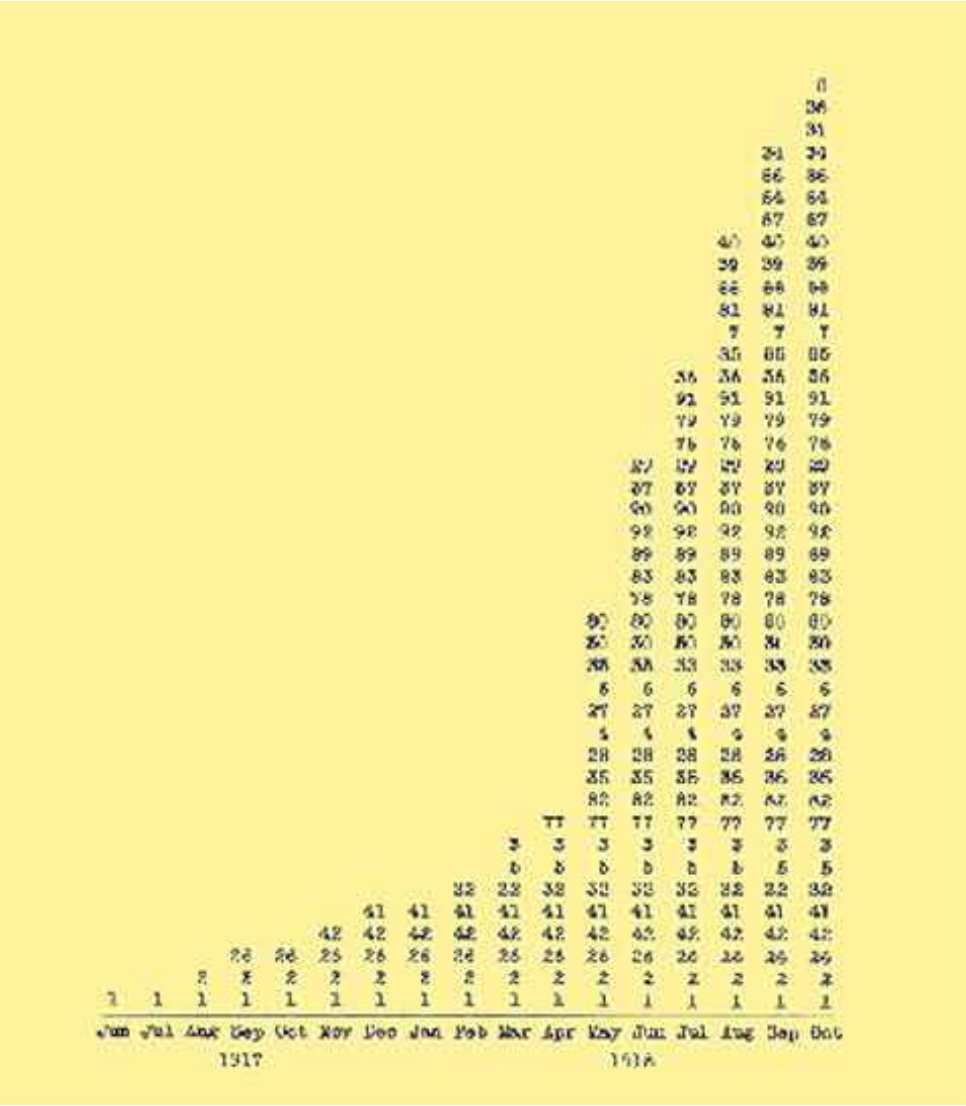


Figura 3.7: "Barra" multifuncional constituída pelos números das divisões americanas presentes no território francês durante o período de Junho de 1917 a Outubro de 1918 na Primeira Guerra Mundial. (Fonte: [50])

princípio de **Maximização de Densidade de Dados e Uso de Pequenos Múltiplos** proposto pelo Tufte, é possível proporcionar aos leitores uma compreensão mais profunda dos dados e *insights* mais claros, tornando as visualizações mais eficazes. Atualmente, os pequenos múltiplos são amplamente usados em gráficos estatísticos, pois têm se mostrado uma ferramenta poderosa para apresentar e analisar conjuntos de dados complexos, possibilitando uma comparação clara e organizada entre diferentes variáveis.

Tufte sustenta a visão de que o *design* gráfico de dados transcende a mera representação clara e eficaz de informações, estendendo-se à criação de visualizações atraentes, cativantes e memoráveis. Ele argumenta que a estética e a técnica no *design* gráfico de dados desempenham papéis fundamentais na comunicação eficaz de dados, pois visualizações bem projetadas têm mais probabilidade de atrair a atenção dos leitores, serem retidas na memória e minimizar erros de interpretação. A estética, relacionada à qualidade visual, abrange a percepção de beleza, atratividade e apelo visual. No contexto

do *design* de visualização de dados, a estética engloba a aparência geral da visualização, incluindo elementos como cores, formas, equilíbrio, proporção e estilo. Já a técnica envolve a aplicação precisa dos princípios de *design*, como a escolha apropriada de tipos de gráficos, o uso de escalas adequadas, a seleção de cores eficazes, bem como a manipulação cuidadosa de tipografia e *layout*.

3.4 Tipos de Dados

Para transformar conjuntos de dados em visualizações perceptualmente eficientes, é crucial compreender os tipos de dados presentes nesses conjuntos. Essa compreensão permite mapeá-los para elementos gráficos distintos, alinhando-se aos princípios de *design* discutidos nas Seções 3.1, 3.2 e 3.3. A classificação dos dados não apenas facilita a sistematização dos mapeamentos de dados em elementos gráficos, mas também abre caminho para generalizá-los, promovendo abordagens mais amplas e reutilizáveis. Embora a classificação dos dados seja desafiadora, versões diferentes são apresentadas em livros de visualização para subsidiar conceitos e algoritmos discutidos. Nesta seção, exploramos duas classificações, uma baseada em valores e relações e outra baseada em técnicas de renderização.

3.4.1 Valores e Relações

A classificação de dados em **valores** e **relações** foi introduzida por Jacques Bertin [9]. Essa distinção é essencial para compreender o modelo de dados relacional desenvolvido por Edgard F. Codd [15]. No contexto desse modelo, os **dados-valores** são representados por **atributos**. Especificamente, os atributos são características associadas a uma **entidade** (como pessoas, lugares ou coisas), e são usados para descrever essa entidade. Já as **relações** se referem às tabelas que organizam e armazenam (as relações entre) esses atributos. No modelo relacional, as relações também capturam as associações entre diferentes entidades por meio de **chaves estrangeiras**, permitindo representar conexões complexas e facilitando a compreensão das interações dentro de um sistema de dados. A Figura 3.8[2] ilustra um conjunto de dados sobre alunos e disciplinas organizados em entidades (linhas), atributos (colunas), relações definidas pelas chaves estrangeiras (em azul e em vermelho).

Em busca de um mapeamento sistemático dos valores dos dados em elementos gráficos durante a análise visual de dados, é comum distinguir dois tipos principais de valores:

Valores quantitativos : representam quantidades numéricas, que podem ser subsivididas em

- **Discretos**: valores contáveis, como número de carros em um estacionamento ou quantidade de livros em uma estante.
- **Contínuos**: valores que podem assumir qualquer valor dentro de um intervalo, como altura, peso ou temperatura.

[2] `https://oracle-patches.com/en/databases/relational-model-and-why-it-doesn't-matter`

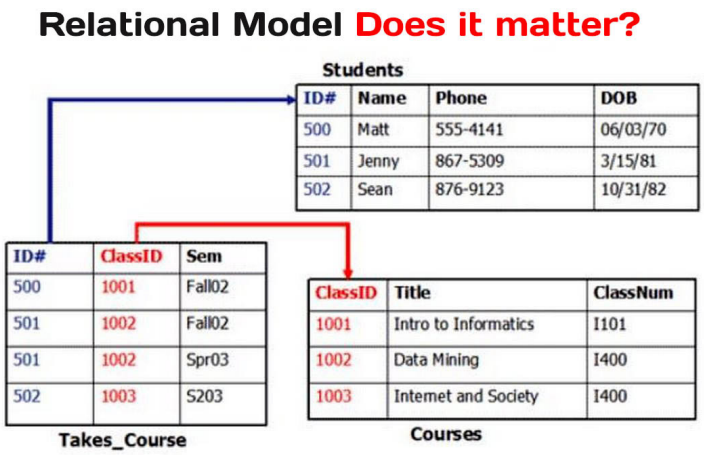


Figura 3.8: Modelo relacional de um conjunto de dados de alunos e disciplinas cursadas.

Valores qualitativos (categóricos) : representam qualidades ou características, que podem ser ainda classificadas em

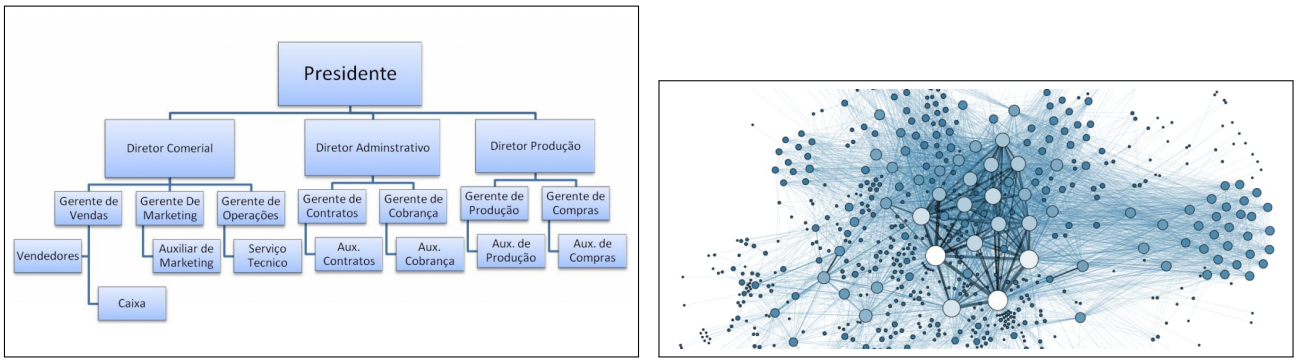
- **Nominais**: categorias sem ordem intrínseca, como cores (azul, verde, vermelho), tipos de frutas (maçã, banana, laranja) ou marcas de carros.
- **Ordinais**: categorias com uma ordem definida, mas sem uma escala numérica precisa, como níveis de escolaridade (fundamental, médio, superior), grau de satisfação (insatisfeito, neutro, satisfeito) ou tamanho de roupa (P, M, G).

As categorias que uma variável pode assumir são chamadas **níveis** (em inglês, *levels*), especialmente no contexto de análise de dados e programação (como em R). Em Python, elas são geralmente chamadas de *categories* ou **valores únicos**.

Para realizar uma análise visual de dados eficaz, é fundamental entender como os valores dos dados são representados graficamente. A escolha da representação gráfica adequada depende do tipo de dado que se pretende analisar e impacta diretamente na clareza e na eficiência da comunicação dos insights extraídos dos dados. Podemos associar os tipos de dados 1D, 2D, 3D e multidimensional, propostos por Schneiderman (Seção 3.1), a entidades com, respectivamente, 1, 2, 3 e mais de 3 atributos associados. Por exemplo, as entidades COURSES e STUDENTS no modelo mostrado na Figura 3.8 são classificadas como tipos de dados 2D e 3D, respectivamente. Suas tuplas podem ser representadas por pontos em gráficos cartesianos, onde cada eixo corresponde a um atributo da entidade. Schneiderman também define o tipo de dado **temporal**, que se aplica quando os valores de um atributo de uma entidade estão relacionados ao aspecto temporal dos dados em estudo. As variações no tempo podem ser visualizadas através de técnicas de **animação**.

Os tipos de dados em árvore e em rede (*networks*), destacados por Schneiderman, permitem visualizar de forma mais direta relações hierárquicas e interconexões entre entidades conectadas por chaves estrangeiras, quando as relações não são lineares. Em **árvores**, a estrutura é adequada para

representar relações de subordinação e dependência, como ilustra a Figura 3.9a. Já em **redes**. **Redes não-estruturadas** apresentam uma variedade de conexões entre nós, sem seguir um padrão definido, permitindo uma representação mais livre e flexível das relações entre os elementos, como mostra a Figura 3.9b. Em contrapartida, **redes estruturadas** aderem a padrões organizacionais específicos, como tabelas ou matrizes na Figura 3.8, proporcionando uma disposição mais ordenada e previsível das interconexões entre os nós. Embora as árvores sejam frequentemente consideradas como um caso particular de redes, a distinção entre árvores e redes é fundamental para escolher abordagens e técnicas adequadas na visualização de dados.



(a) Árvore (b) Rede (*network*)

Figura 3.9: Visualização das relações entre as entidades: (a) hierárquica e (b) arbitrária.

Vale destacar que padrões, tendências e causas-efeitos emergem não apenas das relações entre entidades, mas também das relações entre os atributos dentro de cada entidade. Essas relações intra-entidades, muitas vezes ignoradas, são essenciais para a compreensão dos dados, revelando como as variáveis se relacionam e influenciam mutuamente. Ao explorar tanto as relações entre entidades quanto as relações dentro delas, os analistas obtêm uma visão mais completa e significativa do contexto e da dinâmica dos dados.

3.4.2 Técnicas de Renderização

Sob a **perspectiva das técnicas de renderização**, a representação visual de dados envolve a atribuição de *pixels* na tela a cada conjunto de dados. Nesse cenário, é possível distinguir entre dados que possuem uma associação a uma posição espacial, denominados dados físicos, e dados que estão desvinculados de uma estrutura específica ou um formato físico predeterminado, conhecidos como dados abstratos. **Dados físicos** se referem a medições tangíveis e diretamente observáveis, como comprimento, temperatura, volume ou posição espacial. Esses dados têm uma conexão direta com o mundo físico e suas unidades de medida são claras e específicas. No entanto, dependendo do contexto, até mesmo dados como temperatura ou comprimento podem ser tratados de maneira abstrata, como quando analisamos padrões ou tendências, em vez de dados pontuais. Por outro lado, **dados abstratos** envolvem informações intangíveis ou qualitativas, como sentimentos, tendências,

categorias ou conceitos, que não são diretamente mensuráveis, mas podem ser representados por indicadores ou métricas derivadas.

Seguindo a classificação de Schneiderman na Seção 3.1, os dados com manifestação física podem ser categorizados em 1D, 2D e 3D, dependendo do número de coordenadas espaciais associadas a eles. Na visualização desses dados, o princípio de **Consistência e Padronização** orienta o uso das coordenadas espaciais para posicionar os elementos de forma coerente na tela, garantindo uma interpretação intuitiva e uniforme. Além disso, o princípio de **Organização Perceptual** é frequentemente aplicado, facilitando a identificação de padrões e relações significativas, o que contribui para a resolução eficiente de problemas.

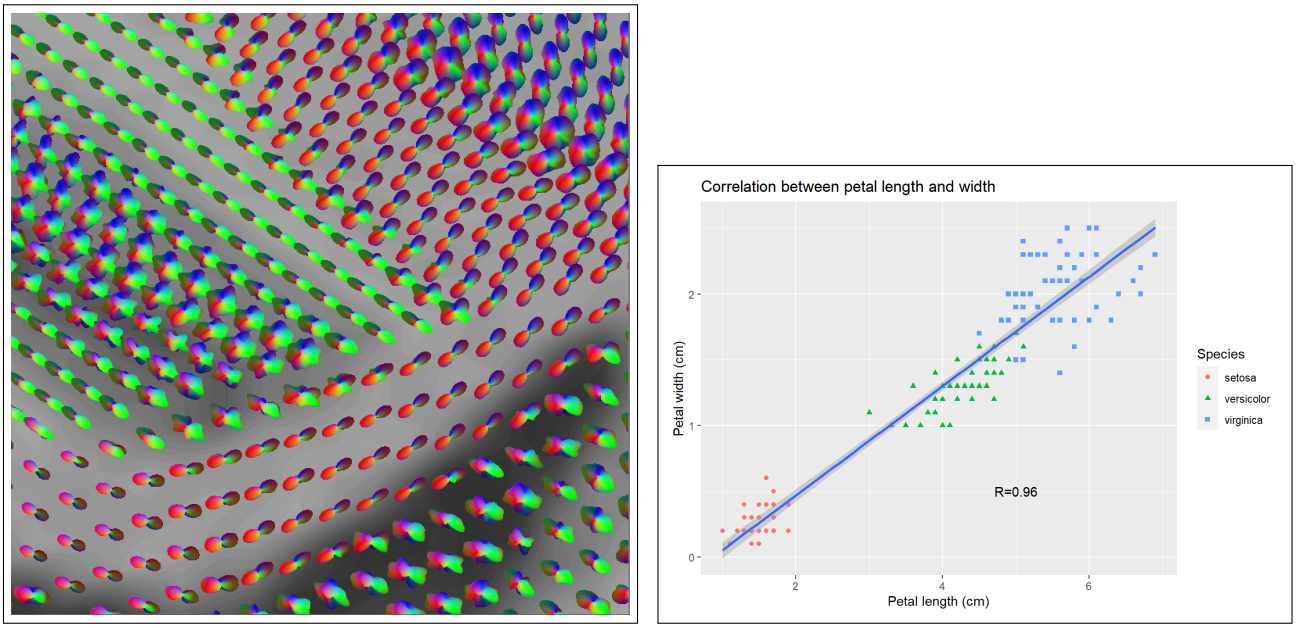
Por outro lado, para visualizar dados abstratos, é necessário criar representações visuais flexíveis. Essa tarefa exige dos cientistas de dados a habilidade de conceber visualizações que revelem ou destaquem padrões complexos, tendências e relações que não são imediatamente evidentes nos dados brutos. Embora esses dados não tenham uma forma física tangível, a escolha de como representá-los visualmente é baseada em princípios de *design* e análise, para garantir que as visualizações comuniquem de forma eficaz os padrões e *insights* presentes nos dados.

A **renderização de dados físicos** frequentemente utiliza técnicas tridimensionais de renderização, representando curvas, superfícies, volumes e estruturas complexas, comn foco em reprodução da aparência e do comportamento dos objetos físicos de forma realista e interativa. A Figura 3.10a ilustra a visualização de sinais de difusão de moléculas de água nos pontos amostrados do cérebro. O conjunto de difusões nas direções espaciais amostradas é sintetizado em uma figura geométrica tridimensional conhecida como glifo.

Por outro lado, a **renderização de dados abstratos** se concentra na comunicação de conceitos, ideias, relações ou informações sem depender das limitações da forma física original dos dados, com foco em apresentá-los de forma clara e concisa, utilizando muitas vezes **gráficos estatísticos** (bidimensionais). A Figura 3.10b ilustra um gráfico de dispersão que mostra a relação entre o comprimento e a largura da pétala de três espécies diferentes de flores íris [18]: setosa, versicolor e virginica. Cada espécie é representada por uma forma geométrica específica: setosa por um círculo, versicolor por um triângulo e virginica por um quadrado. Essas formas não têm relação direta com a geometria das espécies de íris; são apenas convenções geométricas adotadas para a representação de uma flor, junto com a sua espécie, nos gráficos estatísticos. Além disso, os dois atributos, comprimento e largura, são mapeados em coordenadas x e y, respectivamente.

3.5 Gramática dos Gráficos

A Gramática dos Gráficos (em inglês, *Grammar of Graphics (GoG)*), desenvolvida por Leland Wilkinson, oferece uma abordagem sistemática e compreensível para a construção de **gráficos estatísticos**,



(a) Dados físicos (b) Dados abstratos

Figura 3.10: Visualização de dados: (a) direcionais de difusão de moléculas de água no cérebro humano revelados pelo exame de ressonância magnética ponderada em difusão, e (b) relacionais entre o comprimento e a largura da pétala de três espécies de flores íris.

independentemente da complexidade dos **dados abstratos** [108]. A gramática GoG fornece uma estrutura para mapear os dados abstratos em elementos visuais, como pontos, linhas, cores e formas, de maneira clara e consistente, e permite a combinação de elementos gráficos básicos para criar uma ampla variedade de visualizações. A consistência e a padronização na representação de valores e relações, conforme a GoG, facilitam a compreensão e a interpretação de dados, especialmente em cenários complexos e multidimensionais. A biblioteca ggplot2 para a linguagem R [101], desenvolvida por Hadley Wickham, implementou a GoG na prática e teve um papel crucial na sua popularização.

3.5.1 Gramática dos Gráficos (Estatísticos)

Ao utilizar o termo **gramática**, Wilkinson enfatiza a importância de seguir padrões e convenções ao criar visualizações, de modo que a comunicação visual seja clara e compreensível. Assim como as regras gramaticais facilitam a compreensão e a interpretação correta de uma linguagem, os princípios da Gramática dos Gráficos ajudam a garantir a precisão e a interpretação adequada das informações contidas nas representações visuais de dados. Em [109], o autor destaca que os gráficos estatísticos são diferentes de outras visualizações, como mapas, diagramas e representações volumétricas. A Gramática dos Gráficos (estatísticos), fundamentada em conceitos matemáticos, oferece uma estrutura sistemática para representar qualquer tipo de dado, seja físico ou abstrato, em diversas formas visuais. Ela facilita o mapeamento dos dados em elementos gráficos, permitindo ao *designer* se concentrar em selecionar as formas visuais mais adequadas para destacar padrões ou relações relevantes em um contexto específico de análise de dados.

A Gramática dos Gráficos define dois tipos fundamentais de dados: variável e conjunto de variáveis (em inglês, *variable set – varset*). Uma **variável** X é uma função que associa a cada objeto de um conjunto O um valor em um conjunto V. Esse valor é extraído dos dados brutos e pode ser numérico, categórico ou de outro tipo relevante para a análise. Já um **conjunto de variáveis** X é uma função que associa a cada combinação ordenada de objetos de O um conjunto de valores, representando múltiplas características ou propriedades associadas a agrupamentos de objetos em vez de a um único objeto isolado. A gramática estabelece ainda um fluxo sequencial de dados envolvendo 7 classes ortogonais, ou seja, pelo menos uma transformação em cada classe é necessária para transformar um conjunto de dados-valores em gráficos estatísticos significativos. Essas classes proporcionam uma estrutura lógica que guia a criação de gráficos estatísticos, facilitando a comunicação visual de padrões e tendências nos dados:

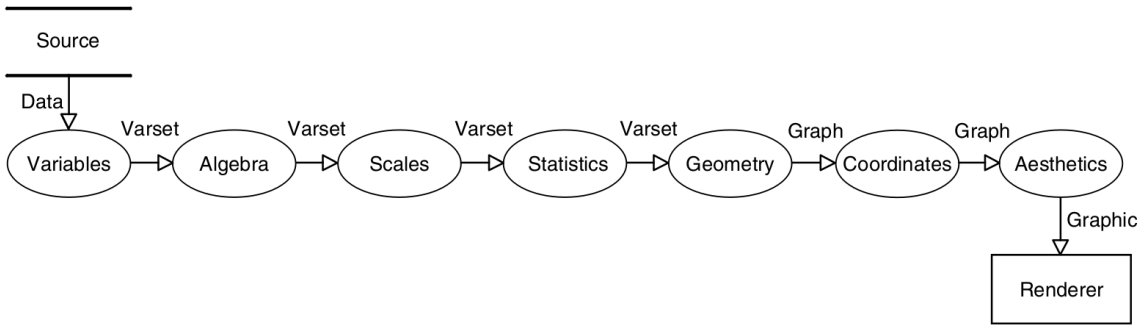


Figura 3.11: Fluxo de dados de acordo com as normativas estabelecidas pela Gramática dos Gráficos proposta por Leland Wilkinson. (Fonte: [109]

Variáveis (*Variables*): Transformar as tabelas de dados-valores em um conjunto de variáveis (*varset*).

Álgebra (*Algebra*): Aplicar operações algébricas, como produto cartesiano (*cross*), divisão (*nest*) ou soma (*blend*), sobre as variáveis para combiná-las em novas tuplas de variáveis.

Escalas (*Scales*): Normalizar ou definir distâncias ou intervalos entre marcas nos eixos, o que auxilia na interpretação e leitura dos dados renderizados, de acordo com as dimensões da área onde os gráficos estatísticos serão representados.

Estatísticas (*Statistics*): Aplicar funções estatísticas, como identidade, cálculo de médias, medianas, desvios padrão, histogramas e regressão, sobre os conjuntos de variáveis, gerando novos conjuntos de variáveis de interesse para visualização.

Geometria (*Geometry*): Especificar os tipos de figuras gráficas, como pontos, arestas, linhas, polígonos, barras e contornos, utilizados para representar os dados.

Coordenadas (*Coordinates*): Determinar como os dados são mapeados no espaço visual, especificando o sistema de coordenadas adotado. O sistema cartesiano é o mais

comumente utilizado.

Estética (*Aesthetics*): Mapear os elementos de um gráfico estatístico para elementos visuais, como posição, tamanho, forma, orientação, brilho, cor e granularidade.

Embora o procedimento permita criar uma ampla variedade de gráficos estatísticos, Ismay e Kim argumentam em [36] que um conjunto básico de cinco gráficos — gráficos de dispersão, gráficos de linha, gráficos de caixa, histogramas e gráficos de barras — é suficiente para visualizar uma vasta gama de variáveis e suas relações de dependência. Esses gráficos são conhecidos como os 5 Gráficos Nomeados (*Five Named Graphs* - 5NG).

3.5.2 Gramática em Camada dos Gráficos

Hadley Wickham introduziu várias melhorias na Gramática dos Gráficos, principalmente por meio do desenvolvimento do pacote ggplot2 na linguagem R [103]. Três das extensões significativas introduzidas por Wickham incluem:

Facetas (*Faceting*): Permitem dividir um gráfico em vários painéis pequenos, tipicamente com base nos níveis de uma ou mais variáveis categóricas. É uma implementação dos pequenos múltiplos propostos pelo Tufte como vimos na Seção 3.3. Vimos também que essa técnica facilita a comparação de padrões e relacionamentos em diferentes segmentos dos dados, melhorando a compreensão e a interpretação das informações apresentadas no gráfico.

Camadas (*Layers*): Permitem adicionar e sobrepor diferentes camadas de informações a um gráfico. Cada camada pode representar diferentes geometrias dos dados, como pontos, linhas, barras, etc. Isso proporciona flexibilidade na construção de gráficos complexos e informativos.

Hierarquia de *defaults* (*Hierarchy of Defaults*): Estabelece uma hierarquia de valores padrão para muitos elementos gráficos, a menos que especificamente modificados pelo usuário. Isso simplifica a criação de gráficos, reduzindo a quantidade de código necessário para gerar visualizações típicas. Ao mesmo tempo, a flexibilidade da Gramática dos Gráficos permite que os usuários substituam ou ajustem esses padrões sempre que desejarem personalizar a visualização de acordo com suas necessidades específicas.

Essas extensões contribuíram para a popularidade e versatilidade do ggplot2 como uma ferramenta poderosa para visualização de dados em R [105]. A **folha de dicas do ggplot2** (*ggplot2 cheat sheet*, em inglês) [25] é um recurso de referência rápida que resume os principais conceitos e funcionalidades do ggplot2. Essas folhas de dicas são úteis para fornecer uma visão geral dos recursos do ggplot2 e

exemplos de código para criar gráficos típicos. Uma descrição completa de todas as funcionalidades de ggplot2 se encontra em [106].

O pacote plotnine [4, 55] implementa a gramática de gráficos sobre o pacote matplotlib[3] em Python, seguindo abordagem similar ao ggplot2 para a criação de visualizações. Ele permite a construção de gráficos ao mapear variáveis de um dataframe do pacote pandas para os elementos visuais, facilitando a composição e personalização de visualizações complexas. Janssen compilou heurísticas [39, 33] para auxiliar na transição de código do ggplot2 (R) para o plotnine (Python). No entanto, nem todas as funções do ggplot2 têm equivalência direta no plotnine. Além disso, extensões populares do ggplot2, como ggthemes e ggalt [76], ainda não foram implementadas como pacotes independentes em Python com os mesmos nomes.

O pacote plotnine [4, 55] é uma implementação da gramática de gráficos sobre o pacote matplotlib[4] em Python, seguindo uma abordagem semelhante à do **ggplot2** para a criação de visualizações. Ele permite a construção de gráficos ao mapear explicitamente variáveis em um dataframe do pacote pandas para os elementos visuais do gráfico, facilitando a composição e personalização de visualizações complexas. Janssen compilou uma série de heurísticas [39] para auxiliar na transição de código do ggplot2 em R para o plotnine em Python, cujas funções são resumidas em [33]. No entanto, é importante observar que nem todas as funções disponíveis no ggplot2 têm uma equivalência direta no plotnine. Além disso, as extensões populares do ggplot2, como ggthemes e ggalt [76], ainda não foram diretamente implementadas para Python como pacotes independentes sob os mesmos nomes. Existem, no entanto, alternativas, como matplotlib, seaborn e altair, para criação de gráficos em Python, sendo matplotlib a biblioteca de visualização mais fundamental em Python e apenas altair possui uma maior aderência aos conceitos da GoG.

Os pacotes ggplot2 e plotnine ilustram o poder da GoG em R e Python, respectivamente. Com eles, é possível construir a partir de dados abstratos gráficos complexos e personalizados de forma incremental e intuitiva, enquanto a criação de gráficos simples permanece direta e descomplicada.

3.5.3 Exemplo

Esta seção demonstra funcionalidades da Gramática dos Gráficos, implementadas no pacote ggplot2, utilizando o *dataset* diamonds[5]. do pacote tidyverse [101]. As linhas de comando em R utilizadas são fornecidas para reprodução.

Primeiramente, o conjunto de dados diamonds do pacote tidyverse é carregado com:

```
library (tidyverse)
data(diamonds)
```

[3] `https://datacarpentry.github.io/python-ecology-lesson/07-visualization-ggplot-python.html`
[4] `https://datacarpentry.github.io/python-ecology-lesson/07-visualization-ggplot-python.html`
[5] Disponível no *github*

A função print permite listar os dados-valores carregados. O conjunto contém 53.940 itens como mostra a Figura 3.12.

```
> diamonds
# A tibble: 53,940 × 10
   carat cut       color clarity depth table price     x     y     z
   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
 7  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47
 8  0.26 Very Good H     SI1      61.9    55   337  4.07  4.11  2.53
 9  0.22 Fair      E     VS2      65.1    61   337  3.87  3.78  2.49
10  0.23 Very Good H     VS1      59.4    61   338  4     4.05  2.39
# ℹ 53,930 more rows
# ℹ Use `print(n = ...)` to see more rows
```

Figura 3.12: Dados-valores brutos do conjunto diamonds carregados no sistema.

Na etapa **Variáveis**, os conjuntos de variáveis (*varsets*) são gerados para possibilitar consultas por valores, como exemplificado na Figura 3.13.

```
ideal <- filter (diamonds, cut == 'Ideal')
premium <- filter (diamonds, cut == 'Premium')
```

Estando os dados preparados para consultas por valor, as operações algébricas e lógicas sobre os valores na etapa **Álgebra** podem ser eficientemente computadas. Por exemplo, o seguinte comando

```
algebra <- select (diamonds, carat, cut, price)
```

gera um novo conjunto de variáveis

A etapa **Escalas** define como os valores das variáveis são mapeados nos eixos e nos elementos visuais do gráfico. Para ilustrar o efeito visual da etapa **Escalas**, é necessário definir a **Estética** que mapeia cada variável a um atributo visual e a **Geometria** que define como os dados são representados geometricamente. Ao mapearmos as variáveis price, carat e cut para os atributos visuais de coordenada x, coordenada y e cor, respectivamente, utilizando a geometria de pontos (geom_point), obtivemos o gráfico apresentado na Figura 3.15.

```
ggplot (data = algebra) +
  geom_point (mapping = aes(x=price,y=carat,color=cut))
```

Os intervalos entre as marcas nos eixos foram definidos automaticamente pelos valores padrão da Hierarquia de *defaults* do sistema. Caso o usuário deseje personalizar esses intervalos, é possível reconfigurá-los na etapa **Escalas**, como ilustrado nas linhas de comando apresentadas a seguir.

```
> premium <- filter (diamonds, cut == 'Premium')
> print(premium)
# A tibble: 13,791 × 10
   carat cut     color clarity depth table price     x     y     z
   <dbl> <ord>   <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
 1  0.21 Premium E     SI1      59.8    61   326  3.89  3.84  2.31
 2  0.29 Premium I     VS2      62.4    58   334  4.2   4.23  2.63
 3  0.22 Premium F     SI1      60.4    61   342  3.88  3.84  2.33
 4  0.2  Premium E     SI2      60.2    62   345  3.79  3.75  2.27
 5  0.32 Premium E     I1       60.9    58   345  4.38  4.42  2.68
 6  0.24 Premium I     VS1      62.5    57   355  3.97  3.94  2.47
 7  0.29 Premium F     SI1      62.4    58   403  4.24  4.26  2.65
 8  0.22 Premium E     VS2      61.6    58   404  3.93  3.89  2.41
 9  0.22 Premium D     VS2      59.3    62   404  3.91  3.88  2.31
10  0.3  Premium J     SI2      59.3    61   405  4.43  4.38  2.61
# ℹ 13,781 more rows
# ℹ Use `print(n = ...)` to see more rows
> ideal <- filter (diamonds, cut == 'Ideal')
> print(ideal)
# A tibble: 21,551 × 10
   carat cut   color clarity depth table price     x     y     z
   <dbl> <ord> <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
 1  0.23 Ideal E     SI2      61.5    55   326  3.95  3.98  2.43
 2  0.23 Ideal J     VS1      62.8    56   340  3.93  3.9   2.46
 3  0.31 Ideal J     SI2      62.2    54   344  4.35  4.37  2.71
 4  0.3  Ideal I     SI2      62      54   348  4.31  4.34  2.68
 5  0.33 Ideal I     SI2      61.8    55   403  4.49  4.51  2.78
 6  0.33 Ideal I     SI2      61.2    56   403  4.49  4.5   2.75
 7  0.33 Ideal J     SI1      61.1    56   403  4.49  4.55  2.76
 8  0.23 Ideal G     VS1      61.9    54   404  3.93  3.95  2.44
 9  0.32 Ideal I     SI1      60.9    55   404  4.45  4.48  2.72
10  0.3  Ideal I     SI2      61      59   405  4.3   4.33  2.63
# ℹ 21,541 more rows
# ℹ Use `print(n = ...)` to see more rows
```

Figura 3.13: Transformação dos dados-valores tabulares em conjuntos de variáveis (*varsets*) organizados em linhas e colunas.

```
> algebra <- select (diamonds, carat, cut, price)
> print (algebra)
# A tibble: 53,940 × 3
   carat cut       price
   <dbl> <ord>     <int>
 1  0.23 Ideal       326
 2  0.21 Premium     326
 3  0.23 Good        327
 4  0.29 Premium     334
 5  0.31 Good        335
 6  0.24 Very Good   336
 7  0.24 Very Good   336
 8  0.26 Very Good   337
 9  0.22 Fair        337
10  0.23 Very Good   338
# ℹ 53,930 more rows
# ℹ Use `print(n = ...)` to see more rows
```

Figura 3.14: Operação de produto entre os conjuntos de variáveis carat, cut e price do conjunto de dados diamonds.

```
ggplot (data = algebra) +
  geom_point (mapping = aes(x=price,y=carat,color=cut)) +
  scale_x_continuous (breaks = seq(2500,17500,by=2500)) +
  scale_y_continuous (breaks = seq(0.5,5,by=0.5))
```

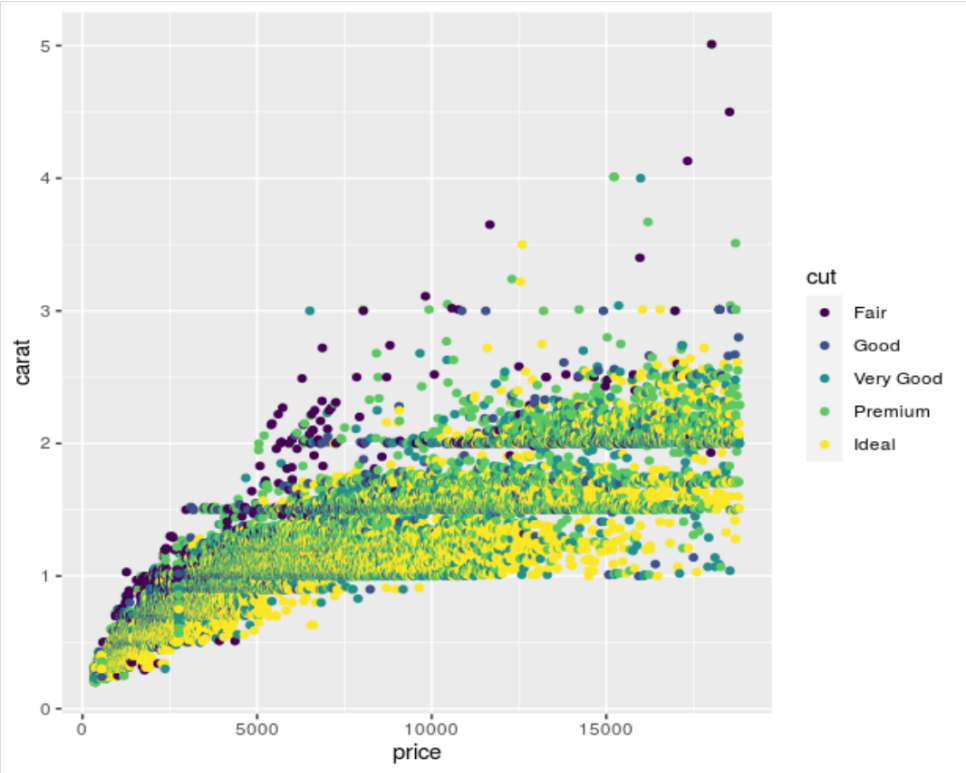


Figura 3.15: Mapeamento dos conjuntos das variáveis carat, cut e price do conjunto de dados diamonds em um gráfico estatístico.

Compare a diferença entre as marcas padrão na Figura 3.15 e as marcas personalizadas na Figura 3.16.

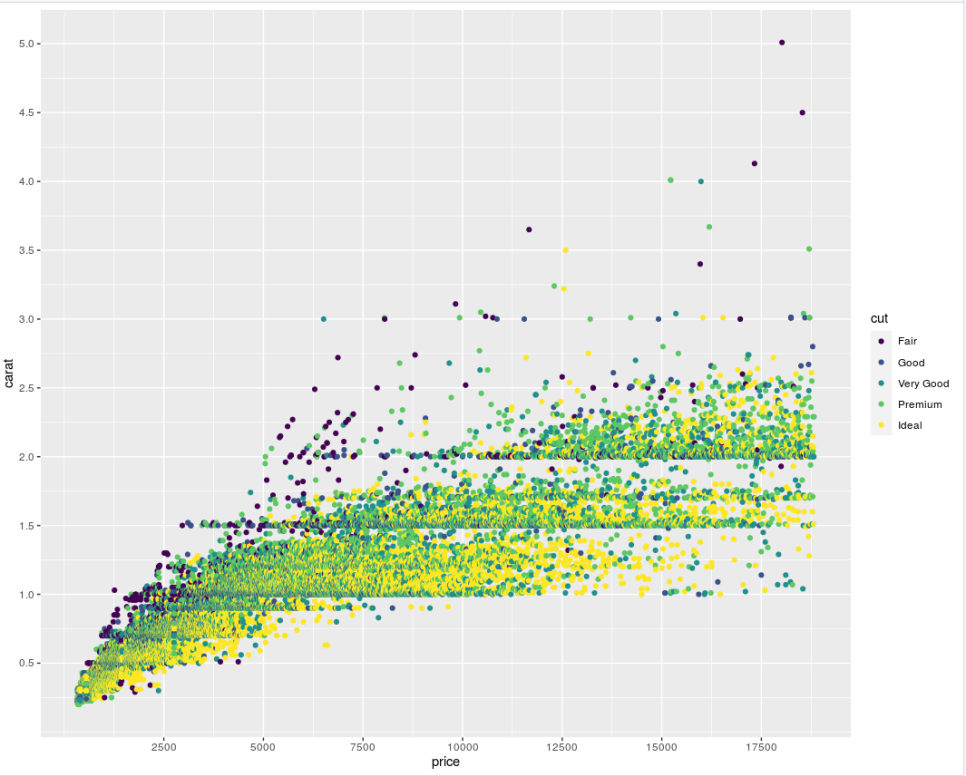


Figura 3.16: Personalização das escalas dos gráficos estatísticos.

Sobre as variáveis pode-se computar os resumos estatísticos ou aplicar transformações estatísticas, como regressão e clusterização, na etapa **Estatísticas**. As seguintes linhas de comando calculam a média, desvio-padrão e a mediana dos quilates e dos preços das 53.940 amostras de diamantes.

```
mean(diamonds$carat)
sd(diamonds$carat)
median(diamonds$carat)
mean(diamonds$price)
sd(diamonds$price)
median(diamonds$price)
```

Os resultados foram $\mu_{carat} = 0.7979397$, $\sigma_{carat} = 0.4740112$, $Md_{carat} = 0.7$, μ_{price} = US$ 3932.8, σ_{price} = US$ 3989.44, Md_{price} = US$ 2401.

A GoG facilita a exploração de dados através de diferentes representações geométricas. Para mudar um gráfico de dispersão para um outro gráfico estatístico, basta substituir a geometria geom_point por outra suportada pelo ggplot2[6]. Por exemplo, para visualizar o histograma e o gráfico de frequências da variável price, segmentados pela variável cut, nas 53.940 tuplas do *dataset* diamonds, substituímos a linha de definição de geometria para a Figura 3.15 pelos seguintes comandos, respectivamente:

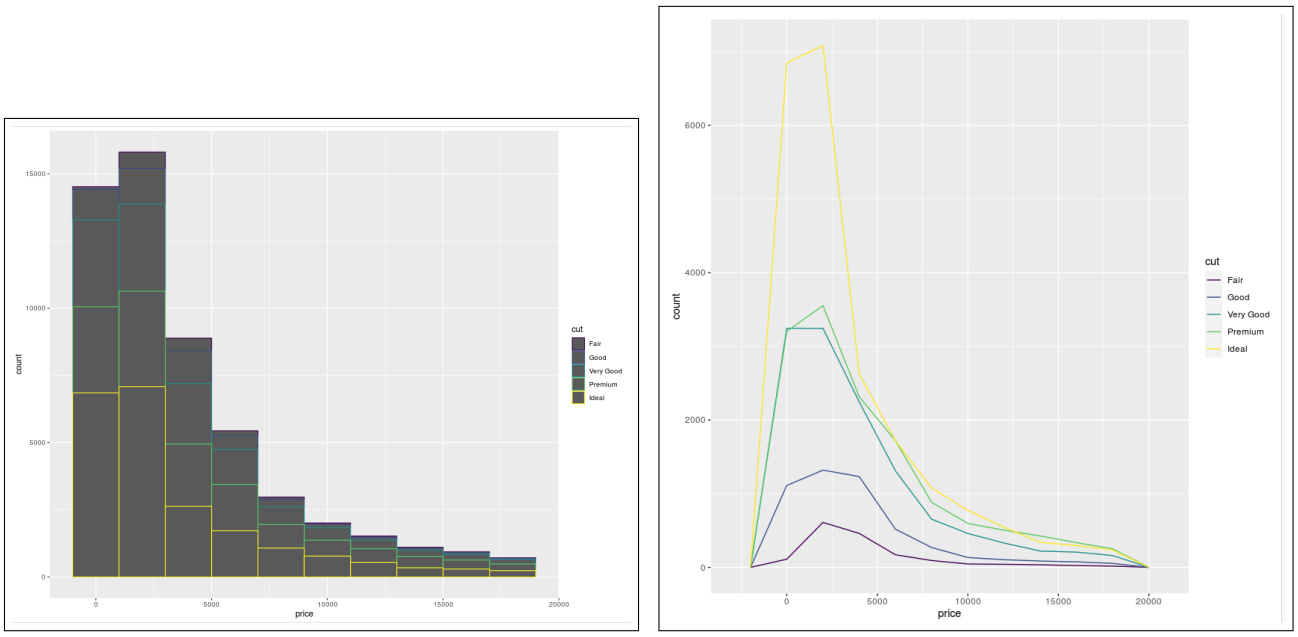
```
ggplot (data = algebra) +
  geom_histogram (mapping = aes(x=price, color=cut), binwidth=2000)
ggplot (data = algebra) +
  geom_freqpoly (mapping = aes(x=price, color=cut), binwidth=2000)
```

Como resultado, temos duas outras formas, apresentadas na Figura 3.17, para visualizar o mesmo conjunto de dados. Enquanto o gráfico de dispersão na Figura 3.15 destaca a relação entre quilates e preço, mostrando como o preço geralmente aumenta com o aumento dos quilates e com o corte, o histograma e o gráfico de frequência na Figura 3.17 permitem comparar a distribuição de preços entre os diferentes cortes de diamantes.

O pacote ggplot2 permite configurar o sistema de referência para as coordenadas das variáveis. O sistema padrão é o cartesiano, mas outros sistemas, como o polar[7], podem ser utilizados. A Figura 3.18 mostra variantes do histograma da Figura 3.17, um com os eixos x e y trocados em um sistema cartesiano e outro utilizando o sistema polar. Esses efeitos são obtidos com os seguintes comandos, respectivamente:

```
ggplot (data = algebra) +
  geom_histogram (mapping = aes(x=price, color=cut), binwidth=2000) +
```

[6] https://ggplot2.tidyverse.org/reference/#geoms
[7] https://ggplot2.tidyverse.org/reference/#coordinate-systems

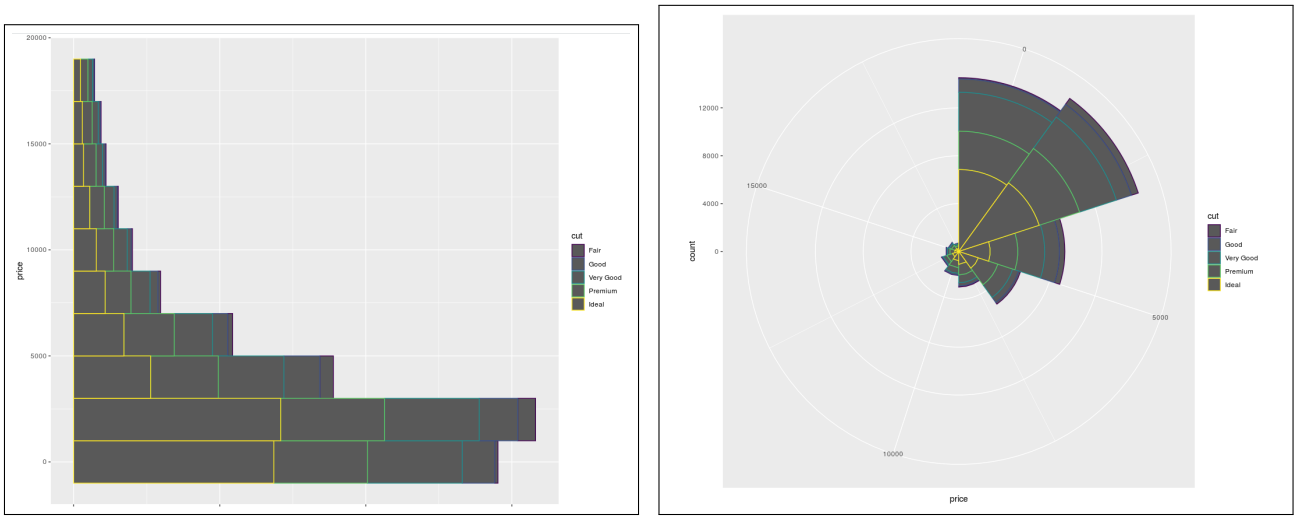


(a) Histograma (b) Gráfico de Frequências

Figura 3.17: Diferentes geometrias para expressar frequências de ocorrência.

```
  coord_flip ()
ggplot (data = algebra) +
  geom_histogram (mapping = aes(x=price, color=cut), binwidth=2000) +
  coord_polar ()
```

Vale destacar que o plotnine não oferece suporte para coordenadas polares [33].



(a) Inversão de Eixos (b) Coordenadas Polares

Figura 3.18: Diferentes sistemas de coordenadas para representar um mesmo conjunto de variáveis.

O uso de **camadas** para composição de diferentes informações em um único gráfico é útil para comparações e para descobrir correlações, tendências e *insights* ocultos, como demonstra a Figura 3.19. Nessa figura são sobrepostas 2 camadas. A primeira camada é a de gráfico de dispersão que

permite identificar a presença de correlação entre os valores das variáveis carat e price, enquanto a segunda camada é o gráfico de uma função que expressa a correlação linear entre essas duas variáveis. O comando que define essas duas camadas é

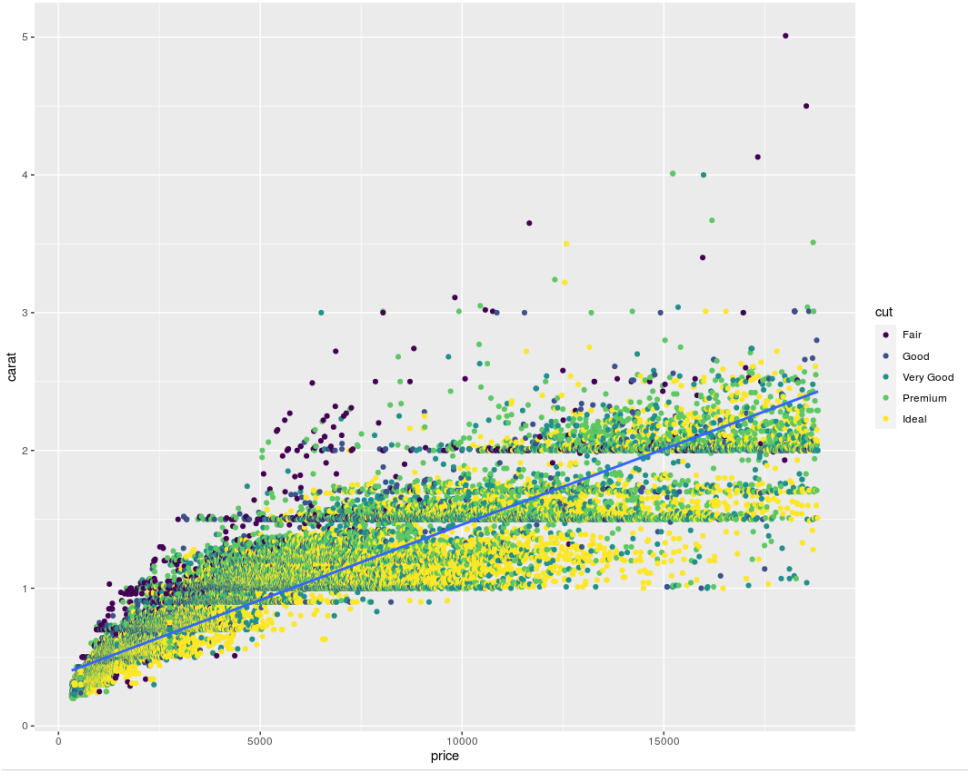


Figura 3.19: Gráfico composto de duas camadas: uma camada de gráfico de dispersão e uma camada de correlação entre o preço e o quilate dos diamantes.

```
ggplot (data = algebra, aes(x=price,y=carat)) +
  geom_point (mapping = aes(color=cut)) +
  geom_smooth(method = "lm", se = FALSE)
```

Para analisar a distribuição ou relação entre variáveis em diferentes categorias de forma mais clara, utiliza-se a visualização multifacetada. O seguinte comando divide o gráfico de dispersão da Figura 3.15 em cinco gráficos menores, cada um representando uma das categorias de corte de diamantes.

```
ggplot (data = algebra, aes(x=price,y=carat, color=cut)) +
  geom_point () +
  facet_wrap (~cut, nrow=2)
```

Os 5 gráficos na Figura 3.20 permitem uma visualização das relações entre preços e quilates por tipo de corte. Elas revelam claramente que a correlação entre carat e price é aproximadamente linear para todos os 5 cortes. Além disso, observa-se que a dispersão dos preços aumenta à medida que o quilate do diamante aumenta para todos os 5 cortes.

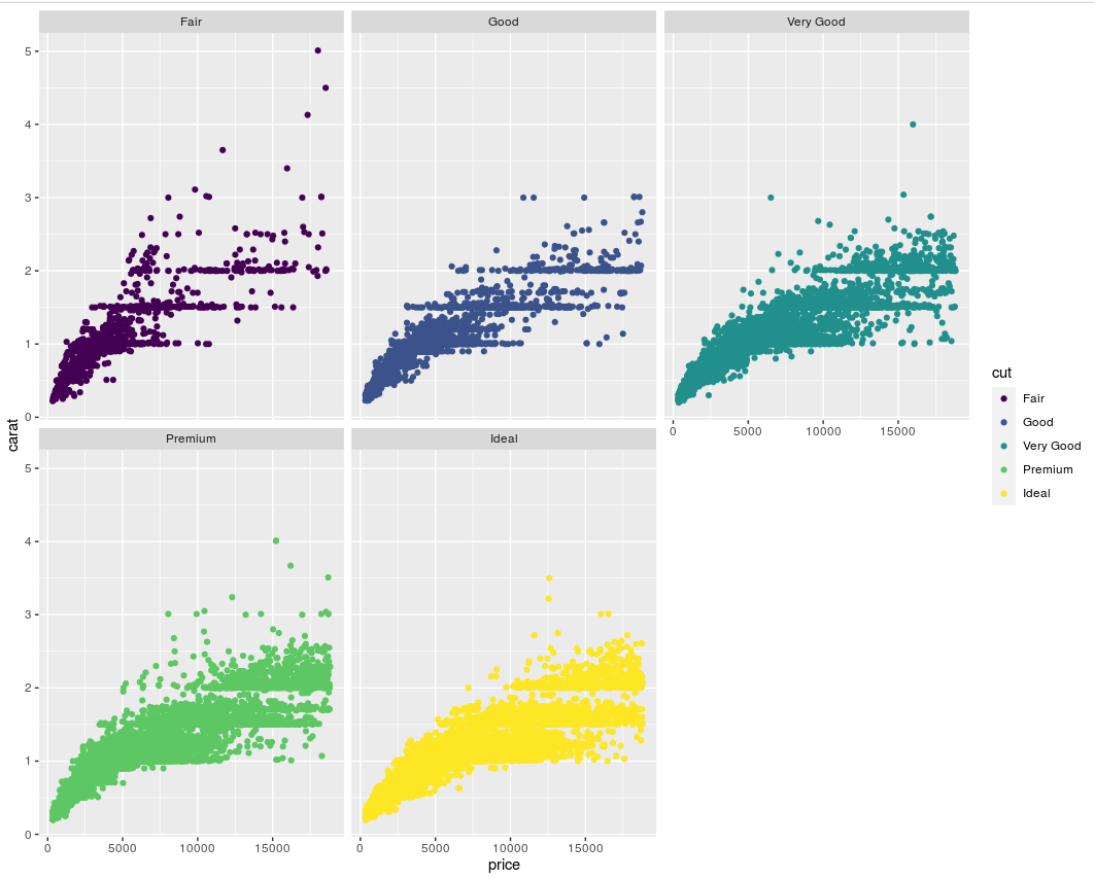


Figura 3.20: Uso de pequenos múltiplos de gráficos de dispersão para identificar padrões específicos em diferentes categorias de cortes de diamantes.

3.6 Considerações Finais

Ao explorar o *design* de interfaces com base nos princípios de Schneiderman, Ware e Tufte, fica evidente a necessidade de uma abordagem cuidadosa e equilibrada entre as áreas de controle e visualização. A clara separação entre essas áreas é essencial para otimizar a experiência do usuário. Schneiderman enfatiza a importância de princípios como usabilidade, *feedback* imediato e interação intuitiva na área de controle. Esses elementos são cruciais para criar interfaces que proporcionem uma experiência eficiente e satisfatória ao usuário, como discutido na Seção 3.1. Com foco em percepção e cognição, Ware se ocupa com a representação e renderização dos dados abstratos (renderização bidimensional) e físicos (renderização tridimensional "fotorealista") na área de visualização (Seção 3.2). Tufte, por sua vez, foca na capacidade informativa da visualização de dados abstratos para análise estatística (Seção 3.2).

Atualmente, somos privilegiados com uma variedade de aplicativos de renderização 2D e 3D de alta qualidade, proporcionando um ambiente propício para a exploração visual de dados físicos [91] e abstratos [58] complexos. No entanto, o verdadeiro desafio reside na habilidade de mapear os dados abstratos em elementos gráficos que não apenas sejam esteticamente agradáveis, mas que também facilitem a extração de informações relevantes e a obtenção de *insights* decisivos. Aqui, a abordagem

minimalista de Tufte é crucial. Ele sustenta que gráficos estatísticos simples, diretos e desprovidos de elementos desnecessários são mais eficazes na transmissão de informações.

Nesse contexto, a Gramática dos Gráficos (GoG) se destaca como uma ferramenta poderosa. Ao adotar seus princípios, o *designer* assegura que os dados sejam representados de forma flexível, independentemente da maneira como foram coletados ou inicialmente apresentados. A GoG permite adaptar as visualizações à natureza dos dados, garantindo representações adequadas e eficazes, alinhadas a princípios fundamentais de autores como Ware e Tufte. Entre as principais implementações da GoG, destacam-se o ggplot2 (R) e o plotnine (Python). Integrar os princípios de *design* de Schneiderman com a GoG resulta em interfaces que permitem aos usuários explorar dados de maneira intuitiva e eficiente.

Tanto o R quanto o Python possuem pacotes poderosos para o desenvolvimento de aplicativos *web* interativos que oferecem uma rica experiência de exploração de dados. O Shiny[8] é o pacote mais popular para criar aplicativos *web* interativos em R, permitindo a renderização de gráficos criados com o ggplot2. No caso do Python, o Dash[9] é a ferramenta mais amplamente utilizada para construir aplicativos *web*, embora sua integração com o plotnine não seja direta. Contudo, o Dash, integrado em Plotly Python[10], oferece suporte ao altair[11], outro pacote de visualização que também segue os princípios da GoG.

Neste curso, focaremos especificamente no uso do ggplot2 (em R) e do plotnine (em Python) para criar visualizações interativas, proporcionando uma base sólida para o *design* de interfaces eficientes e acessíveis para exploração de dados.

3.7 Exercícios

1. Existem vários *sites* e recursos *online* que apresentam exemplos de *designs* de interface gráfica considerados ruins. Alguns deles são:

    - Avoid these 15 bad UI design mistakes (with examples).
    - Interface Hall of Shame.
    - 7 Bad UI Design Examples You Can Learn From.

    Esses recursos podem ser úteis para projetistas de interface de usuário e desenvolvedores, pois oferecem *insights* sobre o que não fazer em seus próprios projetos e inspiração para evitar erros comuns de *design*. Relacione cada um dos erros mencionados nos três *sites* citados aos princípios discutidos ao longo do capítulo que foram violados.

[8] `https://shiny.posit.co/`
[9] `https://dash.plotly.com/`
[10] `https://plotly.com/python/`
[11] `https://altair.com/`

2. Leia o Capítulo 2 em [36] e responda as seguintes LC (do inglês *Learning Checks*) do capítulo: LC2.5, LC2.6, LC2.8, LC2.13, LC2.14, LC2.18, LC2.20, LC2.22, LC2.27, LC2.30, LC2.31, LC2.36.

   Observações:

   - Os conjuntos de dados alaska_flights e early_january_weather são do pacote moderndive de R. Baixe-os como alaska_flights.csv e early_january_weather.csv e carregá-los com o comando

     ```
     import pandas as pd
     alaska_flights = pd.read_csv("<caminho>/alaska_flights.csv")
     weather = pd.read_csv("<caminho>/early_january_weather.csv")
     ```

     Uma outra alternativa para carregá-lo em Python é usar comandos R no ambiente Python via rpy2 conforme mostrado na Seção 2.4.
   - Explique na documentação Markdown os passos aplicados na geração de cada gráfico, quando pertinente.
   - Repositório de conjunto de dados incorporados em R, como nycflights13.
   - Plotnine = ggplot em Python.

Capítulo 4

Percepção Visual e Cognição

A **percepção** se refere à interpretação e organização das informações sensoriais obtidas pelos cinco sentidos primários, visão, audição, tato, olfato e paladar. Além desses cinco sentidos primários, existem outros sentidos que desempenham papéis importantes na percepção e na interação com o ambiente, como o sistema vestibular, responsável pelo equilíbrio e pela orientação espacial, e o sentido proprioceptivo, que fornece informações sobre a posição e movimento do corpo. A **cognição**, por outro lado, envolve processar, interpretar e extrair *insights* a partir dessas informações sensoriais e do conhecimento anterior. Ela abrange processos mais complexos, como raciocínio, inferência, planejamento e a capacidade de usar a linguagem para comunicar ideias.

Os princípios de projeto de interface, discutidos no Capítulo 3, e a cognição estão profundamente interligados, já que a maneira como uma interface é projetada influencia diretamente como os usuários processam, percebem e interagem com as informações apresentadas. Portanto, entender o modo como os seres humanos percebem e processam informações visuais é fundamental para criar representações visuais que sejam intuitivas, informativas, persuasivas e capazes de facilitar a tomada de decisões. Essa compreensão é especialmente valiosa para profissionais de ciência de dados que buscam aprimorar suas habilidades na comunicação de informações ocultas nos dados por meio de representações visuais.

Ao aplicar os princípios fundamentais da percepção visual, é possível desenvolver visualizações mais eficazes, capazes não apenas de cativar a atenção, mas também de transmitir *insights* de maneira clara e impactante. Elementos visuais, como *layout*, cores e tamanho dos elementos, influenciam diretamente a atenção e a percepção dos usuários, enquanto a organização da informação e o *feedback* fornecido têm um impacto significativo na memória e no processamento cognitivo. Além disso, uma interface bem projetada facilita a formação de modelos mentais precisos sobre os dados subjacentes, tornando a interação mais intuitiva e eficiente. Isso transforma dados complexos em narrativas visuais acessíveis, promovendo uma comunicação mais efetiva e proporcionando uma compreensão mais profunda dos padrões e relações presentes nos conjuntos de dados.

Por exemplo, ao alinhar os princípios da percepção visual, como a *Gestalt* (Seção 4.3) e a constância

perceptiva (Seção 4.1), com o *design* da interface gráfica, os desenvolvedores têm a oportunidade de criar representações visuais simples e diretas. Essas representações facilitam a identificação de padrões, tendências e anomalias nos dados. Num contexto onde a capacidade de identificar rapidamente informações relevantes é crucial, uma abordagem pré-atentiva (Seção 4.2) na interface gráfica pode aprimorar significativamente a eficácia da análise.

Este capítulo é organizado na seguinte forma. Apresentamos uma introdução à psicofísica visual na Seção 4.1, abordamos os princípios fundamentais da percepção visual nas Seções 4.2 e 4.3, incluindo a teoria da *Gestalt*, e proporcionamos uma visão abrangente da influência da cognição humana na interpretação de dados visuais na Seção 4.4. Além disso, exploramos na Seção 4.5 como esses conceitos podem ser aplicados no *design* de uma interface gráfica interativa para um sistema de análise visual de dados, com foco na resolução de problemas complexos que ainda não possuem soluções analíticas. Essa abordagem visa criar uma ponte eficaz entre a compreensão das características perceptivas humanas e a implementação prática de ferramentas visuais que potencializem a interpretação e a extração de *insights* a partir de conjuntos de dados desafiadores.

4.1 Psicofísica Visual

A **psicofísica visual** representa uma vertente da psicofísica dedicada a examinar a interação entre estímulos visuais percebidos pelos olhos e as respostas ou percepções subjetivas resultantes desses estímulos. Além disso, seu escopo abrange a exploração das interações entre os processos fisiológicos e perceptuais que moldam a experiência visual humana. Os cohecimentos advindos desse campo são frequentemente aplicados em diversas áreas, incluindo *design* gráfico, ciência cognitiva, psicologia experimental e disciplinas afins.

A visão humana, um sistema intrincado que se inicia nos dois olhos, é um processo complexo em que as radiações luminosas atravessam o cristalino, uma lente biconvexa flexível, e convergem na retina, localizada na porção posterior do olho. A retina contém cerca de centenas de milhões de fotoreceptores. Quando estimulados, esses fotoreceptores desencadeiam transições fotoquímicas, gerando impulsos elétricos. Esses impulsos são, então, transmitidos pelo nervo óptico ao cérebro, resultando em sensações luminosas que podem ser aproximadas por uma convolução gaussiana do sinal, capturado por uma função de espalhamento pontual (em inglês, *point spread function* – PSF). Os **movimentos oculares**, como movimentos sacádicos, são, de fato, os responsáveis pela captura de uma sequência de ângulos diferentes dos objetos no campo visual, permitindo a exploração do ambiente e a percepção inconsciente de detalhes.

A retina é revestida por células fotossensíveis especializadas, os bastonetes e os cones. Os **bastonetes** são mais sensíveis à luz e são responsáveis pela visão em condições de pouca luz (visão escotópica). Eles não são tão eficientes na percepção de cores quanto os cones. Os **cones**, por sua

vez, são responsáveis pela visão de cores e pela percepção de detalhes. Existem três tipos principais de cones, sensíveis a diferentes comprimentos de onda da luz. Eles são chamados de cones de curto (S, tonalidade azul), médio (M, tonalidade verde), e longo (L, tonalidade vermelha) comprimentos de onda. O espaço de cores perceptíveis pode ser decomposto em um canal de luminância (ou intensidade) e dois canais de crominância (ou cromas), permitindo distinguir entre variações de intensidade e variações de cor:

Canal de Luminância : está associado ao brilho ou intensidade da luz. Ele é influenciado pela atividade combinada de todos os três tipos de cones. O canal de luminosidade contribui para a percepção geral da **luminosidade** de uma cena.

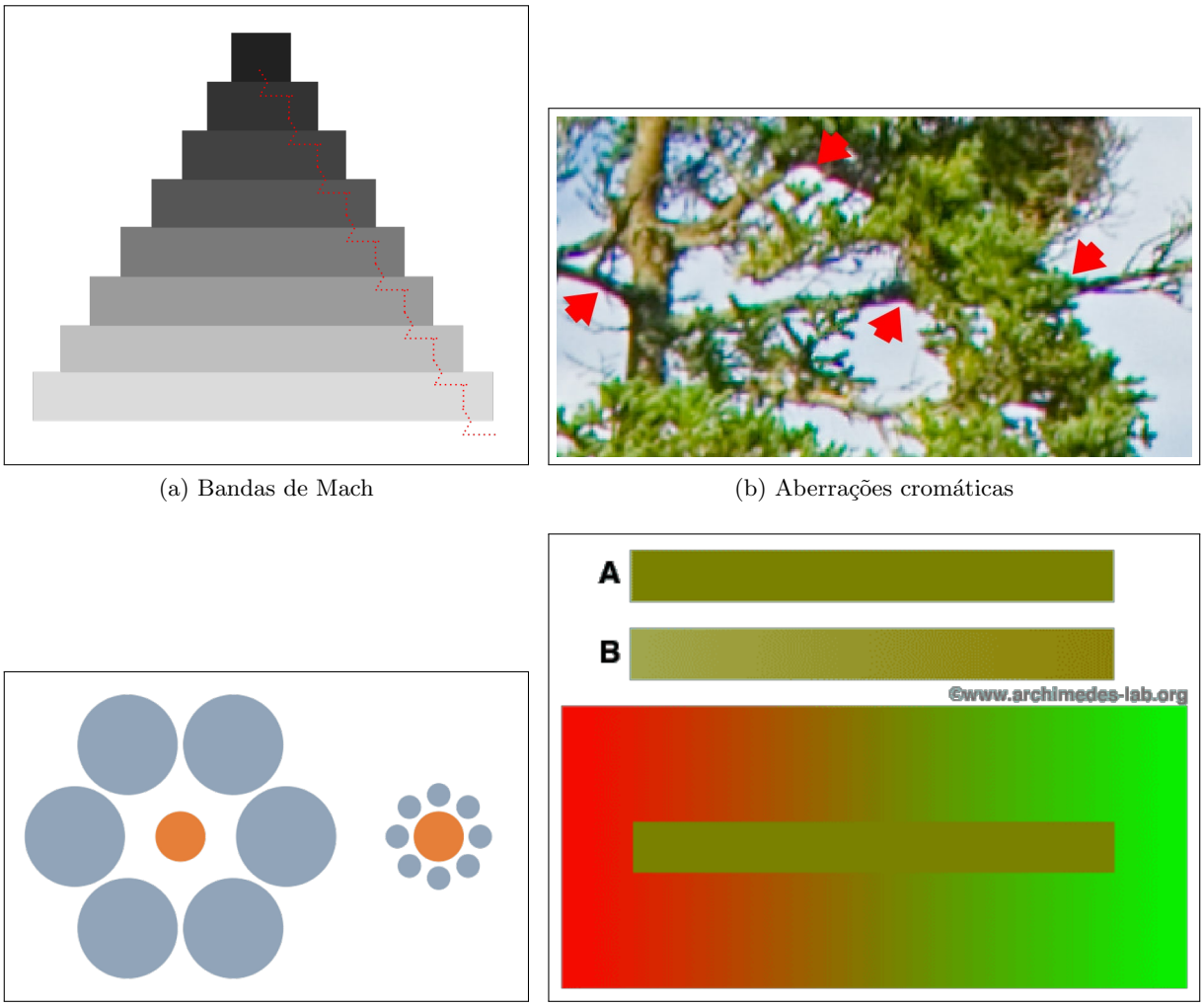
Canal de Cor Vermelha-Verde : é formado pela diferença na atividade dos cones de comprimento de onda longo (L) e médio (M). A variação na atividade desses cones cria uma resposta relativa à luz nas porções vermelha e verde do espectro de cores. O canal de cor vermelha-verde é essencial para discriminar entre essas duas cores, vermelho e verde.

Canal de Cor Azul-Amarelo : é formado pela diferença na atividade dos cones de comprimento de onda curto (S) e a média ponderada dos cones de comprimento de onda longo (L) e médio (M). A variação nessa atividade cria uma resposta relativa à luz nas porções azul e amarela do espectro de cores. O canal de cor azul-amarelo discrimina entre essas duas cores, azul e amarelo.

A imagem formada na retina é real, invertida e menor do que o objeto. Contudo, nossa percepção "corrige" essa visualização graças ao processamento dos sinais visuais. Conforme um modelo de percepção visual de três estágios, a ser explicado na Seção 4.4, os sinais luminosos passam pela transformação inicial em características elementares, como forma, cor, textura e orientação. Essas características simples são, então, agrupadas em padrões (visão de baixo nível). Em seguida, a atenção do observador ativa objetos armazenados na memória para criar a percepção completa de um objeto, seja para reconhecê-lo ou classificá-lo (visão de alto nível). Essa capacidade de perceber um todo é sintetizada nas **leis de Gestalt**, que serão detalhadas na Seção 4.3.

Essa complexa cadeia de eventos destaca a importância da **acuidade visual** para detectar as mínimas diferenças entre elementos em termos de cor, forma ou tamanho. Além disso, a **ação inibidora** fora do centro do campo receptivo de um fotoreceptor contribui para a percepção visual. A **inibição lateral**, como é conhecida, faz com que regiões de variações abruptas de luz pareçam mais claras ou mais escuras quando em contato com áreas contrastantes, resultando em fenômenos ilusórios como as Bandas de Mach (Figura 4.1a). Outra peculiaridade visual intrigante é a **aberração cromática**, originada pela disparidade nos pontos focais de radiações de diferentes comprimentos de onda ao atravessarem o cristalino. Esse fenômeno proporciona a percepção de objetos com cores

distintas em diferentes profundidades. Por exemplo, ao observar um arco-íris, é possível notar que as cores vermelhas parecem estar mais distantes do que as cores azuis, como mostra a Figura 4.1b.



(a) Bandas de Mach

(b) Aberrações cromáticas

(c) Ilusão de Ebbinghaus

(d) Contraste em cores simultâneas

Figura 4.1: Ilusões ópticas: interpretação equivocada do cérebro sobre as informações visuais recebidas dos olhos, revelando as limitações e as peculiaridades do sistema visual humano. (Fontes: Wikipedia, Phototraces, Cleartelligence e Archimedes' Laboratory)

A nossa visão é inerentemente estérea. Quando direcionamos nosso olhar para um objeto, duas imagens retinianas distintas são formadas, uma em cada olho. Devido ao espaçamento entre os olhos, essas imagens não são idênticas. A observação simultânea dessas duas imagens, que são ligeiramente diferentes, desencadeia movimentos musculares nos olhos para criar paralaxe na distância entre dois pontos de alturas distintas. Esses movimentos permitem ao cérebro discernir a distância entre esses pontos, resultando na **percepção de relevo e profundidade**. É interessante notar que estudos indicam que cerca de 32% da população não possui visão estéreo apropriada [34], no entanto, mesmo nesses casos, a percepção de profundidade não é tão comprometida. Outros indícios visuais, como oclusão, tonalidade e perspectiva, desempenham um papel crucial na criação da sensação de profundidade, enriquecendo nossa experiência visual de forma complementar.

A percepção visual se distingue pela capacidade de manter a estabilidade na percepção de atri-

butos fundamentais, como tamanho, forma e cor, independentemente das variações nas condições de observação, como distância, ângulo e iluminação. Esse fenômeno é conhecido como **constância perceptiva**, e reflete a habilidade do sistema visual em filtrar variações externas, garantindo uma percepção estável e confiável desses atributos essenciais. A constância perceptiva é uma característica robusta, mas ela pode ser influenciada por configurações visuais específicas, especialmente elementos discrepantes, que desafiam a percepção relativa, como aquelas encontradas na ilusão de Ebbinghaus (ilusão de tamanho), mostrada na Figura 4.1c, e na ilusão de cores simultâneas (ilusão de cor), apresentada na Figura 4.1d. Essas ilusões ressaltam a complexidade da interpretação visual e mostram como o cérebro considera o contexto ao processar informações visuais.

No entanto, ao integrar deliberadamente essas ilusões ópticas nas visualizações de dados, os *designers* podem aprimorar a compreensão e a interpretação das informações, tornando as visualizações mais elucidativas. Por exemplo, a Banda de Machs pode ser aplicada na visualização de dados, facilitando a percepção de padrões e relações nos dados ao criar uma borda clara entre áreas de diferentes valores ou categorias. Da mesma forma, a ilusão de Ebbinghaus pode ser empregada para destacar a importância relativa de um dado em relação aos outros, ao representar esses dados em tamanhos relativos apropriados. Além disso, a ilusão de cor nos mostra que determinadas combinações de cores podem criar contrastes ilusórios e ser utilizadas em visualizações de dados para realçar padrões ou tendências de forma mais perceptível. E, embora o objetivo principal da visualização de dados seja comunicar informações de maneira clara e precisa e as aberrações cromáticas possam comprometer a clareza se utilizadas indiscriminadamente, essas aberrações podem ser empregadas com moderação. Elas podem oferecer uma abordagem única e interessante para a apresentação visual dos dados, como abordar a incerteza inerente aos conjuntos de dados. Em um gráfico de dispersão, a aberração cromática pode ser aplicada aos pontos com maior incerteza, destacando-os e indicando que seus valores podem não ser tão precisos [35].

Por último, devido às limitações de nossos recursos cognitivos, especialmente na memória, em ambientes visuais saturados de informações, numerosos estímulos competem pela nossa atenção. Apenas alguns, com características visuais marcantes, se destacam e são processados de maneira **pré-atentiva**, ocorrendo de forma inconsciente. Por outro lado, muitos estímulos podem ser descartados antes mesmo de serem reconhecidos. Em ambientes complexos, alterações sutis, mesmo quando significativas, frequentemente passam despercebidas. Esse fenômeno, conhecido como **cegueira às mudanças** (*change blindness* em inglês), demonstra como pequenas modificações em um cenário podem escapar da nossa percepção, especialmente quando estamos focados em outros elementos, como ilustrado pelo vídeo em [14].

4.2 Atributos Pré-atentivos

Para otimizar a busca visual por informações em um conjunto volumoso de dados, é vantajoso codificar suas características em atributos gráficos que as tornem distintas (salientes) e visíveis em um campo visual de abertura menor que 5 graus, alinhado com os movimentos sacádicos dos nossos olhos. A eficácia da ativação de uma estratégia de busca adequada também está intrinsecamente ligada às experiências do usuário. Quanto mais familiarizado o usuário estiver com as cenas visualizadas, mais efetiva se torna a construção de mapas de características que sustentam buscas visuais.

Segundo Ware [98], características como cor, forma (orientação e tamanho), profundidade e movimento são processadas em "canais" visuais distintos. Cada uma delas pode ser associada a um mapa de características que direciona o movimento dos olhos na busca controlada/consciente por uma característica específica. Entretanto, nosso sistema visual também realiza processamentos **pré-atentivos** capazes de destacar alguns elementos em relação aos outros (elementos vizinhos e/ou fundo), de maneira não controlada/inconsciente. Ware ilustra em [98] elementos pré-atentivos e atentivos por meio de uma série de exemplos. Ele classifica como pré-atentivos os seguintes elementos gráficos (Figura 4.2):

- orientação de linha,
- espessura de linha,
- comprimento de linha,
- tamanho,
- curvatura,
- agrupamento espacial,
- borramento,
- marcas adicionais,
- quantidade,
- cor,
- tonalidade,
- movimento (cintilhamento, direção do movimento),
- localização espacial, e
- convexidade.

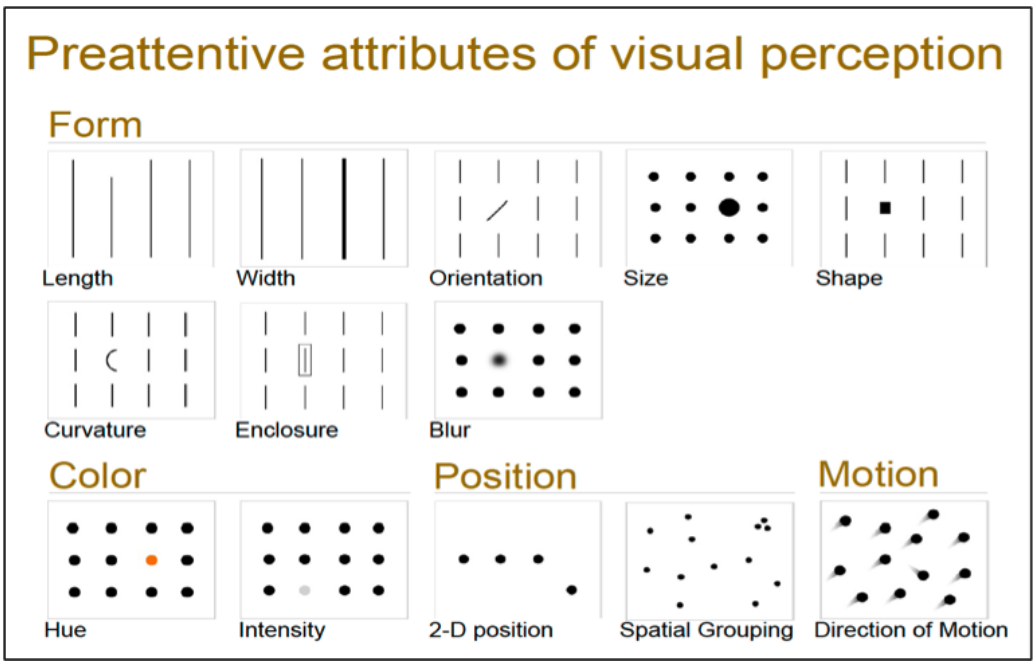


Figura 4.2: Atributos pré-atentivos na percepção visual humana. (Fonte: Relationship|One)

Ware destaca a forte dependência do processamento pré-atentivo em relação ao contexto. Um elemento pode ser destacado pré-atentivamente dos demais mesmo quando variando apenas nos valores de uma mesma característica. Em contraste com a formação de mapas de características em canais distintos, uma marca pode ser considerada pré-atentiva se destacar-se de outras marcas dentro do mesmo canal. Por exemplo, a incorporação de marcas, denominadas por Ware como **assimetrias** (em inglês, *asymmetries*), aos elementos de interesse pode ser eficaz para realçá-los pré-atentivamente. Além disso, se a intenção é intensificar ainda mais esse destaque, podemos empregar a **codificação redundante**, que consiste em associar um mesmo atributo a diferentes **características visuais separáveis**.

Uma observação importante sobre dados físicos, que frequentemente possuem um domínio "nativo" contínuo, é que, mesmo quando amostrados como dados discretos, a visualização contínua de atributos escalares, como a temperatura em um mapa de aquecimento global, tende a ser mais intuitiva. Por isso, é comum utilizar a interpolação para suavizar os valores dos elementos gráficos em que os atributos físicos são mapeados. Assim, há uma preferência por elementos gráficos interpoláveis, como cores e linhas de iso-valores conforme exemplificado em [92], para representar visualmente os valores escalares dos dados físicos.

É raro encontrar dados unidimensionais em aplicações práticas. Na maioria das vezes, os alvos de busca são caracterizados por uma combinação de atributos, o que é denominado uma **busca conjunta**. Uma questão que muitos pesquisadores procuram responder é se existem combinações que podem ser percebidas de forma pré-atentiva. Atualmente, existem evidências indicando que as seguintes combinações podem se tornar pré-atentivas [98]:

1. agrupamento e cor,
2. profundidade e cor ou movimento,
3. luminância e forma,
4. convexidade e cor, e
5. movimento e forma.

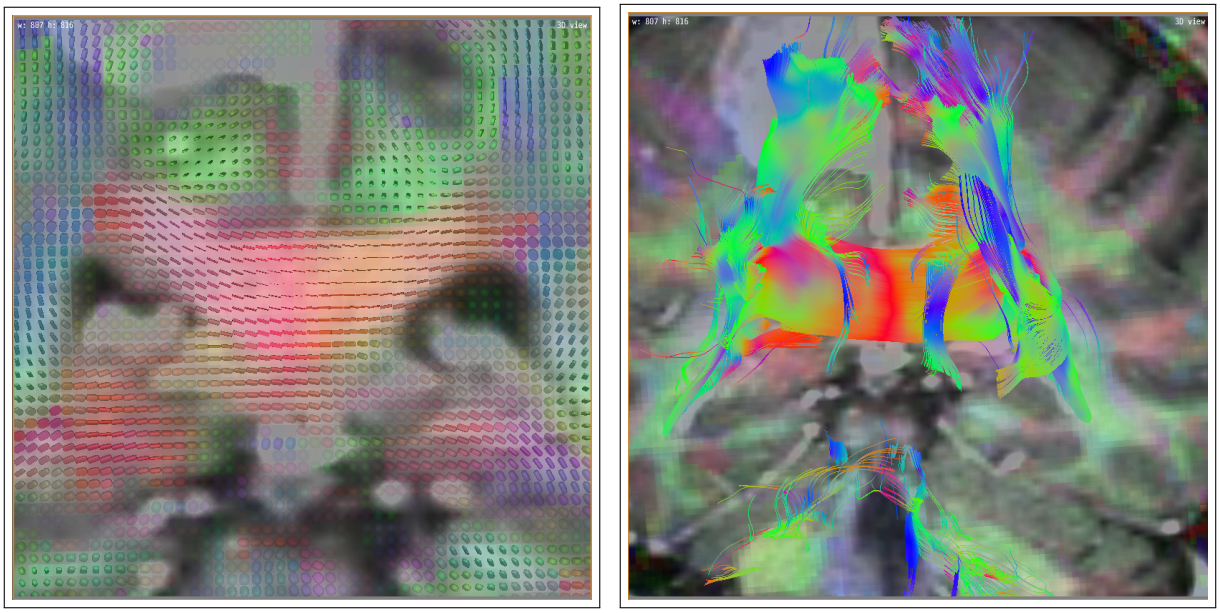
Outra questão explorada pelo Ware é a forma de codificar os valores multidimensionais dos dados em elementos gráficos correspondentes. A prática mais comum adota o uso de elementos discretos conhecidos como **glifos**. Quando as dimensões codificadas nos glifos podem ser visualmente integradas em um único conceito, como a altura e a largura (área) de elipses, torna-se mais fácil perceber padrões do que ao realizar mapeamentos em canais distintos. Para auxiliar no projeto de glifos para uma aplicação específica, Ware apresenta em [98] uma escala de combinações de características visuais para atributos bidimensionais, indo desde 100% integráveis até 100% separáveis. Além disso, ele fornece uma lista de atributos gráficos recomendáveis para o projeto de glifos. Figura 4.3 ilustra o uso de glifos para representar a difusão de água no cérebro, onde as cores vermelho, verde e azul representam, respectivamente, a direção direita-esquerda, anterior-posterior e inferior-superior em relação à posição da cabeça.

4.3 Percepção de Grupos e Correlações

Ao representar dados com atributos multidimensionais por meio de glifos, é possível proporcionar a percepção da continuidade inerente dos dados físicos? Por exemplo, consegue-se discernir a direção dos tratos neurais a partir dos glifos que representam os tensores de difusão de moléculas de água no cérebro, conforme ilustrado na Figura 4.3a? Mais do que simplesmente realçar os dados de interesse, uma análise visual visa descobrir padrões e perceber tendências para aumentar a confiança nas tomadas de decisão. Portanto, é de grande interesse, além dos próprios dados, mapear em atributos gráficos as estruturas subjacentes a esses dados, tornando-as visualmente perceptíveis. No caso da Figura 4.3a, a provável organização direcional das fibras neurais em cada amostra é mapeada no elongamento dos glifos. A expectativa é que o agrupamento consistente desses glifos permita inferir as fibras subjacentes mostradas na Figura 4.3b.

Ao abordar percepções de padrões e grupos, é imprescindível mencionar as **leis de Gestalt**, as quais sintetizam os princípios de percepção de padrões a partir de um conjunto de símbolos gráficos ou glifos:

Proximidade: glifos próximos transmitem a ideia de que os dados que eles representam tem alguma afinidade.



(a) Glifos (b) Fibras

Figura 4.3: Diferentes representações geradas a partir de um mesmo conjunto de dados brutos sobre as fibras neurais: (a) através de amostras (discretas) e (b) através de linhas contínuas.

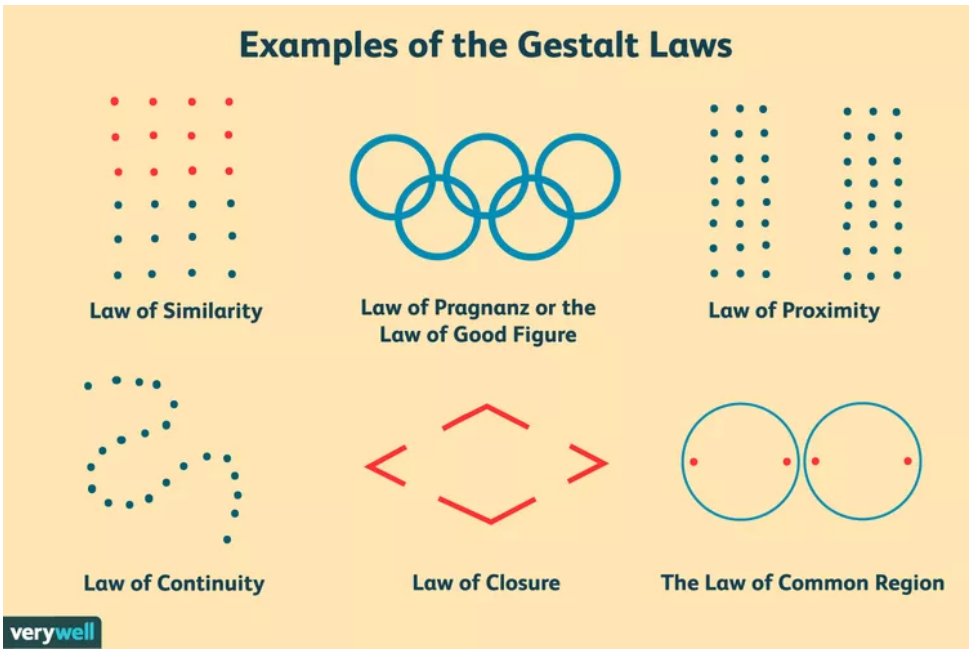


Figura 4.4: Leis de Gestalt: princípios psicológicos que descrevem como os seres humanos percebem e organizam elementos visuais em padrões significativos e estruturados. (Fonte: verywellmind)

Similaridade: glifos similares, processáveis por um mesmo canal visual, geram estímulos pre-atentivamente.

Conectividade: conexões entre os glifos reforçam a ideia de relação entre os dados representados.

Continuidade: formas geométricas suaves, como curvas suaves, proporcionam uma melhor percepção de contiguidade dos dados ou de progressão gradual dos dados repre-

sentados pelos glifos.

Simetria: disposição simétrica dos glifos facilita comparações.

Fechamento: tendência humana de perceber formas completas e fechadas, mesmo quando partes delas estão faltando ou implícitas.

Figura e fundo: organização dos glifos de forma que estes sejam percebidos como figuras e não como o fundo de uma imagem.

Destino comum: glifos que deslocam numa mesma direção são percebidos como um grupo.

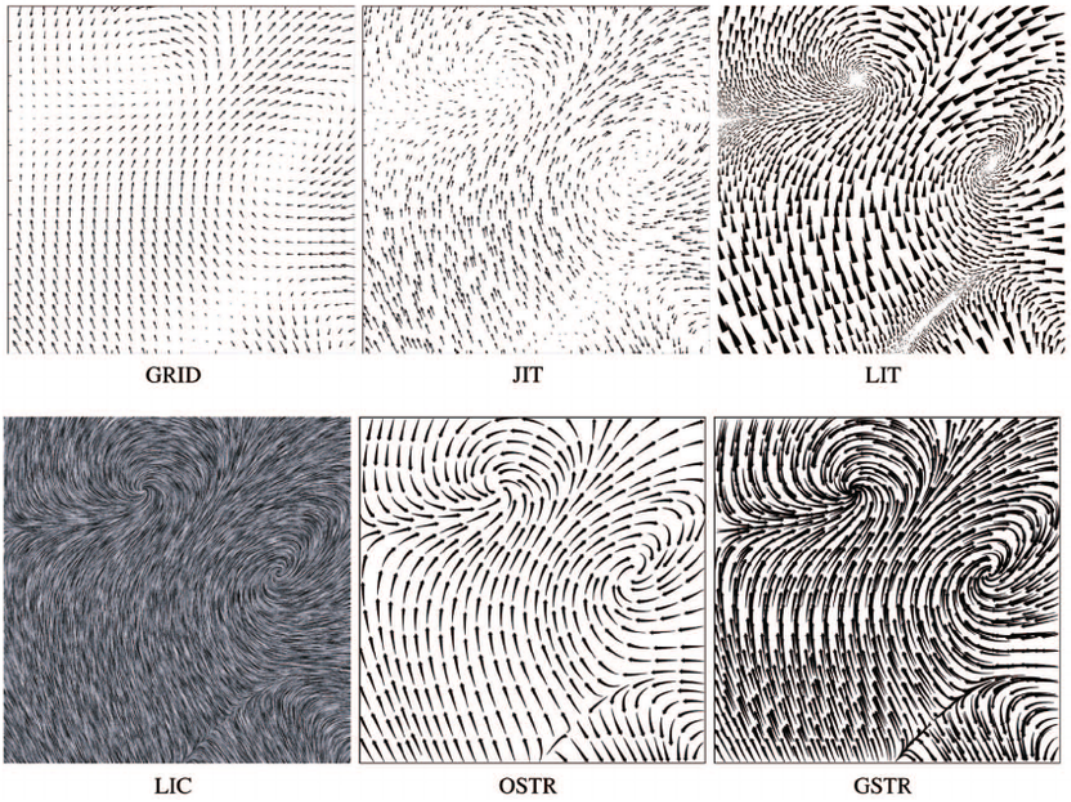


Figura 4.5: Campo vetorial representado por diferentes glifos. (Fonte: Researchgate)

Aplicando esses princípios, Laidlaw et al. usaram diferentes ícones, ilustrados na Figura 4.5), para representar as amostras de um campo (físico) vetorial [48]:

GRID: Ícones de tamanho fixo em uma grade regular.

JIT: Ícones de tamanho fixo com deslocamentos aleatórios sobre uma grade regular.

LIT: Ícones em forma de cunhas triangulares, cujos tamanhos variam com a velocidade do fluxo.

LIC: Imagens com padrões do campo vetorial, que resultam da convolução de uma textura ao longo de linhas de fluxo usando a técnica de convolução de integral de linha (em inglês *Line Integral Convolution*).

OSTR: Linhas de fluxo cujo posicionamento é guiado pela imagem.

GSTR: Linhas de fluxo com pontas semeadas em uma grade regular.

Além disso, Ware mostrou que outros elementos gráficos, como a cor e a forma, podem ser explorados para visualizar simultaneamente a intensidade do campo ao longo das linhas de fluxo [98].

4.4 Psicologia Cognitiva

De acordo com a Wikipedia[19], a **psicologia cognitiva** "estuda a cognição, os processos mentais que estão por detrás do comportamento. É uma das disciplinas da ciência cognitiva. Esta área de investigação cobre diversos domínios, examinando questões sobre a memória, atenção, percepção, representação de conhecimento, raciocínio, criatividade e resolução de problemas. Pode-se definir cognição como a capacidade para armazenar, transformar e aplicar o conhecimento, sendo um amplo leque de processos mentais."

A percepção visual é uma das funções natas que nos permitem captar os estímulos visuais do meio que nos cerca. Ela é intrinsecamente ligada aos princípios e teorias da psicologia cognitiva, proporcionando uma compreensão melhor de como nossa mente organiza e interpreta o mundo visual ao nosso redor. Os estímulos passam por um processamento visual que pode ser dividido, de forma simplista, em três estágios (Figura 3.2) [98]:

Estágio 1: Inicia-se com um processamento altamente paralelo de todos os estímulos, visando extrair atributos básicos, como orientação, cor, textura e padrões de movimento, de uma cena no campo visual. Esses atributos são codificados e retidos brevemente em uma memória sensorial ou icônica.

Estágio 2: Em seguida, ocorre um processamento serial dos dados provenientes da memória sensorial, onde eles são agrupados em regiões ou padrões com base em características como cor, textura e padrões de movimento. Durante este estágio, os padrões identificados são codificados e armazenados nas memórias de trabalho (de curto prazo) por alguns segundos. Dado o limitado espaço de armazenamento na memória de curto prazo, o mecanismo de atenção é acionado para reter apenas os itens de interesse. Experiências repetidas com um mesmo estímulo visual facilitam a transferência da informação da memória de curto prazo para a memória de longo prazo, onde ela pode ser retida por minutos, horas, meses ou até anos.

Estágio 3: Por fim, o processamento direciona-se para tarefas específicas com base nas informações retidas na memória de trabalho pelo mecanismo de atenção e nos conhecimentos recuperados da memória de longo prazo. Este estágio permite uma abordagem mais focada e eficaz nas atividades cognitivas e comportamentais associadas às informações visuais previamente processadas.

Ware mostra que este modelo simplificado de processamento visual oferece uma compreensão clara da distinção entre duas tarefas perceptivas fundamentais: **reconhecimento** (em inglês, *recognition*) e **lembrança** (em inglês, *recall*) [98]. O reconhecimento implica na associação de um evento ou objeto físico a uma informação disponível na memória, sendo uma atividade comparativa que pode ocorrer até de forma inconsciente e imediata. Por outro lado, a lembrança exige a busca de informações na memória de longo prazo, frequentemente sem uma dica explícita para orientar essa busca. Portanto, a solução de um problema não se limita apenas às potencialidades cognitivas individuais, mas também às faculdades cognitivas distribuídas, resultantes das interações do indivíduo com outras pessoas e "ferramentas cognitivas". Este entendimento ampliado de cognição implica que evidências ambientais podem ser úteis na recuperação de conhecimentos da memória de longo prazo, desde que sejam cognitivamente simples, destacando a complexidade e a riqueza das interações cognitivas no processo de reconhecimento e lembrança de um elemento numa interface gráfica.

4.5 Resolução de Problemas

Para criar ou selecionar um sistema eficaz de análise visual de dados que ofereça um suporte adequado à resolução de problemas, é crucial compreendermos como o cérebro aborda o processo para chegar a uma solução. Ter uma compreensão dos mecanismos envolvidos na resolução de um problema nos permite estabelecer diretrizes para os recursos que devem estar presentes no sistema. Em [59], Oz explora, do ponto de vista psicológico, as etapas pelas quais o cérebro passa durante o processo de resolução de problemas, destacando 7 fases distintas:

1. reconhecer ou identificar a existência do problema.
2. definir e representar mentalmente o problema.
3. desenvolver uma estratégia ou um plano de solução.
4. organizar o conhecimento sobre o problema.
5. destinar recursos mentais e físicos à resolução.
6. monitorar o progresso em direção ao objetivo.
7. avaliar a adequabilidade da solução.

A sequência é iterativa, pois ao perceber a falta de progresso na abordagem planejada (item 6) ou a inadequação da solução proposta (item 7), retorna-se ao item 3 para reformular a estratégia de resolução. Em alguns casos, é possível retornar até mesmo aos itens 2 ou 1 para redefinir ou remodelar o problema. A resolução de um problema é verdadeiramente um exercício mental que envolve o desenvolvimento de diversas estratégias cognitivas.

Em geral, quando lidamos com problemas bem condicionados, dotados de modelos analíticos bem estabelecidos como vimos na Seção 1.2, os passos 2 a 7 podem ser sintetizados em procedimentos, protocolos ou algoritmos, facilitando a sistematização e automatização de sua resolução. Contudo, em situações de problemas mal-condicionados, sem precedentes claros também mostrados na Seção 1.2, um **sistema de análise visual de dados** surge como uma ferramenta valiosa para aprofundar a compreensão dos dados em todas as etapas. Ele pode oferecer, como uma extensão da capacidade cognitiva do usuário, suporte na [98]:

1. exploração visual de dados, permitindo a descoberta de padrões e relações não óbvias através de visualizações dinâmicas e interativas.
2. formulação de hipóteses: auxiliando na identificação de tendências e anomalias que podem levar à criação de novas perguntas e hipóteses.
3. validação de hipóteses, fornecendo ferramentas para testar e validar hipóteses através de visualizações comparativas e análises estatísticas.
4. comunicação de resultados, facilitando a criação de visualizações claras e eficazes para comunicar *insights* e descobertas a outros usuários.
5. tomada de decisões, oferecendo suporte visual para a avaliação de cenários e a tomada de decisões informadas.

4.6 Considerações Finais

O avanço digital revolucionou a produção de dados, inundando-nos com uma massa volumosa de dados digitais. Diante desse cenário desafiador, surge a necessidade de extrair *insights* capazes de orientar decisões assertivas. Os sistemas de análise visual de dados se destacam como uma resposta eficaz a esse desafio, explorando a complexa percepção e cognição visual humana para identificar indícios de informações relevantes presentes nessa massa volumosa de dados.

A psicofísica da visão revela como nossos sentidos captam estímulos visuais e os traduzem em percepções conscientes, enquanto os atributos pré-atentivos destacam características visuais que capturam nossa atenção de forma rápida e automática. Por sua vez, as leis de Gestalt nos ensinam como percebemos padrões, formas e relações entre elementos visuais, fundamentando a compreensão de como organizamos informações de maneira significativa e coerente.

Além disso, ao explorarmos os princípios da psicologia cognitiva, entendemos como processamos, armazenamos e recuperamos informações, e como nossas percepções e experiências moldam nossa compreensão do mundo ao nosso redor. Esses conhecimentos são essenciais para o *design* eficaz de visualizações de dados, pois permitem-nos criar representações visuais que se alinham com a maneira como o cérebro humano processa informações, facilitando a compreensão e análise de conjuntos de dados complexos.

Ao integrar os princípios da percepção visual humana com os princípios de *design* de interfaces gráficas para visualização de dados complexos (Capítulo 3), desenvolvedores podem criar representações visuais intuitivas e mecanismos de interação amigáveis. Essas representações e interações facilitam a exploração de dados, permitindo a identificação eficaz de padrões, tendências e anomalias.

4.7 Exercícios

1. Da coleção de ilusões ópticas do Museu Americano de História Natural [6], identifique duas ilusões ópticas que você considera mais promissoras em termos de potencial para aprimorar a visualização de dados. Justifique sua escolha.

2. Nas Seções 9.1 a 9.12 em [71], o estatístico Rafael A. Irizarry discute problemas na interpretação de dados causados por gráficos mal projetados, alguns dos quais gerados com as bibliotecas ggplot2 (R). Discuta, quando pertinentes, soluções em R para contornar tais problemas e apresente as suas equivalentes em Python.

3. Na Seção 10.9 em [71], o estatístico Rafael A. Irizarry explora, sob a perspectiva dos princípios de percepção e cognição, diferentes abordagens para visualizar os dados relacionados aos casos de sarampo nos estados dos Estados Unidos, utilizando o conjunto de dados us_contagious_diseases [96]. O objetivo é demonstrar a efetividade das vacinas na erradicação do sarampo. Porte as alternativas em R, apresentadas por Irizarry, para Python. Faça uma análise crítica dessas alternativas, considerando a possibilidade de outras abordagens mais eficientes e assertivas para alcançar o objetivo proposto.

Capítulo 5

Importação de Dados

A importação de dados é uma etapa fundamental na Análise de Dados, sendo essencial no processo de *data munging* e *data wrangling*. Estes termos se referem ao ciclo de preparação de dados para análise, começando pela importação e seguindo para a limpeza, transformação e enriquecimento dos dados. ***Data wrangling*** é o termo formal que abrange todo o processo, desde a importação de dados de diversas fontes até as etapas de transformação e integração necessárias para a análise. Em contraste, ***data munging*** é uma abordagem mais prática e coloquial, focada especialmente na limpeza e preparação dos dados, corrigindo erros e tratando valores ausentes. A importação de dados é a etapa inicial, estabelecendo a base para todo o processo de *wrangling* e *munging*.

No contexto de *data munging* e *data wrangling*, a importação de dados é crucial para resolver problemas de estruturação dos dados que podem afetar a análise. A importação pode enfrentar desafios inerentes ao integrar dados de diferentes fontes e formatos em um ambiente onde possam ser facilmente manipulados e analisados. Isso frequentemente envolve a integração de dados de arquivos estáticos, que contêm dados fixos, ou seja, dados que não mudam a menos que o arquivo seja atualizado manualmente, dados de bancos de dados, dados obtidos através de APIs (do inglês *Application Programming Interfaces*) de diferentes repositórios ou serviços, e de outras fontes, cada uma com seus próprios formatos e níveis de granularidade.

Garantir que todos os dados estejam corretamente alinhados e sincronizados pode ser desafiador, especialmente com problemas como perda de precisão durante a conversão de formatos, incompatibilidades de esquema, inconsistências de dados e problemas de codificação. Para mitigar essas dificuldades e simplificar a análise, adotar um formato padrão, como o formato "tidy", é altamente recomendável. Amplamente utilizado nas linguagens R e Python, o formato "tidy" organiza os dados em uma estrutura tabular onde cada variável é uma coluna, cada observação é uma linha e cada tipo de unidade é uma tabela. Essa abordagem facilita a implementação de procedimentos unificados para o processamento e visalização de dados, promovendo uma integração e um processamento encadeado (em inglês *pipeline*) mais eficientes.

No entanto, mesmo um formato organizado não garante que os dados estejam completamente limpos ou isentos de problemas. Os dados importados podem conter erros, lacunas e duplicatas. O analista deve garantir a integridade e a qualidade dos dados, além de assegurar que estejam em um formato coerente, completo e estruturado. Isso pode exigir a aplicação de técnicas de limpeza e pré-processamento, como normalização de valores, tratamento de valores ausentes e conversão de tipos de dados, conforme explorado no Capítulo 7.

Neste capítulo, exploraremos a importação de dados para um formato tabular, com o objetivo de alcançar uma estruturação unificada que facilite a conversão para o formato "tidy". Na Seção 5.1, discutiremos o conceito de *tidy datasets*, que são conjuntos de dados organizados em um formato tabular com variáveis como colunas, observações como linhas e tipos de unidade como tabelas. A Seção 5.2 explorará as diversas fontes de dados, como arquivos textuais, como CSV (do inglês *Comma-Separated Velues*, TSV (do inglês *Tab-Separated Velues* e TXT (do inglês *Text*), bancos de dados relacionais, APIs dos servidores de dados e técnicas de *web scraping*. Veremos na Seção 5.3 a transformação dos dados em formatos tabulares para garantir consistência e interoperabilidade entre diferentes fontes. Além disso, abordaremos na Seção 5.4 métodos de validação dos dados importados para assegurar que sejam precisos e úteis para a análise, destacando a importância de uma abordagem sistemática e rigorosa na importação de dados para a construção de modelos e *insights* confiáveis. Por fim, na Seção 5.5, apresentaremos ferramentas para a automação do processo de importação de dados.

5.1 Conjunto de Dados Organizados (*Tidy Data*)

Os conjuntos de dados organizados, também conhecidos como *tidy datasets*, representam uma abordagem formal e sistemática para organizar dados, o que facilita sua manipulação, análise e visualização. Esses conjuntos seguem um formato padronizado que alinha a estrutura física dos dados (seu *layout*) com seu significado e contexto (sua semântica). Em termos de *layout*, a maioria dos dados estatísticos é organizada em tabelas, frequentemente chamadas de *data frames* em linguagens estatísticas e de análise de dados. Os *data frames* são constituídos por linhas e colunas, onde as colunas são quase sempre rotuladas para indicar as variáveis ou atributos, e as linhas, embora nem sempre rotuladas, representam observações ou registros específicos.

Do ponto de vista **semântico**, um conjunto de dados é composto por **valores**, que podem ser numéricos (**quantitativos**) ou textuais (**qualitativos** ou **categóricos**), organizados em torno de variáveis e observações. **Variáveis** definem os atributos que caracterizam uma observação sob diferentes aspectos, como idade e sexo, enquanto **observações** contêm o conjunto completo de valores coletados para uma unidade de análise específica, por exemplo, um indivíduo. A distinção entre variáveis e observações pode parecer simples, mas é altamente dependente do contexto. Por exemplo, os atributos "distância" e "tempo" podem ser tratados como variáveis distintas em uma observação

denominada "Viagem" ou combinados em uma única variável como "Velocidade Média". Neste contexto, cada "Viagem" individual representa uma observação distinta, com seus próprios valores de distância, tempo e, consequentemente, velocidade média.

Além disso, em análises mais complexas, podem existir múltiplos **níveis de observação**, exigindo diferentes *data frames* para representar esses níveis. Por exemplo, em um estudo sobre o desempenho acadêmico, pode-se ter dados individuais dos alunos, informações sobre as turmas e dados de desempenho acadêmico separados em 3 níveis de observações. As variáveis também podem evoluir ao longo da análise: inicialmente, podem corresponder a perguntas específicas, mas, à medida que a análise avança, podem ser combinadas ou refinadas para simplificar o modelo e focar em características específicas, facilitando a interpretação e evitando complexidade desnecessária.

Conforme definido por Hadley Wickham [104], um dos principais defensores dessa abordagem, **conjuntos de dados organizados** representam uma **maneira padronizada** de estruturar dados, garantindo que seu significado seja claramente mapeado na estrutura física dos dados, com base em três princípios fundamentais:

1. Cada variável é representada por uma coluna.

2. Cada observação é representada por uma linha.

3. Cada tipo de unidade observacional é representada por uma tabela.

Embora o conceito de "*tidy data*" se assemelhe à terceira forma normal[1] proposta por Edgar F. Codd [16], "*tidy data*" é adaptado para a análise estatística e Ciência de Dados, focando na organização de dados dentro de uma única estrutura física. Enquanto a terceira forma normal se concentra na eliminação de redundâncias e na integridade de dados em bancos de dados relacionais, o formato *tidy* se concentra na organização clara e uniforme dos dados em uma única tabela, facilitando o acesso e a utilização. Dados que não seguem esses princípios são considerados desorganizados (em inglês, *messy data* ou *untidy data*), como ilustrado na Figura 5.1.

Embora *tidy datasets* sejam projetados para representar dados em um formato tabular simples, eles podem ser usados em conjunto com outras técnicas para lidar com dados hierárquicos ou complexos. Nesses casos, transformações e junções podem ser usadas para "aplanar" os dados e organizá-los em um formato "tidy" para análise. Segundo Hadley Wickham, a organização inicial pode demandar esforço, mas a padronização do *layout* facilita a extração de valores e operações analíticas, otimizando o uso de ferramentas como ggplot2 (R) [105] e plotnine (Python) [4]. Essa eficiência reduz o tempo de manipulação, focando a análise. Em [104], Wickham observa que conjuntos de dados reais frequentemente não aderem aos três princípios fundamentais de dados "*tidy*", violando-os de diversas formas. Embora seja possível encontrar conjuntos de dados que possam ser analisados imediatamente, isso é

[1] A terceira forma normal é uma etapa de normalização em modelos de dados relacionais, projetada para eliminar redundâncias e garantir a integridade dos dados.

	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

Table 1: Typical presentation dataset.

	John Smith	Jane Doe	Mary Johnson
treatmenta	—	16	3
treatmentb	2	11	1

(a) Dados Desorganizados

person	treatment	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

(b) Dados Organizados

Figura 5.1: Dados estruturados em formato tabular: (a) Dados desorganizados (em inglês, *untidy*), porque as colunas correspondem a diferentes valores de um variável, à variável "treatment" na Tabela 1 e à variável "person" na Tabela 2; (b) Dados organizados (em inglês, *tidy*), porque cada linha corresponde a uma observação e cada coluna a uma variável. (Fonte: [104])

uma exceção, não a regra. Ele destaca cinco problemas mais comuns encontrados em conjuntos de dados desorganizados ou "untidy":

1. **Cabeçalhos das Colunas como Valores, Não Nomes de Variáveis**: Às vezes, os valores das células de uma coluna são usados como cabeçalhos, ao invés de nomes claros para as variáveis. Isso pode dificultar a identificação e a análise dos dados.

2. **Várias Variáveis Armazenadas em Uma Única Coluna**: Quando múltiplas variáveis são combinadas em uma só coluna, a análise e a manipulação dos dados se tornam complexas. Cada variável deve ter sua própria coluna.

3. **Variáveis Armazenadas em Linhas e Colunas**: Em alguns conjuntos de dados, as variáveis são distribuídas entre linhas e colunas de forma confusa. Isso quebra a estrutura tabular que facilita a análise.

4. **Mistura de Tipos de Unidades Observacionais na Mesma Tabela**: Quando diferentes tipos de unidades observacionais (por exemplo, pessoas e produtos) são armazenados na mesma tabela, isso complica a análise e a interpretação dos dados.

5. **Uma Única Unidade Observacional Armazenada em Várias Tabelas**: Se uma unidade observacional é dividida entre várias tabelas, isso pode criar desafios na integração e análise dos dados.

Veremos no Capítulo 7 que a maioria dos conjuntos de dados desorganizados, incluindo aqueles que apresentam problemas não descritos, pode ser reorganizada com algumas ferramentas simples antes do processamento, como rotação (para transformar dados de forma mais longa em mais ampla, ou vice-versa) e separação (para dividir dados em colunas ou tabelas apropriadas).

5.2 Fontes de Dados

A etapa inicial da importação de dados envolve a identificação da fonte, que pode incluir arquivos locais, bancos de dados externos, APIs de repositórios ou servidores, ou raspagem de *websites*. A meta é integrar dados de diversas origens em um formato "*tidy*", criando um conjunto coeso e organizado. Cada fonte demanda técnicas específicas para assegurar a transferência correta dos dados para o ambiente de análise, dada a variedade de estruturas que podem apresentar, como listas simples, formatos relacionais, hierárquicos ou aninhados [101].

Além disso, os dados podem estar em diversos formatos textuais, como CSV (do inglês *Comma-Separated Values*), JSON (do inglês *JavaScript Object Notation*), XML (do inglês *eXtensible Markup Language*) e Excel (planilha da Microsoft), cada um com suas peculiaridades e requisitos estruturais. Várias codificações são usadas para representação e armazenamento de valores. Códigos Base64 e Hexadecimal (ou Base16) são métodos de codificação que transformam dados binários em representações textuais, facilitando sua inclusão em formatos textuais como JSON e XML. UTF-8 é amplamente utilizado para garantir a compatibilidade com caracteres globais, enquanto técnicas de compressão, como ZIP e GZIP, reduzem o tamanho dos dados para armazenamento e transmissão. Cada codificação e técnica de compressão atende a necessidades específicas de desempenho, espaço e compatibilidade.

Na prática, a importação de dados nos IDEs, como RStudio ou Jupyter Notebook, envolve transferir conjuntos de dados de diversas fontes para estruturas tabulares, tipicamente data.frames/tibbles em R e DataFrames em Python. Ambas as linguagens oferecem funções para importar e transformar dados estruturados em dados organizados, ou "*tidy*". Essas funções geralmente detectam a codificação e o formato dos arquivos automaticamente, mas permitem ajustes manuais através de parâmetros das função e opções de configuração. Em R, parâmetros como fileEncoding ou encoding ajudam na especificação da codificação, enquanto em Python, parâmetros como encoding e compression são usados para lidar, respectivamente, com a codificação dos arquivos e a leitura de arquivos comprimidos. Pacotes adicionais podem aprimorar a detecção e a manipulação de codificações; por exemplo, a biblioteca chardet em Python oferece detecção avançada de codificação, e a função stri_enc_detect() do pacote stringi em R é uma ferramenta útil para detectar codificações. Quando a detecção automática não é precisa, a leitura com diferentes codificações e o uso da função iconv() em R podem ajudar a lidar com problemas de codificação, garantindo que os dados sejam importados corretamente.

5.2.1 Arquivos Estáticos de Dados Tabulares

Arquivos estáticos de dados tabulares são arquivos que armazenam dados em um formato estruturado e tabular, onde a organização dos dados segue um esquema de linhas e colunas, semelhante a uma planilha ou tabela. Esses arquivos são chamados de **estáticos** porque seu conteúdo não muda automaticamente e precisa ser atualizado manualmente, se necessário. Eles são frequentemente usados

para armazenar e compartilhar conjuntos de dados que podem ser facilmente lidos e processados por diversos programas e ferramentas.

A importação desses arquivos exige atenção cuidadosa a vários aspectos para garantir a integridade e a eficiência do processo. Arquivos, como CSV, TSV, Excel e TXT, representam uma das fontes mais comuns de dados e, embora possam parecer simples, possuem nuances importantes que os desenvolvedores devem considerar. É crucial verificar a **codificação do arquivo**, como UTF-8 ou ISO-8859-1, para evitar problemas de leitura e garantir que caracteres especiais sejam interpretados corretamente. Além disso, o formato e a estrutura dos dados precisam ser bem compreendidos: delimitadores em arquivos CSV, planilhas em arquivos Excel e possíveis cabeçalhos em arquivos TXT devem ser identificados e tratados adequadamente.

Para a importação de arquivos estáticos, como CSV, Excel, TSV ou TXT, tanto em R quanto em Python, existem funções especializadas que simplificam a leitura e a adaptação dos dados conforme o formato do arquivo. Essas funções são projetadas para lidar com as peculiaridades de cada tipo de arquivo, facilitando a importação e o processamento eficiente dos dados.

Em R, funções especializadas da biblioteca readr, que faz parte do pacote tidyverse, são amplamente utilizadas para importar dados de diversos formatos de arquivos estáticos. A biblioteca tidyverse lida com dados da classe de objetos chamada tibble, que é uma versão moderna e aprimorada da classe data.frame. A função read_csv() é empregada para ler arquivos CSV, onde os dados são delimitados por vírgulas, enquanto read_excel() do pacote readxl é utilizada para arquivos Excel. Para arquivos TSV, que utilizam tabulações como delimitadores, a função read_tsv() é a escolha apropriada. E arquivos com extensão .txt podem ser lidos com read_delim(), O pacote tidyverse, que inclui essas funções, pode ser instalado e carregado com os seguintes comandos:

```
install.packages ("tidyverse")
library (tidyverse)
```

Entre os parâmetros dessas funções estão o caminho do arquivo a ser importado e, quando necessário, opções para manipulação de cabeçalhos, configuração de delimitador, formatação de linhas de comentários e especificação dos tipos de dados. O exemplo a seguir demonstra a importação de dados do arquivo caminho/do/seu/arquivo/ex1.csv para um tibble denominado df1 no R:

```
library(readr)
r_df1 <- read_csv("caminho/do/seu/arquivo/ex1.csv", col_names = TRUE, comment="#",
 col_types=col(
   indice = col_character(),
   a = col_integer(),
   b = col_integer(),
   c = col_integer(),
```

```
   d = col_integer(),
   palavra = col_character()
 )
)
```

Segue-se o conteúdo de ex1.csv. A primeira linha do arquivo é uma linha de comentários indicada pelo caractere"#", a segunda linha contém os nomes das colunas e as restantes 3 linhas correspondem a 3 observações sobre os dados:

```
#ex1.csv
indice,a,b,c,d,palavra
um,1,,3,4,cachorro
dois,5,6,,8,gato
tres,9,10,11,12,NA
```

Opcionalmente, pode-se usar a função read_delim do pacote readr, que é parte do tidyverse e é projetada para oferecer melhor desempenho e mais funcionalidades na leitura de arquivos delimitados. Devido à sua implementação eficiente em C++, read_delim é particularmente eficaz ao lidar com grandes arquivos contendo dados estruturados e delimitados por colunas. Qunado está trabalhando com grandes conjuntos de dados ou precisa de um controle mais avançado sobre a leitura dos arquivos, read_delim pode ser a melhor escolha devido à sua capacidade de processar dados de maneira rápida e eficiente, além de oferecer opções adicionais para personalizar a leitura dos dados.

```
library(readr)
r_df1 <- read_delim("caminho/do/seu/arquivo/ex1.csv", col_names = TRUE, comment="#",
 col_types=col(
   indice = col_character(),
   a = col_integer(),
   b = col_integer(),
   c = col_integer(),
   d = col_integer(),
   palavra = col_character(),
   delim = ".",
   locale = locale(decimal_mark=",",grouping_mark=".",enncoding='latin1')
 )
)
```

No R, ao usar funções como read_csv() e read_tsv() do pacote readr, não há um parâmetro direto para especificar a codificação do arquivo. Portanto, é necessário ajustar a codificação do arquivo

antes da leitura. Para isso, pode-se utilizar a função readLines() com o parâmetro encoding para ler o conteúdo do arquivo com a codificação desejada e, em seguida, passar esse conteúdo para a função de importação apropriada. O trecho de código a seguir ilustra como realizar esse processo usando readLines:

```
library(readr)

# Primeiro, leia o arquivo com a codificação especificada usando readLines()
lines <- readLines("caminho/do/seu/arquivo/ex1.csv", encoding = "ISO-8859-1")

# Em seguida, combine as linhas em uma string única
lines_combined <- paste(lines, collapse = "\n")

# Use read_csv() para ler os dados a partir da string
r_df1 <- read_csv(lines_combined, col_names = TRUE, comment="#",
 col_types=col(
   indice = col_character(),
   a = col_integer(),
   b = col_integer(),
   c = col_integer(),
   d = col_integer(),
   palavra = col_character()
 )
)
```

A função read_excel() do pacote readxl não possui um parâmetro para codificação, pois o pacote lida com a codificação internamente, assumindo que o arquivo Excel está codificado corretamente para leitura. Portanto, é necessário garantir que o arquivo Excel esteja salvo no formato padrão para evitar problemas de codificação. Caso encontre problemas de codificação com arquivos Excel, pode ser necessário verificar a origem dos dados ou usar um aplicativo externo para ajustar a codificação.

O pacote readr também oferece funções, como write_csv() e write_tsv(), para exportar dados de tibble para arquivos em formato CSV e TSV, respectivamente. O readr não possui parâmetros diretos para especificar a codificação durante a exportação. Para configurar manualmente a codificação, deve-se ajustar o conteúdo após a exportação utilizando readLines(), ou read_lines() do pacote readr, e iconv() e então salvar o arquivo com a codificação desejada. A exportação de dados para arquivos no formato Excel pode ser feita com o pacote writexl. Para mais detalhes sobre importação e exportação de dados em R, incluindo outras funções e formatos, consulte [88].

Em Python, a importação e manipulação de dados tabulares a partir de diversos formatos são frequentemente realizadas com o pacote pandas, que é comumente importado no ambiente de análise usando o alias pd. Este pacote oferece funções para trabalhar com dados em formatos como CSV, TSV, Excel e TXT. Para importar dados de arquivos de texto no formato CSV e TSV, pode-se usar as funções pd.read_csv() e pd.read_table(), respectivamente. No entanto, pd.read_csv() é mais versátil e pode ser configurado para lidar com diferentes tipos de arquivos de texto ao ajustar o delimitador, como delimiter='\t' para TSV e delimiter=';' para arquivos com delimitadores personalizados. A função pd.read_excel() é utilizada para arquivos Excel, e é necessário ter a biblioteca openpyxl instalada para ler arquivos no formato .xlsx, além da biblioteca xlrd para arquivos no formato .xls.

A configuração de codificação é importante, especialmente se os dados contiverem caracteres especiais. UTF-8 é uma escolha comum, mas pode-se ajustar para outros formatos, como "latin1" ou "iso-8859-1", usando o parâmetro enconding da função. Apenas a função pd.read_excel() não possui um parâmetro de codificação porque os arquivos Excel possuem uma estrutura binária e não texto simples. A instalação e importação do pacote pandas podem ser realizadas com os seguintes comandos no Jupyter Notebook:

```
!pip3 install pandas
import pandas as pd
```

Observe que o ponto de exclamação "!" antes do comando "pip3 install pandas" é uma sintaxe específica do Jupyter Notebook. Ele indica que o comando deve ser executado diretamente no sistema operacional, em vez de dentro do ambiente Python. Isso permite que se execute comandos de *shell*, como a instalação de pacotes, diretamente a partir de uma célula do Jupyter Notebook.

Diferentemente do R, onde os métodos de exportação e importação são frequentemente funções independentes, em Python, com a biblioteca pandas, os métodos de exportação e importação estão associados aos objetos de dados. Por exemplo, o método to_csv() da classe DataFrame é utilizado para exportar dados de um DataFrame para um arquivo no formato CSV. Esses métodos oferecem uma ampla gama de parâmetros de configuração, incluindo delimitadores, formatos de dados e codificação. A seguir estão as linhas de comando que ilustram a importação de dados a partir do arquivo caminho/do/seu/arquivo/ex1.csv e a exportação para outro arquivo caminho/do/seu/arquivo/resultado.csv, excluindo os índices das linhas:

```
py_df1 = pd.read_csv('caminho/do/seu/arquivo/dados.csv', header = 0, comment = "#")
py_resultado = py_df1.to_csv('caminho/do/seu/arquivo/resultado.csv', index = False)
```

Para obter uma compreensão completa sobre o processamento de entrada e saída de dados provenientes de diversas fontes em Python, consulte [87].

5.2.2 Banco de Dados

R e Python oferecem suporte para a importação de dados de bancos de dados relacionais, com pacotes específicos que simplificam a conexão e facilitam a manipulação dos dados utilizando SQL (do inglês *Structured Query Language*). Para conectar-se a bancos de dados relacionais como MySQL, PostgreSQL ou SQLite, diferentes bibliotecas são utilizadas em R e Python. Em R, bibliotecas como DBI e RMySQL facilitam a conexão e a execução de consultas SQL, permitindo a transferência dos resultados diretamente para objetos do tipo data.frame ou tibble. Em Python, bibliotecas como **SQLAlchemy** e pymysql desempenham um papel semelhante, possibilitando a execução de consultas SQL e a importação dos dados para DataFrames do pacote pandas. Essas ferramentas são essenciais para integrar e manipular dados de bancos de dados relacionais em ambos os ambientes.

No R, as instruções básicas para interagir com um banco de dados MySQL são as seguintes:

```
install.packages ("DBI")
   #instalar uma interface comum para interação com banco de dados relacionais
library (DBI)  # carregar a funções definidas no pacote DBI
install.packages ("RMySQL") #instalar o pacote de driver especifico do MySQL
library (RMySQL) #carregar o driver do MySQL
conn <- dbConnect(RMySQL::MySQL(),
                  dbname = <nome_do_banco>,
                  host = <localhost>,
                  user = <seu_usuario>,
                  password = <sua_senha>) #conn é o objeto de conexão com MySQL
```

Para executar uma consulta ou exibir os resultados da consulta, utiliza-se a função dbGetQuery. Os argumentos necessários incluem o objeto de conexão com MySQL, neste caso conn, e a consulta query em SQL, fornecida como uma *string*. O tipo de dado retornado é um data.frame ou um tibble dependendo da biblioteca usada para consultas:

```
query <- "SELECT * FROM nome_da_tabela" # String de consulta SQL
data_frame <- dbGetQuery(conn, query)
     # Consulta ao banco de dados conectado conn
```

Detalhes são fornecidos no Capítulo 11 em [101].

Em Python, as instruções básicas são:

```
!pip3 install sqlalchemy    # instala um pacote de conexão com bancos de dados relacionais
import sqlalchemy as create-engine  # carrega a biblioteca sqlalchemy
engine = create_engine("mysql://usuario:senha@localhost/nome_do_banco")
       #conecta com um banco SQL
```

Para executar uma consulta em SQL e armazenar os resultados do tipo DataFrame, usa-se as funções de manipulação de dados da biblioteca pandas:

```
import pandas as pd
query = "SELECT * FROM nome_da_tabela"    # String de consulta SQL
data_frame = pd.read_sql(query, engine)   # Consulta ao banco de dados conectado denominado en
```

Note que SQLAlchemy é uma biblioteca abrangente e flexível que fornece uma abstração de alto nível para a interação com bancos de dados relacionais em Python, como SQLite, MySQL e PostgreSQL, entre outros. Ela oferece uma API de programação consistente que facilita o trabalho com diversos sistemas de gerenciamento de banco de dados (SGBDs) e inclui recursos avançados, como o mapeamento objeto-relacional (ORM), que permite representar tabelas de banco de dados como objetos Python. Para interagir com bancos de dados instalados localmente, existem pacotes específicos, como sqlite3 para SQLite, mysql.connector para MySQL e psycopg2 para PostgreSQL. Para uma compreensão mais detalhada, consulte a referência em [22].

A codificação de caracteres é gerenciada em dois níveis: no banco de dados e na aplicação que se conecta a ele. No banco de dados, é necessário definir o *charset*, que especifica a codificação de caracteres utilizada para armazenar texto, e a *collation*, que define as regras de comparação e ordenação de texto. Essas configurações garantem que os dados sejam armazenados e manipulados corretamente. Na aplicação, deve-se configurar a codificação da conexão para que corresponda à configuração do banco de dados. Além disso, é importante que as operações de leitura e escrita na aplicação estejam alinhadas com a codificação definida no banco de dados. Uma configuração adequada em ambos os níveis ajuda a evitar problemas com caracteres especiais e assegura a integridade dos dados.

5.2.3 APIs *Web*

O acesso e a troca de informações são otimizados por várias tecnologias que conectam sistemas e aplicações de maneira eficiente. Entre essas tecnologias, as APIs *Web* se destacam por possibilitar a interação e o compartilhamento estruturado de dados entre *softwares*. Essas APIs oferecem uma interface padronizada, permitindo que desenvolvedores acessem funcionalidades e dados de serviços externos por meio de requisições HTTP e processar os resultados recebidos, sem a necessidade de compreender a complexidade interna desses sistemas. Os dados fornecidos frequentemente estão em formatos estruturados e aninhados, como JSON ou XML, e podem abranger uma ampla gama de informações, desde dados financeiros e estatísticos até detalhes sobre produtos e usuários.

Para gerenciar e disponibilizar essas informações, servidores dedicados são utilizados para hospedar APIs *Web* e processar as requisições. Esses servidores garantem que os dados estejam disponíveis em tempo real e podem implementar autenticação e autorização para proteger informações sensíveis. A arquitetura dos servidores de APIs *Web* promove uma comunicação contínua entre diferentes sistemas,

facilitando a integração e o uso de dados em diversos contextos e aplicações.

As APIs *Web* definem o formato dos dados por meio de cabeçalhos HTTP. O cabeçalho Content-Type indica o tipo de mídia do recurso retornado, como text/html, application/json, ou text/xml, e pode incluir a codificação de caracteres (charset). O cabeçalho Content-Encoding especifica o método de compactação aplicado ao corpo da resposta, como gzip ou deflate, enquanto o cabeçalho Accept informa ao servidor quais formatos de resposta o cliente pode processar e prefere receber. A documentação da API geralmente descreve os formatos suportados e como utilizá-los corretamente.

Para conversão automática entre formatos, as APIs utilizam *parsers* e *serializers*. ***Parsers*** traduzem dados de formatos externos para estruturas internas, enquanto ***serializers*** convertem estruturas internas de volta para formatos externos. Algumas APIs oferecem *endpoints*[2] distintos para diferentes formatos e muitas seguem padrões específicos para garantir interoperabilidade e consistência.

Ao contrário dos dados estáticos de arquivos locais, como CSV, Excel ou JSON, que exigem carregamento direto do sistema de arquivos e atualizações manuais, os dados acessados por APIs oferecem **atualização dinâmica**, em tempo real ou em intervalos definidos. Esse **acesso programático** a informações atualizadas elimina a necessidade de *downloads* e reprocessamentos frequentes de arquivos.

Os bancos de dados oferecem acesso a dados estruturados que podem ser consultados e manipulados usando SQL ou outras linguagens de consulta. A interação com bancos de dados envolve uma conexão direta e a execução de consultas, enquanto APIs *Web* **abstraem esses detalhes de conexão** e fornecem dados em formatos específicos como JSON ou XML. Esses formatos estruturados facilitam a interpretação e o processamento por *software*, diferenciando-se da manipulação de dados estáticos ou de interfaces de usuário de bancos de dados.

É importante notar que muitas APIs *Web* exigem **autenticação** para garantir que apenas usuários autorizados possam acessar ou manipular os dados, o que pode incluir o uso de *tokens* de API, chaves de acesso ou outros métodos de autenticação. Essa necessidade de autenticação é uma característica distintiva das APIs em comparação com a manipulação de dados em arquivos locais ou bancos de dados, que podem ter diferentes formas de controle de acesso.

Arquivos locais: O controle de acesso pode ser feito através de permissões do sistema operacional, como permissões de leitura, gravação e execução, ou por meio de criptografia de arquivos.

Bancos de dados: O controle de acesso pode ser implementado através de sistemas de gerenciamento de bancos de dados (SGBDs), que permitem definir usuários, grupos e permissões de acesso a tabelas, visões e outros objetos do banco de dados. Além disso, a segurança pode ser reforçada com criptografia de dados em repouso e em trânsito.

Em R, pacotes como httr e jsonlite simplificam a realização de requições HTTP, enquanto em

[2] *Endpoint* é geralmente uma URL (do inglês *Uniform Resource Locator*) que inclui o endereço-base da API seguido de um caminho que especifica o recurso desejado.

Python, bibliotecas como requests e json desempenham funções semelhantes, facilitando a obtenção e o processamento de dados. A seguir, apresentamos um exemplo prático de como acessar a OpenWeatherMap API, uma interface que fornece dados meteorológicos, incluindo temperatura, condições climáticas, umidade e previsões do tempo para uma localização específica. Pode-se fazer uma requisição HTTP para um *endpoint* da API, como `https://api.openweathermap.org/data/2.5/weather?q=London&appid=YOUR_API_KEY`. A resposta vem em formato JSON contendo informações meteorológicas de temperatura atual (temp), condições do tempo (*weather*), e umidade (*humidity*):

```
{
  "weather": [
    {
      "description": "clear sky",
      "icon": "01d"
    }
  ],
  "main": {
    "temp": 293.25,
    "humidity": 53
  },
  "name": "London"
}
```

Para realizar uma requisição à OpenWeatherMap API, inspecionar a codificação e o formato dos dados retornados, processar a resposta JSON e exibir os resultados em R, execute as instruções a seguir. Valide a chave textsfYOUR_API_KEY antes de prosseguir. O tipo de objeto retornado pela função fromJSON() varia conforme a estrutura do JSON processado, podendo ser list, data.frame e vector. Para dados aninhados, como informações meteorológicas, fromJSON() geralmente retorna list.

```
library(httr)
library(jsonlite)

# Definir a chave da API e a cidade
api_key <- "YOUR_API_KEY"
city <- "London"

# Construir a string de URL
url <- paste0("https://api.openweathermap.org/data/2.5/weather?q=", city, "&appid=", api_key)
```

```
# Fazer a requisição GET
response <- GET(url)

# Verificar o status da requisição
if (status_code(response) == 200) {
  # Verificar os cabeçalhos da resposta
  content_type <- headers(response)$`content-type`
  content_encoding <- headers(response)$`content-encoding`

  # Exibir informações sobre os cabeçalhos
  print(paste("Content-Type:", content_type))
  print(paste("Content-Encoding:", content_encoding))

  # Processar a resposta JSON
  data <- content(response, as = "text")
  data_parsed <- fromJSON(data)

  # Exibir os dados
  print(data_parsed)
} else {
  print(paste("Erro na requisição. Código de status:", status_code(response)))
}
```

A seguir, apresentamos uma versão equivalente em Python. Vale relembrar que em Python os métodos são associados aos objetos, enquanto em R os métodos são genéricos e operam sobre os objetos como parâmetros. A função response.json() retornará um dicionário (dict) se o JSON retornado pela API estiver estruturado como um objeto JSON. Caso o JSON seja uma lista, o resultado será um um objeto da classe list.

```
import requests
import json

# Definir a chave da API e a cidade
api_key = "YOUR_API_KEY"
city = "London"

# Construir a string de URL
```

```
url = 'https://api.openweathermap.org/data/2.5/weather?q={}&appid={}'.format(city, api_key)

# Fazer a requisição GET
response = requests.get(url)

# Verificar o status da requisição
if response.status_code == 200:
    # Verificar os cabeçalhos da resposta
    content_type = response.headers.get('Content-Type')
    content_encoding = response.headers.get('Content-Encoding')

    # Exibir informações sobre os cabeçalhos
    print("Content-Type:", content_type)
    print("Content-Encoding:", content_encoding)
    # Processar a resposta JSON
    data = response.json()

    # Exibir os dados numa estrutura aninhada para facilitar a leitura
    print(json.dumps(data, indent=4))
else:
    print("Erro na requisição. Código de status:", response.status_code)
```

5.2.4 *Web Scraping*

A *internet* é um repositório de dados provenientes de diversas fontes. A **raspagem de dados** (em inglês *web scraping* ou *web harvesting*) se refere ao processo de extração de dados de *sites* que não oferecem APIs ou quando informações específicas não estão disponíveis via APIs. Diferente das APIs, que fornecem dados estruturados, atualizados dinamicamente e acessíveis em tempo real, o *web scraping* exige atenção a aspectos como qualidade dos dados, conformidade legal e impacto sobre os *sites*, uma vez que a extração pode não ser sempre clara ou controlada, e deve respeitar as políticas dos *sites* para evitar sobrecarregá-los.

O *web scraping* é viável porque as informações que um navegador usa para renderizar uma página *web* são enviadas como um arquivo de texto em HTML (do inglês *Hypertext Markup Language*), uma linguagem de marcação usada para estruturar páginas *web*. Cada navegador pode exibir o código-fonte HTML de forma diferente; por exemplo, no Chrome, pode-se visualizar o código-fonte pressionando Control+U em um PC ou Command+Option+U em um Mac [70, 102].

O HTML básico ofereça uma estrutura simples e pouco atraente. O *design* visual moderno das

páginas *web* é aprimorado com CSS (do inglês *Cascading Style Sheets*). O CSS define a aparência e o estilo das páginas, aplicando um mesmo arquivo CSS a várias páginas para garantir consistência. Ele especifica o estilo de elementos como títulos, cabeçalhos, listas, tabelas e *links*, definindo propriedades como fonte, cor, tamanho e margens através de seletores. O uso de CSS pode, por outro lado, complicar o *web scraping* ao ocultar ou estilizar elementos de formas que dificultam a sua localização e extração. Páginas com *layouts* complexos e conteúdo dinâmico podem ter estruturas HTML difíceis de interpretar sem considerar o impacto do CSS. Além disso, muitos *sites* utilizam JavaScript em conjunto com CSS para manipular o conteúdo.

Para superar esses desafios, é útil analisar diretamente o HTML para entender a estrutura dos elementos, sem depender da renderização visual. Simular interações de usuário, como passar o *mouse* sobre elementos ou clicar em abas, pode ser necessário para acessar dados que dependem de ações específicas. Ferramentas que automatizam ações em navegadores, realizam testes, capturam *screenshots*, geram PDFs e fazem *scraping* de dados podem renderizar páginas como um usuário real, permitindo a interação com o conteúdo que só aparece após ações específicas ou é influenciado pelo CSS.

Tanto R quanto Python oferecem ferramentas avançadas para *web scraping* de páginas que utilizam CSS e JavaScript, permitindo a extração de dados de *sites* complexos além do HTML básico. Ambas as linguagens permitem a automação do navegador e a simulação de interações de usuário, o que é essencial para acessar e extrair dados de páginas dinâmicas. No entanto, Python possui uma gama mais ampla e moderna de bibliotecas, como Selenium, Pyppeteer, Playwright e BeautySoup, com suporte ativo e contínuas atualizações. Em comparação, R utiliza pacotes como RSelenium[3] e rvest, que, embora eficazes, podem exigir combinações mais complexas e têm um ecossistema menos extensivo e integrado. Portanto, enquanto Python oferece uma abordagem mais diversificada e atualizada para automação e *scraping*, R depende de uma integração mais manual de pacotes para alcançar resultados similares. Apesar dessas várias abordagens e ferramentas para lidar com páginas dinâmicas que carregam conteúdo por meio de JavaScript, este texto se concentra exclusivamente em técnicas de *scraping* para páginas estáticas. O objetivo é fornecer uma visão introdutória e acessível, sem entrar em detalhes mais avançados sobre a manipulação de páginas que exigem interação dinâmica.

A biblioteca rvest, parte do ecossistema tidyverse em R, é projetada para lidar com documentos XML, incluindo o HTML. É eficaz para extrair dados quando se conhece o seletor CSS que identifica a parte da página com as informações desejadas. Identificar o seletor CSS correto pode ser desafiador, especialmente em páginas sofisticadas. O SelectorGadget é uma ferramenta útil para encontrar o seletor necessário. Essa ferramenta interativa permite clicar na página da *web* e identificar os seletores CSS correspondentes aos elementos desejados. A instalação do SelectorGadget é recomendada para

[3]Como Selenium é uma intercafe para Selenium WebDriver em Python, RSelenium é para R. O Selenium WebDriver é uma ferramenta popular para automação de navegadores *web*. Ele permite que se controle e interaja com navegadores de forma programática, o que é útil para uma variedade de tarefas, como teste automatizado de aplicações *web*, raspagem de dados e interação com páginas dinâmicas.

extração de dados que não sejam tabelas. Existe uma extensão para o Chrome que facilita essa tarefa, destacando elementos e exibindo o seletor necessário. Para mais informações, consulte o artigo sobre SelectorGadget do Hadley Wickham.

O código a seguir demonstra como realizar *web scraping* dos títulos dos melhores filmes disponíveis no *site* IMDb. Para isso, utiliza-se um cabeçalho "User-Agent" configurado para simular um navegador Chrome rodando em um sistema Linux[4]. O processo inclui a extração do texto bruto da página, sua conversão em um objeto HTML estruturado e a filtragem dos títulos usando o seletor CSS h3.

```
# Carregar bibliotecas necessárias
library(httr)
library(rvest)
library(xml2)

# Definir URL e cabeçalho 'User-Agent' da requisição HTTP
url <- "https://m.imdb.com/search/title/?groups=best_picture_winner"
header <- user_agent("Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) C

# Fazer a requisição GET
response <- GET(url, header)

# Extrair o conteúdo da resposta
dataText <- content(response, as = "text")

# Converter para o conteúdo HTML estruturado
dataHTML <- read_html(dataText)

# Encontrar todos os elementos h3 e armazená-los no vetor titleFilms, cada um
# dos elementos corresponde a um h3
titleFilms <- html_nodes(dataHTML, "h3")
```

A função html_nodes em R retorna um objeto da classe xml_nodeset. Este objeto é uma lista de nós XML ou HTML extraídos do documento e é manipulado principalmente usando funções de rvest e xml2.

Abaixo está um código equivalente em Python, que utiliza a função get() da biblioteca requests para obter o texto bruto da página. Em seguida, a função BeautifulSoup() do pacote bs4 é aplicada para fazer o *parsing* e a estruturação do HTML. Por fim, a função find_all() do pacote bs4 é usada para

[4]O cabeçalho "User-Agent" informa ao servidor qual navegador está sendo utilizado para a requisição de *scraping*.

coletar todos os elementos HTML que correspondem ao seletor especificado:

```
# Importar as bibliotecas necessárias
import requests
from bs4 import BeautifulSoup
import re
import pandas as pd

# Definir URL e cabeçalhos 'User-Agent' da requisição HTTP
url = "https://m.imdb.com/search/title/?groups=best_picture_winner"
headers = {
    "User-Agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chro
}

# Fazer a requisição GET
response = requests.get(url, headers=headers)

# Extrair o conteúdo da resposta
data_text = response.text

# Converter para o conteúdo HTML estruturado
soup = BeautifulSoup(data_text, 'html.parser')

# Encontrar todos os elementos h3
title_elements = soup.find_all('h3', class_='ipc-title__text')
```

Em Python, BeautifulSoup usa a classe bs4.element.ResultSet para representar uma coleção de *tags* que correspondem a um seletor ou expressão. ResultSet é essencialmente uma lista de objetos Tag ou NavigableString retornados pela função find_all() do BeautifulSoup, que podem ser manipulados diretamente ou acessados individualmente para extração de dados usando métodos de BeautifulSoup.

Vale ressaltar que, ao utilizar técnicas de raspagem de dados da *internet*, é fundamental considerar os aspectos legais, éticos e de acesso relacionado a essa prática. O *web scraping* pode levantar questões legais complexas, como violação de termos de serviço de sites e leis de proteção de dados, como a Lei Geral de Proteção de Dados Pessoais (LGPD), Lei n° 13.709/2018, no Brasil, o Regulamento Geral sobre a Proteção de Dados (GDPR) na União Europeia e a Lei de Proteção da Privacidade *Online* das Crianças (COPPA) nos Estados Unidos. Muitos *sites* bloqueiam acessos não autorizados ou utilizam

técnicas como CAPTCHAs[5] e mecanismos de bloqueio IP para prevenir *scraping* indevido, refletindo a necessidade de respeitar essas restrições e as políticas de acesso estabelecidas. Além disso, é essencial adotar uma abordagem ética, garantindo que a raspagem não prejudique o desempenho do *site*-alvo ou infrinja os direitos dos proprietários de dados. Respeitar as diretrizes de uso dos *sites* e manter a transparência quanto às intenções e métodos de coleta de dados são práticas importantes para assegurar a conformidade legal e a responsabilidade ética na utilização de *web scraping*.

5.3 Conversão dos Dados

A diversidade de formatos de dados apresenta desafios significativos na análise e exploração de dados, pois dados provenientes de diferentes fontes frequentemente vêm em estruturas variadas que podem não ser diretamente compatíveis. Arquivos estáticos de dados tabulares, como CSV e Excel, podem exigir transformação para alinhar variáveis e observações em um formato consistente, enquanto dados de bancos de dados relacionais podem necessitar de consultas complexas e manipulações para adaptar suas tabelas e relacionamentos a um formato analítico, adequado para exploração e análise. Dados extraídos de APIs dos servidores, geralmente em JSON ou XML, frequentemente possuem estruturas aninhadas que exigem descompactação e reorganização para se tornarem tabulares. Além disso, dados raspados da *web* podem ser não estruturados ou semi-estruturados, exigindo limpeza e estruturação adicional. Esses desafios de diversidade de formatos tornam a harmonização e a transformação de dados essenciais para garantir uma análise eficaz e uma integração fluida das informações.

A conversão para o formato "tidy" é uma estratégia eficaz para resolver inconsistências e tornar os dados mais acessíveis e úteis para análise. Esse formato facilita a manipulação e análise, permitindo a aplicação eficiente de técnicas estatísticas e de aprendizado de máquina. Para utilizar as ferramentas disponíveis em R e Python para essa conversão, é fundamental que os dados estejam inicialmente em estruturas apropriadas, como data.frames ou tibbles em R e DataFrames em Python. Os objetos dessas classes podem servir como base para aplicar funções e métodos que reestruturam e limpam os dados. Nesta seção, oferecemos uma visão geral sobre como converter dados importados de diferentes fontes para os formatos de data.frame e tibble em R, e DataFrame em Python.

Para transformar dados importados de diferentes fontes em estruturas adequadas para pré-processamento em R e Python, é prática comum realizar uma verificação inicial dos dados, para confirmar que a transferência foi bem-sucedida. Isso começa com a verificação da classe dos objetos importados: em R, usa-se a função class(), enquanto em Python utiliza-se a função type(). Para inspecionar a integridade dos dados, é comum visualizar alguns elementos do objeto importado. Em R, a função head() é amplamente utilizada para exibir as primeiras linhas de um objeto. Em Python, métodos e técnicas

[5] CAPTCHAs (do inglês *Completely Automated Public Turing test to tell Computers and Humans Apart*) são mecanismos de segurança usados para distinguir entre usuários humanos e *bots* (programas automatizados). O objetivo principal é impedir que sistemas automatizados realizem ações que devem ser restritas a usuários humanos, como o preenchimento de formulários, a criação de contas ou a realização de *scraping* de dados.

alternativas são necessários para visualizar partes dos dados, dependendo da classe do objeto. A função head() é específica para pandas.DataFrame. Para listas e dicionários, é necessário usar a técnica de fatiamento para acessar os primeiros elementos introduzida na Seção 2.1.5.

Dados provenientes de **arquivos estáticos**, como CSV ou Excel, são carregados diretamente com funções como read.csv() ou read_excel() do pacote readxl em R, resultando em data.frames ou tibbles, e com funções como read_csv() ou read_excel() do pacote pandas em Python, que geram DataFrames. Para dados extraídos de **bancos de dados** relacionais, consultas SQL são realizadas e os resultados são importados como data.frames ou tibbles em R, utilizando pacotes como DBI e RSQLite, e como DataFrames em Python, com a função read_sql() do pacote pandas.

Para dados obtidos via **APIs**, geralmente em formato JSON, utilizamos ferramentas específicas para a conversão desses dados em estruturas de dados manipuláveis. Em R, a função fromJSON() do pacote jsonlite é empregada para converter JSON em data.frames ou tibbles. A classe do objeto retornado por fromJSON() pode variar conforme a estrutura do JSON original, frequentemente resultando em uma lista com hierarquia que reflete a complexidade do JSON, mas também pode ser diretamente um data.frame se o formato for mais simples. Quando lidamos com listas aninhadas, a transformação pode envolver a conversão de sub-listas em um data.frame usando a função as.data.frame(), seguido de manipulações adicionais com pacotes como dplyr e tidyr. O código a seguir ilustra como converter uma sub-lista orders, extraída de um objeto data_parsed com listas aninhadas, num data.frame:

```
# Converter JSON para uma lista de pedidos R
data_parsed <- fromJSON(data)


# Extrair e converter a lista de pedidos em um data frame
orders_df_r <- as.data.frame(data_parsed$orders)
```

Os dados JSON são geralmente convertidos em dicionários e listas em Python, refletindo a estrutura do JSON original. A complexidade do JSON pode influenciar a organização desses dados em Python. Ao trabalhar com esses dados convertidos, é prática comum acessar e transformar listas de objetos em DataFrames utilizando a função DataFrame() da biblioteca pandas. Isso permite realizar manipulações adicionais, como estruturação e limpeza de dados, com as diversas funções oferecidas pelo pandas. Segue-se o código em Python para converter uma sub-lista orders, extraída de um objeto data_parsed com listas aninhadas, num DataFrame:

```
import pandas as pd
import json


# Supondo que json_string é a string JSON contendo a chave 'orders'
data_parsed = json.loads(json_string)
```

```
# Converter uma lista de objetos em um DataFrame
orders_df_py = pd.DataFrame(data_parsed['orders'])
```

Finalmente, para dados raspados da *web*, o HTML extraído com rvest em R ou BeautifulSoup em Python é convertido em data.frames ou DataFrames após o *parsing* e a estruturação dos dados. Esses procedimentos são essenciais para garantir que os dados estejam organizados e prontos para serem transformados em formatos "tidy". O exemplo a seguir demonstra como converter um objeto da classe xml_nodeset, obtido durante a raspagem de dados na Seção 5.2.4, em uma lista e, em seguida, transformar essa lista em um data.frame em R após o laço de iteração:

```
library(stringr)
library (rvest)
library (xml2)

# Definir o padrão de expressão regular para encontrar o texto dentro das tags h3
html_pattern <- "<h3.*?class=\"ipc-title__text\".*?>(.*?)</h3>"

# Inicializar uma lista para armazenar os títulos
titles_list <- list()

# Extrair e armazenar os títulos
for (i in seq_along(titleFilms)) {
    # Converter o nó HTML para texto e aplicar a expressão regular
    film_text <- as.character(titleFilms[i])
    matches <- str_match(film_text, html_pattern)
    if (!is.na(matches[1])) {
        # Adicionar o título à lista
        titles_list[[length(titles_list) + 1]] <- matches[2]
    }
}

# Converter a lista de títulos em um data.frame
titles_df <- data.frame(Title = unlist(titles_list), stringsAsFactors = FALSE)
```

O trecho de código a seguir ilustra como converter um objeto title_elements da classe bs4.element.ResultSet, obtido durante a raspagem de dados na Seção 5.2.4, em uma lista e, posteriormente, transformar essa lista em um DataFrame em Python após o laço de iteração:

```
# Definir o padrão de expressão regular para encontrar o texto dentro das tags h3
html_pattern = re.compile(r'<h3.*?class="ipc-title__text".*?>(.*?)</h3>', re.S)

# Inicializar uma lista para armazenar os títulos
titles_list = []

# Extrair e armazenar os títulos
for element in title_elements:
    film_text = str(element)
    matches = html_pattern.search(film_text)
    if matches:
        titles_list.append(matches.group(1))
# Converter a lista de títulos para um DataFrame
titles_df = pd.DataFrame({'Title': titles_list})
```

5.4 Validação dos Dados

A validação de dados importados e transformados pode garantir a qualidade e a integridade dos dados antes do pré-processamento e análise. A verificação das transformações aplicadas em relação à saída esperada assegura a corretude dos dados. Em R, funções como summary() e, em Python, a função describe() do pacote pandas são úteis para revisar estatísticas descritivas e assegurar que os dados estejam no formato correto. Além disso, a checagem de integridade dos dados é fundamental; é necessário verificar referências cruzadas em conjuntos de dados relacionados para garantir a consistência das relações entre tabelas.

Outro aspecto importante é o teste de subconjuntos de dados, que frequentemente envolve a análise visual de amostras para verificar se as transformações realizadas não introduziram erros. A visualização de subconjuntos específicos de dados pode revelar padrões inesperados, discrepâncias ou problemas que não são facilmente detectáveis apenas por métodos quantitativos, como ilustra a Figura 5.2. Além disso, a checagem de *outliers* e anomalias também é fundamental. A identificação de valores extremos através de gráficos e representações visuais pode indicar erros nas transformações ou na qualidade dos dados. Enfim, as técnicas visuais ajudam a a verificar a integridade e a consistência dos dados, além de facilitar a detecção de erros.

Entre as técnicas de visualização, destacam-se os seguintes gráficos mas usados na validação visual de dados importados:

- **Histogramas**: Utilizados para verificar a distribuição dos dados e identificar valores

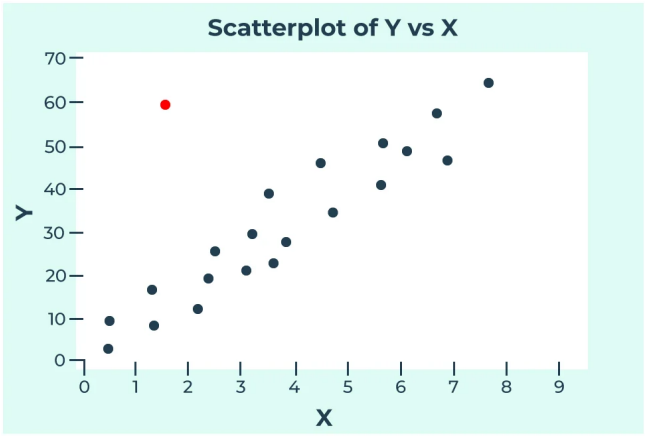


Figura 5.2: Detecção visual de valores extremos (*outliers*).

fora do padrão. Histogramas ajudam a detectar discrepâncias, como valores extremos ou dados faltantes.

- **Gráficos de Dispersão (em inglês *Scatter Plots*)**: Útil para observar relações entre variáveis e identificar *outliers* ou erros de entrada. Pode revelar correlações inesperadas ou padrões de dados inconsistentes.
- **Gráficos de Caixa** (em inglês *Boxplots*): Fornecem uma visão clara das distribuições dos dados e ajudam a identificar *outliers* e variabilidade nos dados.
- **Gráficos de Linha**: Usados para visualizar tendências ao longo do tempo e verificar se os dados seguem um padrão esperado. Pode ajudar a identificar falhas ou anomalias na série temporal.
- **Matrizes de Correlação**: Exibem a relação entre múltiplas variáveis e podem ajudar a identificar dados inesperados ou incoerentes.
- **Gráficos de Pareto**: Mostram a frequência ou o impacto de diferentes categorias de dados, ajudando a identificar as áreas mais significativas para validação.
- **Tabelas de Frequência**: Apresentam a contagem de valores para variáveis categóricas, facilitando a identificação de categorias inesperadas ou erros de codificação.
- **Gráficos de Qualidade dos Dados**: Incluem visualizações que mostram a porcentagem de dados faltantes, duplicados ou inconsistentes, oferecendo uma visão geral da qualidade dos dados importados. Além disso, gráficos de barras empilhadas podem ser utilizados para visualizar as porcentagens de dados faltantes ou duplicados.

Finalmente, a validação de formatos e estruturas é essencial para garantir que os dados estejam corretamente formatados. Em R, a função `str()` permite examinar a estrutura de um `data.frame` ou `tibble`, enquanto em Python, a função `info()` do pacote `pandas` oferece uma visão geral da estrutura do `DataFrame`.

5.5 Automação e Repetibilidade

A automação e a repetibilidade no processo de importação de dados podem melhorar eficiência e consistência no fluxo de trabalho de análise de dados. A motivação para automatizar a importação de dados surge da necessidade de minimizar erros humanos, acelerar processos e melhorar a qualidade dos dados, permitindo que as equipes se concentrem mais na análise e menos em tarefas repetitivas e propensas a erros.

A utilidade da automação é evidente na criação de *pipelines* de dados que podem ser facilmente ajustados e reutilizados. Isso não apenas torna o processo mais eficiente, mas também assegura que os dados sejam importados de maneira consistente e com menos necessidade de intervenção humana. A repetibilidade é alcançada por meio da definição de processos claros e documentados, que garantem que a importação de dados possa ser realizada da mesma maneira em diferentes ocasiões, proporcionando confiabilidade e previsibilidade nos resultados.

Aplicar técnicas de automação no processo de importação envolve a construção de *pipelines* modulares e reutilizáveis que encapsulam todas as etapas necessárias, desde a coleta até a transformação dos dados, como mostra a Figura 5.3. Ferramentas especializadas, como plataformas de ETL (do inglês *Extract – Transform – Load*), são amplamente utilizadas para facilitar esse processo. Ferramentas de código aberto, como Apache Airflow, Luigi e as mencionadas em recursos como o Mindtek ajudam na automação e gerenciamento dos fluxos de trabalho de dados. Estas ferramentas oferecem funcionalidades para agendar e monitorar a execução de tarefas, além de integrar diferentes fontes de dados.

Os desafios associados à automação e repetibilidade incluem a complexidade na configuração inicial dos *pipelines*, a necessidade de atualização regular das configurações, monitoramento constante para garantir que as transformações estejam corretas e a adaptação a mudanças nos formatos e fontes de dados ou nas necessidades do projeto. Para superar esses desafios, é essencial implementar práticas de testes e monitoramento eficazes. Testes automatizados podem ser utilizados para verificar a integridade dos dados após a importação, enquanto sistemas de monitoramento ajudam a identificar e corrigir problemas rapidamente. A documentação detalhada é fundamental para garantir que o processo permaneça eficiente e confiável ao longo do tempo.

Apesar dos desafios, a automação da importação de dados, integrando APIs e bancos de dados locais, representa um avanço significativo na gestão de dados. Ferramentas de automação que extraem dados de APIs, realizam transformações personalizadas e carregam-nos em um banco de dados centralizado, possibilitam uma gestão de dados eficiente. Essa automação reduz a dependência de processos manuais, minimizando erros e inconsistências, e garante que os dados estejam sempre atualizados e prontos para análise. Como resultado, as organizações podem integrar dados de diversas fontes de forma harmoniosa, otimizando a tomada de decisões e fortalecendo a análise estratégica.

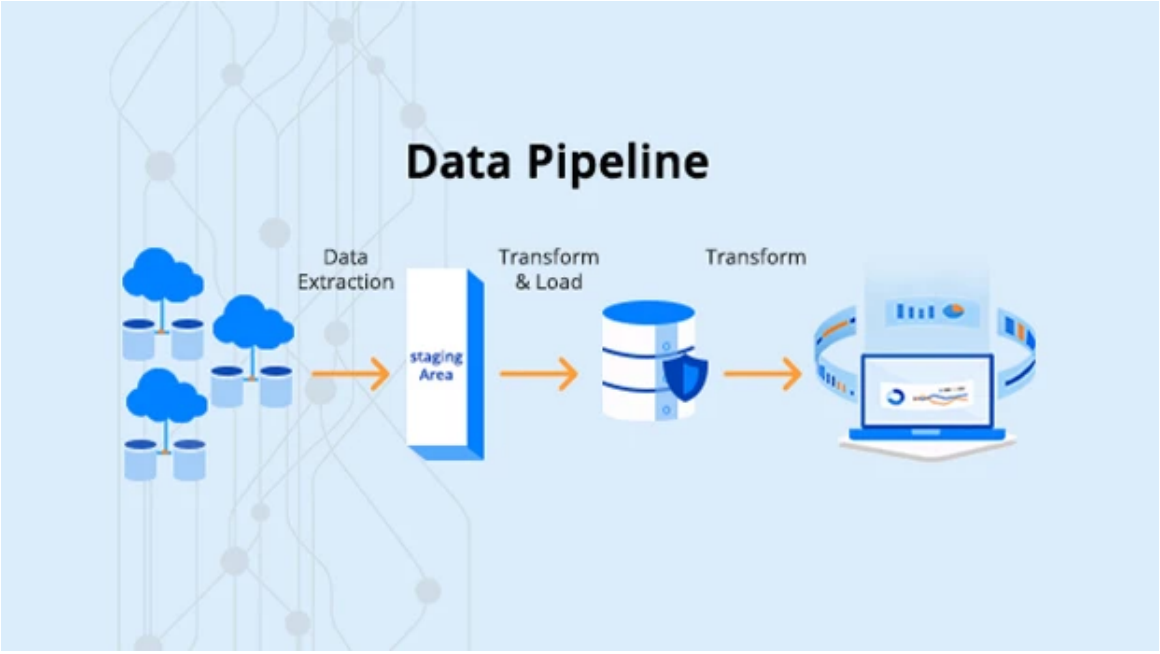


Figura 5.3: *Pipeline* de dados: um conjunto de operações que inclui tarefas periódicas em *batch* para **extrair** dados de uma fonte, **transformar** esses dados para que correspondam ao formato exigido pelo repositório de destino e, por fim, **carregá**-los no destino escolhido, que pode ser um banco de dados ou um *data warehouse*. Esses dados estruturados podem então ser consultados em tempo real.

5.6 Considerações Finais

Abordamos o processo de importação de dados, destacando a importância da conversão para um formato tabular como uma estratégia para mitigar a complexidade associada à diversidade de formatos e fontes, permitindo uma abordagem sistemática e coerente na transformação desses dados em "*tidy datasets*". Na Seção 5.1, detalhamos o conceito de "*tidy datasets*", onde variáveis são representadas como colunas, observações como linhas, valores como células e tipos de unidade como tabelas. Esse formato não apenas simplifica a análise, mas também promove a consistência e a interoperabilidade dos dados. Embora o formato "*tidy*" seja amplamente adotado e eficaz para muitos tipos de análise, é importante reconhecer que alguns modelos de dados são melhor representados por outros formatos visuais. Por exemplo, dados relacionais complexos e redes sociais são frequentemente melhor visualizados e analisados por meio de modelos de grafos (estruturas de dados que representam relações entre entidades). No entanto, a análise e exploração de dados em formatos de grafos carecem de ferramentas tão desenvolvidas quanto aquelas disponíveis para dados tabulares, o que pode limitar sua aplicação prática. Portanto, é essencial considerar as necessidades específicas de cada conjunto de dados e explorar ferramentas adequadas para representar e analisar dados em diferentes formatos. Por exemplo, para visualização de grafos, existem ferramentas como Gephi e Cytoscape.

A Seção 5.2 elucidou as várias fontes de dados, incluindo arquivos textuais como CSV, TSV e TXT, bancos de dados relacionais, APIs e técnicas de *web scraping*. Apresentamos funções em R e Python para importar essas diferentes fontes de dados. Embora essas funções diferem em sintaxe, elas

são conceitualmente muito similares, refletindo a necessidade de uma abordagem flexível e adaptativa para integrar dados provenientes de diversas origens e formatos. A importação de dados de APIs e técnicas de *web scraping* geralmente envolve uma complexidade maior, pois os dados frequentemente são aninhados e hierárquicos, exigindo etapas adicionais de *parsing* e serialização para que possam ser utilizados de forma eficaz no contexto clássico de análise e exploração de dados tabulares. É essencial adaptar as estratégias de importação às características específicas de cada fonte para garantir a integridade e a precisão dos dados, independentemente da linguagem utilizada.

Na Seção 5.3, discutimos técnicas de transformação dos dados para formatos tabulares, essencial para garantir a consistência e interoperabilidade entre diferentes fontes, facilitando a análise e integração dos dados. A validação dos dados importados, abordada na Seção 5.4, é essencial para assegurar que os dados sejam precisos e úteis. Implementar métodos rigorosos de validação, incluindo a validação visual dos dados, ajuda a identificar e corrigir erros, melhorando a qualidade e a confiabilidade dos *insights*. Foram apresentados gráficos aplicados na validação visual, como histigramas, *heatmaps* e gráficos de barras empilhadas, que ajudam a visualizar e entender a distribuição de dados faltantes e duplicados. Na Seção 5.5, apresentamos ferramentas para a automação do processo de importação, destacando como a automação acelera o fluxo de trabalho e reduz a intervenção humana, minimizando erros e melhorando a consistência, especialmente em cenários com grandes volumes de dados e fontes dinâmicas.

Além dos pontos abordados, é importante considerar que a prática contínua de monitoramento e manutenção humana dos *pipelines* de dados é essencial para lidar com mudanças nas fontes de dados e nas necessidades do projeto. A documentação detalhada e a comunicação eficaz entre as equipes envolvidas na importação e análise de dados são fundamentais para manter a eficiência e a confiabilidade dos processos de dados. A automação permite que processos que antes eram feitos manualmente sejam executados de forma mais rápida e com menor chance de erro humano. Embora a automação impulsione a eficiência e permita a gestão de dados em escala expandida, ela também amplifica o potencial de erros. Falhas, quando ocorrem, podem ter impactos significativos e complexos, exigindo esforços substanciais para recuperação.

Adicionalmente, é recomendável a implementação de estratégias de *backup* e recuperação para proteger contra a perda de dados e garantir a continuidade das operações. A integração de práticas de segurança para proteger dados sensíveis também deve ser uma prioridade para assegurar a conformidade com as regulamentações e proteger a privacidade dos dados. A combinação de todos esses esforços é fundamental para a construção de uma base sólida e robusta para análise e exploração dde dados e a tomada de decisões informadas.

5.7 Exercícios

1. Baixe o arquivo de dados CSV age_at_mar disponível no repositório de datasets do R . Importe os dados num objeto da classe tibble em R ou da classe DataFrame em Python. Valide os dados importados verificando as variáveis (colunas) e as observações (linhas) importadas. Utilize a função summarize() no R ou describe() no Python para obter uma visão geral e resumida dos dados importados. Verifique a distribuição dos dados com uso de um histograma e gráfico de caixa. Examine o histograma gerado e identifique um padrão ou característica notável, como distribuição de dados e *outliers*.

2. Acesse o Portal de Dados Abertos do TSE. Selecione qualquer um arquivo CSV da categoria "Resultados". Baixe e descompacte o arquivo. Dentro do arquivo ZIP baixado, localize o arquivo leiame-x.pdf. Abra o leiame-x.pdf para identificar o *encoding* dos dados e os nomes das variáveis presentes no arquivo CSV. Importe os dados, usando essas informações, num objeto da classe tibble em R ou da classe DataFrame em Python. Valide os dados importados verificando as variáveis (colunas) e as observações (linhas) importadas. Utilize a função summarize() no R ou describe() no Python para obter uma visão geral e resumida dos dados importados (Fonte).

3. Faça os itens 1–5 da Seção 15.4 em [71].

4. Extraia os títulos dos filmes listados como "Melhores Filmes" no site do IMDb. Armazene os títulos extraídos em um objeto da classe tibble em R ou da classe DataFrame em Python. Valide os dados importados verificando os primeiros 5 nomes importados (item 13 da Seção 15.4 em [71]).

Capítulo 6

Estatistica Descritiva

A estatística descritiva tem como objetivo resumir, organizar e descrever os dados de maneira significativa. Ao considerar os dados como uma população caracterizada por diversas medidas observadas, ela facilita a compreensão, análise e interpretação, fornecendo **medidas resumo** (*summary statistics*, em inglês) apropriadas para comunicar as características essenciais dos dados. Tais medidas, conhecidas como variáveis, podem ser qualitativas (nominais ou ordinais) ou quantitativas (contínuas ou discretas), oferecendo uma ampla gama de tipos de dados para análise.

Para compreender a variabilidade, tendências e padrões nos dados, uma ferramenta de destaque da estatística descritiva é a **distribuição de frequência** dos dados, mostrando quantas vezes cada valor aparece. Ela fornece uma análise sistemática da ocorrência dos valores em uma população, ajudando a entender como os dados estão distribuídos. Além disso, na análise e compreensão de padrões e relacionamentos nos dados, a estatística descritiva oferece ferramentas como correlação e similaridade. A **correlação** de dados fornece uma descrição do grau e da direção do relacionamento entre duas ou mais variáveis dentro de uma população, enquanto a **similaridade** de dados indica o grau de semelhança entre os dados com base nos valores assumidos por suas variáveis. Essas medidas são fundamentais no processo de clusterização (em inglês, *clustering*), pois ajudam a determinar a proximidade entre diferentes observações ou elementos do conjunto de dados da população.

Embora essas ferramentas auxiliem na revelação das estruturas subjacentes nas populações, os dados que geram muitas vezes são tão volumosos que decifrá-los se torna um desafio. Reconhecidos estatísticos, como Tufte (Seção 3.3) e John Tukey, têm se dedicado a tornar esses dados mais acessíveis. Isso envolve a criação de gráficos que não apenas facilitam a obtenção de *insights*, mas também permitem identificar nuances e padrões que seriam difíceis de discernir apenas com números ou texto. Leland Wilkinson desenvolveu o conceito da gramática de gráficos, uma abordagem sistemática para a construção de gráficos estatísticos, independentemente da complexidade dos dados (Seção 3.5). Por meio de uma sintaxe padronizada, os dados são transformados em uma variedade de representações gráficas compreensíveis, tais como gráficos de barras, gráficos de pizza, gráficos de linhas, bem como

outras representações visuais populares, como gráficos de quantis, gráficos de quantil-quantil, histogramas e gráficos de dispersão. Esse avanço propiciou a proliferação da estatística descritiva para a inspeção visual de grandes volumes de dados. A maioria dos pacotes de análise estatística oferece uma variedade de opções de visualização de dados, permitindo que os usuários explorem e comuniquem suas descobertas de maneira eficaz.

A visualização de dados revela padrões, tendências e relações não óbvias, além de destacar valores discrepantes (em inglês, *outliers*) que podem influenciar a análise. A identificação e o tratamento adequado desses *outliers* asseguram a robustez e confiabilidade dos resultados. Conforme discutiremos no Capítulo 7, a preparação de dados é uma etapa crítica e desafiadora, pois a qualidade dos dados impacta significativamente os resultados da análise. Técnicas estatísticas, tanto numéricas quanto gráficas, são essenciais para essa preparação.

Este capítulo explora as quatro ferramentas de descrições estatísticas e as representações gráficas apropriadas para visualizar e explorar tais descrições diretamente a partir dos dados. A apresentação de cada ferramenta é acompanhada de exemplos práticos em R e Python. Na Seção 6.1, são apresentadas medidas de tendência central, que indicam a localização do meio ou centro de uma distribuição de dados, e uma visão da dispersão dos dados em torno desta tendência. Na Seção 6.2, mostramos as medidas que sintetizam o grau e a direção de relação entre dois conjuntos de dados, além de abordar a construção de um modelo matemático (equação) que representa essa relação. Em seguida, a Seção 6.3 explora medidas que revelam o grau de similaridade ou proximidade entre dois conjuntos de dados. Por fim, a Seção 6.4 detalha a ferramente de clusterização e suas variantes.

6.1 Estatísticas Resumidas

Medidas resumo, também conhecidas como **estatísticas resumidas**, são indicadores que condensam os valores de uma **variável** ou **característica** (observações univariáveis) em uma população. Ao resumir todos os valores que uma variável pode assumir na população, essas medidas oferecem uma visão concisa e informativa de sua distribuição ou comportamento, destacando especialmente a tendência central desses valores e como eles estão dispersos em torno da tendência central.

6.1.1 Organização de Dados

A coleta de dados de uma população de interesse gera um conjunto inicial de **dados brutos** que requer organização para se tornar acessível e compreensível durante a extração das informações desejadas. Geralmente, os dados são organizados em **tabelas de frequência** por variável, onde os valores são listados junto com sua contagem ou frequência. A essa representação que descreve a maneira como os valores de uma variável estão espalhados ou distribuídos em um conjunto de dados, chamamos de **distribuição de frequência** dos dados. No caso de variáveis discretas, a tabela de frequência

consiste em listar os valores possíveis da variável e contar suas ocorrências. Frequentemente, os dados são agrupados em **intervalos**, ou faixas, de valores, em vez de lidar com cada valor individualmente, o que pode simplificar a análise e visualização dos dados.

A distribuição de frequência de dados fornece informações importantes sobre a natureza dos dados, incluindo a tendência central (média, mediana, moda), a presença de *outliers* e a simetria da distribuição, podendo revelar informações ocultas [52]. Uma distribuição pode ser visualizada através de diferentes tipos de gráficos. Por exemplo, o uso de **histogramas**[1], que representam graficamente as tabelas de frequência, oferece uma visão geral e intuitiva da distribuição de todos os dados. Isso permite identificar padrões, valores discrepantes e tendências nos conjuntos de dados de forma visual, contribuindo para uma análise estatística mais eficaz, como demonstrado pelo histograma publicado no *New York Times* sobre os resultados dos exames *Regents* para obtenção do diploma do ensino médio no estado de Nova York [69]. Esse histograma revela que houve arredondamentos das notas ligeiramente abaixo do limite de aprovação para o limite de aprovação.

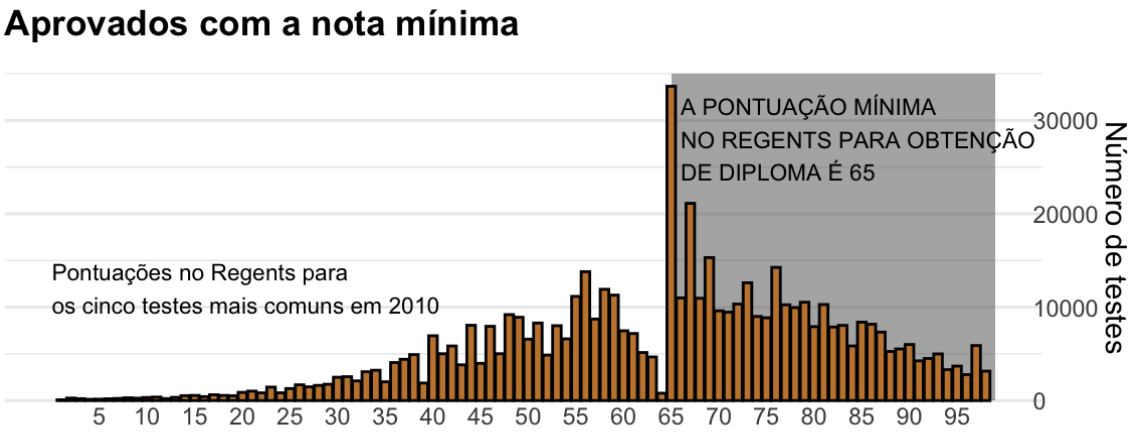


Figura 6.1: Histograma dos resultados de exames *Regents* revela um comportamento pouco esperado em torno da pontuação mínima. (Fonte: [69])

Embora os histogramas sejam excelentes para visualizar a distribuição geral dos dados e identificar a frequência de valores em diferentes faixas, conhecidas por *bins*, eles podem ter algumas limitações. A escolha do número de *bins* pode influenciar significativamente a aparência do histograma. Se houver poucos *bins*, a imagem pode ser muito generalizada; se houver muitos, pode parecer muito ruidosa e granular. Uma alternativa para visualizar a distribuição de dados, que pode ser mais suave e menos dependente da escolha específica dos *bins*, é o **gráfico de densidade**. Em vez de agrupar os dados em intervalos discretos, um gráfico de densidade tenta estimar a função de densidade de probabilidade subjacente dos dados. Ele faz isso colocando uma pequena "função *kernel*" (geralmente uma função gaussiana) em cada ponto de dados individual. Essas funções *kernel* são então somadas para criar

[1]Embora sejam similares visualmente, os histogramas são diferentes do gráficos de barras (em inglês. *bar charts* ou *bar plots*) na natureza das variáveis que eles representam. Um gráfico de barras representa dados categóricos ou quantitativos discretos e um histograma representa dados quantitativos. Histogramas capturam a ideia de intervalos contínuos de valores ordenados.

uma curva suave que representa a densidade dos dados em diferentes pontos. A principal vantagem do gráfico de densidade é que ele suaviza a representação da distribuição, tornando mais fácil identificar a forma geral da distribuição, como a presença de picos (modos), a simetria ou assimetria, e a extensão dos dados, sem a influência direta da escolha dos *bins*.

Muitas distribuições de frequência de ocorrência dos valores de uma variável encontrados na natureza exibem formas de distribuição semelhantes, e muitos dos padrões identificados receberam nomes específicos para facilitar a comunicação e a compreensão entre os estatísticos e pesquisadores. Entre as distribuições de frequências de ocorrência mais conhecidas e amplamente utilizadas na análise estatística de dados estão (Figura 6.2):

Distribuição Uniforme: Esta distribuição descreve situações em que todos os valores dentro de um intervalo têm a mesma frequência de ocorrência (Figura 6.2a).

Distribuição Normal (Gaussiana): Em forma de sino (Figura 6.2b), esta é uma das distribuições mais comuns. Muitos processos naturais e medidas físicas tendem a seguir essa distribuição.

Distribuição Binomial: Esta distribuição descreve o número de sucessos em um número fixo de tentativas independentes de um experimento, onde cada tentativa tem apenas dois resultados possíveis (sucesso ou fracasso). Cada barra na Figura 6.2a corresponde à contagem de sucessos em um número fixo de tentativas independentes de um experimento de Bernoulli.

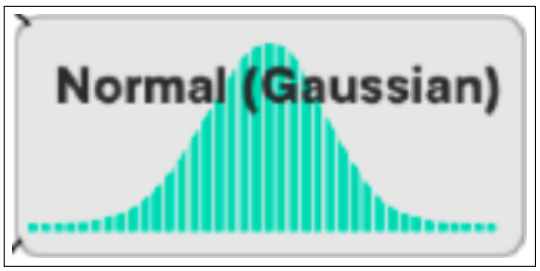
Distribuição de Bernoulli: É um caso especial da distribuição binomial onde há apenas uma tentativa do experimento (Figura 6.2d).

Distribuição de Poisson: Enquanto a distribuição binomial modela o número de sucessos em um número n fixo de tentativas independentes, com probabilidade p de sucesso em cada tentativa, a distribuição de Poisson modela o número de eventos que ocorrem em um intervalo de tempo ou espaço fixo, com uma taxa média constante de ocorrência desses eventos. A distribuição de Poisson é usada quando os eventos são raros e independentes, e o intervalo de observação é fixo. Ela é caracterizada por um único parâmetro λ que é a média de ocorrência de eventos em um determinado intervalo de tempo ou espaço, como ilustra a Figura 6.2e.

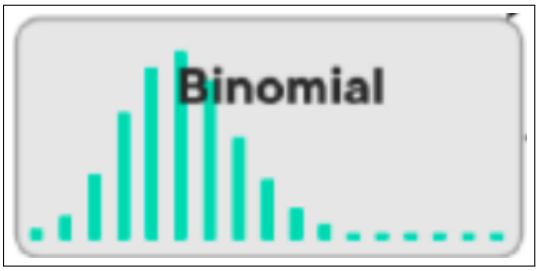
Distribuição Exponencial: Esta distribuição modela o tempo decorrido ou o espaço entre dois eventos consecutivos em um processo de Poisson (Figura 6.2f). É uma distribuição contínua, o que significa que pode modelar qualquer valor de tempo ou espaço.

Quando não se tem certeza sobre a forma da distribuição dos dados, gráficos de barras, histogramas e gráficos de densidade oferecem uma representação visual útil, destacando características importantes

(a) Uniforme

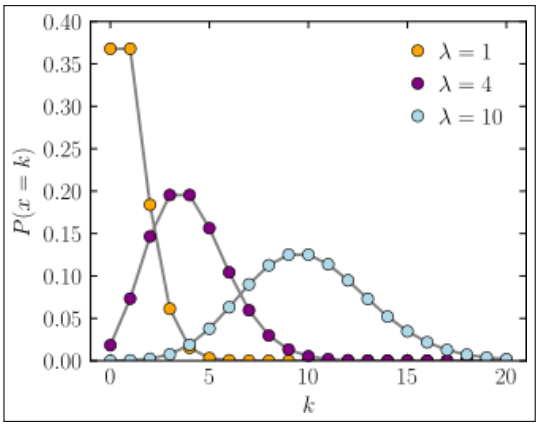


(b) Normal

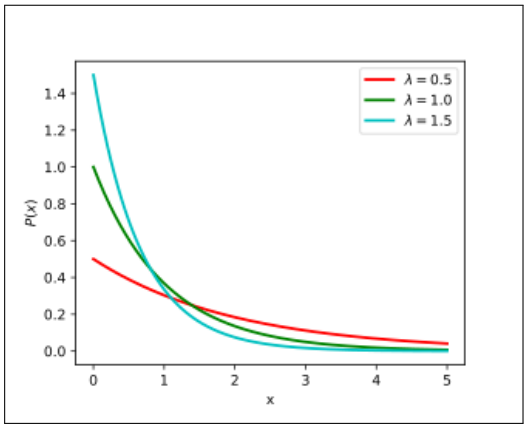


(c) Binormal

(d) Bernoulli



(e) Poisson



(f) Exponencial

Figura 6.2: Exemplos de distribuições de frequência de ocorrência dos valores de uma variável. (Fonte)

e permitindo uma compreensão intuitiva do formato da distribuição, o que facilita a identificação de padrões. Quando os dados são contínuos, é comum que muitos valores sejam únicos. Portanto, relatar a frequência de cada valor individual torna-se ineficaz para resumir os dados. Uma abordagem mais útil para resumir a distribuição de dados numéricos é utilizar uma função que descreva a proporção de dados abaixo de um determinado valor $x_i \in X$, para todos os possíveis valores de uma parte específica dos dados. Essa função é conhecida como **função de distribuição cumulativa** empírica (em inglês, *empirical cumulative distribution function*) e é comumente denotada por F[71]:

$$F(x_i) = \text{ proporção de valores de dados que são menores ou iguais a } x_i,$$

Em outras palavras, $F(x_i)$ mostra a frequência de ocorrência acumulada de valores menores ou iguais a x_i na distribuição de dados. O **gráfico de distribuição acumulada** é uma representação visual da função de distribuição cumulativa. Neste gráfico, o eixo horizontal representa os valores possíveis

da variável de interesse, enquanto o eixo vertical representa a proporção acumulada de observações que são menores ou iguais a cada valor ao longo do eixo horizontal. Cada ponto no gráfico indica a proporção acumulada de observações até aquele valor específico. Esse gráfico é especialmente útil para visualizar a distribuição acumulada dos dados e comparar distribuições entre diferentes grupos. Ele permite identificar rapidamente a posição relativa dos valores e entender a variabilidade dos dados em relação a uma escala de referência. A Figura 6.3 ilustra o gráfico de distribuição acumulada no eixo y dos valores da variável height, expressa em polegadas, provenientes do conjunto de dados heights dos estudantes masculinos do pacote dslabs.

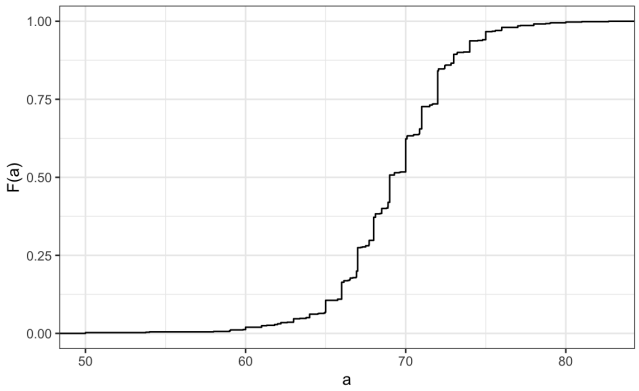


Figura 6.3: Gráfico de distribuição acumulada das alturas em polegadas dos estudantes masculinos do conjunto heights do pacote dslabs. (Fonte: [70])

6.1.2 Medidas de Posição

A **tendência central** é o ponto central ou valor ao redor do qual um conjunto de dados x_i se agrupa. Ela é usada para representar, de maneira resumida e indicativa, a localização central dos dados em um conjunto. Por isso, as medidas relacionadas a ela são também conhecidas como **medidas de posição**. Ela é expressas por meio de estatísticas como:

Média (μ): É a média aritmética dos valores x_i de toda população de tamanho N de uma variável quantitativa

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i. \tag{6.1}$$

A média é sensível a *outliers* e fornece uma medida do "centro de massa" dos dados.

Mediana: é o valor de uma variável quantitativa que divide o conjunto de dados ao meio quando organizado em ordem crescente ou decrescente. Ela não é afetada por *outliers* numa distribuição assimétrica, onde há uma cauda longa de dados em uma direção; portanto, é útil para entender a localização central dos dados mesmo na presença de *outliers*.

Moda: É o valor de uma variável que ocorre com a maior frequência no conjunto de

dados. Pode haver mais de um valor de maior ocorrência, ou seja, mais de uma moda associada a uma variável. Conjuntos de dados com uma, duas, ou três modas são denominados, respectivamente, unimodais, bimodais e multimodais.

Meio da faixa: É a média do maior valor $max(X)$ e menor valor $min(X)$ de um conjunto de dados X

$$meio\_da\_faixa = \frac{max(X) + min(X)}{2}. \tag{6.2}$$

A Figura 6.4 esboça a posição relativa das medidas de tendência central, média, mediana e moda, em distintas distribuições de dados. Vale ressaltar que, ao lidar com uma distribuição de dados, a média e a mediana podem não coincidir com valores específicos que a variável possa assumir. Isso acontece porque tanto a média quanto a mediana representam medidas centrais da distribuição dos valores observados, e não necessariamente valores observados individualmente. Essas medidas fornecem informações distintas sobre a distribuição dos dados, destacando aspectos como tendência central e dispersão, que são fundamentais para compreender a natureza dos dados.

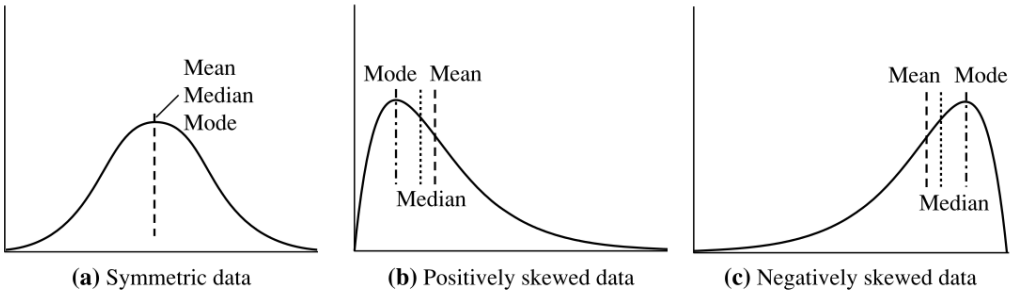


Figura 6.4: Tendências centrais em diferentes distribuições de frequência de dados: (a) distribuição simétrica, (b) distribuição assimétrica com a inclinação para valores mais baixos, (c) distribuição assimétrica com a inclinação para valores mais altos. (Fonte: [31])

6.1.3 Medidas de Dispersão

A **dispersão de dados** é uma medida que indica o quão espalhados ou concentrados estão os valores de uma variável quantitativa, oferecendo informações cruciais sobre a heterogeneidade ou consistência dos dados. Quando os dados estão altamente dispersos, isso significa que eles variam consideravelmente de valor para valor, enquanto uma baixa dispersão indica que os valores estão mais próximos uns dos outros. Para compreender a dispersão e a posição relativa dos dados em uma escala de referência, recorre-se à distribuição cumulativa de frequência em vez da distribuição de frequencia de dados convencional. Pois, ao contrário de simplesmente mostrar a frequência de ocorrência de cada valor individual, a **distribuição cumulativa** soma essas frequências à medida que avança ao longo da escala da variável, oferecendo um melhor entendimento da dispersão e da posição relativa dos valores possíveis de uma variável de interesse ao longo do eixo horizontal do gráfico. Cada ponto do gráfico é um par ordenado ($F(x_i)$,x_i), onde $F(x_i) \times 100\%$ é a percentagem dos dados que estão abaixo do valor

x_i, como ilustra a Figura 6.5. Note que a identificação da mediana é significativamente mais simples no gráfico de distribuição acumulada do que nos gráficos de distribuição de frequência mostrados na Figura 6.4.

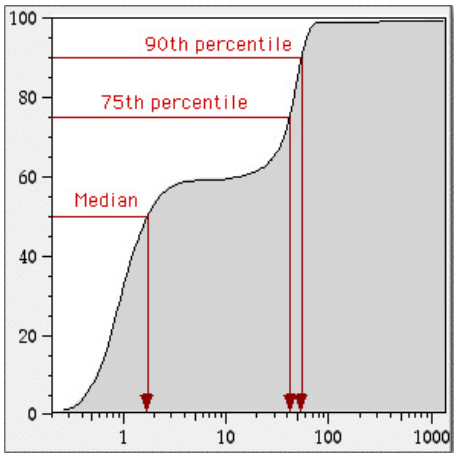
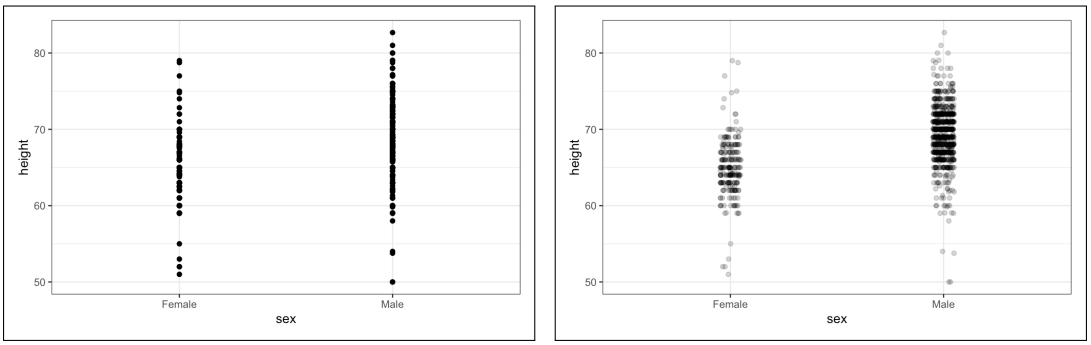


Figura 6.5: Dispersão de dados numa distribuição acumulada de frequência de dados revelada pela identificação direta de estatísticas como a mediana e outros valores que dividem os dados em termos de proporções de ocorrências. (Fonte)

Embora gráficos de dispersão (em inglês, *scatter plots*) sejam comumente usados para variáveis numéricas, eles também são úteis para visualizar a distribuição de valores em relação a variáveis categóricas, principalmente em grandes conjuntos de dados ou para observar a concentração por categoria. A Figura 6.6 ilustra duas formas de visualizar a dispersão da altura por sexo ("feminino" e "masculino").



(a) sem deslocamentos aleatórios (b) com deslocamentos aleatórios

Figura 6.6: Visualização da dispersão de dados por categoria através de gráficos de dispersão. (Fonte)

Para visualizar as peculiaridades de cada medida de dispersão, é recomendado o uso de gráficos específicos, que destacam diferentes aspectos da variabilidade dos dados. As principais medidas de dispersão incluem:

Amplitude: É a diferença entre o maior e o menor valor do conjunto X. É uma medida simples de dispersão, mas sensível a *outliers*.

$$amplitude = max(X) - min(X). \tag{6.3}$$

Essa medida de dispersão pode ser facilmente identificada tanto no **gráfico de distribuição de frequência** quanto no de frequência acumulada.

Quartis e Percentis: Os quantis[2] dividem os dados ordenados, respectivamente, em quatro e 100 partes iguais, ajudando no entendimento da distribuição dos dados. A Figura 6.5 demonstra que **gráficos de distribuição acumulada de frequência**, além de explicitar a distribuição cumulativa, permitem avaliar a posição relativa dos dados em termos de percentis específicos.

Intervalo interquartil (em inglês, *InterQuartile Range*): Corresponde à diferença entre o terceiro quartil (Q_3) e o primeiro quartil (Q_1) em uma distribuição ordenada. Ele é menos sensível a *outliers* do que a amplitude.

$$IQR = Q_3 - Q_1. \tag{6.4}$$

Podemos usar o gráfico de distribuição acumulada para identificar visualmente os quartis Q_1 e Q_3 e, em seguida, calcular o IQR com base nesses valores. No entanto, existe o **gráfico de caixa** (em inglês, *boxplots*) (Figura 6.7) que fornece diretamente informações sobre os quartis e, consequentemente, sobre o intervalo interquartil (IQR).

Resumo de 5 números (amplitude, quantis, quartis, percentis e intervalo interquartil): Consiste de um resumo contendo a mediana (Q_2), os quartis Q_1 e Q_3 , e as menores e maiores observações individuais, dispostas na ordem de Mínimo, Q_1 , Mediana, Q_3, Máximo. Os gráficos de caixa incorporam o resumo de cinco números da seguinte forma (Figura 6.7): (1) as extremidades da caixa estão nos quartis, de modo que o comprimento da caixa é o intervalo interquartil, (2) a mediana é marcada por uma linha dentro da caixa, e (3) duas linhas fora da caixa (*whiskers*, em inglês) se estendem até a menor e maior observações. Além de fornecer uma representação visual compacta e informativa da distribuição de um conjunto de dados e de seus principais pontos estatísticos, esses gráficos são uma das melhores ferramentas para realizar análises comparativas entre diferentes distribuições de uma mesma variável. A Figura 6.7 ilustra o uso de gráficos de caixa para representar os preços unitários dos itens vendidos em quatro filiais. Ao analisar os gráficos, é imediatamente perceptível que 50% dos produtos vendidos nas filiais 1, 2, 3 e 4 possuem preços unitários iguais ou inferiores a US$80, US$100, US$140 e US$80, respectivamente. A variação dos preços é maior na filial 3, enquanto na filial 1 essa variação é pequena.

Variância (σ^2): É a média dos quadrados das diferenças entre cada valor x_i da população

[2]Os **quantis** são pontos específicos que dividem uma distribuição de dados ordenados em partes iguais.

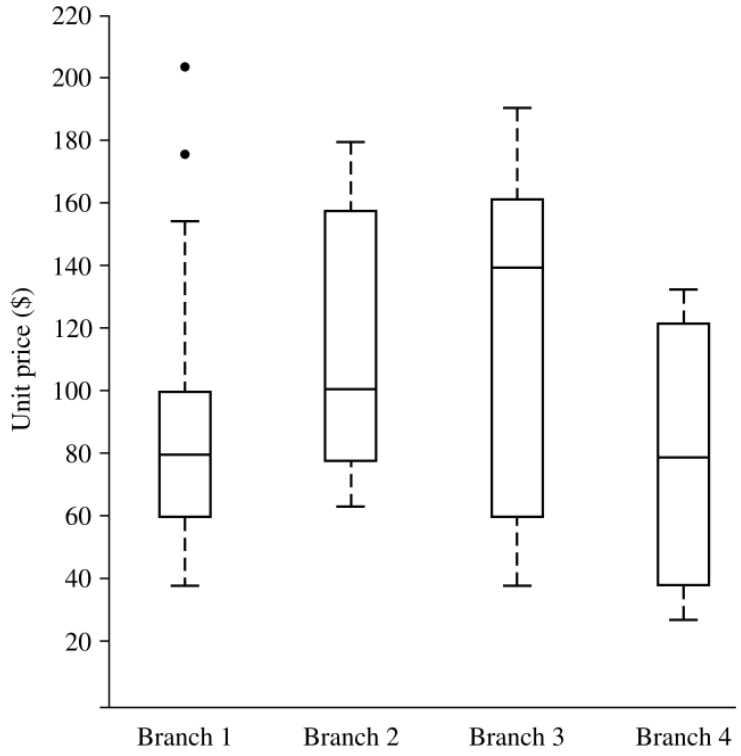


Figura 6.7: Gráficos de caixa representando os preços unitários para itens vendidos em quatro filiais de uma loja *on-line* durante um período de tempo. (Fonte: [31])

de tamanho N e a média desta população.

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2. \tag{6.5}$$

Não há um gráfico específico do qual se leia diretamente a variância dos valores de uma variável. A variância é uma medida de dispersão que requer cálculos específicos para ser determinada. No entanto, gráficos como histograma, gráficos de caixa e gráfico de dispersão podem dar uma ideia visual clara de como os dados estão distribuídos e quão dispersos estão em relação à média ou mediana. Quanto mais espalhados os dados, maior a variância. Combinando as informações do gráfico de caixa com a densidade da distribuição do histograma, como ilustra a Figura 6.8, os **gráficos de violino** tem se mostrado uma excelente opção para explorar a variância de maneira visual.

A forma do "violino" representa a densidade dos dados em diferentes valores. Uma forma mais larga na direção perpendicular ao eixo do violino em uma determinada região indica maior densidade e, consequentemente, maior concentração de valores naquela faixa. A largura geral do violino ao longo do eixo do violino e sua forma fornecem uma indicação visual da dispersão dos dados, que está relacionada à variabilidade dos dados e, indiretamente, à variância.

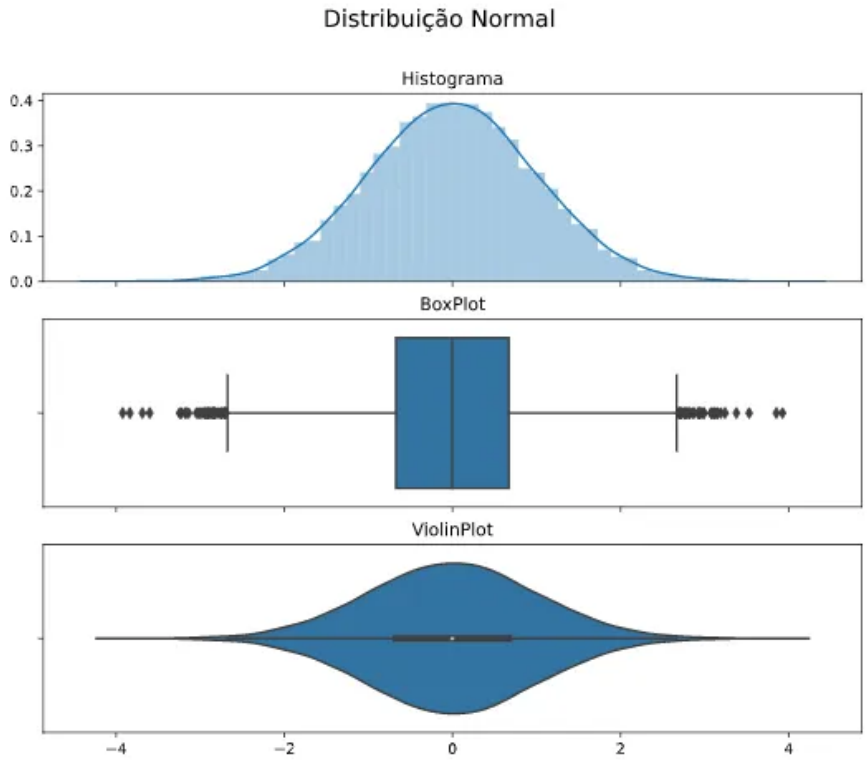


Figura 6.8: Gráficos de violino combinam as informações do gráfico de caixa com a densidade da distribuição do histograma.(Fonte)

Desvio-padrão (σ): É a raiz quadrada da variância e fornece uma medida mais intuitiva da dispersão dos dados, pois está na mesma escala que os próprios dados:

$$\sigma = \sqrt{\sigma^2} \tag{6.6}$$

Por estar na mesma escala dos dados, o desvio padrão é uma medida de dispersão que pode ser visualizada de forma mais intuitiva no histograma, conforme ilustrado na Figura 6.9. Em uma distribuição normal, aproximadamente 68,3% dos valores da variável se encontram dentro de um desvio padrão da média.

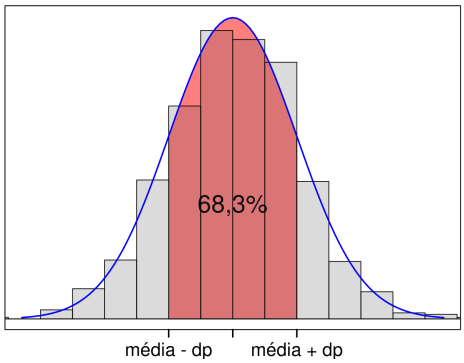


Figura 6.9: Desvio-padrão: uma medida de dispersão dos dados em relação à média numa distribuição de frequência normal, indicando o quanto os valores tendem a se afastar da média.

Além disso, o **gráfico de barras de erro** permite a representação visual direta

do desvio padrão, facilitando a comparação da dispersão entre diferentes grupos ou pontos de dados. Um **gráfico de barras de média com barras de erro**, frequentemente conhecido informalmente como **gráfico de dinamite** (em inglês, *dynamite plots*), é uma representação visual que exibe as médias de conjunto de dados e a variabilidade ou incerteza de cada conjunto. Ele consiste em barras que representam os valores médios de diferentes grupos ou categorias, e "barras de erro" (linhas verticais ou horizontais) que se estendem acima e abaixo (ou para os lados) de cada barra, como ilustra a Figura 6.10.

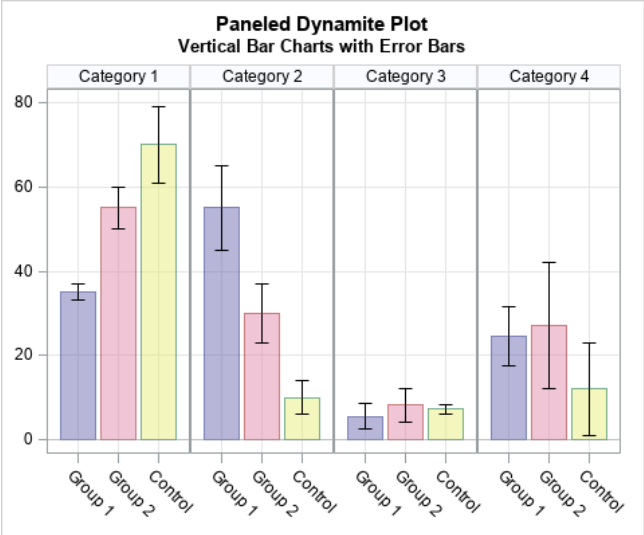


Figura 6.10: Gráficos de barras de média (coloridas) com barras de erro que são as linhas verticais representando as extensões dos "erros" (ou desvios). (Fonte)

Coeficiente de dispersão: Descreve a variabilidade ou a dispersão dos dados em torno da média. Ele fornece uma ideia de quanto os dados estão espalhados ou dispersos. É adimensional e proporciona uma maneira normalizada de avaliar a dispersão em relação à média. Esta medida é útil para comparar a variabilidade relativa entre diferentes conjuntos de dados, especialmente quando as unidades de medida ou as escalas dos conjuntos de dados podem ser diferentes. Um dos coeficientes de dispersão mais comuns é o **coeficiente de variação** (cv):

$$cv = \frac{\sigma}{\mu}. \tag{6.7}$$

Embora não haja um gráfico específico para o coeficiente de dispersão, a dispersão relativa dos dados pode ser visualizada em diversos tipos de gráficos. Opções gráficas incluem gráficos de barras com barras de erro representando o coeficiente de variação, gráficos de caixa para comparação intercategorias da dispersão, gráficos de dispersão com codificação de cor/tamanho dos pontos para o coeficiente de variação por categoria e gráficos de

violino. A seleção do gráfico apropriado será determinada pelo tipo de dados, pelo número de categorias a serem comparadas e pelos *insights* que se deseja extrair.

Quando estamos analisando dados estatísticos, é comum querermos saber onde um valor específico se encaixa em relação à média da população. Para isso, usamos o **escore-z**, ou **escore padrão**. Ele nos diz, em termos de desvios padrão, quão distante um valor x está da média da população (μ). Basicamente, o escore-z nos dá uma ideia de onde esse valor se posiciona na distribuição dos dados:

$$z = \frac{x - \mu}{\sigma}. \tag{6.8}$$

6.1.4 Verificação da Normalidade

Agora que já conhecemos a diversidade de distribuição de frequência que os dados podem apresentar e as ferramentas para descrever suas características centrais e sua variabilidade, sabemos que a análise da distribuição de um conjunto de dados pode ser realizada através de estatísticas resumidas, como a média e o desvio padrão, ou através de visualizações de dados, como gráficos de barras, histogramas e gráficos de densidade. Ambas as abordagens têm suas vantagens e desvantagens, tornando-as complementares na análise de dados.

As estatísticas resumidas oferecem uma maneira concisa de descrever os dados usando poucos números. No entanto, estas estatísticas podem ocultar características importantes da distribuição, como assimetria, múltiplos picos e a presença de *outliers*, sendo a média especialmente sensível a estes últimos. Por outro lado, as visualizações de dados fornecem uma representação intuitiva da forma da distribuição, revelando facilmente características como centro, dispersão, simetria, multimodalidade e *outliers*. Em muitos cenários, a combinação de ambas as abordagens é ideal, a menos a distribuição de frequência de uma variável que se aproxima da curva normal ou Gaussiana, conforme ilustrado na Figura 6.2b.

Uma das propriedades marcantes da distribuição normal é sua definição completa por apenas duas estatísticas resumidas: a média (posição central) e o desvio-padrão (dispersão). Essa simplicidade é o motivo de sua grande relevância e aplicação prática na análise estatística. Por isso, muitos métodos estatísticos e modelos de inferência partem da premissa de que as proporções em intervalos podem ser estimadas usando apenas a média e o desvio-padrão, ou seja, assumindo a normalidade dos dados. No entanto, essa condição nem sempre é atendida. Assim, a verificação da normalidade é necessária antes de prosseguir com a análise.

Uma ferramenta prática para essa verificação é o **gráfico quantil-quantil**, ou Q-Q *plot*. Ele compara os quantis dos dados observados (eixo vertical) com os quantis esperados de uma distribuição normal (eixo horizontal). Se os dados seguem a normal, os pontos plotados devem formar uma linha diagonal, conforme ilustrado na Figura 6.11. A proximidade à diagonal sugere normalidade. Caso contrário, outras distribuições ou transformações devem ser consideradas.

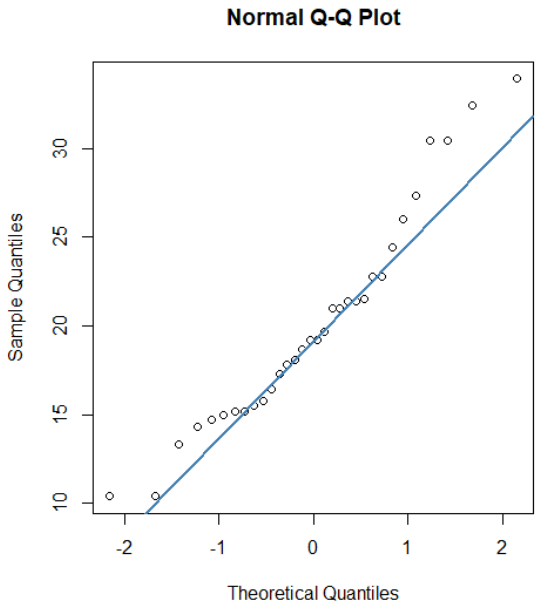


Figura 6.11: Gráfico quantil-quantil em que uma distribuição de frequência no eixo vertical é comparada com a distribuição de frequência normal no eixo horizontal. (Fonte)

6.1.5 Programação

Na biblioteca de visualização de dados, ggplot2 (R) ou plotnine (Python), as estatísticas resumidas são fundamentais para construir certos tipos de gráficos, como *boxplots*, gráficos de dispersão, e histogramas. Para entender como as estatísticas são tratadas, precisamos distinguir entre aquelas que são computadas automaticamente, as que precisam ser informadas explicitamente para a camada de estatísticas (Figura 3.11) e as que exigem manipulação manual antes de passar para o ggplot2.

O ggplot2 simplifica a análise de dados ao calcular automaticamente diversas estatísticas a partir dos dados fornecidos, sem a necessidade de intervenção manual do programador. Essa funcionalidade é implementada através da camada de estatísticas, que aplica funções de agregação e cálculo como médias e medianas. Por exemplo, ao utilizar geom_histogram(), o ggplot2 calcula automaticamente a distribuição e a frequência dos dados em diferentes intervalos. Da mesma forma, para geom_boxplot(), ele computa automaticamente os quartis (Q1, Q3), a mediana e identifica valores atípicos (*outliers*).

Em situações onde estatísticas mais específicas ou personalizadas são necessárias, que não são calculadas automaticamente, o usuário pode informá-las diretamente na camada de Estatísticas (ou stat_*) do ggplot2, como acontece com geom_smooth(), que utiliza um modelo de regressão subjacente para ajustar curvas aos dados. No entanto, para algumas estatísticas, o cálculo e a manipulação dos dados precisam ser realizados manualmente antes de serem passados para o ggplot2. Isso ocorre quando as estatísticas desejadas não podem ser facilmente derivadas pelas camadas do ggplot2 ou quando um cálculo mais complexo e específico é requerido.

Os pacotes dplyr em R e pandas e numpy em Python oferecem diversas funções de sumarização para resumir conjuntos de valores de uma variável, tanto para o conjunto de dados completo quanto

para grupos específicos. Isso facilita a implementação de análises com funções personalizadas. No dplyr, a função summarize() é utilizada para essa finalidade, enquanto em numpy, a função aggregate() desempenha um papel similar. Essas funções abrangem cálculos como média, mediana, soma e desvio padrão, conforme ilustrado na Tabela 6.1.

Tabela 6.1: Funções de resumo mais populares para uma variável X de uma população (NA = *Not aAailable*).

Função	R (summarize)	Python (aggregate)
Contagem total	n(X)	(X, 'size')
Contagem total sem NA	sum(!is.na(X))	X.isnull().sum()
Contagem total de valores distintos	n_distinct(X)	(X, 'nunique')
Média	mean(X)	('X', mean)
Mediana	median(X)	('X', median)
Quantil	quantile(X)	('X', quantile)
Mínimo	min(X)	('X', min)
Máximo	max(X)	('X', max)
Primeiro elemento	first(X)	('X', 'first')
n-ésimo elemento	nth(X)	('X', 'nth')
último elemento	last(X)	('X', 'last')
Desvio padrão	sd(X)	('X', std)
Intervalo interquartil	IQR(X)	('X', stats.iqr)
Desvio absoluto da mediana	mad(X)	('X', stats.median_absolute_deviation)

Como um exemplo de uso de camada de estatísticas, temos a construção de gráficos como o gráfico de distribuição acumulada e o gráfico quantil-quantil. Na primeira construção, é preciso calcular a distribuição acumulada ou os quantis dos dados. Esse cálculo pode ser simplificado com as camadas de **Estatísticas** em ggplot() (R) ou plotnine (Python), que aplicam funções estatísticas diretamente (Figura 3.11). Em R, a função stat_ecdf() [29] facilita a criação de gráficos de distribuição acumulada, e em Python, stat_ecdf() em plotnine cumpre a mesma função [64]. Para a segunda construção, utilizamos a função stat_qq() em R [28] ou a versão correspondente em Python [65].

Exemplos de cálculos estatísticos que exigem manipulação manual antes de serem visualizados no ggplot2 incluem a visualização do coeficiente de variação (CV) e a média do logaritmo natural do tempo de resposta. No primeiro caso, o ggplot2 não possui uma função direta na camada de estatísticas (stat_*) para calcular e exibir o CV nas barras de erro; portanto, essa estatística precisa ser calculada externamente e seus valores fornecidos para a camada gráfica. No segundo exemplo, embora o ggplot2 calcule a média dos dados brutos, não há uma função direta para calcular e visualizar a média de uma transformação específica, como o logaritmo natural. Assim, a transformação e o cálculo da média devem ser realizados fora do ggplot2, com os resultados transformados sendo utilizados na criação do gráfico.

Uma vez que os dados estejam preparados, a visualização pode servir como uma ferramenta poderosa para explorar padrões, tendências e relações entre variáveis. Quanto à criação de gráficos, tanto o pacote ggplot2 em R quanto o plotnine em Python oferecem uma ampla gama de **geometrias**

para mapeamento de dados [30]. As geometrias incluem pontos (geom_point (), geom_jitter ()), barras (geom_bar(), geom_histogram() e geom_freqpoly) e linhas (geom_hline(), geom_vline() e geom_path), entre outras, que são utilizadas conforme o tipo de análise e o formato dos dados.

6.2 Medidas de Relação

A estatística conjunta de múltiplas variáveis é essencial na análise estatística, explorando a relação simultânea entre duas ou mais variáveis. Quando lidamos com duas variáveis, isso envolve a construção e interpretação de uma tabela de dupla entrada, conhecida como **tabela de frequência conjunta**. Esta tabela contém informações sobre a ocorrência conjunta de todas as possíveis combinações dos valores das variáveis, proporcionando uma visão abrangente da distribuição conjunta.

A Figura 6.12 ilustra a tabela de frequência conjunta das variáveis X e Y. A última coluna e a última linha contêm os totais de ocorrências de cada variável, separadamente, e correspondem, respectivamente, às **tabelas marginais de frequência** da variável X e da variável Y. Pela tabela, a frequência relativa da combinação $(\text{não}, 2)$ é $\frac{6}{20}$ e as frequências relativas marginais de X são $\frac{8}{20}$ para sim e $\frac{12}{20}$ para não.

X / Y	1	2	3	total
sim	4	2	2	8
não	5	6	1	12
total	9	8	3	20

Figura 6.12: Tabela de frequência conjunta. (Fonte: [52])

A correlação e a covariância são medidas que proporcionam *insights* sobre a **relação entre as duas variáveis**. A **covariância de Pearson**, em particular, **quantifica** como duas variáveis variam conjuntamente, indicando se elas tendem a aumentar ou diminuir juntas, ou se não apresentam uma relação aparente. Considerando duas variáveis qualitativas, A e B, e uma população de N observações em valores reais $\{(a_1, b_1), \cdots, (a_N, b_N)\}$, as médias de A e B são dadas pelas equações:

$$\mu_A = \frac{\sum_{i=1}^{N} a_i}{N}$$
$$\mu_B = \frac{\sum_{i=1}^{N} b_i}{N}$$

A covariância de Pearson entre A e B é expressa por

$$Cov(A, B) = \frac{1}{N} \sum_{i=1}^{N} (a_i - \mu_A)(b_i - \mu_B). \tag{6.9}$$

Para as duas variáveis A e B que têm a tendência de variar juntos, se um valor $a_i \in A$ é maior do que μ_A e o valor correspondente $b_i \in B$ é também maior doe que μ_B, dizemos então que a covariância entre A e B é positiva. Por outro lado, se uma variável tem a tendência de estar acima de seu valor esperado quando o outro atributo está abaixo de seu valor esperado, então a covariância de A e B é negativa.

No entanto, a covariância de Pearson não é diretamente interpretável, pois depende das escalas das variáveis. Para contornar isso, usa-se a **correlação**, normalizando a covariância para o intervalo $[-1, 1]$. Segue-se a definição do popular **coeficiente de correlação de Pearson**, ou **coeficiente de correlação produto-momento**, como "normalização" da covariância de Pearson pelos desvios-padrão:

$$r_{A,B} = \rho_{A,B} = \frac{Cov(A, B)}{\sigma_A \sigma_B} \tag{6.10}$$

Uma correlação próxima de 1 indica uma relação positiva forte, enquanto uma correlação próxima de -1 sugere uma relação negativa forte.

Para computar esses valores em R, podemos empregar a função padrão cor(). Devemos fornecer como argumentos as duas variáveis para as quais queremos calcular a correlação e, como terceiro argumento, o tipo de correlação desejado [101, 41]. No exemplo abaixo, utilizamos o coeficiente de correlação Pearson para calcular a correlação entre as variáveis V1 e V2 do conjunto mv_normal :

```
cor(mv_normal$V1, mv_normal$V2, method = "pearson")
```

Em Python, a função de correlação é implementada nos pacotes numpy, scipy e pandas [56]. O exemplo que se segue usa o coeficiente de Pearson implementado no pacote pandas:

```
import pandas as pd
x = pd.Series(range(10, 20))
y = pd.Series([2, 1, 4, 5, 8, 12, 18, 25, 96, 48])
xy = pd.DataFrame({'x-values': x, 'y-values': y})
x.corr(y)                    #correlação entre x e y
corr.matrix = xy.corr()      #correlação entre todas as combinações x e y.
```

A melhor maneira de visualizar a relação entre duas variáveis, subjacente à covariação ou correlação de Pearson, é por meio de **gráficos de dispersão**, onde cada ponto representa um par de observações das duas variáveis. A posição do ponto é determinada pelos valores das variáveis correspondentes. Se os pontos no gráfico têm uma tendência geral ascendente da esquerda para a direita, isso sugere uma correlação positiva. Se os pontos têm uma tendência geral descendente da esquerda para a direita, indica uma correlação negativa. Se esses pontos estiverem aleatoriamente espalhados, sugere uma correlação fraca ou inexistente, como ilustra a Figura 6.13. Se os pontos estiverem concentrados em torno de um padrão de curva, a forma da curva pode sugerir um tipo específico de função que governa a relação entre as duas variáveis.

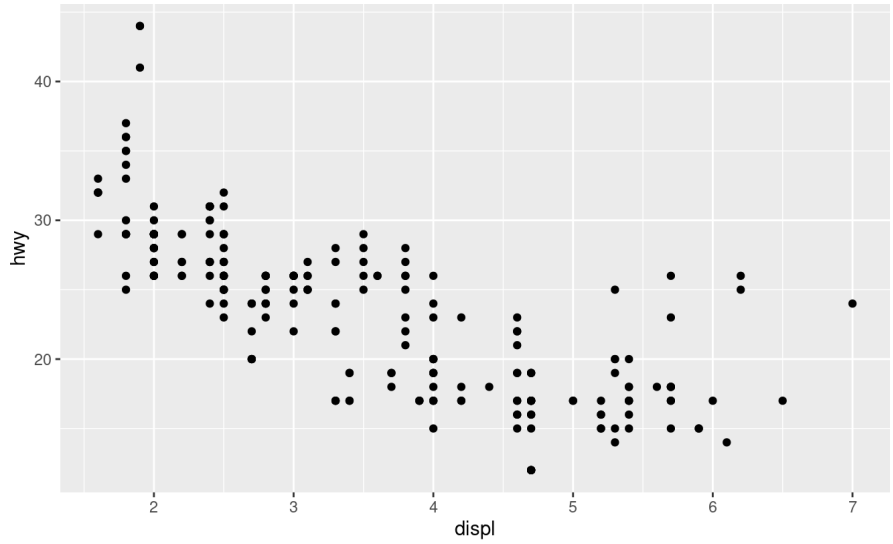


Figura 6.13: Gráfico de dispersão representado a relação entre o tamanho do motor (displ) e a eficiência de combustível (hwy). (Fonte: [101]

Tanto em R quanto em Python, uma maneira de visualizar esses gráficos de dispersão é empregar a função ggplot() disponível nos pacotes ggplot2 (R) ou plotnine (Python). O seguinte código exemplifica a visualização da relação entre a largura e o comprimento das pétalas das flores do conjunto de dados iris, incluído no pacote base do R e no pacote plotnine [96]:

```
library(ggplot2)
ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width)) +   #ponto = (Sepal.Length,Sepal.Width)
    geom_point()
```

Os códigos equivalentes em Python são:

```
from plotnine import ggplot, aes, geom_point
from plotnine.data import iris


ggplot(iris, aes(x='Sepal.Length', y='Sepal.Width')) +   #ponto = (Sepal.Length,Sepal.Width)
    geom_point()
```

Para visualizar as relações entre múltiplas variáveis simultaneamente, as **matrizes de gráficos de dispersão** (*scatterplot matrix*) são uma ferramenta eficaz. As bibliotecas de visualização ggplot2 no R e sua equivalente em Python, plotnine, oferecem uma maneira de gerar essas matrizes. Elas utilizam a função facet_wrap para dividir o gráfico em múltiplos painéis (em inglês, *facets*), exibindo os gráficos de dispersão de todos os pares de variáveis em uma única camada. No entanto, as bibliotecas ggally em R (com ggpairs()) e seaborn em Python (com pairplot()) fornecem funções que automatizam essa transformação e a criação da matriz de dispersão de forma mais direta. Elas geram pequenos múltiplos de gráficos de dispersão de duas variáveis para todos os possíveis pares (Seção 3.5.2), acompanhados

de histogramas ou gráficos de densidade para cada variável na diagonal principal da matriz, como ilustra a Figura 6.14.

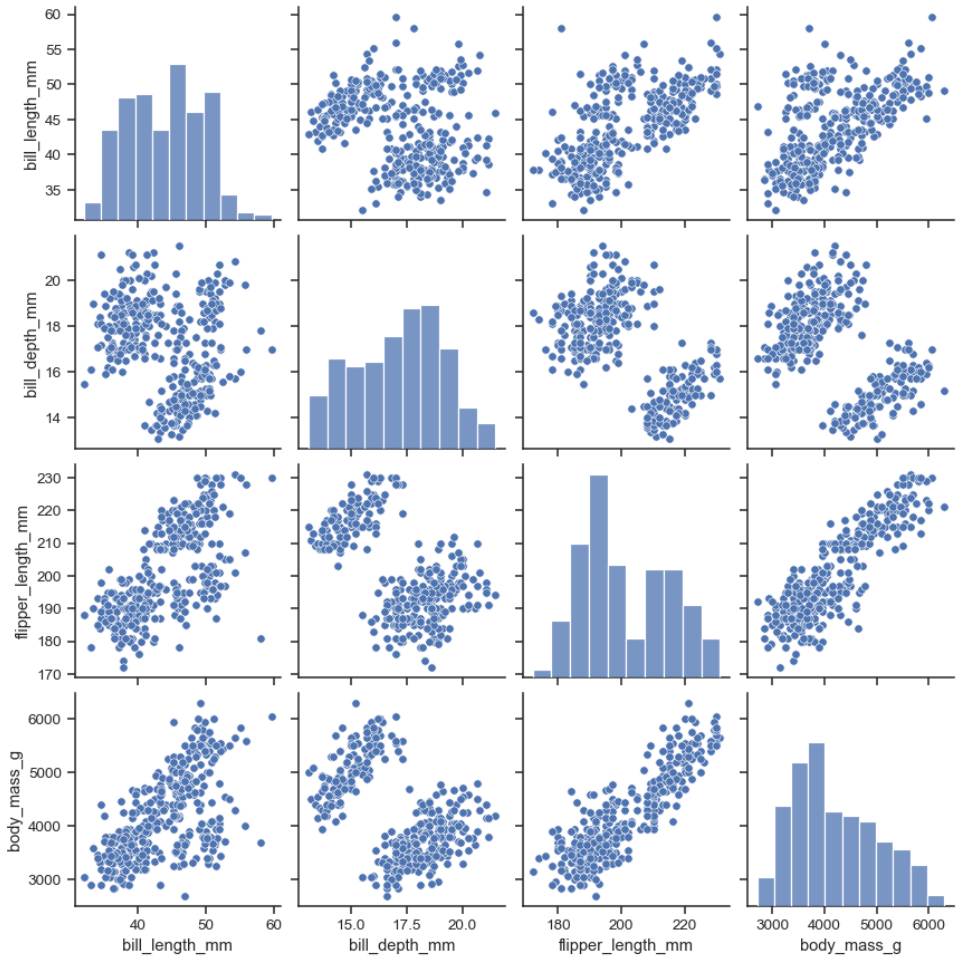


Figura 6.14: Uso de uma matriz de gráficos de dispersões para mostrar as relações entre as quatro características do conjunto de dados pinguins do pacote seaborn: comprimento e profundidade do bico e comprimento e massa da nadadeira. (Fonte: [83])

Para visualizar as correlações entre todas as variáveis de uma população, construímos uma **matriz de correlação**, onde cada elemento representa a correlação do par de variáveis (i, j), com i e j indicando linha e coluna. Essa matriz pode ser representada graficamente por geom_tile() (R) ou geom_rect() (Python), colorindo cada célula de acordo com o valor de correlação, como ilustra a Figura 6.15. Isso permite uma visão abrangente das relações entre as variáveis.

6.3 Similaridade

A Seção 6.1.2 introduziu as medidas resumo calculadas para os valores de uma variável quantitativa em um experimento, enquanto a Seção 6.2 explora técnicas relacionadas a duas variáveis quantitativas de uma população. Em contraste com as medidas resumo e de covariância/correlação, que fornecem resumos estatísticos ou avaliam relações lineares entre duas variáveis específicas, as **medidas de similaridade** são abordagens mais abrangentes, levando em consideração todas as variáveis, ou

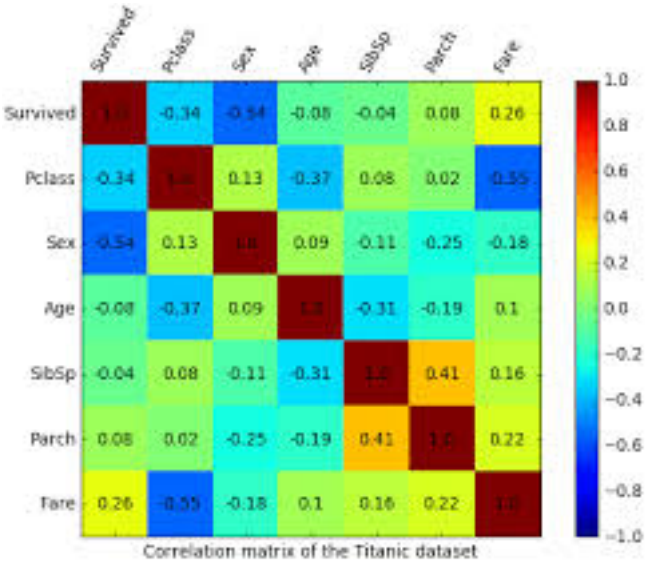


Figura 6.15: Matriz de correlação das variáveis observadas no conjunto de dados do Titanic.

características, quantitativas presentes nas populações. Um conjunto de características é denominado um **vetor de características**. Essas medidas visam capturar a semelhança entre n observações com p variáveis, onde a i-ésima observação é representada por $x_i = (x_{i1}, x_{i2}, ..., x_{ip})$.

A medida de similaridade é geralmente expressa como um valor numérico, aumentando à medida que as amostras de dados se tornam mais semelhantes. Frequentemente, é representada por um número entre 0 (baixa similaridade) e 1 (alta similaridade). Diversas técnicas para calcular similaridade operam sobre uma **tabela relacional**, onde as linhas e as colunas correspondem, respectivamente, às observações/vetores de observações e variáveis/características. Essa tabela é também conhecida por **matriz de dados**, e os resultados do cálculo de dissimilaridade, par a par, são geralmente organizados em uma estrutura conhecida como **matriz de dissimilaridade**, onde cada elemento $d(i, j)$ é a **medida de dissimilaridade**, ou diferença, entre as observações i e j. É importante notar que $d(i, j) = d(j, i)$, e que a **medida de similaridade** é o complemento da medida de dissimilaridade:

$$sim(i, j) = 1 - d(i, j) \tag{6.11}$$

Tipicamente, o valor numérico que representa uma medida de similaridade corresponde à distância entre cada par de observações. Este cálculo é viabilizado ao modelar os vetores de características como vetores-posição multidimensionais, em que cada componente do vetor representa o valor de uma variável específica. Essa abordagem permite a utilização de uma **métrica de distância** no espaço vetorial, a cujos vetores-posição são atribuídas os vetores de características. Por exemplo, se estivermos trabalhando com uma população de duas variáveis, cada vetor de características será representada por um ponto em um plano cartesiano, onde as coordenadas do ponto correspondem aos valores das duas variáveis de uma observação da população. No contexto de dados tridimensionais, cada observação seria representada como um ponto num espaço tridimensional.

Considere um espaço métrico (M, d) onde M é o espaço dos vetores e d é uma métrica de distância aplicada em M. Para que a métrica $d : M \times M \longrightarrow \Re$ seja válida, ela deve satisfazer três propriedades fundamentais:

Não-negatividade: A distância entre dois pontos é sempre não-negativa, ou seja,

$$d(x, y) \geq 0,$$

onde $d(x, y) = 0$, se e somente se, $x = y$.

Simetria: A distância entre dois pontos é igual, independentemente da ordem em que esses pontos são considerados, isto é,

$$d(x, y) = d(y, x)$$

Desigualdade Triangular: A distância entre dois pontos e o comprimento do caminho mais curto entre eles é sempre menor ou igual à soma das distâncias entre os pontos e um terceiro ponto arbitrário.

$$d(x, z) \leq d(x, y) + d(y, z)$$

Considere dois pontos correspondentes às observações com n variáveis, $P = (p_1, p_2, \cdots, p_n)$ e $Q = (q_1, q_2, \cdots, q_n)$. Seguem-se algumas das métricas de distância mais aplicadas no cômputo da distância entre dois vetores de características [53]:

Distância Euclideana: É a medida da menor distância entre dois pontos em um espaço. Ela é frequentemente utilizada em problemas onde a geometria do espaço é essencial.

$$d(P, Q) = \parallel P - Q \parallel = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{6.12}$$

Distância Manhattan: É também conhecida como distância de cidade, pois é uma métrica que considera apenas movimentos horizontais e verticais entre pontos. Ela é apropriada para espaços onde os movimentos são restritos a uma grade.

$$d(P, Q) = \parallel P - Q \parallel = \sum_{i=1}^{n} |p_i - q_i| \tag{6.13}$$

Distância Minkowski: É uma generalização que inclui tanto a distância euclidiana quanto

a de Manhattan. É definida como:

$$d(P,Q) = \parallel P_i - Q_i \parallel_m = (\sum_{i=1}^{n} |p_i - q_i|^m)^{\frac{1}{m}}. \tag{6.14}$$

A distância euclidiana é um caso especial quando m=2 e a distância de Manhattan é um caso especial quando m=1.

As medidas de similaridade têm aplicações em uma variedade de contextos. No agrupamento de dados (em inglês, *clustering*), elas são essenciais para agrupar observações semelhantes em *clusters* distintos, facilitando a organização e compreensão dos dados. Em sistemas de recomendação, essas medidas desempenham um papel fundamental ao identificar usuários ou itens semelhantes, contribuindo para recomendações personalizadas e precisas. Além disso, na análise de padrões e detecção de anomalias, as medidas de similaridade são de extrema importância, pois oferecem uma abordagem sistemática para comparar observações com base em suas características, permitindo a identificação de padrões ocultos e a detecção de observações anômalas.

Tanto R quanto Python oferecem suporte para o cálculo de matrizes de similaridade entre observações. Os dados são geralmente representados em uma estrutura tabular, onde as linhas correspondem às observações e as colunas representam as diferentes variáveis ou características.

No ambiente R, pode-se usar o pacote proxy para calcular matrizes de dissimilaridade ou similaridade. Abaixo está um exemplo simples usando a distância Euclidiana:

```
# Instale o pacote se ainda não o tiver
# install.packages("proxy")

library(proxy)

# Crie um exemplo de dados
dados <- matrix(rnorm(100), ncol = 5)

# Calcule a matriz de dissimilaridade Euclidiana, depois de converter dados para o tipo matrix
matriz_dissimilaridade <- proxy::dist(as.matrix(dados))
matriz_similaridade <- 1 / (1 + matriz_dissimilaridade)
```

Em Python, podemos utilizar a função pairwise_distances() do módulo metrics do pacote sklearn para computar as matrizes de similaridade e dissimilaridade, como demonstra o seguinte trecho de códigos que usou a métrica "euclideana":

```
%pip install sklearn  # se não estiver instalado
```

```
import numpy as np
import sklearn
from sklearn.metrics import pairwise_distances
from sklearn.preprocessing import MinMaxScaler

# Crie um exemplo de dados, contendo 20 observações com 5 variáveis
dados = np.random.randn(20, 5)

# Calcule a matriz de dissimilaridade Euclidiana
matriz_dissimilaridade = pairwise_distances(dados, metric='euclidean')
normalizador = MinMaxScaler(feature_range=(0, 1))
matriz_dissimilaridade_normalizada = normalizador.fit_transform(matriz_dissimilaridade)
matriz_similaridade_normalizada = 1-matriz_dissimilaridade_normalizada
```

Note que foi usada a função normalizadora MinMaxScaler() do módulo preprocessing para normalizar as distâncias no intervalo [0,1].

Os gráficos mais comuns para visualizar a similaridade entre dados incluem a matriz de dissimilaridade, o mapa de calor e o gráfico de dispersão. A **matriz de dissimilaridade** mostra como cada observação se compara a todas as outras em termos de distância ou similaridade, sendo útil para analisar as relações entre os dados de maneira geral. Para tornar essa visualização mais intuitiva, o **mapa de calor** pode ser utilizado, onde a matriz é representada por uma grade de cores, com as tonalidades mais escuras indicando maior similaridade ou menor dissimilaridade, o que facilita a identificação de padrões de proximidade entre os dados. Já o **gráfico de dispersão** pode ser usado quando se deseja representar dados em duas ou três dimensões, permitindo que a proximidade entre os pontos mostre visualmente a similaridade, sendo eficaz quando o número de variáveis é reduzido. Nesse caso, técnicas de redução de dimensionalidade, como a Análise de Componentes Principais (em inglês, *Principal Component Analysis*) [5, 115], são muito utilizadas para projetar dados de alta dimensão em um espaço bidimensional ou tridimensional constituído pelas primeiras duas ou três componentes, preservando ao máximo possível a estrutura e a proximidade relativa dos pontos originais.

6.4 Clusterização

Um **grupo** (em inglês, *cluster*) é uma coleção de objetos de dados que compartilham semelhanças entre si dentro do próprio grupo, mas são diferentes de objetos em outros grupos [32]. Mesmo que esse grupo não tenha sido explicitamente definido ou rotulado, a similaridade intrínseca entre os objetos de um grupo sugere implicitamente uma classe subjacente. Nesse sentido, a **clusterização**, ou agrupamento, é às vezes chamado de **classificação automática** ou **classificação não-supervisionada**.

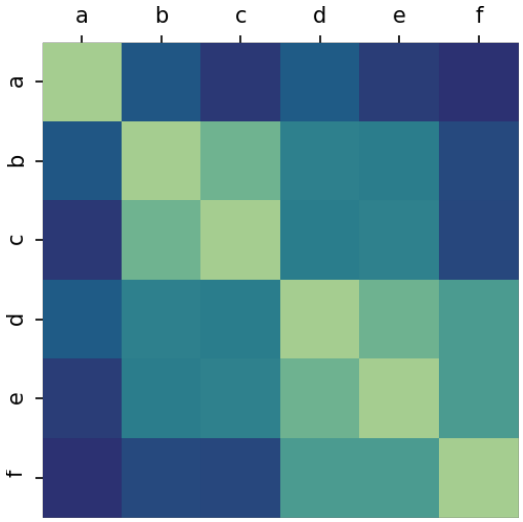


Figura 6.16: Mapa de calor para visualizar as dissimilaridades entre os pares de observações (i,j), onde i representa a linha e j representa a coluna de uma matriz de dissimilaridades. As dissimilaridades são codificadas em cores. (Fonte)

No âmbito de aprendizado de máquina, a clusterização é considerada uma forma de **aprendizado por observação**, em contraste com o aprendizado por exemplos. A análise de *clusters* visa identificar padrões e similaridades inerentes aos dados. Por essa razão, a clustrização também é denominada **segmentação de dados** em algumas aplicações, pois particiona grandes conjuntos de dados em grupos de acordo com sua similaridade. Aplicações de **detecção de valores discrepantes** também podem se beneficiar da análise de *cluster* para distinguir *outliers*.

No contexto da estatística descritiva, a clusterização se destaca como uma ferramenta para explorar e descrever a estrutura subjacente dos dados. Ao agrupar observações similares, permite uma representação mais concisa e interpretável de um conjunto de dados, facilitando a compreensão de suas características distintivas. Contrariamente à estatística de inferência, que busca fazer inferências sobre populações a partir de amostras, a clusterização está focada na organização e descrição direta dos dados disponíveis.

Almejando uma minimização dos requisitos de conhecimento de domínio a fim de facilitar a aplicação da clusterização em diversas áreas, os algoritmos populares de clusterização são essencialmente baseados na relação espacial entre as observações. Limitando-nos às variáveis quantitativas, podemos adequar os dados a esses algoritmos, mapeando diretamente os vetores de características, correspondentes a todas observações, em vetores-posição. Com isso, conferimos às observações uma informação espacial e uma medida de similaridade baseada numa métrica de distância e organizada em matrizes de similaridade/dissimilaridade como vimos na Seção 6.3. Para esses dados providos de relação espacial, que chamamos de **pontos de dados**, destacam-se os seguintes métodos de clusterização [32]:

Método de Particionamento: Envolve a divisão de um conjunto de dados em um número predeterminado de *clusters*, onde cada ponto de dados pertence a exatamente um *clus-*

ter. O critério de divisão é tipicamente a distância entre os pontos. O **k-means** é um algoritmo de particionamento bem popular, onde os dados são agrupados em k *clusters* com base na minimização da soma dos quadrados das distâncias entre os pontos e o centro de seus *clusters* aos quais esses pontos são atribuídos.

Método Hierárquico : Envolve a construção de dendogramas, um tipo de diagrama de árvore, que representam a hierarquia de *clusters*. Os dados podem ser aglomerados (*bottom-up*) ou divididos (*top-down*) sucessivamente até que todos estejam em um *cluster*. Os dados não precisam ter necessariamente uma estrutura hieráquica. A organizaçãpo hierárquica é apenas para simplificar a sumarização e representação dos dados. Os critérios de proximidade aplicados podem ser por distância, densidade e conectividade entre os pontos. A **clusteriação hierárquica aglomerativa** (em inglês, *Agglomerative Hierarchical Clustering*) é um exemplo desse método, onde os dados são unidos iterativamente com base em sua proximidade.

Método Baseado em Densidade: É considerada a densidade de pontos de dados para formar *clusters*. Pontos em áreas densas são considerados parte do mesmo *cluster*, separados por regiões menos densas. Embora a medida de densidade não seja diretamente uma métrica de distância, ela depende da proximidade dos pontos entre si. A proximidade é um indicativo da densidade local. O DBSCAN (do inglês *Density-Based Spatial Clustering of Applications with Noise*) é um método baseado em densidade, identificando *clusters* como regiões densas conectadas.

Método Baseado em Grade: O espaço de dados é organizado em uma grade e os pontos de dados são atribuídos a células específicas. As células com um grande número de pontos indicam a presença de *clusters*. CLARA (do inglês *Clustering Large Applications*) é uma implementação de clusterização baseada em grade.

A principal força da clusterização reside na sua capacidade de revelar a estrutura dos dados através da visualização em grupos distintos. Em dados bidimensionais ou tridimensionais, **gráficos de dispersão** são uma ferramenta excelente para observar a formação de *clusters*, utilizando tamanho ou cor para identificar a que grupo cada ponto pertence. No caso da clusterização hierárquica, os **dendrogramas**, mostrados na Figura 6.17, são indicados para visualizar a organização dos dados em diferentes níveis de granularidade.

Segue uma apresentação dos quatro algoritmos de clusterização, juntamente com suas implementações em R e Python. Também exploramos uma abordagem visual para facilitar a compreensão dos agrupamentos através de **mapas de calor**.

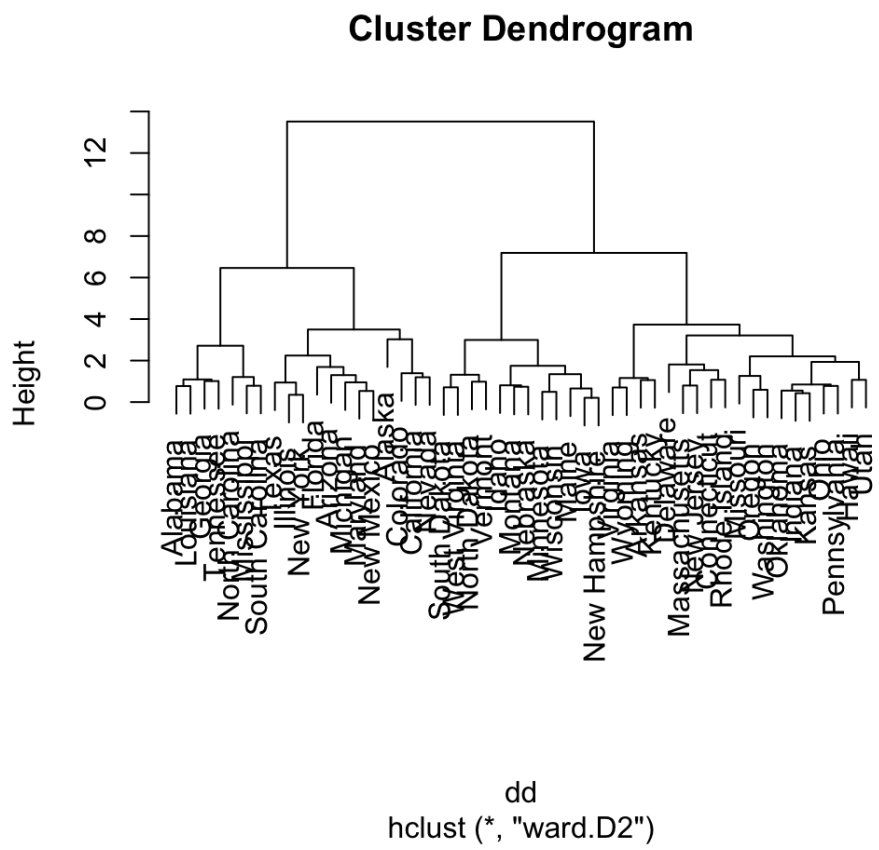


Figura 6.17: Dendrograma do conjunto de dados USArrests que contém estatísticas, em número de prisões por 100.000 residentes, para agressão, assassinato e estupro em cada um dos 50 estados dos EUA em 1973. (Fonte)

6.4.1 K-means

O **k-means** é um algoritmo de clusterização que visa particionar um conjunto de dados em k *clusters*, onde cada ponto de dado pertence ao *cluster* cujo centróide é o mais próximo. O procedimento ocorre da seguinte maneira a partir da definição de uma quantidade k de *clusters*. Primeiro, seleciona-se aleatoramente k pontos de dados como os centróides iniciais dos *clusters*. O algoritmo *k-means*, então, melhora iterativamente a variação dentro dos *clusters*. Para cada ponto de dado no conjunto, atribua-o ao *cluster* cujo centróide é o mais próximo. Isso é feito com base numa métrica de distância, tipicamente euclidiana ou Manhattan, entre o ponto e os centróides. Atualize-se os centróides c_j da iteração anterior para cada novo *cluster* S_j com m_j pontos, tomando a média dos pontos x_{ji} atribuídos a esse *cluster*. O novo centróide c_j é calculado como:

$$c_j = \frac{1}{|S_j|} \sum_{i=1}^{m_j} x_{ji}$$

As iterações continuam até que um número máximo de iterações seja atingido ou que não haja mais mudanças nas atribuições dos *clusters*. Note que as novas atribuições não devem aumentar a soma dos quadrados das distâncias entre os pontos de dados e os centróides de seus *clusters* atribuídos,

ou seja, aumentar variações intra-*cluster*,

$$J = \sum_{j=1}^{k} \sum_{i=1}^{m_j} \omega_{i,j} \cdot d(x_{ji}, c_j)^2, \tag{6.15}$$

onde $\omega_{i,j}$ é uma variável indicadora que é 1 se x_{ji} está no *cluster* j e 0 caso contrário. O resultado é uma partição dos dados em k *clusters*, cada um representado pelo seu centróide. O método k-means não garante a convergência para a melhor solução global e, muitas vezes, termina em um ótimo local. Os resultados podem ser influenciados pela escolha inicial aleatória dos centros dos *clusters*. Para obter resultados mais confiáveis, é comum na prática aplicar o algoritmo *k-means* várias vezes com diferentes escolhas iniciais para os centros dos *clusters*. A clusterização que resulta na menor variação dentro dos *clusters* é então considerado como o resultado final.

O pacote cluster do R oferece funções para realizar a clusterização por meio do algoritmo *k-means* e visualizar os centróides dos *clusters* detectados, como ilustrado no seguinte trecho de código [75]):

```
install.packages("cluster")
library(cluster)
library(ggplot2)            # renderizar os pontos

# Criar um conjunto de dados fictício
set.seed(123)               # assegurar a reprodutibilidade, gerando sempre a mesma semente
dados <- data.frame(x = rnorm(200), y = rnorm(200))

# Aplicar o algoritmo k-means com k=3
resultado_kmeans <- kmeans(dados[, c("x", "y")], centers = 3)

# Adicionar a informação do cluster aos dados
dados$cluster <- as.factor(resultado_kmeans$cluster)

# Criar um gráfico de dispersão com ggplot2
ggplot(dados, aes(x = x, y = y, color = cluster)) +
  geom_point() +
  geom_point(data = as.data.frame(resultado_kmeans$centers), aes(x = x, y = y), color = "red",
  ggtitle("Resultado do k-means em ggplot2")
```

Segue abaixo um trecho de código equivalente em Python. Observe que é necessário importar os pacotes numpy, sklearn, pandas e plotnine para, respectivamente, realizar manipulação de estruturas de dados matriciais, clusterização por k-means [47], e renderização.

```
from plotnine import ggplot, aes, geom_point, ggtitle
from pandas import DataFrame
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Criar um conjunto de dados fictício
np.random.seed(123)
dados = np.random.randn(200, 2)

# Aplicar o algoritmo k-means com k=3
kmeans = KMeans(n_clusters=3, random_state=123)
resultado_kmeans = kmeans.fit_predict(dados)

# Criar um gráfico de dispersão com plotnine
(ggplot(dados, aes(x='x', y='y', color='cluster')) +
 geom_point() +
 geom_point(data=DataFrame(resultado_kmeans.cluster_centers_, columns=['x', 'y']), aes(x='x',
 ggtitle("Resultado do k-means em plotnine")
)
```

6.4.2 Clusterização Hierárquica Aglomerativa

A **Clusterização Hierárquica Aglomerativa** (CHA) é um método de agrupamento que constrói uma hierarquia de *clusters*. O algoritmo inicia considerando cada observação como um *cluster* individual e, em seguida, combina iterativamente os *clusters* mais similares até que todos estejam agrupados em um único *cluster*. No início, cada observação é considerada um *cluster* independente. O algoritmo CHA mescla os *clusters* iterativamente com base na medida de ligação (em inglês, *linkage measure*) selecionada. Em cada iteração, calcula-se a ligação entre todos os pares de *clusters*. Algumas medida de ligação comumente utilizadas entre dois *clusters* S_j com n_j pontos, x_{ji}, e S_k com n_k pontos, x_{kl}, incluem (Seção 8.3.2 em [32]):

Distância mínima : $dist_{min}(S_j, S_k) = min|x_{ji} - x_{kl}|$

Distância máxima : $dist_{max}(S_j, S_k) = max|x_{ji} - x_{kl}|$

Média das distâncias : $dist_M(S_j, S_k) = |m_j - m_k|$ com $m_j = \frac{\sum x_{ji}}{n_j}$ e $m_k = \frac{\sum x_{kl}}{n_k}$

Distância média : $dist_m(S_j, S_k) = \frac{\sum_{x_{ji} \in S_j, x_{kl} \in S_k} |x_{ji} - x_{kl}|}{n_j n_k}$

A métrica usada para computar as distâncias entre os pontos é tipicamente euclideana. Os *clusters* com menores medidas de ligação são mesclados se essa fusão resultar na minimização da variação intra-*cluster*, promovendo a formação de *clusters* mais coesos. As medidas de ligação entre os *clusters* são atualizadas e o processo é repetido até que todos os pontos de dados estejam em um único *cluster*. Um critério específico para minimizar a variação intra-*cluster* é conhecido como **critério de Ward**. Esse critério avalia a diferença entre as distâncias médias de dois *clusters*, S_j e S_k, combinados, e é expresso por

$$W(S_j, S_k) = \frac{n_j n_k}{n_j + n_k} |m_j - m_k|^2. \tag{6.16}$$

Os **dendrogramas** são comumente utilizados para visualizar a estrutura de agrupamentos hierárquicos e para ajudar na interpretação da similaridade entre diferentes elementos no conjunto de dados. Figura 6.17 ilustra um dendrograma em R que revela a clusterização hierárquica das observações do conjunto USArrests [96].

A técnica CHA é implementada tanto em R quanto em Python. O seguinte trecho de códigos ilustra a clusterização por CHA em R, usando uma variante do critério Ward para monitorar variações intra-*cluster* [74], e o uso do pacote ggdendro para a visualização de dendrogramas dentro do *framework* do ggplot2 em R [63]:

```
# Instale o pacote ggplot2 se ainda não estiver instalado
# install.packages("ggplot2")

# Carregar bibliotecas
library(plotly)
library(ggplot2)
library(ggdendro)

# Criar dados fictícios
set.seed(123)
dados <- matrix(rnorm(30), ncol = 3)

# Calcular a matriz de dissimilaridade
matriz_dissimilaridade <- dist(dados)

# Executar o algoritmo de CHA
modelo_cha <- hclust(matriz_dissimilaridade, method = "ward.D2") # CHA com o critério Ward

# Converter o resultado do CHA para um formato de dendrograma
```

```
dendrograma <- as.dendrogram(modelo_cha)

# Plotar o dendrograma usando ggplot2
p <- ggplot(data = dendrograma, labels = TRUE) +
  geom_segment(aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_text(aes(x = x, y = y, label = label, hjust = 0), vjust = 1) +
  theme_minimal() +
  labs(title = "Dendrograma - CHA usando ggplot2")

ggplotly(p)
```

Segue abaixo um código equivalente usando o pacote scipy [80] e o pacote plotnine em Python. Note que, com o plotnine, pode-se criar dendrogramas diretamente usando sua própria gramática de gráficos, sem precisar de pacotes adicionais:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import linkage, dendrogram

# Criar dados fictícios
np.random.seed(123)
dados = np.random.randn(10, 3)

# Calcular a matriz de dissimilaridade
matriz_dissimilaridade = linkage(dados, method='ward')

# Plotar o dendrograma
(ggplot()
 + geom_segment(aes(x='x', y='y', xend='xend', yend='yend'), data=dendrogram(matriz_dissimilar
 + geom_text(aes(x='x', y='y', label='label'), data=dendrogram(matriz_dissimilaridade), hjust=
 + theme_minimal()
 + labs(title='Dendrograma - CHA usando plotnine')
)
```

6.4.3 DBSCAN

A **densidade** de um objeto é quantificada pelo número de objetos próximos a ele. O algoritmo DBSCAN (do inglês, *Density-Based Spatial Clustering of Applications with Noise*), um método de

clusterização, agrupa pontos de dados com base em sua densidade. Ele identifica objetos principais e suas vizinhanças para formar regiões densas como *clusters*. Similar à CHA, ele é capaz de descobrir *clusters* de diferentes formas e tamanhos, além de identificar *outliers*. Para iniciar uma clusterização com o DBSCAN, é necessário definir a distância máxima, ε, que especifica o raio ao redor de um ponto e o número mínimo MinPts de pontos dentro do raio ε para que um ponto seja considerado como parte de um *cluster*. Inicialmente, todos os pontos são marcados como "não visitado". Para cada ponto x_{ji} não visitado, marque x_{ji} como um *outlier*, se o número de pontos dentro do raio ε ao redor de x_{ji} for menor que MinPts; caso contrário, crie um novo *cluster* e inclua x_{ji} e todos os pontos alcançáveis a partir de x_{ji} dentro do raio ε e com pelo menos MinPts vizinhos. Marque todos esses pontos como visitados e atribua-os ao *cluster*. O procedimento se repete até que todos os pontos sejam visitados.

Segue um exemplo de como utilizar o algoritmo DBSCAN para clusterização em R, demonstrando o uso da função dbscan disponível no pacote dbscan [54]:

```
# Instalar os pacotes necessários
# install.packages("dbscan")
# install.packages("ggplot2")

# Carregar bibliotecas
library(dbscan)
library(ggplot2)

# Gerar dados de exemplo
set.seed(123)
dados <- data.frame(
  x = c(rnorm(50, mean = 0, sd = 1), rnorm(50, mean = 5, sd = 1)),
  y = c(rnorm(50, mean = 0, sd = 1), rnorm(50, mean = 5, sd = 1))
)

# Executar o algoritmo DBSCAN
resultado_dbscan <- dbscan(dados, eps = 1, minPts = 5)

# Adicionar rótulos de clusters ao conjunto de dados original
dados$cluster <- as.factor(resultado_dbscan$cluster)

# Plotar os resultados usando ggplot2
ggplot(dados, aes(x = x, y = y, color = cluster)) +
  geom_point(size = 3) +
```

```
  theme_minimal() +
  labs(title = "DBSCAN Clustering")
```

Segue-se o mesmo exemplo usando a implementação de DBSCAN no pacote sklearn em Python [81]:

```
%pip install scikit-learn matplotlib

from plotnine import ggplot, aes, geom_point, labs
import numpy as np
from sklearn.cluster import DBSCAN
from sklearn.datasets import make_blobs

# Gerar dados de exemplo
np.random.seed(123)
dados, rotulos = make_blobs(n_samples=200, centers=3, cluster_std=1.0, random_state=0)

# Adicionar outliers aos dados de exemplo
outliers = np.array([[8, 6], [10, 5]])
dados = np.concatenate([dados, outliers])

# Executar o algoritmo DBSCAN
modelo_dbscan = DBSCAN(eps=1.0, min_samples=5)
rotulos_dbscan = modelo_dbscan.fit_predict(dados)

# Plotar os resultados usando plotnine
(ggplot(dados, aes(x='x', y='y', color='cluster')) +
 geom_point(size=3) +
 labs(title="DBSCAN Clustering"))
```

6.4.4 CLARA

CLARA (do inglês, *Clustering Large Applications*) foi desenvolvida por Kaufman e Rousseeuw, em 1990 [26]. Ela é uma extensão do método de clusterização PAM (do inglês, *Partitioning Around Medoids*). Foi concebida com o objetivo de reduzir o tempo de computação e o uso de memória em casos de conjuntos de dados extensos. Como quase todos os algoritmos de particionamento, exige que o usuário especifique um número apropriado de *clusters* a serem gerados. A apresentação do algoritmo nesta seção visa a mostrar que ainda não há uma implementação direta deste algoritmo no

pacote scikit-learn em Python, embora seja uma técnica popular em R. Os detalhes do algoritmo foge do escopo desse texto.

O pacote cluster em R é uma implementação da técnica CLARA amplamente utilizada. O seu uso é demonstrado no seguinte trecho de código [73].

```
# Instalar o pacote ggplot2 (se ainda não estiver instalado)
# install.packages("ggplot2")

# Carregar os pacotes necessários
library(cluster)
library(ggplot2)

# Gerar dados de exemplo
set.seed(123)
dados <- matrix(rnorm(200), ncol = 2)

# Executar CLARA
resultado_clara <- clara(dados, k = 3, samples = 5, metric = "euclidean")

# Atribuir clusters ao conjunto de dados original
clusters <- predict(resultado_clara)

# Converter os dados para um data frame
dados_df <- data.frame(x = dados[, 1], y = dados[, 2], cluster = as.factor(clusters$clustering

# Visualizar os resultados com ggplot2
ggplot(dados_df, aes(x = x, y = y, color = cluster)) +
  geom_point(size = 3) +
  geom_point(data = as.data.frame(resultado_clara$centers), aes(x = x, y = y), color = "black"
  labs(title = "CLARA Clustering with ggplot2") +
  theme_minimal()
```

6.4.5 Agrupamentos por Percepçao Visual

Vimos na Seção 4.3 que nossos cérebros são muito eficientes em identificar semelhanças e agrupamentos, mesmo sem cálculos complexos. A Lei de Gestalt sugere que os seres humanos tendem a perceber elementos semelhantes (em cor, forma, tamanho, proximidade, etc.) como pertencentes a um mesmo grupo ou unidade. Quando usamos cores ou coordenadas geométricas de maneira cuidadosa e

consistente em uma visualização (por exemplo, um *heatmap* ou um gráfico de dispersão), pode haver agrupamentos visuais intuitivos, ou seja, os observadores podem perceber grupos de dados como se fossem "*clusters*". A esse tipo de percepção visual chamamos de clusterização visual natural.

A Seção 6.3 demonstra como os mapas de calor (*heatmaps*) são utilizados para visualizar similaridades ou dissimilaridades entre as observações. Cada célula do mapa é colorida de acordo com o valor correspondente de similaridade ou dissimilaridade. Aplicando `geom_tile()` (R) ou `geom_rect()` (Python), essa mesma técnica de representação visual pode ser aplicada para exibir os valores de um conjunto de dados organizado (*tidy*). Nesse caso, cada célula da grade representa a combinação de uma observação e uma variável, e sua cor indica o valor dessa variável para aquela observação específica. Geralmente, as variáveis de cada observação são exibidas ao longo do eixo horizontal. A intensidade da cor segue uma convenção comum: tons mais escuros para valores maiores e tons mais claros para valores menores. A Figura 6.18 ilustra essa abordagem com um mapa de calor do conjunto de dados `mtcars` [96], onde a gradação de cores de amarelo (menor valor) a vermelho (maior valor) é utilizada.

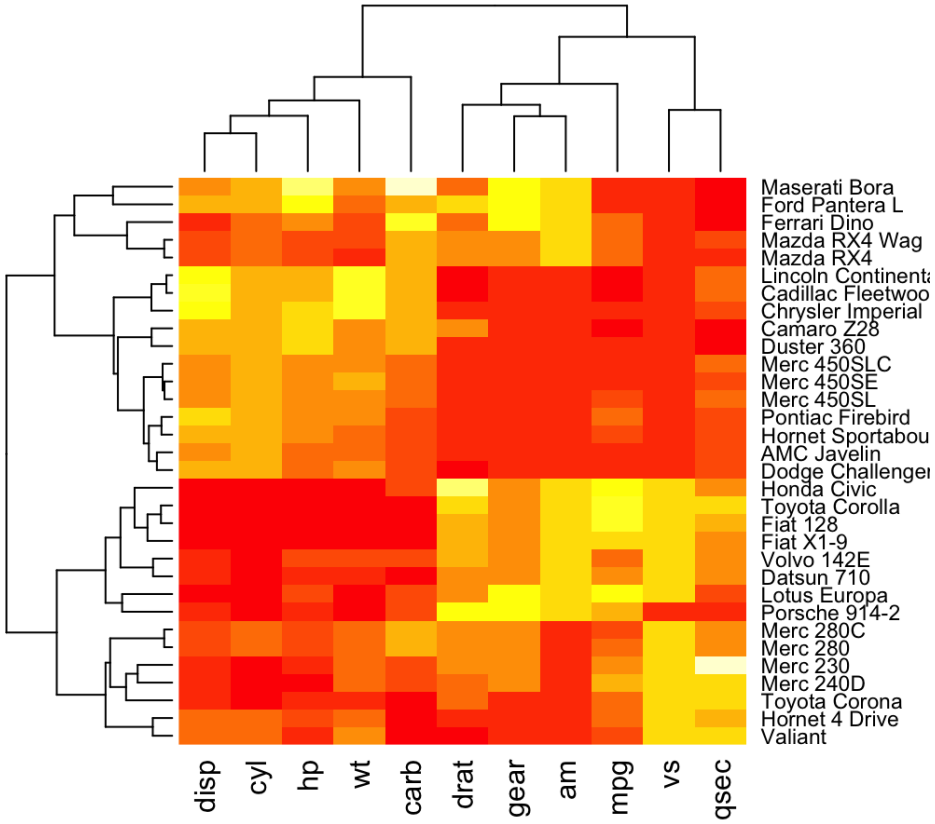


Figura 6.18

Porém, essa forma de clusterização visual não é equivalente à clusterização algorítmica, que envolve o uso de métodos matemáticos e computacionais para definir agrupamentos de dados com base em métricas objetivas, como distância ou densidade. Algoritmos como k-means (Seção 6.4.1) ou DBSCAN (Seção 6.4.3) formalizam a clusterização, proporcionando uma divisão precisa e quantitativa dos dados,

enquanto a clusterização visual depende da percepção subjetiva do observador que é propensa a ilusão visual (Seção 4.1). Portanto, embora a clusterização visual possa sugerir agrupamentos de forma rápida e intuitiva, ela carece da precisão e da robustez da clusterização algorítmica, que permite a validação dos padrões identificados.

As duas abordagens de clusterização se complementam: a clusterização visual serve como uma ferramenta exploratória, auxiliando na identificação de padrões e na formulação de hipóteses. A clusterização algorítmica, por outro lado, oferece uma análise mais objetiva e quantitativa, validando ou refinando os agrupamentos percebidos visualmente. Assim, embora a análise visual natural forneça um ponto de partida valioso, a clusterização algorítmica é essencial para uma análise precisa e baseada em métricas. A Figura 6.18 ilustra a visualização conjunta de um mapa de calor dos dados e um dendograma gerado por clusterização algorítmica. Os mapas de calor destacam visualmente as regiões de maior densidade de dados, sugerindo a presença de *clusters*. O dendograma, por sua vez, oferece uma validação formal dessas observações.

6.5 Considerações Finais

Este capítulo tem como objetivo explorar a estatística descritiva, fornecendo uma compreensão das medidas resumo e dos padrões de distribuição que caracterizam variáveis numéricas. Para facilitar a visualização dessas medidas, apresentamos gráficos utilizando ggplot2 (R) e plotnine (Python), com seus respectivos códigos. Destacamos que, embora as estatísticas resumidas busquem descrever completamente as distribuições, apenas a distribuição normal possui essa propriedade. Além disso, a normalidade dos dados é uma premissa fundamental para muitos algoritmos estatísticos. Portanto, deve-se verificar a normalidade dos dados antes de aplicar técnicas estatísticas que a pressupõem.

Discutimos os conceitos de correlação e covariância, analisando como essas medidas fornecem *insights* sobre a relação entre duas ou mais variáveis. Destacamos os gráficos de dispersão como uma forma de visualizar as relações entre duas variáveis e as matrizes de gráficos de dispersão para a análise de relações multivariáveis. Apresentamos também as medidas de similaridade, que permitem a comparação entre as observações de um determinado fenômeno. As técnicas de clusterização, juntamente com as representações visuais de dendrogramas e mapas de calor, tornaram-se ferramentas básicas para explorar e identificar grupos inerentes nos dados, revelando estruturas e padrões que podem não ser imediatamente evidentes.

Entretanto, é imperativo ressaltar algumas considerações. O coeficiente de correlação de Pearson, amplamente utilizado devido à sua interpretação simples, facilidade de cálculo e capacidade de medir relações lineares. Ele é bem estabelecido teoricamente, frequentemente ensinado em cursos de estatística e disponível na maioria dos *softwares* estatísticos. **É particularmente adequado para variáveis normalmente distribuídas** (Seção 8.5.2) e é amplamente aplicado em várias áreas de pes-

quisa. No entanto, sua sensibilidade a *outliers*, a limitação a relações lineares (e a limitação implícita a normalidade) podem levar ao uso de outros coeficientes, como Spearman ou Kendall, quando necessário.

Tanto as técnicas de análise de similaridade quanto de agrupamento dependem de comparações numéricas. Embora essa abordagem seja útil, ela pode não capturar completamente a complexidade semântica subjacente. O mapeamento inadequado dos valores numéricos pode distorcer a análise visual, tornando essencial que usuários experientes apliquem mapeamentos apropriados para revelar padrões e tendências significativas.

Nosso foco está em variáveis numéricas organizadas em tabelas, deixando de lado as variáveis categóricas. Embora tenhamos abordado amplamente as técnicas estatísticas para dados quantitativos, reconhecemos que a estatística descritiva é uma área ampla, com muitos outros aspectos a explorar, especialmente no que diz respeito às variáveis categóricas. Além disso, embora as representações tabulares de dados sejam altamente eficazes e versáteis para a maioria das análises, muitos fenômenos do mundo real podem envolver complexidades que vão além das limitações das tabelas bidimensionais. A organização inicial dos dados em tabelas, como etapa de pré-processamento, tem se mostrado extremamente útil, mas, em alguns casos, pode ser necessário considerar abordagens mais sofisticadas ou multidimensionais para capturar melhor a riqueza dos dados.

Apesar dessas ressalvas, este capítulo forneceu uma introdução aos fundamentos da estatística descritiva para dados numéricos. Entretanto, a análise de dados abrange uma gama muito mais ampla de tipos e técnicas. É fundamental avançar para a análise de variáveis categóricas, a compreensão da semântica dos dados, a exploração de estruturas complexas e o domínio de métodos mais sofisticados. A estatística, uma ferramenta essencial para a tomada de decisões, continua a evoluir, oferecendo inúmeras oportunidades de aprendizado e descoberta.

6.6 Exercícios

1. Consulte a Seção 2.4 da referência [72] e defina *outliers* conforme proposto por Tukey.

2. Faça LCA.1 e LCA.2 do Apêndice A da referência [36].

3. Resolva os itens 1 – 17 da Seção 1.11 da referência [72], usando os conjuntos de dados do repositório [96].

Capítulo 7

Preparação de Dados

Após importar e converter os dados para estruturas tabulares, é essencial realizar uma verificação minuciosa da qualidade dos dados antes de iniciar a exploração e análise em ambientes de desenvolvimento integrado (em inglês, *Integrated Development Environment* – IDE). De acordo com Han e colegas (Seção 2.4.1 em [31]), a **qualidade dos dados** está relacionada com a capacidade de atender aos requisitos de uso pretendidos. Existem vários fatores que compõem a qualidade global dos dados para garantir uma base sólida e confiável na tomada segura de decisões, incluindo:

Precisão (em inglês, *Accuracy*): Refere-se à exatidão dos dados em relação à realidade.

Completude (em inglês, *Completeness*): Diz respeito à presença de todos os dados necessários, sem omissões significativas.

Consistência (em inglês, *Consistency*): Envolve a uniformidade e coesão dos dados, garantindo que não haja contradições ou discrepâncias.

Atemporabilidade (em inglês, *Timeliness*): Relaciona-se à atualidade dos dados, ou seja, quão recentes são em relação a qualquer momento em que são necessários.

Credibilidade (em inglês, *Believability*): Refere-se à confiabilidade e a origem dos dados, ou seja, se as informações apresentadas são confiáveis e provenientes de fontes seguras.

Interpretabilidade (em inglês, *Interpretability*): Diz respeito à facilidade com que os dados podem ser compreendidos e utilizados de maneira eficaz para atender aos objetivos específicos.

Na Ciência de Dados (tabulares), o *data wrangling* é o procedimento de preparação de dados que transforma conjuntos de dados brutos em conjuntos limpos, organizados e otimizados (***tidy***) para exploração, análises estatísticas e visualização. Cerca de 80% do tempo dos analistas é dedicado a essa etapa, conforme observado por Wickham[104], em vez da análise propriamente dita,

A preparação de dados *tidy* compreende três etapas principais: importação, organização e transformação. A importação e formatação inicial dos dados foram discutidas no Capítulo 5. No entanto,

para otimizar a análise exploratória, visualização e modelagem em RStudio ou Jupyter Notebook, os dados devem passar por:

- análise exploratória (Seção 7.1): para identificar padrões e problemas iniciais.
- organização e reestruturação (Seção 7.2): para garantir a consistência e a estrutura adequada dos dados.
- transformação (Seção 7.3): para converter valores brutos em formatos adequados, mantendo a integridade dos dados originais.

Durante essas etapas, ferramentas estatísticas são utilizadas para identificar e corrigir discrepâncias, filtrar dados relevantes e reduzir redundâncias, preparando os dados para análise e interpretação eficazes.

A reestruturação e transformação dos dados são altamente adaptáveis, ajustando-se conforme os objetivos da análise planejada. Embora existam diversas ferramentas que suportam esses processos, sua combinação é personalizada para atender às nuances de cada cenário. Para análises de menor complexidade, pode ser suficiente reorganizar e filtrar os dados ou até mesmo transpor linhas em colunas nos conjuntos tabulares originais para tornar a representação dos dados "tidy". Em situações mais complexas, a integração de dados de fontes variadas pode exigir uma reestruturação mais profunda e a inclusão de informações ausentes. Em alguns casos, a transformação dos valores dos dados, por meio de sumarizações ou expansões, é necessária. Essa flexibilidade significa que a ordem de apresentação dos estágios de preparação de dados neste capítulo não reflete necessariamente a sequência exata de manipulações que um analista de dados adotaria em cada situação. A decisão sobre a sequência específica permanece um julgamento crítico que ainda não pode ser plenamente automatizado.

7.1 Análise Exploratória de Dados

A Análise Exploratória de Dados (em inglês, *Exploratory Data Analysis – EDA*) é fundamental para explorar dados sistematicamente, utilizando técnicas de visualização e transformação [102]. Este processo iterativo visa aprofundar a compreensão dos dados através de três etapas principais:

- Formulação de perguntas: Direciona a exploração, focando nos aspectos de maior interesse.
- Busca por respostas: Utiliza visualizações (gráficos e tabelas), transformações (reorganização e limpeza) e modelagem (representações matemáticas ou algorítmicas, modelos preditivos, relações e hipóteses).
- Refinamento contínuo: Ajusta as perguntas com base nos *insights* obtidos, criando um ciclo de exploração progressiva.

Este ciclo iterativo é o cerne da EDA, permitindo uma investigação adaptável e aprofundada dos dados.

Estatísticas descritivas, como médias, medianas e desvios padrão, podem ser calculadas para resumir a distribuição e a tendência central dos dados. Além disso, gráficos como histogramas, gráficos de caixa (em inglês, *boxplots*) e gráficos de dispersão são utilizados para revelar a distribuição subjacente a essas estatísticas resumidas e as relações entre variáveis. Vimos no Capítulo 6 que essas representações visuais são muito úteis para identificar padrões, tendências e possíveis *outliers* que podem afetar a qualidade da análise. A análise de correlação é outra técnica importante durante a EDA, permitindo aos analistas examinar como diferentes variáveis estão relacionadas. Uma matriz de correlação pode ajudar a identificar associações entre variáveis e a avaliar a força dessas associações, melhorando o nosso entendimento da estrutura dos dados.

EDA não segue um conjunto rígido de regras formais. É mais uma abordagem flexível e adaptativa, onde devemos nos sentir livres para explorar e investigar qualquer ideia que surja. Algumas dessas investigações levarão a descobertas valiosas, enquanto outras poderão não trazer os resultados esperados, mas todas contribuem para um entendimento mais profundo dos dados. Mesmo que as perguntas principais da pesquisa sejam fornecidas, é ainda recomendável examinar a qualidade dos dados por meio de EDA antes de aplicar processamentos mais complexos.

O tratamento de dados faltantes é uma etapa crítica na EDA. A identificação das ausências e a compreensão de suas causas são essenciais para decidir entre remover ou imputar os dados, ou adaptar as técnicas de análise. Essas decisões afetam diretamente a qualidade dos dados, sendo cruciais para a preparação para modelagem. A EDA também facilita a transformação e reestruturação dos dados. Funções como pivot_longer() e pivot_wider() da biblioteca tidyr em R, ou melt() e pivot() da biblioteca pandas em Python, auxiliam na conversão de dados entre formatos largos e longos, adequando-os aos diferentes objetivos de análise. Além disso, durante a EDA, os analistas podem testar hipóteses sobre as relações entre variáveis e os padrões observados, o que orienta a seleção de variáveis para a modelagem. Assim, a EDA aprimora a análise e garante a precisão das decisões baseadas nos dados.

Em [102], Wickham apresenta uma gama de ferramentas para ajudar a compreender a variação nos dados antes de iniciar a exploração e análise detalhada. Ele enfatiza a importância das técnicas que demonstram como lidar com variáveis individuais e pares de variáveis. Em seus comentários sobre os exemplos utilizados para ilustrar cada técnica, Wickham observa

> *You've seen techniques that work with a single variable at a time and with a pair of variables. This might seem painfully restrictive if you have tens or hundreds of variables in your data, but they're the foundation upon which all other techniques are built.*

Esta observação destaca a importância das abordagens fundamentais na análise de dados, mesmo quando se lida com conjuntos de dados mais complexos e extensos.

Tanto R [45] quanto Python [113] forncem ferramentas adequadas para levantar perfis dos dados (em inglês, *data profiling*). O ***data profiling*** é o processo de analisar dados para entender sua estrutura, conteúdo e qualidade. Isso envolve a coleta de estatísticas descritivas, a identificação de padrões e anomalias, e a avaliação da conformidade dos dados com regras de negócios e padrões de qualidade. Desenvolvidas com o principal objetivo de proporcionar uma análise exploratória de dados (EDA) consistente e rápida, as bibliotecas funModeling (R) e ydata-profiling (Python) geram relatórios completos com apenas uma linha de código.

No R, a biblioteca funModeling simplifica a EDA de conjuntos de dados quantitativos com a função profiling_num(), que gera um relatório detalhado das variáveis numéricas. O código a seguir demonstra como utilizar essa função para obter o relatório:

```
library(funModeling)


# Carregue o seu conjunto de dados
data = read_csv("caminho/para/o/arquivo.csv")


# Gere o relatório
profiling_num(data)
```

No Python, a biblioteca ydata-profiling oferece uma funcionalidade similar com a classe ProfileReport, que gera um relatório completo sobre o DataFrame. O código a seguir mostra como gerar um perfil de dados em Python:

```
import pandas as pd
from ydata_profiling import ProfileReport


# Carregue o seu conjunto de dados
data = pd.read_csv('caminho/para/seu/arquivo.csv')


# Gere o relatório
profile = ProfileReport(data) profile.to_notebook_iframe()
```

7.2 Organização e Reestruturação de Dados

O processo de organização e reestruturação de dados importados de fontes heterogêneas envolve várias etapas essenciais. Primeiramente, é necessário realizar a **limpeza dos dados** e **eliminação de duplicatas**, que inclui ajustes para lidar com valores ausentes, erros de digitação e formatos inconsistentes.

Em seguida, deve-se **reorganizar os dados** no formato "tidy", o que implica estruturar os dados de forma que cada variável se torne uma coluna, cada observação uma linha e cada tipo de unidade de observação uma tabela. Essa organização facilita o processamento eficiente dos dados pelas ferramentas de Ciência de Dados. Além disso, a **integração de dados** de diferentes fontes deve garantir que as informações relacionadas a uma mesma unidade de observação sejam corretamente unificadas.

7.2.1 Limpeza de Dados

É comum que os dados brutos apresentem erros ou falhas. Três tipos de erros comuns na aquisição de dados brutos são [31]:

Dados faltantes (em inglês, *data missing*): Referem-se à ausência de informações em determinadas observações ou variáveis em um conjunto de dados. Essa ausência de dados pode ocorrer por diversas razões, como erros na coleta de dados, falhas nos instrumentos de medição, recusa dos participantes em fornecer determinadas informações, entre outros motivos.

Dados inconsistentes (em inglês, *inconsistent data*): Referem-se a informações que estão em desacordo com as regras, padrões ou expectativas estabelecidas para um determinado conjunto de dados. Essa inconsistência pode ocorrer por diversos motivos, incluindo erros humanos durante a coleta ou entrada de dados, falhas nos processos de integração de dados de diferentes fontes (Seção 7.2.4), ou mesmo devido a alterações nas regras de transsformações que não foram refletidas de maneira consistente nos dados.

Dados ruidosos (em inglês, *noisy data*): Referem-se a informações que contêm variações indesejadas, imprecisões ou perturbações que não seguem o padrão esperado ou real do fenômeno que está sendo medido. Esse "ruído" pode ser causado por diversos fatores, incluindo erros de medição, interferências externas, falhas nos instrumentos de coleta de dados, ou até mesmo características inerentes à natureza do processo em estudo. Em muitos casos, os valores discrepantes (em inglês, *outliers*) podem ser considerados como uma forma específica de ruído nos dados, representando valores que não seguem o padrão típico do restante do conjunto.

Segundo Han e colegas, o primeiro passo no processo de limpeza de dados é a **detecção de *outliers*** com base na exploração dos dados, utilizando [31]:

- o conhecimento prévio sobre as propriedades dos dados. Esse conhecimento, também chamado de "dados sobre dados", é referido como **metadados**.
- Técnicas de estatística descritiva para obter valores como média, mediana e moda, e identificar valores discrepantes.

Os dados devem ser analisados quanto à univocidade (unicidade do valor associado a um atributo), consecutividade (presença de todos os valores entre o mínimo e o máximo especificados para um

atributo) e condição nula (representação e tratamento dos valores que indicam a condição nula ou ausência de valor para um determinado atributo).

Após identificar valores atípicos nos dados, é necessário selecionar a técnica mais adequada para tratá-los. Alguns desvios podem ser corrigidos manualmente, seja pela exclusão de observações, inserção manual de valores ausentes ou atribuição automática de um valor padrão. No entanto, a maioria dos erros exige transformações nos dados (Seção 7.3). Entre as técnicas para lidar com dados faltantes, a mais popular é a imputação de valores prováveis, computados por meio de árvores de decisão, regressão e inferência Bayesiana. Para suavizar dados ruidosos, técnicas como *binning* (segmentação) e regressão são amplamente empregadas. A abordagem interativa de limpeza e transformação de dados é uma estratégia promissora [31].

Tanto R quanto Python oferecem ferramentas para tratamento de dados faltantes. Em R, os dados faltantes são representados explicitamente por NA (do inglês *Not Available*) ou implicitamente por omissão (Capítulo 9 em [101]). Essas entradas podem ser detectadas usando a função filter do pacote dplyr com o argumento is.na() para remover as entrada com valores faltantes. Aplicando no *tibble* r_df1 definido na Seção 5.2.1:

```
r_df1_semfaltantes_1 <- r_df1 %>%
filter(!is.na(b), !is.na(c))
```

ou com o argumento complete.cases() para separar apenas as observações sem dados faltantes, como ilustra a seguinte linha de comando equivalente à anterior:

```
r_df1_semfaltantes_2 <- r_df1 %>%
filter(complete.cases(.))
```

] ou usar a função na.omit() para omitir as observações com dados faltantes:

```
r_df1_omitfalltantes <- na.omit(r_df1)
```

] Além do descarte das observações com dados faltantes, pode-se substituir os dados faltantes por valores válidos usando a função mutate. Esta função permite combinar colunas existentes em uma tabela utilizando qualquer tipo de operação condicional e matemática desejada. Para r_df1 especificamente, as seguintes linhas de comando preenchem os dados faltantes nas colunas b e c com 100 e 200, respectivamente:

```
r_df1_manual <- r_df1 %>%
     mutate(b = ifelse(is.na(b),100,b), c = ifelse(is_na(c),200,c))
```

Outra alternativa é preencher os dados faltantes com os valores interpolados com a função approx(). A seguinte sequência de instruções mostra o preenchimento do dado faltante da coluna c pelo valor interpolado dos outros 2 valores presentes:

```
vetor <- pull(r_df1, c);      #separar a coluna c como um vetor
indices_nao_nulos <- which(!is.na(vetor))
     # identificar indices das entradas que não são NA
vetor_approx <- approx(
     x=indices_nao_nulas,
     y=vetor[indices_nao_nulas, seq_along(vetor),
     n=3, method="linear")$y       #resultado com 3 elementos
r_df1_interpola_1 <- mutate(r_df1, c = vetor_approx)
     #substituir os valores da coluna c pelos valores interpolados
```

Em Python, todos os dados faltantes (NA do inglês *Not Available*) nas classes de objetos Series e DataFrames são representados por NaN (acrônimo de *Not a Number*) ou None. Esses valores podem ser facilmente detectados utilizando os métodos isnull() (entradas ausentes)[1]. Entre os métodos para tratar dados faltantes nas duas classes de objetos, destacam-se dropna(), que remove entradas com NA, seja por unidade em Series ou por linhas/colunas em DataFrames; fillna(), que substitui entradas NA por um valor especificado, incluindo estatísticas descritivas como a média; e interpolate(), que preenche valores faltantes com valores interpolados com base nos dados presentes [61, 60] [100].

Para identificar dados inconsistentes ou ruidosos, é possível aplicar **estatísticas resumidas** (Figura 7.1) para identificar *outliers*. Uma abordagem simples para tratá-los é substituí-los por valores ausentes (NA) e, em seguida, aplicar as técnicas desenvolvidas para lidar com dados faltantes.

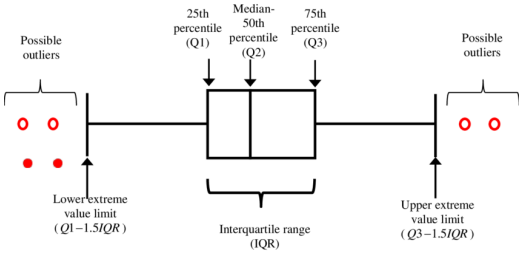


Figura 7.1: Detecção dos *outliers* através do gráfico de caixa (*boxplot*) (Fonte).

7.2.2 Organização no Formato *tidy*

A padronização dos dados no formato *tidy* é cada vez mais comum, e a maioria das funções prontas para uso pressupõe essa organização. O formato *tidy* estrutura os dados em tabelas, com cada observação em uma linha e cada variável em uma coluna (Figura 7.2), como explorado na Seção 5.1. Historicamente, os usuários dos primeiros aplicativos de processamento estatístico desenvolviam conversores para o formato *tidy* com linguagens de programação de propósito geral ou ferramentas de processamento de texto UNIX, como sed ou awk.

[1] A função complementar é notnull() (entradas presentes)

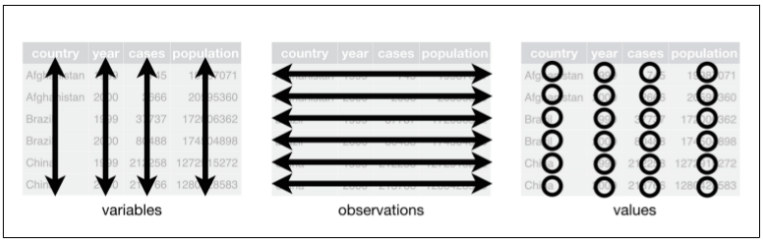


Figura 7.2: Dados no formato *tidy*. (Fonte: [101]

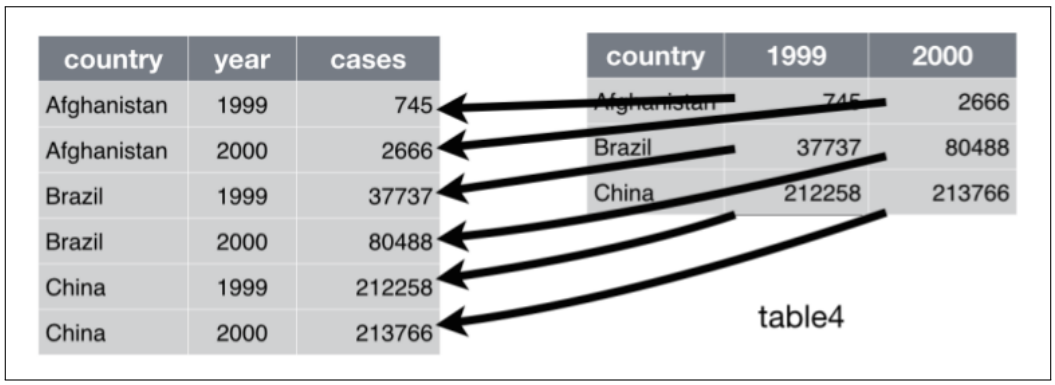
Contudo, Wickham demonstra no artigo que os problemas identificados em dados desorganizados, descritos na Lista 5.1, podem ser facilmente resolvidos com as funções do pacote tidyr em R. Vale ressaltar que todas as funções mencionadas no artigo possuem equivalentes no pacote pandas em Python. Ambas as bibliotecas dispõem funções que facilitam a reorganização dos dados para o formato "tidy", onde os valores de uma variável das observações são organizadas em uma única coluna. Essa reorganização torna os dados compatíveis com as funções de manipulação e transformação de dados nas bibliotecas dplyr (R) e pandas, além de facilitar a renderização de gráficos com ggplot2 (R) e plotnine (Python).

Entre as funções do pacote tidyr, destacam-se pivot_longer() e pivot_wider(), que transformam dados entre os formatos largo e longo. Essas funções, também conhecidas como **pivotamento de dados**, foram introduzidas como substitutas mais intuitivas e flexíveis para gather() e spread(), respectivamente. A função gather() combina várias colunas, cujos cabeçalhos são valores de uma variável, em uma única coluna, criando duas novas colunas: uma para armazenar os nomes originais das colunas (agora representadas como linhas) e outra para os valores correspondentes. A Figura 7.3a ilustra a junção dos nomes das colunas "1999" e "2000", que são valores da variável "year", em uma nova coluna "year". Com isso, todos os valores da variável "cases" são organizados em uma única (segunda) coluna, em conformidade com o formato *tidy*.

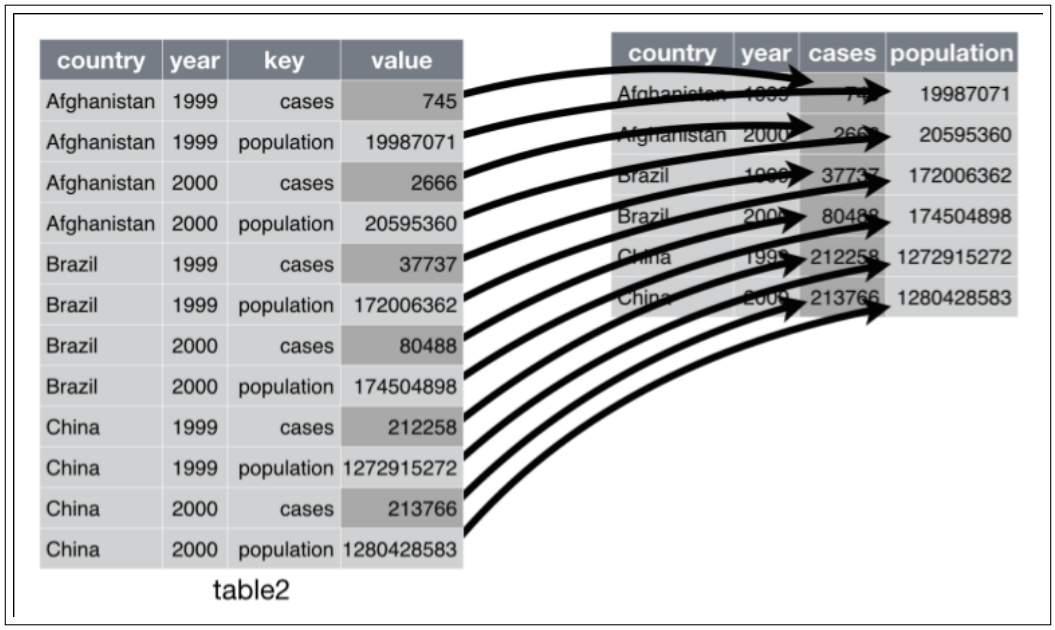
A função spread(), por outro lado, distribui os valores de várias variáveis armazenadas em uma coluna em múltiplas colunas, utilizando os valores de outra (segunda) coluna como cabeçalhos. A Figura 7.3b exemplifica essa operação, mostrando a separação dos valores de duas variáveis em duas colunas distintas. Os cabeçalhos dessas novas colunas, "cases" e "population", correspondem aos valores da coluna "key". Com isso, os valores das variáveis "cases" e "population", originalmente misturados na coluna "value", são separados em colunas individuais. Geralmente, gather() transforma tabelas amplas (*wide*) em tabelas estreitas e longas (*long*), organizando variáveis e seus valores em duas colunas. A função spread() realiza o processo inverso, convertendo tabelas longas em tabelas curtas e amplas.

Segue-se um trecho de código que transforma dados organizados em tabelas *untidy*, dados_mat e dados_port, em dados *tidy*:

```
#Carregar bibliotecas
```



(a) gather()



(b) spread()

Figura 7.3: Reestruturação de linhas e colunas, tal que as linhas e as colunas sejam, repectivamente, observações e características: (a) agrupamento de valores em diferentes colunas, resultando em duas colunas distintas, uma para as chaves e outra para os valores (*long*), (b) dispersão dos valores de uma coluna para diversas colunas, onde cada coluna representa um valor de chave da coluna "key" de chaves (*wide*). (Fonte: [101])

```
library(dplyr)
library(tidyr)


# Criando dados frame, contendo vetores de notas por ano
dados_mat <- data.frame(
  ID = c(1, 2, 3),
  Sexo = c("M", "F", "M"),
  '2020' = c(90, 80, 70),
  '2022' = c(75, 80, 60),
  '2023' = c(65, 95, 85)
)
```

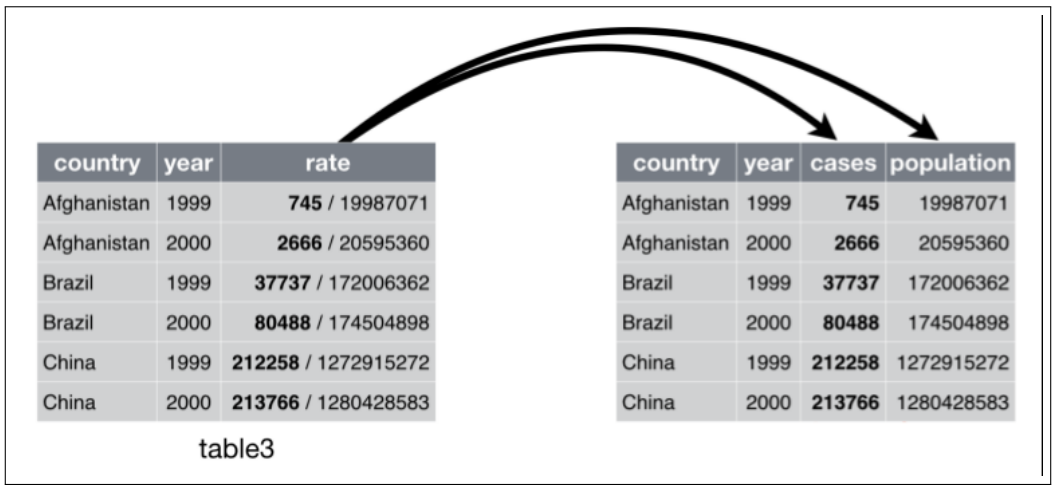
```
dados_port <- data.frame(
  ID = c(1, 2, 3),
  Sexo = c("M", "F", "M"),
  '2021' = c(70, 50, 75),
  '2022' = c(55, 75, 85),
  '2023' = c(50, 80, 85)
)

# Convertendo para o formato tidy usando duas alternativas equivalentes
dados_mat_tidy <- dados_mat %>%
  gather(key="Ano", value="Mat", -ID, -Sexo) #mantenha as colunas ID e Sexo
dados_port_tidy <- dados_port %>%
  gather(key="Ano", value="Port", 3:5) #junte os valores das colunas 3 e 5
```

Para variáveis categóricas ou texto, as funções separate() e unite() são aplicadas na reorganização das colunas dentro de um *data.frame* ou *tibble*. A função separate() divide os valores de uma única coluna em múltiplas colunas (*wide*), permitindo a criação de várias colunas a partir de uma coluna existente. Figura 7.4a mostra o desmembramento de um valor da variável categórica "rate" em dois valores que são armazenados em duas novas colunas distintas, "cases" e "population". A função unite(), por sua vez, combina os valores de várias colunas em uma única coluna (*long*), agregando múltiplas colunas em uma única nova coluna. Figura 7.4b) apresenta a concatenação dos valores das colunas "century" e "year" em um único texto, que é armazenado no nova coluna "year".

O seguinte trecho de códigos em R ilustra o uso da função separate() para separar o valor de data de aniversário numa coluna "Data" em três colunas, "Dia", "Mes" e "Ano":

```
dados_aniv <- data.frame(
  ID = c(1, 2, 3),
  Sexo = c("M", "F", "M"),
  Data = c("20/03/2002","30/07/2001","15/11/2001") #tokens da data separados por /
)

dados_aniv_discr <- dados_aniv %>%
  separate (Data, into=c("Dia","Mes","Ano"), sep="/")
```

Em Python, as bibliotecas pandas e numpy são amplamente utilizadas para reformatar os dados no formato *tidy*. A função melt () do pandas é equivalente à função pivot_longer() do tidyr em R, enquanto a função pivot() tem função análoga ao pivot_wider() em R. Por exemplo, a seguinte sequência de

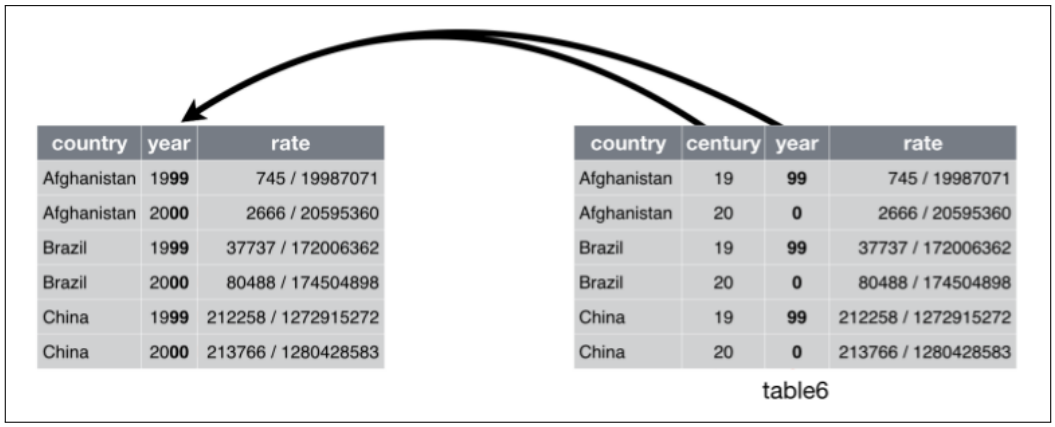


country	year	rate
Afghanistan	1999	**745** / 19987071
Afghanistan	2000	**2666** / 20595360
Brazil	1999	**37737** / 172006362
Brazil	2000	**80488** / 174504898
China	1999	**212258** / 1272915272
China	2000	**213766** / 1280428583

table3

country	year	cases	population
Afghanistan	1999	**745**	19987071
Afghanistan	2000	**2666**	20595360
Brazil	1999	**37737**	172006362
Brazil	2000	**80488**	174504898
China	1999	**212258**	1272915272
China	2000	**213766**	1280428583

(a) separate()



country	year	rate
Afghanistan	**1999**	745 / 19987071
Afghanistan	**2000**	2666 / 20595360
Brazil	**1999**	37737 / 172006362
Brazil	**2000**	80488 / 174504898
China	**1999**	212258 / 1272915272
China	**2000**	213766 / 1280428583

country	century	year	rate
Afghanistan	19	**99**	745 / 19987071
Afghanistan	20	**0**	2666 / 20595360
Brazil	19	**99**	37737 / 172006362
Brazil	20	**0**	80488 / 174504898
China	19	**99**	212258 / 1272915272
China	20	**0**	213766 / 1280428583

table6

(b) unite()

Figura 7.4: Recodificação dos valores de variáveis: (a) separação de um valor numa coluna em diferentes partes, cada uma das quais é alocada a uma coluna, (b) concatenação de partes de um valor alocadas em diferentes colunas num único valor alocado numa única coluna. (Fonte: [101])

instruções em Python cuja tabela *tidy* py_dados_mat_tidy gerada é equivalente à tabela dados_mat_tidy gerada pelos códigos em R:

```
# Exemplo de uso do pandas em Python
import pandas as pd


# Criando um DataFrame
py_dados_mat = pd.DataFrame({
    'ID': [1, 2, 3],
    'Sexo': ['M', 'F', 'M'],
    '2020': [90, 80,70],
    '2022': [75, 80, 60],
    '2023': [65, 95, 85]
})
```

```
py_dados_port = pd.DataFrame({
    'ID': [1, 2, 3],
    'Sexo': ['M', 'F', 'M'],
    '2021': [70, 50, 50],
    '2022': [55, 75, 85],
    '2023': [50, 80, 85]

})

# Convertendo-o para o formato tidy longer
py_dados_mat_tidy = py_dados_mat.melt(id_vars=['ID', 'Sexo'], var_name='Ano',
    value_name='Mat')
py_dados_port_tidy = py_dados_port.melt(id_vars=['ID', 'Sexo'], var_name='Ano',
    value_name='Port')
```

As funções equivalentes às funções separate() e unite() no pacote pandas são, respectivamente, split() e join(). O trecho a seguir demonstra o uso do split() para dividir a coluna "Data" em três colunas: "Dia", "Mês" e "Ano", semelhante à tabela dados_aniv_discr gerada com o código R. Vale notar que em Python são utilizadas, de fato, duas funções: split() para dividir uma coluna em três e concat() para unir as novas colunas à tabela, após remover a coluna original "Data".

```
py_dados_aniv = pd.DataFrame({
  'ID': [1, 2, 3],
  'Sexo': ['M', 'F', 'M'],
  'Data': ["20/03/2002","30/07/2001","15/11/2001"] #tokens da data separados por /
})

new_columns = (py_dados_aniv.
  Data.str.split("/", expand = True).
  rename(columns = {0: "Dia", 1: "Mes", 2: "Ano"})
  )

py_dados_aniv_discr = pd.concat([py_dados_aniv.drop(columns = 'Data'), new_columns], axis=1)
```

7.2.3 Eliminação de Redundâncias

Durante a fase de preparação dos dados, redundâncias podem surgir de diversas fontes, como registros duplicados ou informações repetidas em diferentes colunas. No formato *tidy*, é normal que algumas

entradas apareçam várias vezes, como parte da estrutura de dados, mas redundâncias no sentido de dados desnecessários ou duplicados devem ser removidas. Dados redundantes podem inflacionar artificialmente as estatísticas, comprometer a validade dos *insights* e resultar em modelos de aprendizado de máquina que não generalizam bem para novos dados. Ao remover redundâncias para Análise de Dados, asseguramos que **cada observação e cada variável representem informações únicas e não repetidas**. Isso não só melhora a eficiência das análises, reduzindo o tempo de processamento e armazenamento, mas também contribui para a qualidade dos resultados.

Tanto R quanto Python oferecem funções eficazes para a remoção de duplicatas em conjuntos de dados. Em R, podemos utilizar as funções duplicated(), unique() e distinct(). A função duplicated() identifica quais linhas são duplicadas, enquanto unique() remove as duplicatas, retornando apenas valores únicos. A função distinct(), do pacote dplyr, remove, por sua vez, duplicatas para colunas específicas. No Python, utilizando a biblioteca pandas, podemos alcançar resultados semelhantes com as funções duplicated() e drop_duplicates(). A função duplicated() detecta a ocorrência de duplicatas, e drop_duplicates() é empregada para eliminar essas duplicatas do DataFrame.

7.2.4 Integração de Dados

A análise de dados frequentemente exige a **integração de dados**, ou seja, a fusão de informações provenientes de múltiplos repositórios de dados (Seção 2.4.3 em [31]). A heterogeneidade semântica e a estrutura dos dados apresentam grandes desafios na integração de dados. Problemas, como identificação de entidades, dados conflitantes, redundância em valores associados a uma variável e duplicação de registros de dados, constituem desafios para integração. Uma integração cuidadosa pode contribuir para reduzir redundâncias e evitar inconsistências no conjunto de dados resultante, aprimorando a precisão e a velocidade do subsequente processo de análise de dados.

O problema de **identificação de entidades** (em inglês, *entity identification problem*) se refere ao desafio de associar a uma mema entidade observações equivalentes de diferentes fontes de dados heterogêneos. Cada fonte tem o seu próprio esquema de dados e identificadores únicos para um propósito específico. Ao integrar essas fontes distintas para formar uma visão abrangente dos dados, podem surgir dificuldades na determinação de quais observações em uma fonte de dados correspondem às observações de mesmas entidades em outras fontes, especialmente quando seus identificadores e esquemas são diferentes. Considere, por exemplo, a integração de diversos exames, imagiológicos, laboratorias e neuropsicológicos, de um paciente para a tomada de decisões sobre o tratamento mais eficaz. Resolver o problema de identificação de entidades envolve o uso de metadados que revelam as propriedades dos esquemas a serem fundidos.

Diferenças na representação, escala ou codificação entre fontes de dados levam a valores de atributos (caracerísticas ou variáveis) divergentes para uma mesma entidade, um problema comum na integração de dados. Essa disparidade causa inconsistências e conflitos nos atributos da entidade.

Amenizar esse problema de dados conflitantes pode envolver práticas como transformar os dados para unidades comuns, incluindo padronização do formato de dados (em inglês, *data standardization*), normalização dos valores para uma mesma escala (em inglês, *data normalization*), harmonização de diferentes codificações ou nomenclaturas para os mesmos elementos (em inglês, *data reconciliation* ou *data harmonization*), e desambiguação de representações ligeiramente diferentes (em inglês, *data disambiguation*).

Na integração de dados, a **redundância**, repetição desnecessária de informações, é também um problema comum. Ela ocorre quando um atributo (característica ou variável) é derivado de outro, ou devido a inconsistências na nomenclatura. A remoção de redundâncias é uma prática comum em bancos de dados. Vimos na Seção 5.2.2 que bancos de dados relacionais aplicam a técnica de normalização para reduzir a redundância e evitar problemas como inconsistências e anomalias de atualização. O processo de normalização envolve a decomposição de tabelas complexas em tabelas mais simples interligadas por chaves.

A normalização de bancos de dados e o formato *tidy*, apesar de parecerem contraditórios, na verdade cumprem papéis diferentes dentro da análise de dados. A normalização garante a eficiência e a consistência dos dados no armazenamento, enquanto o formato *tidy* facilita a análise e a modelagem dos dados. Portanto, ao importarmos os dados dos bancos de dados relacionais, deve-se prepará-los para análise estatística e modelagem reorganizando-os numa única tabela de formato *tidy*, especialmente em linguagens como R e Python. Ferramentas de Ciência de Dados facilitam essa transformação. O formato *tidy* é o formato esperado para a maior parte das ferramentas de análise de dados.

Existem duas classes de funções de junção de tabelas [102]: as **junções baseadas na combinação** das variáveis das observações (*mutating joins*) e as **junções baseadas na filtragem** das observações de uma tabela em relação às observações contidas na segunda tabela (*filter join*). Em linguagem matemática, as tabelas podem ser abstraídas em **conjuntos de dados** e os diferentes tipos de junção em diferentes tipos de **operações de conjunto**. Os elementos das junções baseadas na combinação das variáveis das observações são as variáveis/características de uma unidade observacional (colunas das tabelas), enquanto os elementos das junções baseadas em filtragem das observações são as observações de uma população (linhas das tabelas). Cabe observar que as técnicas de integração muitas vezes se restringem a manipulações sintáticas, concentrando-se nos nomes das variáveis (colunas) e na comparação sintática dos valores dessas variáveis (linhas).

Dentre as **junções baseadas na combinação** das variáveis das observações de duas tabelas A e B, destacam-se

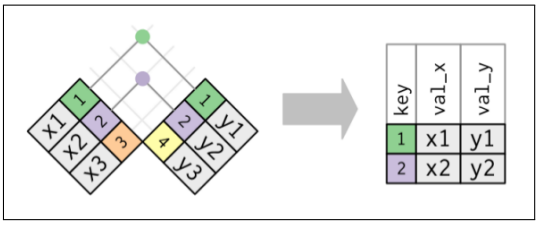
"$A \cap B$" (em inglês, *inner join*): a nova tabela contém as observações das duas tabelas, cujos valores das variáveis especificadas estão contidos nas observações de ambas as tabelas (Figura 7.5a).

"$A \cup B$" (em inglês, *full join*): a nova tabela contém as observações das duas tabelas,

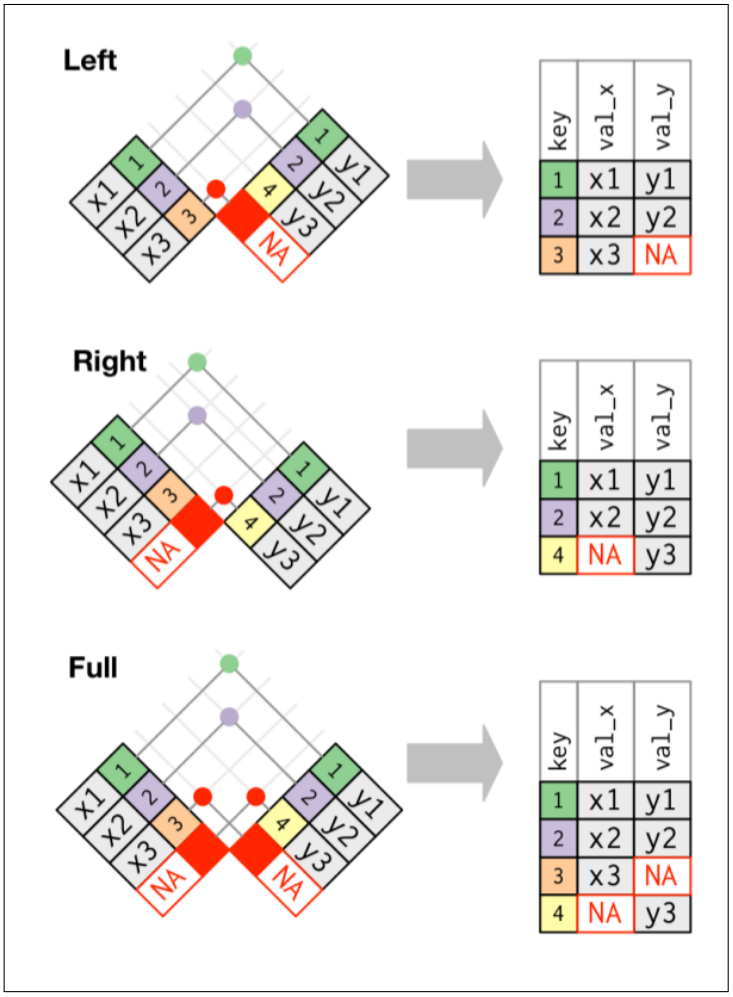
cujos valores das variáveis especificadas estão contidos nas observações da tabela A ou da tabela B (Figura 7.5b).

"$(A - B) \cup (A \cap B)$**"** (em inglês, *left join*): a nova tabela contém as observações das duas tabelas, cujos valores das variáveis especificadas estão contidos na tabela A (Figura 7.5b).

"$(B - A) \cup (A \cap B)$**"** (em inglês, *right join*): a nova tabela contém as observações das duas tabelas cujos valores das variáveis especificadas estão contidos nas observações da tabela B (Figura 7.5b).



(a) *Inner Join*



(b) *Outer Join*

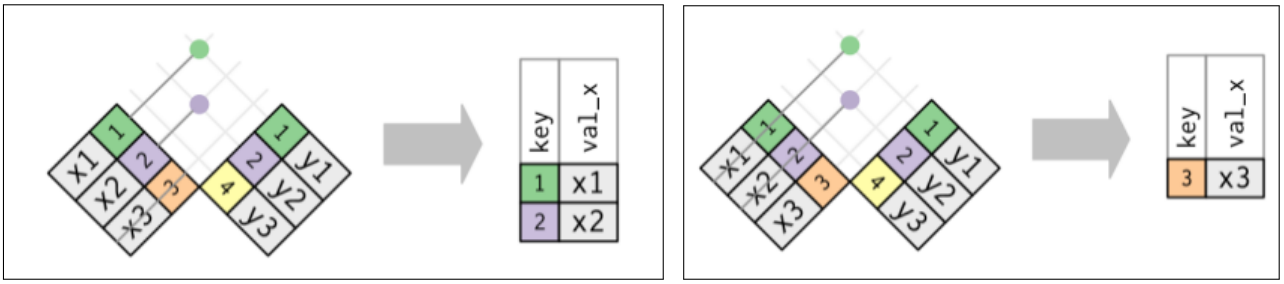
Figura 7.5: Integrações baseadas na junção das variáveis das duas tabelas: (a) a nova tabela contém apenas as observações que possuem os mesmos valores em ambas as tabelas para as variáveis especificadas, e (b) a nova tabela inclui as observações que têm valores em pelo menos uma das tabelas para as variáveis especificadas. (Fonte: [101])

E das **junções baseadas em filtragem** das observações em A em relação aos valores das variáveis especificadas na tabela B, distinguem-se

filtragem por inclusão (em inglês, *semi-join*): a nova tabela contém as variáveis de A e

contém todas as observações em A que tem os mesmos valores em B para as variáveis especificadas (Figura 7.6a).

filtragem por exclusão (em inglês, *anti-join*): a nova tabela contém as variáveis de A e contém as observações em A que **não** tem os mesmos valores em B para as variáveis especificadas (Figura 7.6b).



(a) *Semi Join* (b) *Anti Join*

Figura 7.6: Integrações baseadas em filtragem das observações na tabela A em relação aos valores das variáveis especificadas na tabela B: (a) a nova tabela contém apenas as observações que possuem os mesmos valores que a tabela B para as variáveis especificadas, e (b) a nova tabela exclui as observações que têm os mesmos valores que a tabela B para as variáveis especificadas. (Fonte: [101])

A escolha entre esses métodos depende da estrutura dos dados e dos objetivos de análise, sendo essencial compreender as relações entre as tabelas para garantir uma integração adequada e evitar redundâncias ou perda de informações.

As seguintes instruções, em R, ilustram as diferentes ações de integração sobre as tabelas dados_port_tidy e dados_mat_tidy geradas na Seção 7.2.2, considerando as variáveis "ID", "Sexo" e "Ano":

```
dados_notas_inner <- inner_join (dados_port_tidy, dados_mat_tidy,
       c=("ID","Sexo","Ano")
dados_notas_full <- full_join (dados_port_tidy, dados_mat_tidy,
       c=("ID","Sexo","Ano")
dados_notas_left <- left_join (dados_port_tidy, dados_mat_tidy,
       c=("ID","Sexo","Ano")
dados_notas_right <- right_join (dados_port_tidy, dados_mat_tidy,
       c=("ID","Sexo","Ano")
dados_notas_semi <- semi_join (dados_port_tidy, dados_mat_tidy,
       c=("ID","Sexo","Ano")
dados_notas_anti <- anti_join (dados_port_tidy, dados_mat_tidy,
       c=("ID","Sexo","Ano")
```

Em Python, o método merge() do pacote pandas trata todas as junções baseadas na combinação das variáveis. O tipo de junção é diferenciado pelo valor atribuído ao argumento how do método (Tabela 7.1 em [100]):

```
py_dados_notas_inner = py_dados_port_tidy.merge (py_dados_mat_tidy,
       on=["ID","Sexo","Ano"],how='inner')
py_dados_notas_full = py_dados_port_tidy.merge (py_dados_mat_tidy,
       on=["ID","Sexo","Ano"],how='outer')
py_dados_notas_left = py_dados_port_tidy.merge (py_dados_mat_tidy,
       on=["ID","Sexo","Ano"],how='left')
py_dados_notas_right = py_dados_port_tidy.merge (py_dados_mat_tidy,
       on=["ID","Sexo","Ano"],how='right')
```

O pacote numpy dispõe a função concat() que empilha DataFrames verticalmente (ao longo das linhas, axis=0) ou horizontalmente ao longo das colunas, axis=1), como mostra a Tabela 7.2 em [100].

Não há funções correspondentes a semi_join() and anti_join() em Python. Pode-se, porém, usar as funções do pacote pandas e numpy para implementá-las. Aplicando o procedimento a seguir sobre as tabelas py_dados_port_tidy e py_dados_port_tidy, chega-se às mesmas tabelas dados_notas_semi e dados_notas_anti computadas em R:

```
py_dados_notas_full = py_dados_port_tidy.merge (
      py_dados_mat_tidy, how='outer', indicator=True)
      # combina as duas tabelas por full_join
py_dados_notas_semi=py_dados_notas_full[
      (py_dados_notas_full._merge=='both')][ #inclui linhas que tem valores comuns
      list(py_dados_port_tidy.columns)] # seleciona as linhas que tem em py_dados_port_tidy
py_dados_notas_anti=py_dados_notas_full[
      (py_dados_notas_full._merge=='left_only')][ #inclui linhas que so tem em A
      list(py_dados_port_tidy.columns)] # seleciona as linhas que tem em py_dados_port_tidy
```

Tabelas provenientes de fontes diversas podem apresentar desafios, como ter colunas com nomes distintos, mas conteúdo equivalente, ou colunas com valores semelhantes ou "derivados", mas não idênticos. Para superar essas disparidades, são necessários ajustes prévios nas tabelas individuais, como a **renomeação de colunas** ou a **adição de colunas** auxiliares. No R, para renomear e adicionar colunas em um *data.frame* ou *tibble*, podemos utilizar as funções rename() e mutate() da biblioteca dplyr [102]. A função rename() é usada para alterar o nome das colunas existentes, enquanto mutate() permite adicionar novas colunas ou modificar as existentes. No Python, com a biblioteca pandas, podemos renomear colunas usando a função rename() [100], e para adicionar novas colunas, utilizar assign().

A instrução a seguir renomeia a coluna "Mat" por "Matemática" da tabela dados_mat_tidy:

```
rename (dados_mat_tidy, Matemática = Mat)
```

Em Python, uma instrução equivalente à renomeação de colunas é:

```
py_dados_mat_tidy.rename(columns={'Mat':'Matemática'})
```

O seguinte código em R ilustra a adição da coluna "Media" na tabela dados_notas_full

```
mutate(dados_notas_full,
     Media = (Por + Mat)/2)
```

Um resultado equivalente pode ser obtido em Python usando a função assign:

```
 py_dados_notas_full.assign (
     Media = lambda x:(x.Port+x.Mat)/2
   )
```

Uma abordagem alternativa em Python é criar e atribuir valores a uma coluna chamada "Media", que ainda não existe no py_dados_notas_full. Isso pode ser feito calculando a média dos valores das colunas "Port" e "Mat" em cada observação e atribuindo o resultado à nova coluna "Media":

```
py_dados_notas_full["Media"]=(py_dados_notas_full["Port"]+py_dados_notas_full["Mat"])/2
```

7.3 Transformação de Dados

A transformação de dados modifica e consolida dados tabulares para exploração e análise [31], facilitando a análise visual e a compreensão de padrões. As técnicas incluem filtragem, reorganização, normalização, redução de dados e redução de dimensionalidade. A **filtragem de dados** (Seção 7.3.1) seleciona observações; a **reorganização** (Seção 7.3.2) reordena dados; a **normalização** (Seção 7.3.3) escala valores (diferente da normalização relacional, Seção 7.2.4); a **redução de dados** (Seção 7.3.4) reduz o volume de dados; e a **redução de dimensionalidade** (Seção 7.3.5) remove variáveis irrelevantes.

7.3.1 Filtragem

A Seção 7.2.1 abordou a limpeza de dados, destacando a importância das ações imediatas realizadas logo após a importação dos dados. Essas ações visam lidar com problemas comuns, como valores ausentes, erros de digitação, inconsistências e outras imperfeições nos dados brutos. A **filtragem condicionak de dados** é uma etapa subsequente, que ocorre após a integração dos dados ou em momentos específicos do processo de análise. Nessa fase, **seleciona**-se as observações (linhas) ou as variáveis (colunas) com base em critérios determinados, como valores presentes em determinadas colunas, intervalos de tempo específicos ou outras condições relevantes. O objetivo da filtragem é simplificar o processo de exploração, focando no que é relevante para a análise, o que torna as investigações mais claras e eficazes.

Na Seção 7.2.3 vimos que a remoção de dados duplicados é uma tarefa comum na limpeza de dados. Essa tarefa pode ser vista, de fato, como um caso especial de filtragem. Nesse contexto, o critério para a filtragem condicional de dados é a identificação e exclusão de linhas que são réplicas de outras linhas no conjunto de dados. Essa filtragem é conhecida por **filtragem de duplicatas**, cujo objetivo específico é manter apenas as observações únicas, eliminando a redundância que pode distorcer as análises. Note que essas observações duplicadas podem surgir na integração de conjuntos de dados, mesmo que as duplicatas tenham sido eliminadas nas tabelas individuais. A identificação e remoção dessas duplicatas garantem que cada observação seja única, evitando distorções nos resultados e proporcionando uma base sólida para análises confiáveis.

Além das funções de limpeza de dados que vimos na Seção 7.2.3, R oferece duas funções básicas relacionadas com a filtragem de dados que replicam os efeitos da limpeza de dados [101]:

filter() que seleciona subconjuntos específicos de linhas com base em condições lógicas predefinidas.

select() que seleciona subconjuntos específicos de colunas com base nos nomes predefinidos.

Dada uma tabela com duplicatas:

```
dados_duplicados <- data.frame(
  ka = rep(c('um', 'dois'), times = c(3, 4)),
  kb = c(1, 1, 2, 3, 3, 4, 4)
)
```

Podemos remover as observações duplicadas com a função distinct()

```
library(dplyr)
dados_sem_duplicatas <- distinct (dados_duplicados)
     # Retorna um vetor ou uma matriz com os elementos únicos de kb.
```

ou com uso das 3 funções básicas de R duplicated (), filter() e select():

```
variaveis_originais <- colnames(dados_duplicados)
     #salvar os nomes das colunas antes de adicionar "duplicado"
dados_duplicados$duplicado <- duplicated(dados$kb)
     # adicione uma nova coluna duplicado do tipo booleano
dados_duplicados_filtrados <- dados_duplicados %>%
     filter (duplicado == FALSE) %>%
     # seleciona as linhas que satisfaz duplicado==FALSE
     select (all_of(variaveis_originais))
     # seleciona as colunas em variaveis_originais
```

Em Python, uma função equivalente a distinct() é a função drop_duplicates() do pacote pandas que faz descarte direto de todas as linhas duplicadas [100]), como demonstra o seguinte bloco de instruções:

```
import pandas as pd


py_dados_duplicados = pd.DataFrame({'ka': ['um'] * 3 + ['dois'] * 4,
                                    'kb': [1, 1, 2, 3, 3, 4, 4]})
py_dados_duplicados_filtrados = py_dados_duplicados.drop_duplicates('kb')
```

O seguinte trecho de código demonstra que o efeito da função drop_duplicates() pode ser alcançado com o uso da função query() para a seleção condicional de linhas, baseada em expressões booleanas [37], da função filter() para a seleção de colunas (características ou variáveis) com base em seus rótulos e da função loc para a seleção de linhas (observações) ou colunas (variáveis) por rótulos. Ambas as funções são métodos disponíveis no pacote pandas.

```
import pandas as pd


py_dados_duplicados = pd.DataFrame({'ka': ['um'] * 3 + ['dois'] * 4,
                                    'kb': [1, 1, 2, 3, 3, 4, 4]})
#salvar os nomes das colunas antes de adicionar "duplicado"
variaveis_originais = dados_duplicados.columns.tolist()
#inserir a coluna duplicated
py_dados_duplicados["duplicated"]=py_dados_duplicados.duplicated('kb')
#selecionar linhas pela condição duplicated==False
#e sobre o resultado selecionar apenas as colunas listadas em variaveis originais
dados_duplicados_filtrados = py_dados_duplicados \
.query ('duplicated == False') \
.filter (variaveis_originais)
```

Enquanto filter() e loc() são utilizados para selecionar linhas e colunas com base em critérios específicos, a função drop() em pandas é utilizada para remover linhas ou colunas específicas de um *DataFrame* com base em rótulos ou índices [37]). A seleção das colunas que pertencem apenas à tabela original py_dados_duplicados na última linha do código acima equivale ao descarte da coluna auxiliar duplicated, que foi adicionada para facilitar a remoção das duplicatas. Esse procedimento pode ser implementado com o uso de drop(), conforme ilustrado no seguinte trecho de código:

```
dados_duplicados_filtrados = py_dados_duplicados \
.query ('duplicated == False') \
.drop (columns=['duplicated'])
```

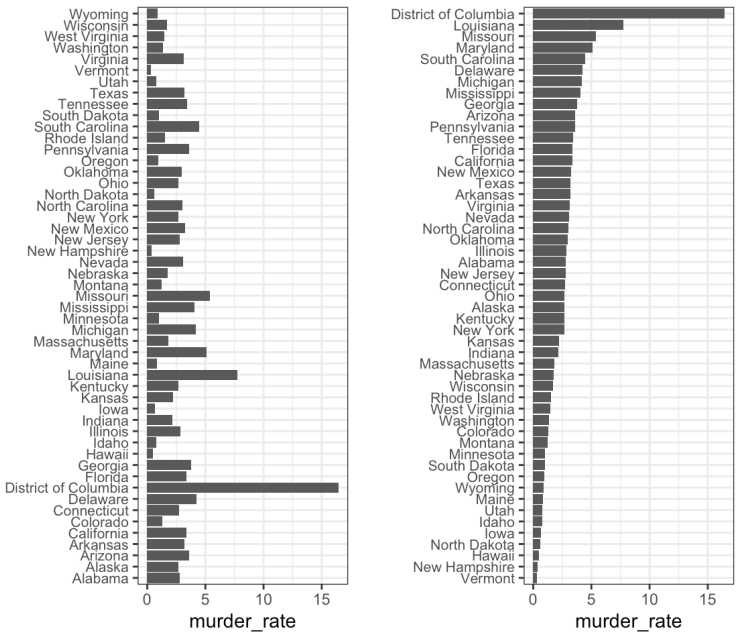


Figura 7.7: Reordenação dos valores de uma variável melhora a visualização dos dados, destacando padrões relevantes. (Fonte)

7.3.2 Reordenação de Dados

A **ordenação dos dados** facilita a interpretação visual de padrões, tendências ou comportamentos atípicos, tornando-os mais evidentes. Visualizações de dados, como gráficos de linha ou séries temporais, podem se beneficiar significativamente de uma ordenação adequada. Além disso, a ordenação pode **realçar tendências temporais e sequências lógicas** nos dados, contribuindo para uma compreensão mais clara. Em análises estatísticas, especialmente em técnicas como a regressão, a ordenação pode ajudar a **identificar relações lineares** entre variáveis independentes e dependentes. A organização dos dados também pode ser útil na **detecção de valores extremos** (em inglês, *outliers*), que podem se destacar mais quando os dados estão dispostos em ordem. Se a intenção é **segmentar os dados em grupos** com base em uma variável, a ordenação pode simplificar esse processo. É, porém, importante considerar o contexto específico da análise e os objetivos da investigação ao decidir pela ordenação, pois em algumas situações a ordem original dos dados pode ser mais relevante, enquanto em outras, uma reordenação proporciona *insights* adicionais. A Figura 7.7 compara a visualização da taxa de assassinatos antes e depois da reordenação, destacando a melhoria.

R e Python oferecem, respectivamente, a função arrange() e sort_values() para simplificar as ordenações em relação a um conjunto de variáveis. Por exemplo, para ordenar na ordem crescente em R [102] as observações em relação à nota "Port" na tabela dados_notas_full, pode-se usar a seguinte instrução:

```
arrange (dados_notas_full, Port)
```

Para ordem decrescente, basta explicitar a ordem:

```
arrange (dados_notas_full, desc(Port))
```

Para ordenações envolvendo mais de uma variável, cada variável adicional será usada para desempatar os valores associados às variáveis anteriores:

```
arrange (dados_notas_full, desc(Mat),desc(Port))
```

As instruções equivalentes em Python são [100]:

```
py_dados_notas_full.sort_values (by = ['Port'])
py_dados_notas_full.sort_values (by = ['Port'], ascending=False)
py_dados_notas_full.sort_values (by = ['Mat','Port'], ascending=False)
```

Em ambas as linguagens, os valores NA são sempre colocados no final de cada sequência.

7.3.3 Normalização de Valores

Para evitar dependências na escolha das unidades de medida, é uma prática comum normalizar ou padronizar os dados. A **normalização** se refere à transformação dos dados de forma que se ajustem a uma faixa menor ou comum, como $[-1,0; 1,0]$ ou $[0,0; 1,0]$. Dentre os diversos métodos para normalização de dados, destacam-se os seguintes, conforme abordado na Seção 2.5.1 de Han e Kamber [31]:

Normalização Min-max: Aplica-se uma transformação linear sobre os dados originais $v_i \in [min_A, max_A]$ de um atributo A para os valores $v_i\prime \in [new\_min_A, new\_max_A]$

$$v_i\prime = \frac{v_i - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A \tag{7.1}$$

Normalização por escore-z: Os valores de um atributo, A, são normalizados com base na média ($\overline{A}$) e no desvio-padrão (σ_A) de A. Um valor v_i de A é normalizado para $v_i\prime$ através de

$$v_i\prime = \frac{v_i - \overline{A}}{\sigma_A} \tag{7.2}$$

Normalização por escala decimal: Essa normalização movendo a vírgula dos valores do atributo A. O número de casas decimais movidas, j, depende do valor absoluto máximo ($max(|v_i|)$ de A, isto é, j deve ser tal que o valor normalizado

$$v_i\prime = \frac{v_i}{10^j} \tag{7.3}$$

satisfaça $0.1 < max(|v_i\prime| < 1$, que equivale a

$$10^{-1} \leq \frac{max(|v_i\prime|}{10^j} \leq 10^0.$$

Aplicando logaritmos na base 10 em todos os termos, temos

$$-1 \leq log_{10}(\frac{max(|v_i\prime|}{10^j}) \leq 0$$
$$-1 \leq log_{10}(max(|v_i\prime|) - log_{10}(10^j) \leq 0$$
$$-1 \leq log_{10}(max(|v_i\prime|) - j \leq 0$$
$$log_{10}(max(|v_i\prime|) - 1 \leq j \leq log_{10}(max(|v_i\prime|). \tag{7.4}$$

É importante destacar que para os três métodos de normalização apresentados, é útil calcular estatísticas resumidas, como médias, máximos e mínimos, conforme detalhado na Seção 6.1. Além disso, é válido observar que a normalização pode resultar em alterações significativas nos dados originais. Recomenda-se a preservação dos parâmetros de normalização, tais como média e desvio padrão (no caso da normalização por escore-z) e os valores mínimo e máximo (no caso da normalização Min-max e por escala decimal), para possibilitar a reconstrução dos dados originais ou a criação de novas variáveis para os valores normalizados. Esta prática é crucial, uma vez que as técnicas de normalização continuam a evoluir, e os dados normalizados pelas técnicas atuais podem não ser tão eficazes em futuras análises.

Para ilustrar, vamos normalizar as notas na coluna 'Mat' de dados_notas_full pelas 3 técnicas em R. As primeiras duas equações usam os valores, mínimo, máximo, média e desvio padrão, computáveis pela função summarize do pacote dplyr:

```
resMat <- summarize (dados_notas_full,
          min = min(Mat, na.rm=TRUE),
          max = max(Mat, na.rm=TRUE),
          media = mean(Mat, na.rm=TRUE),
          std = sd(Mat, na.rm=TRUE))
```

Note o uso do argumento na.rm em R. Esse argumento pode ser encontrado em muitas funções estatísticas em R. É um argumento lógico (booleano) que especifica se os valores ausentes devem ser removidos antes do cálculo. Se na.rm for definido como TRUE, os valores ausentes serão removidos; se for definido como FALSE (o padrão), os valores ausentes resultarão em NA. Em Python, as funções da biblioteca pandas são projetadas para lidar com valores NA de maneira específica, e muitas delas incluem o argumento skipna, que é por padrão definido como True, ignorando automaticamente os valores NA.

Segue-se a implementação em R da Eq. 7.4

```
calcular_j <- function(x) {
  j_min <- log10(x) - 1
  j_max <- log10(x)
```

```
  return(c(j_min, j_max))
}
```

Para preservar a coluna original "Mat", três novas colunas "N1", "N2" e "N3" são criadas no seguinte bloco de instruções que implementa o cômputo dos valores normalizados definidos, respectivamente, pelas Eqs. 7.1, 7.2 e 7.3.

```
dados_notas_full_n<-copy(dados_notas_full) #fazer uma copia independente
j_intervalo <- calcular_j(resMat$max) #computar o intervalo de j
j <- round(j_intervalo[2],digits=0)   #arredondar para um valor inteiro
mutate (dados_notas_full_n,
      N1 = ((Mat-resMat$min)/(resMat$max-resMat$min))*(1.0-0), #normalização 1
      N2 = (Mat-resMat$media)/resMat$std,  #normalização 2
      N3 = Mat/10^j     #normalização 3
)
```

Caso queira substituir os valores da coluna "Mat" por valores normalizados calculados pela Eq. 7.3, basta usar a seguinte instrução:

```
mutate(dados_nots_full_n, 'Mat' = ifelse('Mat' == Mat,  Mat/10^j, 'Mat'))
```

Em Python, a Eq. 7.4 é implementada com o seguinte bloco de instruções usando a função log10 do pacote math

```
import math
def calcular_j(x):
    j_min = math.log10(x) - 1
    j_max = math.log10(x)
    return j_min, j_max
```

e a criação de novas colunas com os valores normalizados em Python pode ser implementada com o seguinte bloco de códigos:

```
py_resMat = py_dados_notas_full.agg (
   {'Mat':['min','max','mean','std']})   #sumariza variavel Mat em min, max, mean e std
)
py_dados_notas_full_n=py_dados_notas_full.copy() #fazer uma copia independente
py_dados_notas_full_n["N1"]=((py_dados_notas_full_n['Mat']-py_resMat.loc['min','Mat'])/(py_res
py_dados_notas_full_n["N2"]=(py_dados_notas_full_n['Mat']-py_resMat.loc['mean','Mat'])/py_resM
py_dados_notas_full_n["N3"]=py_dados_notas_full_n['Mat']/int(10**j)
```

Em Python, a substituição dos valores da coluna "Mat" por valores normalizados calculados pela Eq. 7.3 pode ser feita pela seguinte linha de atribuição:

```
py_dados_notas_full_n['Mat'] = py_dados_notas_full_n['Mat']/(10**j)
```

7.3.4 Redução de Observações

A redução do volume de observações é uma estratégia essencial para lidar com grandes volumes de dados sem sacrificar a riqueza da informação. No entanto, ao empregar técnicas como discretização e amostragem, os praticantes de análise de dados podem enfrentar desafios ao tentar preservar a integridade essencial dos dados durante o processo de redução. As estratégias desenvolvidas visam aprimorar a eficiência computacional e facilitar análises subsequentes, sem comprometer a qualidade da informação contida nos conjuntos de dados.

A **discretização** é uma técnica comum para redução de observações, transformando variáveis contínuas em intervalos discretos (Seção 2.5.2 em [32]). As técnicas de discretização variam em como realizam a divisão, se usam informações da variável alvo (supervisão ou sem supervisão) e a direção do processo (cima para baixo ou baixo para cima):

discretização supervisionada: usa informações da **variável alvo** (**variável supervisora**) durante a divisão dos dados. Ou seja, a forma de dividir os dados contínuos é influenciada pela variável alvo, com o objetivo de otimizar a divisão para prever ou explicar melhor essa variável alvo.

discretização não supervisionada: sem levar em conta uma variável alvo (variável supervisora). Os dados são agrupados de acordo com a sua distribuição ou características, como a amplitude dos valores, ou o número de grupos desejados, sem se preocupar com uma variável de interesse.

discretização de cima para baixo (divisão): começa com um ou mais **pontos de divisão**, ou **pontos de corte**, para dividir a faixa dos valores do atributo (variável) e repete o processo recursivamente nos intervalos resultantes.

discretização de baixo para cima (fusão) considera todos os valores contínuos como pontos de divisão potenciais, mescla valores vizinhos para formar intervalos e repete o processo recursivamente.

Quanto ao princípio adotado em discretizações, destacam-se a discretização por intervalos (*binning*), análise de histograma, análise de *cluster*, análise de árvore de decisão e análise de correlação. Dentre essas, a discretização por análise de *cluster*, análise de árvore de decisão e análise de correlação são técnicas supervisionadas, pois levam em conta a estrutura de classe dos dados.

Análise de *Cluster*: Agrupa os dados com base em semelhanças (Seção 6.4).

Análise de Árvore de Decisão: Usa a classe como critério para dividir os dados em subgrupos.

Análise de Correlação (Seção 6.2): Embora não seja diretamente uma técnica de discretização, identifica relações entre variáveis, o que pode ser relevante para a discretização em alguns contextos.

Essas técnicas exigem um bom conhecimento dos dados e interação com os analistas, que são responsáveis por decidir qual técnica aplicar, com base em sua experiência e intuição. O suporte computacional oferece ferramentas para clusterização, construção de árvores de decisão e análise de correlação, mas a escolha da abordagem fica a cargo dos especialistas.

Por outro lado, a discretização por intervalos (*binning*) e a análise de histograma (Seção 6.1.1) são técnicas não supervisionadas, pois não dependem das informações de classe dos dados. A técnica de ***binning*** divide o intervalo contínuo dos dados em intervalos discretos ou *bins* com base nos valores observados, sem considerar as classes. Já a **análise de histograma** pode ser vista como uma forma de *binning*, pois também agrupa os dados em intervalos e conta as ocorrências. A principal diferença é que o histograma apresenta visualmente a distribuição dos dados, enquanto o *binning* gera uma representação tabular da frequência dos dados em cada intervalo.

Em técnicas não supervisionadas, os intervalos podem ser definidos de duas maneiras:

Largura igual (*equal-width*): Os intervalos têm a mesma largura.

Frequência igual (*equal-frequency*): Os intervalos são criados de forma que cada um tenha aproximadamente o mesmo número de amostras.

Após a discretização, cada valor do intervalo pode ser substituído pela média ou mediana do intervalo, processo conhecido como suavização por médias de intervalo ou suavização por medianas de intervalo, respectivamente.

Outra abordagem para reduzir o volume de dados é a **amostragem** (Seção 2.5.4 em [32]), considerada eficaz especialmente em grandes conjuntos, onde o processamento de todas as observações pode ser impraticável. Essa técnica envolve a seleção de um subconjunto representativo dos dados. O custo de obter uma amostra é proporcional ao tamanho da amostra n, e não ao número total de observações N. Por isso, a complexidade da amostragem tende a ser sublinear em relação ao tamanho dos dados. Isso contrasta com outras técnicas de redução de dados, que podem exigir pelo menos uma passagem completa por todas as N observações. Quando o número de atributos (características ou variáveis) k aumenta, a complexidade da amostragem cresce linearmente, enquanto técnicas como histogramas podem ter um aumento exponencial na complexidade, devido à expansão do espaço de variáveis.

A amostragem é comumente usada para estimar respostas a uma **consulta sumarizada**. Aplicando o **Teorema Central do Limite**, é possível determinar o tamanho da amostra, com n observações, necessário para estimar o valor médio de uma característica (variável) com um erro espe-

cificado e um nível de confiança determinado. O teorema afirma que, para um grande número M de amostras, cada uma com n observações, a distribuição das médias dessas M amostras tende a ser normal, independentemente da distribuição subjacente dos dados. Isso permite fazer inferências estatísticas sobre a média populacional de cada variável a partir de um número reduzido de observações, conforme abordado no Capítulo **??**.

Embora a estatística de inferência seja uma abordagem comum para reduzir volumes de dados, pois permite extrair conclusões sobre uma população com base em uma amostra representativa, as **estatísticas descritivas** também desempenham um papel importante na redução da quantidade de dados. Ao contrário da inferência, que se foca em estimar parâmetros populacionais a partir de uma amostra, as estatísticas descritivas resumem e descrevem as características essenciais de um conjunto de dados, como média, mediana, moda, variância e desvio padrão. Essa abordagem permite categorizar, agrupar ou resumir os dados de forma que se possam obter *insights* significativos sem a necessidade de analisar cada dado individualmente. Em muitos casos, essa **resumização** pode fornecer uma visão geral do conjunto de dados, permitindo simplificar a análise sem perder informações essenciais.

Tanto R quanto Pyhton oferecem funções de categorização dos valores numéricos, e sumarização das observações agrupadas. A **categorização** se refere à capacidade de reduzir dados por meio de operações que categorizam valores numéricos, e o **agrupamento com sumarização** agrega observações e realiza operações de estatísticas resumidas sobre os valores agrupados com base em critérios específicos.

Em R, as funções cut() (divisão por valores)/qcut() (divisão por quantis) são utilizadas para criar intervalos e categorizar um conjunto de valores numéricos em sub-intervalos, com o extremo esquerdo aberto e o extremo direito fechado. Essa transformação converte uma variável do tipo numérico para o tipo fator (Seção 2.2.2). Além disso, R oferece a função group_by(), que, em conjunto com a função summarize() (Seção 7.3.3), permite reduzir as observações de uma tabela, agrupando-as com base em um conjunto predefinido de variáveis, e calcular estatísticas resumidas para os valores da variável de interesse em cada grupo.

O exemplo a seguir ilustra a aplicação dessas técnicas, categorizando os valores nas colunas "Mat" e "Port" da tabela dados_notas_full em conceitos e, em seguida, agrupando em estatísticas de resumo os valores de "Port" por conceito na nova tabela dados_notas_agrupados:

```
intervalos <- c(0,60,70,80,90,100)             #valores numericos
conceitos <- c("Reprovado","Regular","Bom","Ótimo","Excelente")
         #conceitos correspondentes
dados_notas_full$catPort <- cut(dados_notas_full$Port, breaks=intervalos, labels=conceitos)
         #numerico->categorico
dados_notas_full$catPort <- cut(dados_notas_full$Port, breaks=intervalos, labels=conceitos)
dados_notas_agrupados <- group_by(dados_notas_full, catPort),  #agrupamento por conceito
```

```
        summarize(dados_notas_agrupados,                    #de português
           mediaPort=mean(Port, na.rm=TRUE).                #média por conceito
           stdPort=sd(Port, na.rm=TRUE)                     #desvio-padrão por conceito
        )
```

Segue-se a tradução do bloco de instruções em R para Python. Note que as funções cut() e group_by() correspondem às funções cut()/cut() e groupby() do pacote pandas em Python.

```
import pandas as pd


intervalos = [0, 60, 70, 80, 90, 100]
conceitos = ["Reprovado", "Regular", "Bom", "Ótimo", "Excelente"]
py_dados_notas_full['catPort'] = pd.cut(py_dados_notas_full['Port'], bins=intervalos,
        labels=conceitos)
py_dados_notas_full['catMat'] = pd.cut(py_dados_notas_full['Mat'], bins=intervalos,
        labels=conceitos)
py_dados_notas_agrupados_full = py_dados_notas_full.groupby('catPort').agg(['mean','std'])
        # as estatísticas de resumo são calculadas para todos os valores numéricos
py_dados_notas_agrupados = py_dados_notas_agrupados_full['Port']
        #selecionar apenas os resumos de interesse
```

Podemos visualizar a distribuição dos dados em função da variável numérica "Port" através de um histograma, usando a função ggplot() do pacote ggplot2 (R), ou plotnine (Python) como mostra o seguinte trecho de códigos:

```
ggplot (data = dados_notas_full +
    geom_histogram (mapping = aes(x=Port), binwidth = 5)

py_dados_notas_full['Port'].hist(bins=5) #renderizar histograma de Port
```

7.3.5 Redução de Dimensionalidade

A dimensionalidade de dados se refere à quantidade de atributos, características, variáveis ou dimensões que estão presentes em um conjunto de dados. No contexto de dados relacionais, a redução de dimensionalidade visa identificar e **remover variáveis** ou dimensões irrelevantes, fracamente relevantes ou redundantes, para eliminar a complexidade desnecessária nos dados e permitir que os analistas se concentrem nos elementos mais significativos (Seção 2.6 em [32]). Essa abordagem é particularmente valiosa para analistas de dados, pois proporciona uma maneira mais acessível de explorar e interpretar padrões em conjuntos de dados com uma grande quantidade de características (variáveis). O resultado são *insights* mais claros e eficazes. No entanto, o desafio surge no tamanho do espaço de

busca, que pode se tornar excessivamente amplo. Para n variáveis, a busca por um subconjunto ótimo dentre as 2^n possíveis combinações pode se tornar proibitivamente demorada.

Entre os métodos de redução de dimensionalidade, destaca-se a **Análise de Componentes Principais** (em inglês *Principal Component Analysis* – PCA). O **PCA** é uma técnica linear que projeta os dados em um espaço de dimensões reduzidas, preservando a maior parte da variância original. O processo consiste em padronizar os dados, aplicando por exemplo a normalização por escore-z (Eq. 7.2), e calcular os autovalores e autovetores da matriz de covariância entre as variáveis. Oa autovetores e autovalores representam, respectivamente, as direções principais e as variâncias ao longo dass respectivas direções. Tipicamente, considera-se que quanto maior a contribuição de uma direção, mais informação ele retém sobre a distribuição dos dados originais. Portanto, os primeiros autovetores correspondentes aos maiores autovalores são escolhidos como os componentes principais que definem novos espaços sobre os quais os dados são projetados.

O PCA é frequentemente usado como uma etapa de pré-processamento em problemas de aprendizado de máquina para reduzir a dimensionalidade dos atributos (características ou variáveis). Isso pode simplificar a entrada para algoritmos de aprendizado de máquina, reduzir a complexidade computacional e remover multicolinearidade entre as características. Tanto R quanto Python oferecem funções robustas para realizar redução de dimensionalidade. Em R, a função `prcomp()` do pacote básico realiza a Análise de Componentes Principais (PCA) [24], enquanto em Python, a função `PCA()` do módulo `sklearn.decomposition` executa a mesma técnica [82].

Apesar da popularidade do PCA na redução de dimensionalidade, as técnicas de aprendizado de máquina se destacam por sua capacidade de identificar padrões e estruturas complexas nos dados. Ao extrair características (variáveis) informativas, essas técnicas preservam as informações essenciais para análises e modelagens subsequentes. Um exemplo clássico de técnica de aprendizado de máquina que pode identificar padrões e extrair características informativas é o *Autoencoder*. O ***Autoencoder*** é uma rede neural projetada para aprender uma representação compacta (ou codificada) dos dados, o que pode ser útil para redução de dimensionalidade, além de identificar e extrair características relevantes dos dados de entrada para tarefas subsequentes, como classificação ou análise de *clusters*. Ao contrário do PCA, que apenas projeta os dados em componentes principais, o *Autoencoder* aprende representações não-lineares dos dados, o que pode ser mais eficaz em capturar padrões complexos que o PCA pode não identificar.

7.4 Considerações Finais

Neste capítulo dedicado à preparação de dados na Ciência de Dados, exploramos de maneira abrangente três fases cruciais: análise exporatória, reorganização e reestruturação, e transformação de dados. Cada fase tem a função de garantir que os dados estejam prontos para análise. Ao longo do capítulo,

proporcionamos uma visão prática, fornecendo exemplos e ferramentas tanto em R quanto em Python predominantemente com base em [101, 37][2], para capacitar os profissionais de dados a enfrentar desafios e extrair o máximo de informação nos seus conjuntos de dados. Cabe ressaltar que o capítulo foca em **dados *tidy*** organizados em estruturas tabulares. Existem **estruturas hierárquicas** (árvores) e **grafos**, amplamente aplicados, que podem demandar abordagens distintas de preparação de dados. R e Python tem bilbiotecas, como igraph (R/Python), tree (R) e scikit-learn (Python), que facilitam a manipulação, análise e visualização de dados em formato de árvores e grafos, tornando mais fácil a preparação de dados para análises específicas, mas não serão exploradas no contexto deste texto.

Na Seção 7.1, mostramos a importância de realizar uma boa **análise exploratória dos dados** (EDA) em Ciência de Dados. A EDA permite adquirir um melhor conhecimento dos dados e gerar *insights* que podem orientar a modelagem dos dados, impactando diretamente o desempenho dos algoritmos de análise, como os de aprendizado de máquina. Além disso, é importante destacar que as funções fornecidas pelos bancos de dados relacionais são igualmente valiosas no processo de limpeza e organização dos dados, devido à sua eficiência de processamento, capacidade de lidar com grandes volumes de dados, integridade dos dados, recursos de transformação e suporte a transações atômicas.

A **reorganização e reestruturação dos dados** (Seção 7.2) compreendem a segunda fase do processo, onde nos concentramos na limpeza, formatação em *tidy* e integração dos dados. Mostramos, por meio de exemplos, como organizar os dados de maneira eficiente, garantindo sua consistência e tornando-os adequados para análises mais avançadas. Apresentamos algumas das diversas ferramentas disponíveis, como pacotes em R (dplyr, tidyr) e bibliotecas em Python (pandas, numpy), que facilitam essa etapa. Vale ressaltar que as três etapas, limpeza, formatação e integração, não ocorrem necessariamente em uma sequência rígida, podendo se entrelaçar dependendo da situação. Por exemplo, após a integração de dados, podem surgir valores NA, que precisariam ser removidos. As ferramentas fornecem uma variedade de funções e alternativas, cabendo ao analista de dados combiná-las de maneira eficaz para realizar tarefas específicas.

A **transformação de dados** (Seção 7.3), nossa terceira fase, busca a padronização e simplificação dos dados. Isso é feito por meio de processos como filtragem, redução de observações ou diminuição de variáveis, sempre com o cuidado de preservar as informações essenciais para a análise desejada. A complexidade dessa etapa está na diversidade de técnicas de aprendizado de máquina aplicadas para resolver problemas nos quais soluções analíticas tradicionais não são viáveis. Embora o foco deste texto seja a preparação de dados, apresentamos algumas funções simples disponíveis tanto em R quanto em Python, ilustrando partes dessa tarefa complexa e mostrando como preparar os dados para análises mais avançadas. Vale lembrar que as técnicas de aprendizado de máquina não são temas deste texto.

[2]Alguns ajustes foram necessários para compatibilizar com a versão de python (2.7.12), ipython (7.9.0), R (4.3.2) e RStudio (4.3.2) instalados no meu ambiente de testes.

7.5 Exercícios

1. Execute o notebook e leia o texto explicativo. Sintetize os principais passos de preparação de dados para aprendizado de máquina. Compare-os com os passos apresentados neste capítulo.

2. Reproduza todos os exemplos apresentados no Capítulo 10 de [102] se estiver utilizando R, ou no Capítulo correspondente em [37] se estiver utilizando Python, conforme a linguagem escolhida. Para cada exemplo reproduzido, comente brevemente uma informação que você extraiu dos dados.

3. Faça todos os itens do *Learning Check* do Capítulo 3 em [36].

Capítulo 8

Probabilidade

A estatística descritiva, como discutido no Capítulo 6, fornece uma variedade de medidas estatísticas que resumem os valores de uma variável, ou uma característica, em uma população. No entanto, na prática, é comum se deparar com dificuldades, ou até mesmo impossibilidades, de obter todos os dados, o que introduz uma incerteza inerente à representação dos valores disponíveis das características ao calcular tais medidas.

Na análise de dados, a **probabilidade** emerge como uma ferramenta que nos permite compreender, quantificar e computar as variáveis de uma população, considerando a incerteza inerente aos fenômenos observados. Isso é alcançado ao associarmos a cada possível observação uma probabilidade correspondente à sua ocorrência. Embora muitos possam ver a probabilidade e a estatística como disciplinas intercambiáveis, é fundamental reconhecer que são campos distintos, apesar de serem profundamente interligados. A probabilidade é considerada uma disciplina autônoma das ciências, podendo complementar a estatística descritiva no tratamento da aleatoriedade, fornecendo um arcabouço teórico e prático para lidar com a incerteza nos dados coletados e inferir informações plausíveis desses dados.

A probabilidade é uma medida da chance de ocorrência de um resultado em um experimento ou fenômeno sujeito à aleatoriedade. Essa medida é expressa como um número entre 0 e 1, onde valores mais próximos de 1 indicam uma maior certeza na ocorrência do evento. Para explorar a relação entre probabilidade e estatística descritiva, é essencial primeiro estabelecer conceitos fundamentais de probabilidade. Na Seção 8.1, abordamos esses conceitos básicos. Em seguida, na Seção 8.2, discutimos as operações fundamentais para calcular e manipular probabilidades, que são cruciais para compreender e aplicar princípios probabilísticos em análises estatísticas de inferência. Na Seção 8.3, definimos variáveis aleatórias e exploramos seu papel na quantificação das incertezas nos dados coletados de uma população. Na Seção 8.3.1, apresentamos as funções de distribuição de probabilidades associadas aos eventos em um espaço de probabilidade, que estão intimamente ligadas à modelagem de incertezas nos valores das variáveis aleatórias. A seguir, na Seção 8.4, discutimos uma das mais

importantes funções de distribuição: a função de distribuição das médias amostrais. Esta função é fundamental para a estatística inferencial e está no cerne do Teorema Central do Limite. Na Seção 8.3.2, adotamos uma abordagem probabilística para calcular as estatísticas descritivas discutidas no Capítulo 6. Em seguida, na Seção 8.5, exploramos modelos de probabilidade comumente utilizados para atribuir probabilidades aos valores de uma variável aleatória. Por fim, na Seção 8.6, introduzimos uma metodologia de simulação de experimentos sob diversas condições, permitindo capturar a variabilidade inerente a um sistema e compreender seu comportamento.

8.1 Conceitos Básicos

Quando recorremos à análise de dados a partir de um subconjunto de **observações representativas** de uma população, geralmente de dimensões consideravelmente menores, para fins de estudo, dizemos que estamos conduzindo um **experimento aleatório** ou observando um **fenômeno aleatório**, utilizando as amostras de uma população [31]. Uma **amostra** é um conjunto de observações, ou instâncias de medição, obtidas num experimento aleatório, num estudo estatístico ou numa pesquisa. O **espaço amostral** é o conjunto de todos os elementos possíveis de um experimento aleatório, frequentemente representado pela letra grega Ω. Em geral, a letra grega minúscula ω é usada para representar um resultado específico de um experimento aleatório.

Os subconjuntos do espaço amostral, que representam resultados específicos ou combinações dos resultados de um experimento, são denominados **eventos**. Esses eventos são tipicamente representados por letras latinas maiúsculas, como A, B, $\cdots$, com o conjunto vazio representado por $\emptyset$. Por exemplo, considere o lançamento de um dado. O espaço amostral é o conjunto $\Omega = \{1, 2, 3, 4, 5, 6\}$, onde cada elemento representa um possível resultado do experimento (o número que aparece na face superior do dado). Os eventos podem ser diversos, como $A = \{2, 4, 6\}$ ("obter um número par") ou $B = \{1, 3, 5\}$ ("obter um número ímpar"). Cada um desses eventos é um subconjunto do espaço amostral Ω que representa uma característica específica do resultado do experimento.

Uma amostra, em termos de eventos, pode ser considerada uma coleção de observações específicas de eventos de um espaço amostral. Estes eventos podem ou não repetir-se durante um experimento aleatório. Por exemplo, em uma amostra de lançamentos de dados, pode haver repetições de resultados, como obter "6" mais de uma vez em uma amostra. Para um experimento com N observações, dizemos que é uma amostra de tamanho N, onde a frequência de ocorrência de cada evento pode variar de acordo com a natureza do fenômeno físico de interesse. Isso significa que diferentes eventos dentro da amostra podem ocorrer com frequências diferentes, refletindo a distribuição de probabilidade do fenômeno em questão. Por exemplo, em um experimento de lançamento de um dado, se quisermos uma amostra de tamanho N, a frequência de ocorrência de cada número (evento) de "1" a "6" pode variar dependendo da aleatoriedade do processo. Essas frequências observadas na amostra podem

então ser usadas para fazer inferências sobre a distribuição de probabilidade subjacente do fenômeno físico de interesse.

O conjunto $\mathcal{F}$ que compreende todos os eventos possíveis de um experimento, acompanhado de uma **medida de probabilidade** P atribuída a esses eventos, define um **espaço de probabilidades** $(\Omega, \mathcal{F}, P)$ do experimento. A **medida de probabilidade**, ou **probabilidade de evento**, é, por sua vez, uma função $P : \mathcal{F} \longrightarrow [0, 1]$ que atribui valores numéricos que expressam a chance de ocorrência de cada evento $A \in \mathcal{F}$ do espaço amostral Ω em um experimento aleatório, de maneira que os seguintes três **axiomas de Kolmogorov** sejam satisfeitos:

1. $0 \leq \mathrm{P(A)} \leq 1, \forall\ \mathrm{A} \in \ \ \Omega$.
2. $\mathrm{P}(\Omega) = 1$.
3. $\mathrm{P}(\cup_{j=1}^{n} A_j) = \sum_{j=1}^{n} P(A_j)$, sendo A_j eventos mutuamente exclusivos.

Existem duas abordagens principais para atribuir uma probabilidade a eventos no espaço amostral. Ela pode ser feita com base em características teóricas da realização do experimento aleatório, ou através das frequências de ocorrência de cada valor da variável de interesse em diversas repetições do experimento em que ocorre a variável [52]. Na abordagem de **modelo teórico**, a probabilidade de cada evento é atribuída com base em considerações teóricas sobre a natureza do experimento aleatório. Isso pode envolver modelos matemáticos, leis físicas, ou outras características fundamentais do sistema em estudo. Por exemplo, ao lançar um dado justo, onde todas as faces têm a mesma chance de ocorrer, podemos atribuir uma probabilidade igual de $\frac{1}{6}$ a cada uma delas. Na Seção 8.5 são apresentados alguns modelos matemáticos usados na representação das incertezas dos eventos. Na abordagem de **frequência empírica**, a probabilidade de cada evento é determinada pela **frequência relativa** de sua ocorrência em um grande número de repetições do experimento. Ou seja, a probabilidade é estimada com base em observações empíricas. Por exemplo, ao lançar um dado várias vezes e contar o número de vezes que cada face aparece, podemos usar essas frequências para estimar as probabilidades de ocorrência de cada uma das faces.

8.2 Operações Fundamentais de Probabilidade

Para construir um espaço de probabilidades $(\Omega, \mathcal{F}, P)$ de um experimento aleatório a partir das medidas de probabilidade P de uma coleção de eventos $\mathcal{F}$, a estrutura algébrica σ-álgebra é amplamente utilizada. Uma coleção $\mathcal{F}$ de subconjuntos de Ω é uma σ-álgebra sobre o espaço amostral Ω se, e somente se, $\mathcal{F}$ possui as seguintes propriedades:

Propriedade do universo: $\Omega \in \mathcal{F}$.

Propriedade de complementação: Se um evento $A \in \mathcal{F}$, então o seu evento complementar $A^c \in \mathcal{F}$.

Propriedade de fechamento sob união enumerável: Se os eventos $A_i \in \mathcal{F}$ para qualquer i, então a união dos eventos $\cup_{i=1}^{\infty} A_i \in \mathcal{F}$.

Dessas propriedades, derivam-se duas propriedades úteis:

1. Das propriedades de universo e complementação, segue-se $\emptyset \in \mathcal{F}$.
2. Das propriedades de complementação e de fechamento sob união enumerável, segue-se que a σ-álgebra $\mathcal{F}$ também é fechada sobre intrsecções via leis de De Morgan.

Note que as operações definidas sobre Ω na σ-álgebra $\mathcal{F}$ são as seguintes operações de conjunto:

União de Eventos ($\cup$): A união de dois eventos, denotada por $A \cup B$, consiste em todos os resultados que pertencem a pelo menos um dos eventos A ou B.

Interseção de Eventos ($\cap$): A interseção de dois eventos, denotada por $A \cap B$, consiste em todos os resultados que pertencem simultaneamente aos eventos A e B.

Eventos Complementares: O complemento de um evento A, A^c ou $\overline{A}$, consiste em todos os resultados que não pertencem ao evento A e $A \cup A^c$ abrange todo o espaço amostral.

Essas **operações de conjunto** permitem a combinação e manipulação de eventos para calcular as probabilidades de eventos compostos e analisar as relações probabilísticas entre diferentes eventos em um experimento aleatório, ao aplicarmos as seguintes propriedades de probabilidade, que são análogas às propriedades de conjuntos [52]:

Probabilidade do Evento: Probabilidades dos eventos sempre estarão entre (e incluindo) 0 e 1. Uma probabilidade de 0 significa que o evento é impossível. Uma probabilidade de 1 significa que um evento é garantido de acontecer. Uma probabilidade próxima a 0 significa que o evento é "pouco provável" e uma probabilidade próxima a 1 significa que o evento é "altamente provável" de ocorrer. Denotamos a probabilidade do evento A como P(A) e

$$0 \leq P(A) \leq 1. \tag{8.1}$$

Probablidade do Complemento: A probabilidade do complemento A^c de um evento A é igual a 1 menos a probabilidade de A, ou seja,

$$P(A^c) = 1 - P(A). \tag{8.2}$$

Probablidade do Conjunto Vazio: A probabilida de um conjunto vazio é 0, ou seja,

$$P(\emptyset) = 0. \tag{8.3}$$

Probabilidade de União de Eventos: Sejam A e B eventos de Ω, então a probabilidade da união de A e B é

$$P(A \cup B) = P(A) + P(B) - P(A \cap B). \tag{8.4}$$

Propriedade de Eventos Independentes: Dois eventos A e B são **independentes** se a probabilidade do evento A ocorrer não é afetada pela ocorrência ou não ocorrência do evento B, e vice-versa, ou seja,

$$P(A \cap B) = P(A)P(B). \tag{8.5}$$

Propriedade de Eventos Mutuamente Exclusivos: Dois eventos são **mutuamente exclusivos**, se a ocorrência de um evento impede a ocorrência do outro, ou seja, $A \cap B = \emptyset$. Portanto,

$$P(A \cap B) = 0 \quad \text{e} \tag{8.6}$$

$$P(A \cup B) = P(A) + P(B). \tag{8.7}$$

Quando lidamos com eventos que não são independentes ou não são mutuamente exclusivos, recorremos frequentemente à técnica da **probabilidade condicional** para calcular a chance de um evento ocorrer, tendo conhecimento da ocorrência de outros eventos. Tomemos dois eventos A e B, a probabilidade condicional de A dado que B ocorreu, denotado por $P(A|B)$, é

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \quad P(B) > 0. \tag{8.8}$$

Combinando as Eqs. 8.5 e 8.8, podemos observar que, em eventos independentes, a probabilidade condicional de A dado B, ou de B dado A, é simplesmente igual à probabilidade do próprio evento. Isso ocorre porque a ocorrência do outro evento não afeta a sua própria ocorrência, como expressam as seguintes igualdades:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A)P(B)}{P(B)} = P(A), \quad P(B) > 0$$
$$P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{P(A)P(B)}{P(A)} = P(B), \quad P(A) > 0$$

De forma análoga, ao combinar as Eqs. 8.6 e 8.8, podemos concluir que a probabilidade condicional entre dois eventos mutuamente exclusivos é

$$P(A|B) = P(B|A) = \frac{P(A \cap B)}{P(B)} = \frac{P(A \cap B)}{P(A)} = 0 \quad P(A) > 0 \text{ e } P(B) > 0.$$

Isso acontece porque a ocorrência de um evento mutuamente exclusivo torna impossível a ocorrência do outro evento, como indicado por $P(A \cap B) = 0$.

Tanto em R quanto em Python, é possível associar probabilidades aos eventos. O seguinte trecho de código em R exemplifica a associação dos seis eventos correspondentes às seis faces (Face 1, Face 2, Face 3, Face 4, Face 5 e Face 6) de um dado justo às suas respectivas probabilidades de ocorrência, definidas no vetor prob_evento. As correspondências estabelecidas são armazenadas na estrutura de dados probabilidades:

```
prob_evento <- rep(1/6, 6)
nomes_eventos <- c("Face 1", "Face 2", "Face 3", "Face 4", "Face 5", "Face 6")
probabilidades <- data.frame(Evento = nomes_eventos, Probabilidade = prob_evento)
```

Um código equivalente em Python que associa cada evento do lançamento de um dado justo a uma probabilidade e armazena as correspondências em probabilidade_evento é apresentado a seguir:

```
espaco_amostral = ["Face 1", "Face 2", "Face 3", "Face 4", "Face 5", "Face 6"]
probabilidade_evento = {"Face 1": 1/6, "Face 2": 1/6, "Face 3": 1/6,
"Face 4": 1/6, "Face 5": 1/6, "Face 6": 1/6}
```

8.3 Variáveis Aleatórias

Uma **variável aleatória** X é uma função matemática que atribui a cada observação individual ω do espaço amostral Ω de um experimento aleatório, com determinada medida de probabilidade P, um valor numérico $X(\omega)$. Em outras palavras, X transforma uma observação individual do espaço de probabilidade $(\Omega, \mathcal{F}, P)$ em um valor numérico. Esse valor numérico atribuído pela variável aleatória representa um conceito de interese para o contexto do problema em questão. A convenção usual para representar uma variável aleatória consiste em usar letras latinas maiúsculas como X e Y. Um valor específico desta variável é representado pela letra latina minúscula correspondente, como x e y. Observe que utilizamos a mesma notação das variáveis populacionais introduzidas no Capítulo 6 para representar as variáveis aleatórias, a menos que isso possa gerar ambiguidade no texto. Quando necessário, distinguimos as variáveis aleatórias das variáveis populacionais adicionando o símbolo $\sim$ sobre as letras.

A **quantificação dos resultados de um experimento facilita a análise estatística e a modelagem probabilística de fenômenos incertos**. Por exemplo, se estivermos estudando a altura das pessoas em uma determinada população, podemos definir uma variável aleatória X que mapeia cada indivíduo ω para sua altura $X(\omega) = x$ em centímetros. Aqui, a altura de cada pessoa é o **conceito de interesse**. Da mesma forma, em um experimento de lançamento de moedas, podemos definir uma variável aleatória Y que mapeia cada resultado w do lançamento (cara ou coroa) para um

valor numérico, como 1 para cara e 0 para coroa. Neste caso, o valor numérico y representado pela função $Y(\omega)$, que pode assumir 1 ou 0, é o conceito de interesse. Por exemplo, $(Y(\omega) = 1)$ indica que o evento ocorreu (cara), enquanto $(Y(\omega) = 0)$ indica que não ocorreu cara (coroa).

Dentro dos valores numéricos aos quais as variáveis aleatórias são associadas, distinguimos dois tipos principais: variáveis aleatórias discretas e variáveis aleatórias contínuas. As **variáveis aleatórias discretas** são aquelas que assumem um conjunto enumerável de possíveis resultados. Por exemplo, o resultado de lançar um dado é uma variável aleatória discreta, pois só pode assumir valores específicos, como 1, 2, 3, 4, 5 ou 6. Já as **variáveis aleatórias contínuas** assumem um conjunto não-enumerável de valores dentro de um intervalo em uma reta real. Por exemplo, a altura de uma pessoa é uma variável aleatória contínua, pois pode assumir qualquer valor dentro de um intervalo contínuo, como entre 1,50m e 2,00m.

Podemos classificar as variáveis aleatórias com base na quantidade de características envolvidas. Quando uma variável aleatória envolve apenas um valor escalar, ela é denominada **variável aleatória univariada**. Isso significa que a variável aleatória pode assumir apenas um único valor em cada realização do experimento aleatório. Por exemplo, no estudo da altura de uma pessoa em uma amostra, a variável aleatória correspondente seria univariada, pois estamos medindo apenas uma característica (altura) em cada observação. As variáveis aleatórias univariadas são frequentemente representadas por letras latinas maiúsculas, como X, Y, Z, etc. Por outro lado, se estivermos interessados em fazer inferências sobre múltiplas variáveis relacionadas, como altura e peso, podemos modelá-las como uma **variável aleatória multivariada** e representá-las como um vetor aleatório cujos elementos são as características individuais. Isso nos permite modelar a relação conjunta entre essas variáveis e realizar análises estatísticas mais complexas, a fim de examinar não apenas as características individuais, mas também como elas se relacionam entre si, oferecendo uma compreensão mais abrangente do conjunto de dados.

8.3.1 Funções de Distribuição de Variáveis Aleatórias

A **função de probabiliudade** P do espaço de probabilidades $(\Omega,\mathcal{F},P)$ é uma função que atribui probabilidades aos eventos definidos na sigma-álgebra $\mathcal{F}$. Como um evento pode incluir múltiplos valores $x \in X$ da variável aleatória, é necessário distribuir a probabilidade associada a cada evento entre os possíveis valores que pertencem a esse evento, por meio da **função de distribuição** $f(x)$. Esta função atribui a cada valor x uma medida de probabilidade $p(X = x)$, respeitando as seguintes propriedades:

1. $f(x) \geq 0$, $\forall x \in X$ (não-negatividade), e
2. a soma de todas as probabilidades $p(X = x)$ para todos os valores possíveis em X é igual a 1.

Dessa forma, se associarmos às variáveis representadas em uma tabela *tidy* com distribuições de

probabilidades, podemos considerá-las como variáveis aleatórias e analisá-las de forma semalhante às variáveis, como vimos nos capítulos anteriores.

Com base no tipo de variáveis aleatórias, a função de distribuição é classificada em função de massa de probabilidade e função de densidade de probabilidade[1]. A **função de massa de probabilidade** (PMF, do inglês *probability mass function*) é utilizada para variáveis aleatórias discretas. Ela descreve a relação entre cada valor discreto $x_i \in X$ e sua respectiva probabilidade, ou seja, é uma função $f(x)$ que atende às seguintes condições:

1. $0 \leq f(x_i) \leq 1$, e
2. $f(x_i) = p(X = x_i)$.

No caso de uma variável aleatória contínua X, envolvendo um conjunto infinito de valores, a chance de obter um valor específico é extremamente baixa, praticamente zero. Portanto, atribuir probabilidades pontuais para valores individuais, como a PMF, é impraticável. Ao invés da PMF, a **função de densidade de probabilidade** (em inglês *probability density function* – PDF) é amplamente utilizada para descrever como a densidade de probabilidade está distribuída ao longo de um intervalo da variável aleatória, tal que a probabilidade $p(c \leq X \leq d)$ de um intervalo $[c, d] \in X$ seja a área definida por $f(x)$, como ilustra a Figura 8.1:

$$p(c \leq X \leq d) = \int_c^d f(x)dx \tag{8.9}$$

A PDF satisfaz, portanto, duas propriedades:

1. $f(x) \geq 0, \quad \forall x \in (-\infty, \infty)$, e
2. $\int_{-\infty}^{\infty} f(x)dx = 1$

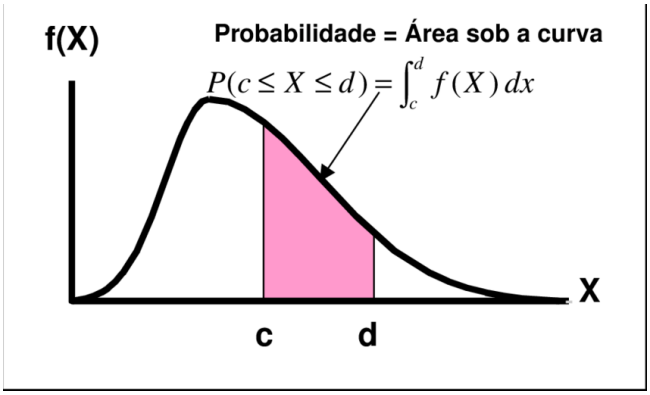
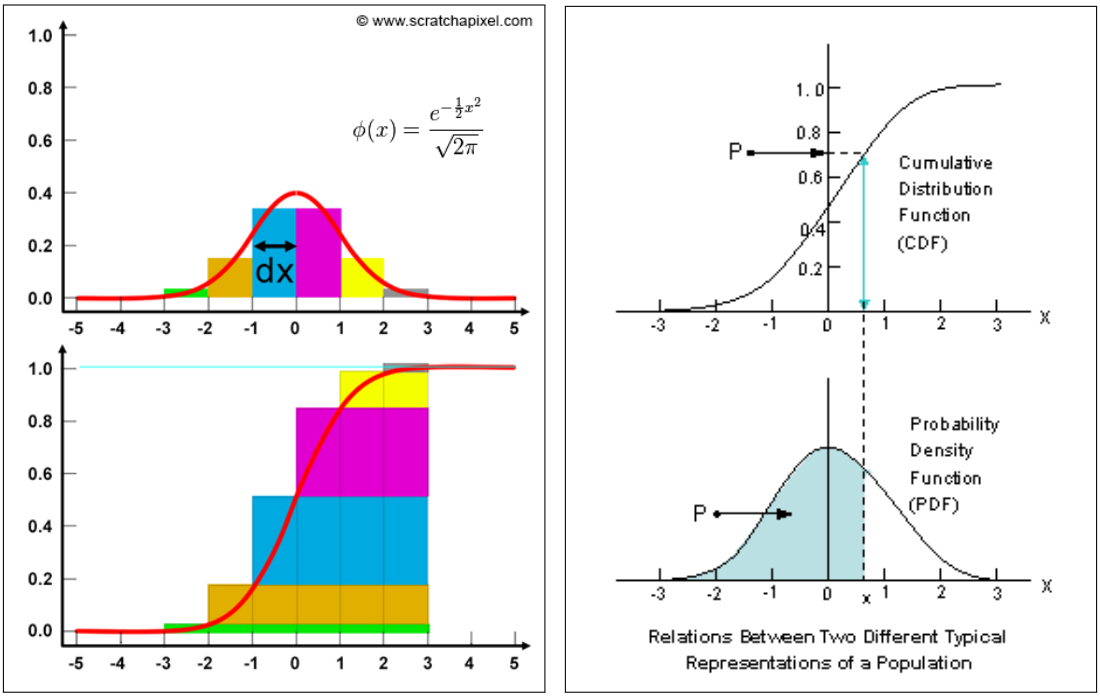


Figura 8.1: Função de densidade de probabilidade $f(x)$ e probabilidade $p(c \leq X \leq d)$. (Fonte: Researchgate)

[1]Muito autores usam **função de probabilidade** para se referir à distribuição de probabilidade discreta e **função de densidade de probabilidade** para se referir à distribuição de probabilidade contínua. Para evitar confusão, utilizamos, respectivamente, função de massa e função de densidade de probabilidade.



(a) Discreta (b) Contínua

Figura 8.2: Função de distribuição acumulada $CDF(x)$: (a) Cada degrau em x_i corresponde a $p(X = x_i)$. $CDF(x)$ equivale a $p(X \leq x) = \sum_{x_i \leq x} p(X = x_i)$, a área definida pela escada da função de massa de probabilidade (PMF) (Fonte: Scratchpixel), (b) $CDF(x)$ equivale a $p(X \leq x) = \int_{-\infty}^{x} f(x)dx$, a área definida pela curva da função de densidade de probabilidade (PDF) (Fonte: Stackexchange).

A partir da função de distribuição $f(x)$, podemos definir a **função de distribuição acumulada** (em inglês *cumulative distribution function* – CDF) $F(x)$ que representa a probabilidade acumulada de que a variável aleatória X seja menor ou igual a um determinado valor x, ou seja,

$$F(x) = p(X \leq x) = \sum_{x_i \leq x} f(x_i) = \sum_{x_i \leq x} p(X = x_i), \tag{8.10}$$

onde x_i são todos os valores de X menores ou iguais a x. Esta função fornece uma visão mais completa da distribuição de probabilidade da variável aleatória X, que pode ser tanto discreta quanto contínua. Para variáveis discretas, a CDF é uma função de escada que salta apenas nos valores possíveis da variável aleatória, enquanto para variáveis contínuas, a CDF é uma função contínua, não uma função de escada como vimos na Seção 6.1.1, e fornece a probabilidade acumulada para todos os valores de $x \in X$ no intervalo real como apresentada na Figura 8.2. A Seção 8.5 apresenta algumas funções de probabilidade mais aplicadas para descrever o comportamento das variáveis aleatórias em situações práticas.

Em situações reais, as observações sobre uma população são quase sempre multidimensionais, isto é, relacionadas a várias características ou variáveis. Por exemplo, em registros médicos eletrônicos, cada paciente pode ser representado por várias características, como idade, sexo, pressão arterial, níveis de colesterol, etc. Na Seção 6.2, vimos que, em situações envolvendo observações bidimensionais com

duas variáveis X e Y, organizamos todos os pares (x, y) que ocorrem em uma tabela de frequência de ocorrência desses pares. Para aplicar conceitos de probabilidade, é necessário considerar essas variáveis X e Y como variáveis aleatórias, cada uma com sua própria distribuição de observações reais, que são chamadas de **funções de distribuição marginais**. Ao atribuir uma probabilidade a cada combinação possível (x, y) de valores das variáveis aleatórias, indicando a chance de cada combinação ocorrer no conjunto de dados, estaremos definindo uma **função de distribuição conjunta**, denotada por

$$p[(X = x) \cap (Y = y)] = p(X = x, Y = y),$$

onde x e y são os valores assumidos pelas variáveis aleatórias X e Y, respectivamente. Isso nos permite quantificar a incerteza subjacente às combinações dos dados e entender a distribuição de possíveis resultados de um evento ou experimento.

Retomando o problema da impossibilidade de coletar todos os dados de uma população, conforme mencionado na introdução deste capítulo, as variáveis aleatórias se destacam como opções ideais para representar amostras representativas. Isso se deve à sua capacidade de quantificar as incertezas dos dados por meio das funções de distribuição. Fundamentadas na teoria da probabilidade e estatística, essas variáveis aleatórias viabilizam análise, interpretação e inferência de dados, incluindo o cálculo de médias, desvios padrão, intervalos de confiança e a realização de testes de hipóteses.

8.3.2 Estatísticas Resumidas em Abordagem Probabilística

Na análise de dados provenientes de amostras retiradas de uma população maior, é fundamental utilizar conceitos de probabilidade e variáveis aleatórias para medir e quantificar a incerteza associada às estimativas estatísticas. Uma abordagem probabilística para calcular estatísticas descritivas utiliza variáveis aleatórias, cujas distribuições de probabilidade são conhecidas ou modeladas, proporcionando um rigor matemático robusto na análise da variabilidade e incerteza dos dados.

Essa metodologia permite uma avaliação precisa das inferências, assegurando consistência e precisão nos métodos estatísticos. Além disso, a abordagem probabilística é altamente generalizável e aplicável a diversos domínios, tornando-se uma ferramenta versátil para a análise de dados. Ao compreender as funções de probabilidade associadas aos dados, os especialistas podem tomar decisões mais informadas, considerando tanto os resultados mais prováveis quanto os cenários de risco. A linguagem probabilística oferece uma forma clara e precisa de comunicar resultados e incertezas, facilitando a interpretação e a apresentação das análises estatísticas para diferentes públicos.

Segue-se uma abordagem fundamentada em métodos probabilísticos, contrastando com a abordagem anteriormente apresentada na Seção 6.1.2, que se baseava em conjuntos de dados e médias aritméticas. Esta nova abordagem implica o uso de funções de probabilidade, variáveis aleatórias e outros conceitos estatísticos no cômputo das medidas de resumo de uma população de tamanho N:

Média: Para uma variável discreta X, a média μ_X, também conhecida por **valor esperado** ou **esperança**, é calculada multiplicando cada valor x_i possível de X pela correspondente probabilidade p_i e somando todos os produtos

$$E(X) = \mu_X = \mu = \sum_{i=1}^{N} x_i p_i. \tag{8.11}$$

Para uma variável contínua X, a expressão para a média é análoga, mas a soma é substituída por uma integral, e função de massa de probabilidade p_i por $f(x)dx$, onde $f(x)$ é a função de densidade de probabilidade de X e dx representa uma variação infinitesimal da variável:

$$E(X) = \mu_X = \mu = \int_{-\infty}^{\infty} x f(x) dx. \tag{8.12}$$

A **propriedade de linearidade do valor esperado** afirma que o valor esperado de uma soma de variáveis aleatórias é igual à soma dos valores esperados de cada variável aleatória individualmente:

$$E(a_1X_1 + a_2X_2 + \cdots + a_nX_n) = a_1E(X_1) + a_2E(X_2) + \cdots + a_nE(X_n)$$

Variância: A variância σ^2 de uma variável X é calculada como a média dos quadrados das diferenças entre cada valor possível da variável e a média, ponderada pelas probabilidades de ocorrência desses valores. Para uma variável discreta X, ela é dada por

$$\sigma^2 = Var(X) = \sum_{i=1}^{N} (x_i - \mu)^2 p_i, \tag{8.13}$$

enquanto para uma variável contínua X, as substituições análogas às da média de X são feitas:

$$\sigma^2 = Var(X) = \int_{-\infty}^{\infty} (x - \mu)^2 f(x) dx. \tag{8.14}$$

A **propriedade de linearidade da variância** nos diz que a variância da soma de variáveis aleatórias é a soma das variâncias individuais, desde que as variáveis sejam independentes:

$$Var(a_1X_1 + a_2X_2 + \cdots + a_nX_n) = a_1^2Var(X_1) + a_2^2Var(X_2) + \cdots + a_n^2Var(X_n)$$

Desvio padrão: O desvio padrão é a raiz quadrada da variância populacional:

$$\sigma = \sqrt{\sigma^2}. \tag{8.15}$$

Mediana: A mediana de uma função de distribuição, representada por M_d, é o valor em que a probabilidade acumulada até esse ponto é igual a 0,5. Em outras palavras, uma metade das observações de uma variável X é menor ou igual a M_d, e a outra metade é maior or igual a M_d. Isso é expresso pelas seguintes condições:

$$p(X \geq M_d) = 0.5 \;\text{ e }\; p(X \leq M_d) = 0.5. \tag{8.16}$$

Essas condições garantem que a mediana divide a probabilidade acumulada da função de distribuição em duas partes iguais, proporcionando uma medida de centralidade que não é influenciada por *outliers*.

Quartis e Percentis: Os quartis e percentis podem ser calculados de maneira semelhante à mediana, mas com diferentes pontos de corte de probabilidade acumulada na função de distribuição, ou seja, podemos expressar os quartis da seguinte forma:

Primeiro quartil (Q1) : $P(X \leq Q1) = 0.25$.

Segundo quartil (Q2) : $P(X \leq Q2) = 0.5$.

Terceiro quartil (Q3) : $P(X \leq Q3) = 0.75$.

Essas expressões garantem que os quartis dividam a probabilidade acumulada da função de distribuição em quatro partes iguais, fornecendo uma medida adicional de centralidade e dispersão dos dados. Observe que Q_2=M_d. Os percentis de uma função de distribuição, por sua vez, são valores específicos $P_{percentil}$ que dividem a probabilidade acumulada em 100 partes iguais. O valor de $p_{percentil}$ é tal que a soma de probabilidades até este ponto corresponde a $percentil \times 0.01$:

$$P(X \leq P_{percentil}) = percentil \times 0.01. \tag{8.17}$$

Por exemplo, $P(X \leq P_3) = 0.03$.

Moda: A moda M_o pode ser identificada como os máximos da função de distribuição. Para uma variável discreta X em N observações, em que cada valor possível x_i tenha uma probabilidade de ocorrência p_i associada, ou seja,

$$P(X = x_i) = p(x_i) = p_i, \tag{8.18}$$

a moda pode ser expressa como

$$P(X = M_o) = max(p_1, p_2, \cdots, p_N). \tag{8.19}$$

Para uma variável contínua X, deve-se substituir a função de massa de probabilidade

$p(x_i)$ por uma função de densidade de probabilidade $f(x)$ e a moda é o valor M_o que satisfaz a condição

$$f(M_o) = max f(x). \tag{8.20}$$

A **covariância** quantifica como duas variáveis variam conjuntamente (Seção 8.3), indicando se elas tendem a aumentar ou diminuir juntas, ou se não apresentam uma relação aparente. Considerando duas variáveis, X e Y, e um conjunto de N observações em valores reais $\{(x_1, y_1), \cdots, (x_N, y_N)\}$, a covariância entre X e Y pode ser transformada da Eq. 6.9 para uma expressão envolvendo esperanças, aplicando a propriedade de linearidade do valor esperado:

$$\begin{aligned} Cov(X,Y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_X)(y_i - \mu_Y) &= E[(X - \mu_X)(Y - \mu_Y] \\ &= E[XY - X\mu_Y - Y\mu_X + \mu_X\mu_Y] \\ &= E[XY] - E[X\mu_Y] - E[Y\mu_X] + E[\mu_X\mu_Y] \\ &= E[XY] - \mu_Y E[X] - \mu_X E[Y] + \mu_X\mu_Y \\ &= E[XY] - E[Y]E[X] - E[X]E[Y] + E[X]E[Y] \\ &= E[XY] - E[Y]E[X]. \end{aligned}$$

Quando X e Y não apresentam nenhuma relação linear entre elas, ou quando o conhecimento sobre uma variável não fornece nenhuma informação sobre a outra variável

$$Cov(X,Y) = E[XY] - E[Y]E[X] = 0 \Longrightarrow E[XY] = E[X]E[Y].$$

De maneira análoga ao que foi discutido na Seção 6.2, utilizamos a **correlação**, uma medida que normaliza a covariância em relação aos desvios padrão populacionais, a fim de padronizar a relação entre as variáveis X e Y na abordagem probabilística:

$$\rho(X,Y) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}. \tag{8.21}$$

A abordagem probabilística permite uma análise detalhada e rigorosa dos dados, considerando as incertezas associadas a eles por meio de variáveis aleatórias e distribuições de probabilidade. No entanto, as funções do pacote dplyr[2] em R e as funções dos pacotes numpy e pandas em Python calculam estatísticas descritivas diretamente sobre os dados observados, como a média aritmética, sem levar em consideração o modelo subjacente ou a distribuição dos dados. A média aritmética,

[2] As funções de estatísticas descritivas, como mean() e median(), são funções básicas de R. Para cálculos simples dessas estatístics, não é necessário carregar o pacote dplyr. O dplyr oferece uma função chamada summarize() que pode ser usada para calcular estatísticas resumidas em conjuntos de dados, incluindo a média e a mediana, além das operações de filtragem, seleção e agregação dos dados.

que é uma estimativa de tendência central, pode ser interpretada como uma esperança matemática em contextos simples, mas o uso de esperanças matemáticas requer um conhecimento prévio sobre a distribuição dos dados ou um modelamento estatístico mais preciso. Este tipo de modelagem é mais comum em etapas posteriores da análise, como a modelagem estatística, e não é tão usual na prática da análise exploratória de dados.

8.4 Teorema Central do Limite

A variável aleatória que mapeia amostras em médias amostrais é o cerne do Teorema Central do Limite. Esta variável aleatória representa a média de um conjunto de observações (amostras) extraídas de uma população e, à medida que o tamanho da amostra aumenta, a distribuição dessas **médias amostrais** tende a se aproximar de uma **distribuição normal**, como ilustra a Figura 8.3 (Seção 8.5.2), e sua média se aproxima da média populacional. Na figura, cada barra no histograma representa uma média amostral específica, com a altura da barra indicando a frequência ou probabilidade associada a essa média. O padrão resultante dessas barras ao longo do eixo horizontal se assemelha ao formato de um sino, que é a forma característica da distribuição normal.

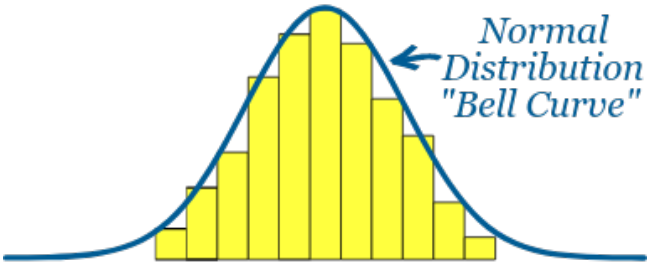


Figura 8.3: Teorema Central do Limite: distribuição normal das médias amostrais para qualquer população, se as observações forem independentes e a quantidade de repetições de amostragem for suficientemente grande. (Fonte: Mathsisfun)

O **Teorema Central do Limite** estabelece que, independentemente da forma da distribuição da população original, a média das amostras, quando repetidamente calculada, exibirá uma distribuição normal à medida que o número de amostras se torna suficientemente grande. Podemos confiar na normalidade das médias. Essa propriedade é fundamental para a inferência estatística, pois permite que os estatísticos façam previsões e estimativas sobre a população com base nas médias das amostras. Veremos no Capítulo **??** que a aproximação à distribuição normal facilita a construção de intervalos de confiança e a realização de testes de hipóteses, oferecendo uma base sólida para a tomada de decisões em diversas disciplinas, desde ciências sociais até ciências naturais.

8.5 Modelos de Probabilidade

Uma função de probabilidade é uma ferramenta sistemática e precisa para atribuir probabilidades aos valores de variáveis aleatórias, permitindo-nos calcular a probabilidade de ocorrência de eventos específicos e entender o comportamento das variáveis aleatórias em diferentes cenários.

Os estatísticos empregam uma variedade de ferramentas e modelos matemáticos para explorar as funções de probabilidade associadas às variáveis aleatórias. Identificar padrões nessas funções é fundamental, pois não apenas condensa dados complexos em representações matemáticas compreensíveis, mas também enriquece a interpretação dos dados e possibilita inferências estatísticas mais robustas.

Na Seção 6.1.1 destacamos que muitos fenômenos aleatórios apresentam comportamentos semelhantes, frequentemente descritíveis por um conjunto comum de modelos de probabilidade. Hoje, uma variedade de modelos matemáticos está disponível, cada um adaptado para diferentes situações e fenômenos aleatórios. A diversidade desses modelos oferece uma gama rica de ferramentas matemáticas para descrever uma ampla variedade de fenômenos aleatórios. Ao reconhecer e aplicar as propriedades conhecidas desses modelos, não apenas simplificamos a análise estatística, mas também extraímos informações relevantes de dados empíricos, promovendo uma compreensão mais profunda dos processos subjacentes e facilitando as tomadas de decisões.

Assim, em vez de atribuir probabilidades empiricamente, muitos estudos se concentram na caracterização dos dados disponíveis e na seleção de um modelo mais apropriado para descrevê-los. A escolha desse modelo depende das características específicas dos dados e da natureza do fenômeno em estudo. Nesta seção, apresentamos, tanto para variáveis discretas quanto para as contínuas, alguns dos modelos mais reconhecidos e amplamente utilizados na prática estatística e probabilística.

É importante ressaltar que, embora existam distribuições de frequência, ilustradas na Figura 6.2, que se assemelham a certos modelos de probabilidade, eles não são equivalentes. Enquanto as distribuições de frequência representam a contagem de valores específicos em um conjunto de dados observados, os modelos de probabilidade descrevem probabilidades teóricas associadas aos valores das variáveis aleatórias.

8.5.1 Variáveis Discretas

Para variáveis aleatórias discretas, alguns exemplos comuns de modelos de probabilidade, análogos às distribuições de frequência de ocorrência de valores de uma variável apresentadas na Seção 6.1.1, incluem [52]:

Função de probabilidade uniforme: Atribui-se a mesma probabilidade a todos os k possíveis valores de uma variável aleatória X:

$$P(X = x_j) = \frac{1}{k}, \quad \forall j = 1, 2, 3, \cdots, k \tag{8.22}$$

Função de probabilidade de Bernoulli: Essa função modela um experimento que resulta em apenas dois possíveis resultados: sucesso (geralmente denotado por "1") ou fracasso (geralmente denotado por "0") por meio da função de probabilidade

$$P(X = k) = p^k(1-p)^{(1-k)}, \quad x = 0, 1, \tag{8.23}$$

onde p representa a probabilidade de sucesso.

Função de probabilidade binomial: É expressa através da equação

$$P(X = k) = \begin{pmatrix} n \\ k \end{pmatrix} p^k(1-p)^{(n-k)}, \quad k = 0, 1, \cdots, n \tag{8.24}$$

a probabilidade de k sucessos em uma sequência fixa de n tentativas independentes, cada uma com apenas dois resultados possíveis (sucesso e fracasso), como em ensaios de Bernoulli, e todas as tentativas com a mesma probabilidade de sucesso p.

Função de probabilidade de Poisson: É usada para modelar a probabilidade de um número discreto de eventos ocorrer em um intervalo fixo de tempo ou espaço, quando esses eventos ocorrem de forma independente e a uma taxa média conhecida. A equação que descreve a função de probabilidade de Poisson é:

$$P(X = k) = \frac{e^{-\lambda}\lambda^k}{k!}, \tag{8.25}$$

onde λ é a taxa média de ocorrência de eventos no intervalo considerado.

8.5.2 Variáveis Contínuas

Entre os exemplos dos modelos de probabilidade para variáveis contínuas, temos [52]:

Função de probabilidade uniforme: Atribui-se a mesma densidade de probabilidade a todos os valores de uma variável aleatória no intervalo $[a, b]$:

$$f(x) = \begin{cases} \frac{1}{b-a}, & a \leq x \leq b \\ 0, & \text{caso contrário.} \end{cases} \tag{8.26}$$

Assim, a média e a variância de uma distribuição uniforme são, respectivamente,

$$\begin{aligned} \mu &= \int_a^b x \cdot f(x)dx = \frac{b+a}{2} \\ \sigma^2 &= \int_a^b (x-\mu)^2 \cdot f(x)dx = \frac{(b-a)^2}{12} \end{aligned} \tag{8.27}$$

Função de probabilidade exponencial: É utilizada para modelar o tempo entre eventos em um processo de Poisson, onde a taxa de ocorrência de eventos é constante e independente do tempo.

$$f(x) = \begin{cases} \alpha e^{-\alpha x}, & x \geq 0 \\ 0, & \text{caso contrário.} \end{cases} \tag{8.28}$$

É possível demonstrar através de cálculos de integração que, para uma distribuição exponencial,

$$\begin{aligned} \mu &= \int_0^\infty x \cdot f(x) dx = \frac{1}{\alpha} \\ \sigma^2 &= \int_0^\infty (x-\mu)^2 \cdot f(x) dx = \frac{1}{\alpha^2}. \end{aligned} \tag{8.29}$$

Função de probabilidade normal: É representada através da expressão

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, -\infty < x < \infty \tag{8.30}$$

A distribuição normal é uma das mais importantes na Estatística. Ela é totalmente caracterizada pelos dois parâmetros (Seção 6.1.4):

$$\begin{aligned} \mu &= \int_{-\infty}^\infty x \cdot f(x) dx \\ \sigma^2 &= \int_{-\infty}^\infty (x-\mu)^2 \cdot f(x) dx \end{aligned} \tag{8.31}$$

Isso implica que, ao conhecer μ e σ, podemos descrever completamente a forma da distribuição normal. Além disso, **se os dados seguem uma distribuição normal, a média da população, a média amostral e a média do gráfico de distribuição coincidem**. As variáveis aleatórias de muitos fenômenos se comportam próximas a distribuição normal. É frequente encontrar a notação $N(X; \mu, \sigma^2)$ para indicar que uma variável X segue uma distribuição normal com média μ e variância σ^2.

Para simplificar o cálculo da probabilidade de ocorrência de determinados valores de uma variável aleatória, a técnica de tabela de busca (em inglês, *lookup table*) tem sido amplamente utilizada até a popularização dos computadores. Por meio das **tabelas de distribuição normal padrão** $N(Z; 0, 1)$ [8], é possível consultar a probabilidade de qualquer valor Z que resulta da normalização de uma variável aleatória z, conhecido como escore-z (Eq. 6.8).[3].

[3] A transformação $X \leftrightarrow Z$ não altera a forma da distribuição de probabilidade, pois apenas redimensiona e move a escala dos valores de X para uma nova escala representada por Z. Essa operação mantém a estrutura relativa dos dados,

Função de probabilidade binomial: Descreve a distribuição de probabilidade de k sucessos em um número total n de tentativas independentes, onde cada tentativa tem apenas dois resultados possíveis (geralmente chamados de "sucesso"e "fracasso") com a probabilidade $\mathcal{P}$ de sucesso em uma única tentativa

$$f(k) = p(X = k) = \begin{pmatrix} n \\ k \end{pmatrix} \cdot \mathcal{P}^k \cdot (1 - \mathcal{P})^{n-k} \tag{8.32}$$

A função de probabilidade binormal é fundamental em muitos contextos, como experimentos binários repetidos, testes de hipóteses e modelagem de eventos discretos com apenas dois resultados possíveis. Uma propriedade fundamental da distribuição binomial é que a média da distribuição é o produto do número n de tentativas pelo sucesso individual, que é a probabilidade $\mathcal{P}$. Para a variância, usamos o fato de que a variância de uma soma de variáveis aleatórias independentes é a soma das variâncias individuais, que segue a fórmula geral para variância de uma distribuição de Bernoulli, $\mathcal{P}(1 - \mathcal{P})$. Ou seja,

$$\begin{aligned} \mu &= n \cdot \mathcal{P} \\ \sigma^2 &= Var(X_1 + X_2 + \cdots + X_n) = nVar(X) = n \cdot \mathcal{P} \cdot (\mathcal{P} - 1) \end{aligned} \tag{8.33}$$

Essas expressões podem ser derivadas a partir das propriedades da média e da variância de uma soma de variáveis aleatórias independentes e identicamente distribuídas. Por exemplo, para a variância da distribuição de Bernoulli, em que a variável aleatória X assume apenas dois valores, x_1=0 e x_2=1, com as respectivas probabilidades $p_1 = (1 - p)$ e $p_2 = p$, temos

$$\begin{aligned} Var(X) &= \sum_{i=1}^{2} (x_i - \mu)^2 \mathcal{P}_i = (0 - \mathcal{P})^2(1 - \mathcal{P}) + (1 - \mathcal{P})^2\mathcal{P} = \mathcal{P}^2(1 - \mathcal{P}) + (1 - 2\mathcal{P} + \mathcal{P}^2)\mathcal{P} \\ &= \mathcal{P}^2 - \mathcal{P}^3 + \mathcal{P} - 2\mathcal{P}^2 + \mathcal{P}^3 = \mathcal{P} - \mathcal{P}^2 = \mathcal{P}(1 - \mathcal{P}). \end{aligned} \tag{8.34}$$

Função de probabilidade Student (ou distribuição t de Student): É caracterizada por ser simétrica em torno de zero e tem caudas mais pesadas do que a distribuição normal, especialmente para amostras pequenas. A distribuição t é controlada por um parâmetro conhecido como graus de liberdade ν, que é determinado pelo tamanho da amostra n menos 1. Quanto maior o número de graus de liberdade, mais próxima a distribuição t se aproxima de uma distribuição normal padrão. A função é definida

preservando as relações de ordem e as proporções entre os valores.

pela expressão:

$$f(t \mid \nu) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\Gamma(\frac{\nu}{2})}(1 + \frac{t^2}{\nu})^{-\frac{\nu+1}{2}}, \tag{8.35}$$

onde t é a variável aleatória (geralmente chamada de estatística t), ν é o número de graus de liberdade, e Γ é uma função gama generalizada de Euler. Essa função gama é, por sua vez, definida por

$$\Gamma(t) = \int_0^\infty t^{z-1}e^{-t}dt, \tag{8.36}$$

onde z é o parâmetro da função gama. Quando z é um número inteiro positivo, $z > 0$, a função gama assume o valor de $(z-1)!$. Para valores negativos de z ou para $z = 0$, a integral pode divergir e a função gama não é válida.

Pode-se demonstrar que a média da distribuição t-Student é 0 (zero), e sua variância é igual a $\frac{\nu}{\nu-2}$ para $\nu > 2$.

Função de probabilidade Qui-Quadrado (ou distribuição qui-quadrado): É usada em estatísticas para determinar a probabilidade de uma certa quantidade de variação em um conjunto de dados, assumindo que os dados seguem uma distribuição qui-quadrado. Esta distribuição é comumente aplicada em várias áreas da estatística, especialmente em testes de hipóteses e análises de variância. A forma da distribuição qui-quadrado depende do número de graus de liberdade d, e ela é não negativa e assimétrica à direita.

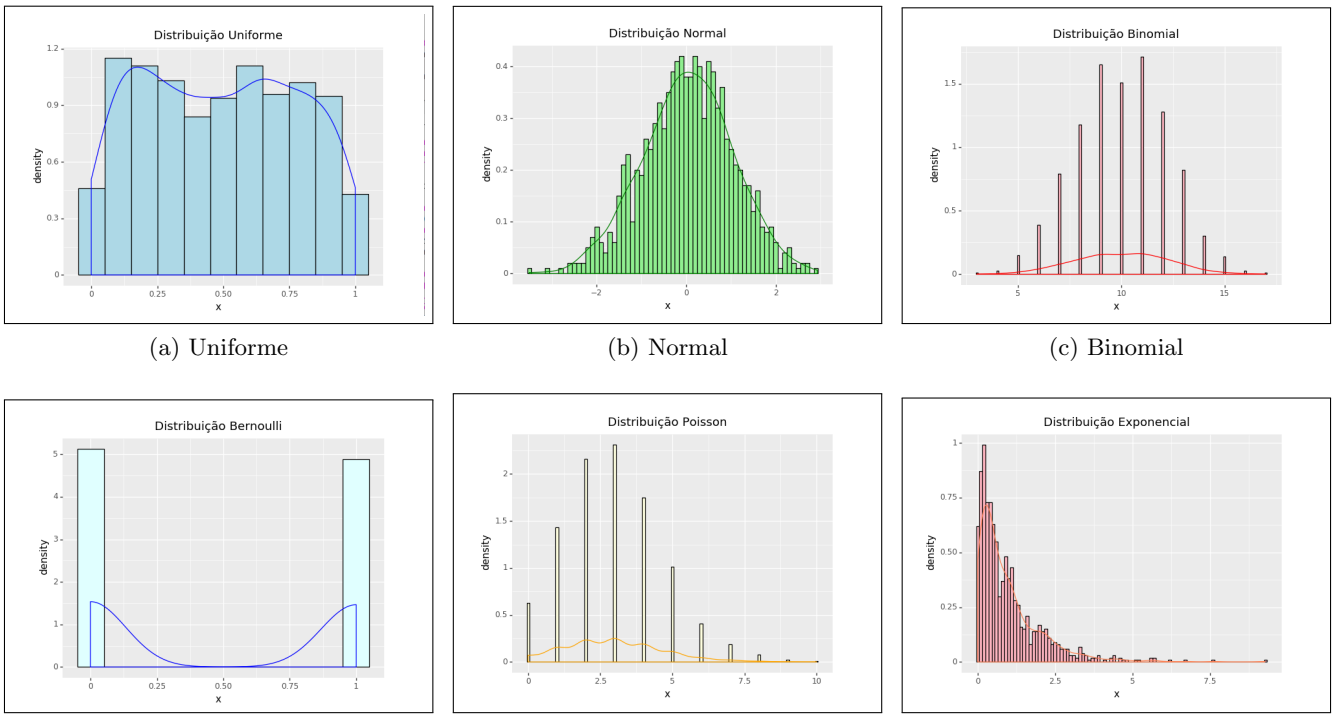
$$f(x) = \frac{1}{2^{\frac{d}{2}}\Gamma(\frac{d}{2})}x^{\frac{d}{2}-1}e^{-\frac{x}{2}}, \quad x \geq 0. \tag{8.37}$$

A sua média e o desvio-padrão são dados pelas seguintes equações:

$$\begin{aligned} \mu &= d \\ \sigma^2 &= 2d. \end{aligned} \tag{8.38}$$

8.5.3 Gráficos de Distribuições em R e Python

Tanto em R quanto em Python, é possível criar gráficos que representam as distribuições de frequência dos valores de uma variável e traçar curvas de densidade de probabilidade que melhor se ajustam a essas distribuições. Para realizar essa tarefa, podemos utilizar bibliotecas como ggplot2 em R e plotnine em Python. Abaixo, apresentamos instruções para gerar 1000 pontos aleatórios de seis distribuições diferentes – uniforme, normal, binomial, de Poisson, de Bernoulli e exponencial – e para traçar as curvas de densidade de probabilidade correspondentes que melhor se ajustem a essas distribuições em ambas as linguagens. Os resultados são apresentados na Figura 8.4.



(a) Uniforme (b) Normal (c) Binomial

(d) Bernoulli (e) Poisson (f) Exponencial

Figura 8.4: Visualização das funções de densidade de probabilidade que melhor se aproximam das distribuições de frequência dos 1000 pontos gerados aleatoriamente com uso de 6 modelos de distribuição.

R

Com exceção do pacote ggplot2 (Seção 3.5), todas as funções relacionadas com distribuições de frequência ou de probabilidade dos valores de uma variável são funções básicas de R.

```
library(ggplot2)


# Distribuição Uniforme
data_uniform <- runif(1000)
ggplot(data.frame(x = data_uniform), aes(x)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.1, fill = "lightblue", color = "black") +
  geom_line(stat = "density", color = "blue") +
  ggtitle("Distribuição Uniforme")


# Distribuição Normal
data_normal <- rnorm(1000)
ggplot(data.frame(x = data_normal), aes(x)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.1, fill = "lightgreen", color = "black") +
  geom_line(stat = "density", color = "green") +
```

```
  ggtitle("Distribuição Normal")

# Distribuição Binomial
data_binomial <- rbinom(1000, size = 20, prob = 0.5)
ggplot(data.frame(x = data_binomial), aes(x)) +
  geom_histogram(aes(y = ..density..), binwidth = 1, fill = "lightpink", color = "black") +
  geom_line(stat = "density", color = "red") +
  ggtitle("Distribuição Binomial")

# Distribuição de Poisson
data_poisson <- rpois(1000, lambda = 3)
ggplot(data.frame(x = data_poisson), aes(x)) +
  geom_histogram(aes(y = ..density..), binwidth = 1, fill = "lightyellow", color = "black") +
  geom_line(stat = "density", color = "orange") +
  ggtitle("Distribuição de Poisson")

# Distribuição de Bernoulli (discreta, só terá duas barras)
data_bernoulli <- rbinom(1000, size = 1, prob = 0.5)
ggplot(data.frame(x = data_bernoulli), aes(x)) +
  geom_bar(aes(y = ..density..), fill = "lightcyan", color = "black") +
  geom_line(stat = "density", color = "blue") +
  ggtitle("Distribuição de Bernoulli")

# Distribuição Exponencial
data_exponential <- rexp(1000, rate = 1)
ggplot(data.frame(x = data_exponential), aes(x)) +
  geom_bar(aes(y = ..density..), fill = "lightpink", color = "black") +
  geom_line(stat = "density", color = "coral") +
  ggtitle("Distribuição Exponencial")
```

Python

Para plotar gráficos de distribuições usando o pacote plotnine, é essencial carregar as distribuições disponíveis no scipy.stats. Ao importar scipy.stats, que faz parte da biblioteca SciPy, também é importante incluir o NumPy para garantir o funcionamento adequado de todas as funcionalidades da SciPy. Isso porque a SciPy depende do NumPy para representações de matrizes, vetores e outras estruturas de dados numéricos, além de operações de manipulação de dados e cálculos subjacentes. Adicionalmente,

é necessário carregar o pandas para transformar os dados gerados aleatoriamente em uma estrutura DataFrame, que pode ser processada pela função ggplot.

```
import numpy as np
import pandas as pd
from plotnine import *
from scipy.stats import uniform, norm, binom, poisson, bernoulli expon

# Distribuição Uniforme
data_uniform = uniform.rvs(size=1000)
df_uniform = pd.DataFrame({'x': data_uniform})
(
ggplot(df_uniform, aes(x='x'))
+ geom_histogram(aes(y='stat(density)'), binwidth=0.1, fill='lightblue', color='black')
+ geom_density(color='blue')
+ ggtitle("Distribuição Uniforme")
)

# Distribuição Normal
data_normal = norm.rvs(size=1000)
df_normal = pd.DataFrame({'x': data_normal})
(
ggplot(df_normal, aes(x='x'))
+ geom_histogram(aes(y='stat(density)'), binwidth=0.1, fill='lightgreen', color='black')
+ geom_density(color='green')
+ ggtitle("Distribuição Normal")
)

# Distribuição Binomial
data_binomial = binom.rvs(n=20, p=0.5, size=1000)
df_binomial = pd.DataFrame({'x': data_binomial})
(
ggplot(df_binomial, aes(x='x'))
+ geom_histogram(aes(y='stat(density)'), binwidth=0.1, fill='lightpink', color='black')
+ geom_density(color='red')
+ ggtitle("Distribuição Binomial")
)
```

```
# Distribuição de Poisson
data_poisson = poisson.rvs(mu=3, size=1000)
df_poisson = pd.DataFrame({'x': data_poisson})
(
ggplot(df_poisson, aes(x='x'))
+ geom_histogram(aes(y='stat(density)'), binwidth=0.1, fill='lightyellow', color='black')
+ geom_density(color='orange')
+ ggtitle("Distribuição Poisson")
)

# Distribuição de Bernoulli (discreta, só terá duas barras)
data_bernoulli = bernoulli.rvs(p=0.5, size=1000)
df_bernoulli = pd.DataFrame({'x': data_bernoulli})
(
ggplot(df_bernoulli, aes(x='x'))
+ geom_histogram(aes(y='stat(density)'), binwidth=0.1, fill='lightcyan', color='black')
+ geom_density(color='blue')
+ ggtitle("Distribuição Bernoulli")
)

# Distribuição Exponencial
data_expon= expon.rvs(scale=1, size=1000)
df_expon = pd.DataFrame({'x': data_expon})
(
ggplot(df_expon, aes(x='x'))
+ geom_histogram(aes(y='stat(density)'), binwidth=0.1, fill='lightpink', color='black')
+ geom_density(color='red')
+ ggtitle("Distribuição Exponencial")
)
```

8.6 Simulações de Monte Carlo

As simulações de Monte Carlo são uma técnica computacional amplamente utilizada em várias áreas, desde a física e engenharia até a finança e biologia, devido à sua versatilidade, eficácia e capacidade de lidar com problemas complexos e não-lineares. Essa abordagem recebe o nome da famosa cidade de

Monte Carlo, conhecida por seus cassinos e jogos de azar, devido à natureza probabilística da técnica. Numa **simulação de Monte Carlo**, um modelo ou sistema é analisado por meio da geração de múltiplos conjuntos de valores aleatórios, conforme ilustrado na Seção 1.2.1. Cada conjunto representa uma realização possível do experimento. Ao repetir a simulação diversas vezes, as estatísticas e distribuições das saídas são utilizadas para estimar as propriedades do sistema ou modelo em questão, bem como os resultados em sistemas complexos e determinísticos. É **uma prática comum em simulações estatísticas e estudos de probabilidade**.

A prática de repetir o experimento com diferentes conjuntos de valores aleatórios é fundamental nas simulações de Monte Carlo. Isso permite capturar a variabilidade inerente ao sistema e compreender melhor os possíveis resultados sob diferentes condições. Quanto maior o número de repetições (ou seja, a quantidade de conjuntos de valores aleatórios), mais precisas serão as estimativas e análises resultantes da simulação de Monte Carlo. Por exemplo, se estivermos interessados na média de uma população, podemos aplicar o Teorema Central do Limite. Este teorema estabelece que, quando o tamanho da amostra é grande o suficiente, a distribuição das médias amostrais de uma população se aproxima de uma distribuição normal. Portanto, a média das médias amostrais se aproxima da média da população. Assim, espera-se que em simulações de Monte Carlo, onde múltiplas amostras são geradas e os resultados são agregados ou analisados, possamos estimar a média de qualquer população independentemente da distribuição subjacente.

A relevância das simulações de Monte Carlo reside em sua capacidade de lidar com problemas complexos e não-lineares, para os quais não existem soluções analíticas diretas ou são computacionalmente proibitivas de obter. Ao invés de depender de equações matemáticas complicadas ou modelos determinísticos, as simulações de Monte Carlo fornecem uma abordagem prática e eficaz para estimar resultados através da repetição de experimentos aleatórios. Por exemplo, na física, as simulações de Monte Carlo são usadas para modelar o comportamento de partículas subatômicas, sistemas físicos complexos e fenômenos de transporte. Na engenharia, são empregadas para analisar a confiabilidade de estruturas, otimizar o *design* de produtos e simular o desempenho de sistemas dinâmicos. Na finança, são utilizadas para avaliar riscos e simular o comportamento de mercados financeiros. Na biologia, são aplicadas para estudar interações entre moléculas, simular a evolução de populações e prever a propagação de doenças.

As principais etapas de uma simulação de Monte Carlo incluem:

Definição do Problema: Identificar e definir o problema ou sistema que será simulado, incluindo a especificação das variáveis de interesse, dos parâmetros do modelo e dos objetivos da simulação.

Modelagem do Sistema: Desenvolver um modelo matemático ou computacional do sistema que represente as relações entre as variáveis e descreva o comportamento do sistema ao longo do tempo ou espaço.

Geração de Amostras Aleatórias : Gerar conjuntos de valores aleatórios que representem diferentes cenários ou realizações possíveis do experimento. Isso pode envolver a seleção de modelos de probabilidade apropriados para as variáveis do modelo.

Execução da Simulação: Realizar a simulação computacional, onde os conjuntos de valores aleatórios são usados como entrada para o modelo e o sistema é simulado repetidamente para cada conjunto de valores.

Análise dos Resultados: Analisar os resultados da simulação para extrair informações úteis sobre o sistema, podendo envolver calcular estatísticas descritivas, identificar tendências ou padrões, ou avaliar o desempenho do sistema em relação aos objetivos definidos.

Validação e Interpretação: Validar os resultados da simulação comparando-os com dados reais, modelos analíticos ou resultados de outras simulações. Interpretar os resultados à luz dos objetivos da simulação e das suposições subjacentes ao modelo.

8.6.1 Exemplo

Um exemplo simples de simulação de Monte Carlo para estimar a frequência de ocorrência de cara (C) no lançamento de uma moeda justa envolve determinar a distribuição das proporções de cara (C) de N amostras, cada uma consistindo de n lançamentos (observações). Nessa simulação, cada lançamento tem uma probabilidade de 0,5 de resultar em cara (C) e 0,5 de resultar em coroa (K). Para cada amostra de n lançamentos, calculamos a proporção de caras obtidas. Repetimos esse processo N vezes para obter uma distribuição das proporções amostrais de caras (C) obtidas e "generalizar" a partir delas a frequência de ocorrência de C nos lançamentos da moeda.

Definição do Problema : Estimar a média das porporções de cara (C) de uma série de lançamentos de uma moeda justa.

Modelagem do Sistema : Cada lançamento de uma moeda justa é um evento de Bernoulli. Estamos interessados na contagem do número de vezes que obtemos cara em um número fixo n de lançamentos, e os lançamentos são independentes. Portanto, o número total de caras em n lançamentos (observações) nas N repetições segue uma distribuição binomial.

Geração de Amostras Aleatórias : Usamos funções disponíveis em R/Python para gerar N amostras aleatórias, cada uma das quais contém n observações que seguem uma distribuição binomial.

Execução da Simulação : Para cada amostra, lançamos a moeda n vezes e registramos os resultados de cada lançamento. Calculamos a proporção amostral das caras obtidas. Repetimos esse processo N vezes.

Análise dos Resultados : Calculamos a média das médias amostrais para obter uma estimativa da média da frequência de ocorrência de caras em uma moeda justa, cujo valor esperado é 0.5.

Programação em R e Python

Segue abaixo a implementação em R e Python da simulação de Monte Carlo descrita. A média de probabilidades de ocorrência de "cara" nos lançamentos da moeda é 0.504599, o que está muito próxima da percentagem de frequência esperada.

R

```
# Definição da quantidade de amostras e tamanho de amostra
N = 1000   # Quantidade de amostras
n = 10     # Tamanho de amostra
mean_samples = rep(0, N)  # Proporções das caras obtidas/lancamento
sample = rep(0,n) # Resultados dos 10 lancamentos

# Distribuição Binomial
# Execução da simulação de Monte Carlo
for (i in 1:N) {
    # Gerar uma amostra de 10 lançamentos da moeda (Cara = 1, Coroa = 0)
    sample <- rbinom(n, 1, 0.5)
    # Calcular a frequência de C e armazená-la
    mean_samples[i] <- mean(sample)
}

# Calcular a média das proporções amostrais
mean_estimate <- mean(mean_samples)

# Exibir resultado
cat("Estimativa da média da população:", mean_estimate, "\n")
```

Python

```
import numpy as np

# Definição da quantidade de amostras e tamanho de amostra
N = 1000   # Quantidade de amostras
```

```
n = 10     # Tamanho de amostras
mean_samples = np.zeros(N)  # Proporção das caras obtidas/lancamento
sample = np.zeros (n) # Resultados dos 10 lancamentos

#Distribuição Binomial
# Execução da simulação de Monte Carlo
for i in range(N):
    # Gerar uma amostra de 10 lançamentos da moeda (Cara = 1, Coroa = 0)
    sample = np.random.binomial(1,0.5,size=n)
    # Calcular a frequência de C e armazená-la
    mean_samples[i] = np.mean(sample)
# Calcular a média das médias amostrais
mean_estimate = np.mean(mean_samples)

# Exibir resultado
print("Estimativa da média da população:", mean_estimate)
```

8.7 Considerações Finais

Ao longo deste capítulo, buscamos fornecer uma base sobre a probabilidade e sua relação com as estatísticas descritivas. Iniciamos explorando os conceitos essenciais de probabilidade na Seção 8.1, delineando as noções básicas que são fundamentais para qualquer análise subsequente. Na sequência, na Seção 8.2, abordamos as operações básicas para calcular e manipular probabilidades. Esses princípios são essenciais para computar incertezas inerentes às análises estatísticas de inferência. Na Seção 8.3, introduzimos as variáveis aleatórias, cujos valores possuem probabilidades de ocorrência representadas pelas funções de distribuição discutidas na Seção 8.3.1. Ao definir e explorar o papel dessas variáveis, pudemos compreender sua importância na representação e no cálculo das incertezas em fenômenos aleatórios. Em particular, mostramos na Seção 8.4 que a variável aleatória que mapeia as amostras em médias amostrais é o cerne do Teorema Central do Limite, um dos pilares fundamentais da estatística de inferência. Essa conexão reforça a relevância das variáveis aleatórias na análise estatística e na modelagem de incertezas.

Na Seção 8.3.2, introduzimos uma abordagem probabilística para calcular as estatísticas descritivas mencionadas no Capítulo 6. Em vez de calcular as estatísticas resumidas diretamente sobre os dados observados, são usadas as esperanças matemáticas que consideram as incertezas associadas aos dados. Essa abordagem nos permite compreender como os conceitos probabilísticos podem ser aplicados na análise e interpretação de conjuntos de dados, ampliando o escopo das estatísticas para lidar com dados

permeados de incerteza. A visualização da frequência dos dados através dos gráficos de funções de distribuição ajuda na análise de dados, pois permitem aos analistas compreender a estrutura dos dados, identificar padrões e anomalias, escolher modelos estatísticos apropriados e comunicar os resultados de forma clara e eficaz. A Seção 8.5.3 introduziu funções em R e Python que determinam a função de densidade de probabilidade que melhor se ajusta à distribuição de frequência dos dados observados.

Os modelos de probabilidade, apresentados na Seção 8.5, desempenham um papel relevante na modelagem estatística, lidando com a incerteza inerente aos dados e na inferência sobre a distribuição de uma variável aleatória. Sua importância se estende ao campo do aprendizado de máquina, onde técnicas como redes neurais e algoritmos de classificação dependem fortemente de modelos predefinidos. Em áreas emergentes, como o aprendizado profundo e o aprendizado por reforço, as máquinas podem descobrir padrões complexos nos dados e ajustar os modelos para se adaptarem a eles. No entanto, vale ressaltar que essa capacidade de **inovação** e **adaptação** é **restrita aos dados de treinamento e aos modelos predefinidos**. As máquinas não possuem intuição para incorporar novos conhecimentos sem exposição aos dados. Portanto, embora sejam eficazes na descoberta e modelagem de padrões existentes, não têm a capacidade de gerar conhecimento novo ou incorporar informações ausentes nos dados ou nos modelos predefinidos.

Concluímos este capítulo introduzindo uma técnica poderosa, a simulação estatística de Monte Carlo, na Seção 8.6. Essa técnica oferece uma maneira de **simular estatisticamente** experimentos sob diversas condições para capturar a variabilidade inerente a um sistema e compreender seu comportamento de maneira mais abrangente. Além disso, a simulação de Monte Carlo oferece uma ferramenta poderosa para identificar modelos probabilísticos subjacentes aos dados. Ao simular experimentos sob diferentes condições e observar os resultados, os praticantes podem iterativamente ajustar e validar modelos probabilísticos que melhor representam o comportamento do sistema em estudo. Essa capacidade de identificar e validar modelos probabilísticos pode ser especialmente útil em cenários onde os processos subjacentes são complexos e não podem ser facilmente descritos por modelos analíticos tradicionais. Assim, a simulação de Monte Carlo não apenas fornece uma compreensão mais abrangente do comportamento do sistema, mas também desempenha um papel importante no contexto do aprendizado de máquina, fornecendo ferramentas e técnicas para lidar com a incerteza, otimizar modelos e validar sua eficácia.

8.8 Exercícios

1. Faça os *Learning checks* LC7.1 – LC7.7 propostos em [36].

2. Qual é a probabilidade teórica e empírica da ocorrência do evento (cara, coroa, cara) ao lançarmos simultaneamente três moedas? Realize simulações de Monte Carlo para comparar os resultados obtidos por ambas as abordagens. Dica: A probabilidade teórica de ocorrência do evento (cara,

coroa, cara) ao lançar simultaneamente três moedas pode ser calculada utilizando o princípio da multiplicação para eventos independentes. Cada lançamento é independente e a probabilidade de obter cara/coroa em cada moeda justa é 0.5.

3. O problema de Monty Hall e o problema dos aniversários são frequentemente estudados em contextos de teoria das probabilidades devido à sua natureza intrigante e contra-intuitiva, que desafia a intuição inicial. Ambos os problemas envolvem situações em que a probabilidade aparente da ocorrência de um eventos pode ser enganosa, levando muitas pessoas a tirarem conclusões incorretas. Leia a Seção 3.5 em [71], que aborda como esses desafios podem ser investigados empiricamente utilizando a técnica de Monte Carlo antes de explorarmos os dois problemas:

   (a) Defina o contexto de cada problema e identifique o conceito de interesse associado a eles.

   (b) Identifique a variável aleatória em cada problema, levando em consideração que, no problema de Monty Hall, existem dois cenários alternativos, enquanto no problema dos aniversários há uma quantidade potencialmente infinita de cenários.

   (c) Realize 10.000 simulações estatísticas para as duas estratégias stick e switch do problema de Monty Hall.

   (d) Para cada uma das simulações, verifique se os resultados seguem uma distribuição normal utilizando um gráfico quantil-quantil (*qq-plot*). Se a distribuição for normal, plote o gráfico da função de distribuição do "conceito de interesse" para visualizar a probabilidade de ganhar o prêmio com as estratégias stick e switch. Obs.: Para um conjunto de dados normalmente distribuídos, você pode calcular a média e considerá-la uma boa estimativa da média populacional.

   (e) Realize 10.000 simulações estatísticas para quatro grupos de pessoas, de 10, 30, 50 e 70 pessoas, do problema dos aniversários.

   (f) Para cada simulação, verifique se os resultados seguem uma distribuição normal utilizando um gráfico quantil-quantil (*qq-plot*). Se a distribuição for normal, plote o gráfico de distribuição do "conceito de interesse" para visualizar a probabilidade de que duas pessoas compartilhem o mesmo aniversário em grupos de 10, 30, 50 e 70 pessoas

   (g) Discuta o impacto da visualização da distribuição dos dados na compreensão dos dois problemas.

   Use a função ggplot na renderização dos gráficos quando pertinentes.

Capítulo 9

Estatística de Inferência: Estimativas Pontual e Intervalar

Embora a estatística descritiva, explorada no Capítulo 6, seja uma ferramenta valiosa para resumir e visualizar conjuntos de dados, sua aplicação direta em populações muito grandes se torna impraticável ou impossível devido ao tamanho, custo ou complexidade dos dados envolvidos. Imaginem tentar calcular estatísticas descritivas, como a média de altura, o desvio padrão de renda ou os quartis de idade, para a população inteira de um país com centenas de milhões de habitantes. A tarefa de coletar dados de cada indivíduo seria não apenas extremamente demorada e dispendiosa, consumindo vastos recursos financeiros, humanos e temporais, mas também logística e operacionalmente inviável. Organizar, armazenar e processar um volume tão massivo de informações representaria um desafio.

Em vez de tentar examinar toda a população, os estatísticos recorrem a técnicas probabilísticas, introduzidas no Capítulo 8, para extrair informações e conclusões válidas de uma população a partir de amostras representativas. Uma **amostra** é considerada **representativa** quando reflete fielmente as características importantes da população de onde foi retirada. A Seção 8.1 introduz dois conceitos fundamentais: o **espaço amostral** Ω, que é o conjunto de todos os resultados possíveis em um experimento aleatório, e a **amostra**, que é um subconjunto de observações retiradas de uma população. A **amostragem** é o processo de seleção de observações de uma população para compor as amostras de um experimento. Esse processo é crucial para garantir que as conclusões tiradas das amostras possam ser generalizadas com segurança para a população inteira.

Enquanto a estatística descritiva nos proporciona ferramentas para resumir e explorar dados observados de uma população, e a probabilidade nos oferece uma medida da certeza/incerteza de ocorrência de eventos específicos em um conjunto de resultados possíveis, a **estatística de inferência** foca em fazer inferências abrangentes sobre as estatísticas da população, como média, variância, proporção e outras, utilizando apenas uma coleção de N amostras, cada uma das quis contendo n observações. Essa capacidade de **extrapolar** informações de um conjunto de amostras para toda a população se

revela muito útil no processamento preciso de grandes volumes de dados, especialmente quando a distribuição subjacente é desconhecida. Uma ampla aplicação em diversos campos, como ciências sociais, saúde, negócios e engenharia, pode se beneficiar dessa capacidade. No cerne da estatística inferencial reside o conceito de **estimativa**, que envolve o cálculo de valores aproximados, denominados **estimativas**, para os parâmetros populacionais (estatísticas resumidas da população, conforme a Seção 6.1), com base em amostras coletadas.

A estatística de inferência permite que profissionais de diversas áreas obtenham *insights* relevantes, realizem análises significativas, tomem decisões informadas, façam previsões confiáveis e desenvolvam estratégias eficazes com base em informações limitadas, mas representativas, da população em estudo. Muitas formas de inferência da distribuição da população dependem do conhecimento sobre a variabilidade das amostras revelada pelas distribuições de estatísticas amostrais, que são abordadas na Seção 9.2. Na Seção 9.1, veremos como a estimativa pontual busca prever um único valor representativo de um parâmetro específico da população, como a média populacional derivada das médias amostrais. Já na Seção 9.3, os intervalos de confiança fornecem uma faixa de valores plausíveis para o parâmetro populacional, capturando a incerteza associada à estimativa pontual.

Nesse contexto, a visualização dos dados desempenha um papel importante na compreensão das características subjacentes e na identificação das distribuições dos dados. Ao analisar graficamente as distribuições das estatísticas, somos capazes de **identificar padrões, assimetrias** e outras características relevantes que influenciam a escolha do método mais apropriado para realizar inferências sobre a população. Além disso, a visualização ajuda na **validação de pressupostos estatísticos**, como normalidade e homogeneidade de variâncias entre as amostras, que são fundamentais para a aplicação correta das técnicas de inferência estatística. Ao longo do capítulo, são explorados os recursos gráficos que proporcionam uma compreensão melhor do comportamento dos dados subjacentes aos resultados de uma estatística de inferência.

9.1 Estimativas Pontuais

Em estatística, o objetivo frequentemente reside em compreender as características de uma população. Um **parâmetro** (populacional) é uma quantidade numérica fixa e teórica que resume um aspecto específico da população inteira, tipicamente associado a uma função estatística. Ele representa o verdadeiro valor dessa característica na totalidade do grupo. Teoricamente, um parâmetro pode ser calculado diretamente a partir dessa função estatística que mapeia todos os dados da população em uma única estatística populacional. Costuma-se representar os parâmetros populacionais com a letra grega, tais como θ, μ e σ. Quando se pode acessar e analisar os dados de toda a população, as estatísticas populacionais resultantes correspondem diretamente aos respectivos parâmetros populacionais.

No entanto, na maioria das situações práticas, obter dados da população na sua totalidade é inviável devido a limitações de tempo, custo ou acessibilidade. Nesses casos, recorremos à coleta de uma amostra de n observações, que é um subconjunto representativo da população. A partir dos dados dessa amostra, utilizamos um estimador para inferir ou aproximar o valor do parâmetro populacional desconhecido. Um **estimador amostral** é uma função estatística que mapeia os dados de uma amostra em uma **estatística amostral**, que chamamos de **estimativas**. Geralmente, os estimadores são representados pelos símbolos dos parâmetros correspondentes, acrescidos de um acento circunflexo, como $\hat{\theta}$, $\hat{\mu}$ e $\hat{\sigma}$.

As estimativas podem variar de uma amostra para outra devido à aleatoriedade inerente à amostragem de observações. É, portanto, conveniente tratar cada valor individual x_i da variável como uma variável aleatória X_i. Assim, definimos um estimador $\hat{\theta}$ como uma função de n variáveis aleatórias correspondentes às n observações de uma amostra:

$$\hat{\theta} = f(X_1, X_2, \cdots, X_n). \tag{9.1}$$

Por exemplo, se estamos interessados no parâmetro "média populacional" μ_A de uma variável aleatória A, o estimador $\hat{\mu}$ pode ser a média amostral e a estimativa será o valor específico dessa média calculado a partir das n variáveis aleatórias da amostra

$$\hat{\mu}_1(X_1, X_2, \cdots, X_n) = \frac{\sum_{i=1}^{n} X_i}{n}.$$

O exemplo apresentado na Seção 8.6 demonstra uma estimativa da frequência de ocorrência de cara (C) no lançamento de uma moeda. Para isso, foram realizadas 1000 amostras de 10 lançamentos cada. Utilizou-se a frequência de ocorrência de C em cada uma dessas amostras como estimador, resultando em 1000 estimativas do parâmetro de frequência populacional.

Embora frequentemente se utilize a mesma função estatística tanto para definir o parâmetro populacional quanto para estimar seu valor a partir da amostra, isso não é uma exigência. A função usada no estimador pode ser diferente, dependendo do contexto ou das propriedades desejadas da estimativa. Nada impede que alguém defina um estimador de média populacional como uma função que sempre assume o valor do primeiro, segundo ou i-ésimo valor sorteado na amostra

$$\hat{\mu}_2(X_1, X_2, \cdots, X_n) = X_i. \;\; i \in \{1, \cdots, n\}$$

A lógica fundamental por trás da utilização de estimadores reside na esperança de que a amostra seja representativa da população. Se a amostra for selecionada de forma adequada (idealmente, através de amostragem aleatória), as características da amostra (resumidas pelas estatísticas amostrais) tendem a refletir as características da população. Consequentemente, espera-se que o valor da

estimativa obtida a partir do estimador seja próximo do verdadeiro valor do parâmetro populacional.

Na estatística de inferência, os estimadores são ferramentas básicas para fazer inferências sobre parâmetros desconhecidos da população, principalmente com base em amostras. Um estimador $\hat{\theta}$ pode ser avaliado quanto a características que determinam sua qualidade e utilidade em estimar o parâmetro θ de interesse. Seguem-se as descrições das principais propriedades pelas quais os estimadores são avaliados [52]:

Imparcialidade: $\hat{\theta}$ é considerado **não-viciado**, ou **não-viesado**, se, em média, ele acerta o valor real do parâmetro θ. Isso significa que o valor esperado do estimador é igual ao verdadeiro valor do parâmetro populacional:

$$E(\hat{\theta}) = \theta.$$

Em termos práticos, se coletarmos muitas amostras da mesma população e aplicarmos $\hat{\theta}$ a cada uma, a média dessas estimativas tenderá a ser igual ao valor verdadeiro de θ. Portanto, um **estimador imparcial** não sistematicamente superestima nem subestima o parâmetro.

Eficiência: $\hat{\theta}$ é considerado **eficiente**, se ele possui a menor variância possível entre todos os estimadores imparciais do mesmo parâmetro. Ou seja, entre dois estimadores imparciais $\hat{\theta}_1$ e $\hat{\theta}_2$, diz-se que $\hat{\theta}_1$ é mais eficiente se:

$$Var(\hat{\theta}_1) < Var(\hat{\theta}_2).$$

Dizemos que um estimador eficiente fornece estimativas mais precisas, pois suas estimativas tendem a estar mais concentradas em torno do valor verdadeiro do parâmetro.

Consistência: $\hat{\theta}$ é **consistente** se, à medida que o tamanho n da amostra aumenta indefinidamente, a probabilidade do estimador se aproxima do valor verdadeiro do parâmetro θ também aumenta. Em outras palavras, o estimador converge em probabilidade para o valor real do parâmetro populacional à medida que o tamanho da amostra aumenta

$$\begin{aligned} \lim_{n\to\infty} E(\hat{\theta}) &= \theta. \\ \lim_{n\to\infty} Var(\hat{\theta}) &= 0. \end{aligned}$$

Invariância: $\hat{\theta}$ é **invariante** a transformações da variável de interesse, ou seja

$$\hat{\theta}(T(X)) = T(\hat{\theta}(X)).$$

Isso significa que se aplicarmos uma transformação T à variável de interesse, o estimador deve refletir essa transformação na mesma relação.

Robustez: $\hat{\theta}$ é considerado **robusto** quando continua fornecendo boas estimativas mesmo sob pequenas violações dos pressupostos do modelo estatístico, como não normalidade dos dados ou presença de valores extremos (em inglês, *outliers*). Ele não é excessivamente sensível a observações anômalas ou desvios moderados dos pressupostos, e tende a manter sua validade em situações do mundo real, onde os dados raramente seguem perfeitamente o modelo teórico.

Suficiência: Um estimador é suficiente para um parâmetro θ se ele retém toda a informação da amostra relevante para estimar esse parâmetro. Formalmente, uma estatística $T(X)$ é suficiente para θ se a distribuição condicional da amostra X, dado $T(X)$, não depende de θ. Em outras palavras, uma vez que conhecemos o valor da estatística suficiente, nenhuma informação adicional da amostra contribui para melhorar a estimação de θ.

Escolher um bom estimador é um desafio para os estatísticos, pois envolve equilibrar essas características para obter estimativas precisas e confiáveis do parâmetro de interesse. Um bom estimador garante a qualidade das análises estatísticas e das conclusões derivadas delas. Tipicamente, os estimadores "naturais" são derivados das estatísticas descritivas amostrais, que consistem nos n valores observados em uma amostra (de tamanho n) de uma população, modelados como variáveis aleatórias $\{X_1, X_2, \cdots, X_n\}$, como ilustram os seguintes estimadores.

Média amostral: É calculada como a média aritmética dos valores observados em uma amostra específica:

$$\hat{\mu} = \overline{X} = \frac{\sum_{i=1}^{n} X_i}{n}. \tag{9.2}$$

Variância amostral: É uma medida da dispersão dos valores observados em relação à média amostral $\overline{X}$:

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{n} (X_i - \overline{X})^2}{n}.$$

Quando a variância amostral é considerada como uma estimativa da variância populacional, o divisor $n-1$ é usado para corrigir o viés na estimativa da variância porque a média amostral $\overline{X}$ usada na variância amostral não é exatamente igual à média μ da população completa:

$$s^2 = \frac{\sum_{i=1}^{n} (X_i - \overline{X})^2}{n-1}. \tag{9.3}$$

Desvio padrão amostral: É simplesmente a raiz quadrada da variância amostral. Ele nos dá uma medida mais fácil de entender sobre a dispersão dos dados em torno da

média amostral:

$$s = \sqrt{s^2}. \tag{9.4}$$

Proporção amostral: É a frequência relativa de um determinado resultado X_i em uma amostra, ou seja, o número de ocorrências de um evento dividido pelo tamanho da amostra n:

$$\hat{p} = \frac{\text{número de ocorrências de } X_i \text{ na amostra}}{n}. \tag{9.5}$$

Numa distribuição Bernoulli, onde temos dois resultados possíveis (por exemplo, sucesso e falha), se a proporção de sucesso é p, então a proporção de falha é $1 - p$. A **dispersão** (*spread*, em inglês) entre essas duas proporções $p - (1 - p)$, que é equivalente a $2p - 1$, representa de fato a diferença entre as probabilidades de sucesso e falha na amostra [70].

Embora os estimadores baseados em estatísticas descritivas amostrais sejam comumente utilizados na prática estatística devido às propriedades desejáveis que frequentemente apresentam, existem situações em que outros tipos de estimadores são preferidos para estimar parâmetros desconhecidos da população. Por exemplo, estimadores fundamentados em modelos teóricos, que se baseiam em princípios já estabelecidos, e estimadores funcionais, que não necessariamente dependem de uma sustentação teórica, podem ser aplicados sem a necessidade direta de amostras.

9.2 Distribuições Amostrais

A distribuição de estatísticas amostrais descreve a variabilidade das estatísticas calculadas a partir de todas as possíveis amostras que poderiam ser extraídas de uma população. Essa distribuição é essencial para a estatística inferencial, pois permite fazer inferências sobre parâmetros populacionais com base em amostras limitadas, além de fornecer *insights* sobre a precisão das estimativas e a incerteza associada aos resultados. Dois tipos comuns de distribuições amostrais são: a **distribuição amostral sem reposição**, em que as amostras são retiradas de uma população sem que cada unidade possa ser escolhida mais de uma vez, e a **distribuição amostral com reposição**, na qual cada unidade da população pode ser escolhida múltiplas vezes. A primeira abordagem é uma prática estatística tradicional, enquanto a segunda é comum em métodos como o *bootstrap*, que visa estimar a variabilidade de uma estatística a partir da geração de amostras independentes de uma amostra original.

Ao realizarmos uma amostragem de uma população, obtemos N estimativas $\overline{X}_i$ para diferentes amostras i de tamanho n. Ao reunir essas estimativas, formamos uma distribuição de frequência que descreve a ocorrência de cada valor entre as N amostras de tamanho n extraídas da população por um estimador específico (Seção 6.1.1). Essa frequência pode ser interpretada como uma medida empírica da probabilidade associada a cada valor, aproximando-se da distribuição amostral, como ilustra a

Figura 9.1. Em outras palavras, as **distribuições amostrais** representam probabilisticamente as estimativas obtidas de diferentes amostras da mesma população, apresentando características próprias, como forma, tendência central e variabilidade. Embora não reproduzam exatamente a distribuição de probabilidade da população, as distribuições amostrais são amplamente utilizadas na estatística inferencial para estimar parâmetros populacionais, evidenciando como essas amostras podem variar em torno desses parâmetros.

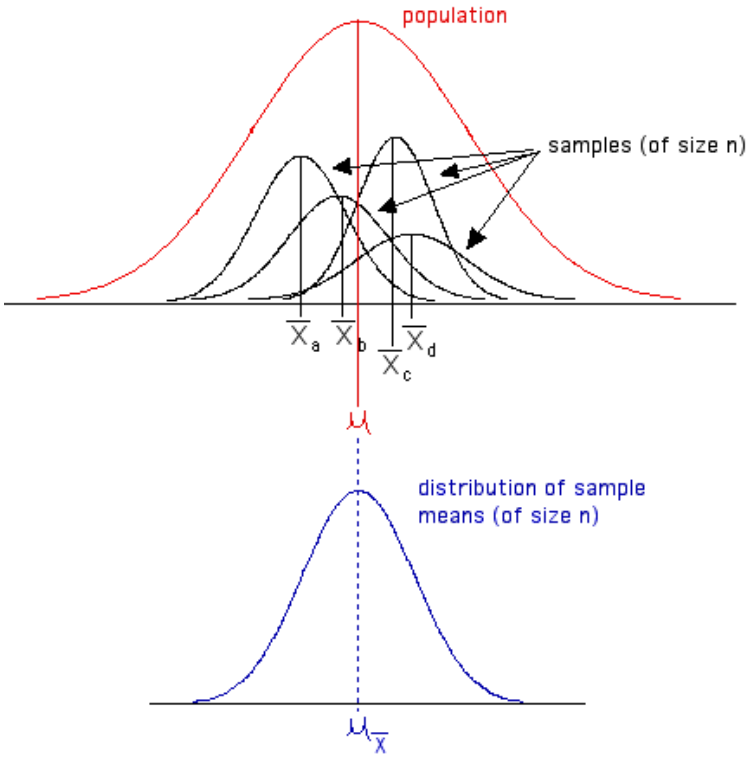


Figura 9.1: Variabilidade das estatísticas das diferentes amostras de mesmo tamanho de uma população refletidas numa distribuição amostral (Fonte)

As distribuições amostrais fornecem informações importantes sobre as propriedades de um **estimador** (Seção 9.1). Elas são, de fato, as **distribuições de probabilidade de um estimador**. Ao analisar a distribuição amostral de um estimador, os estatísticos podem entender a variabilidade do estimador, fazer inferências sobre a confiabilidade e a eficácia do estimador em estimar o verdadeiro valor do parâmetro populacional. Por exemplo, considerando uma característica populacional representada por X, com as estatísticas, média μ e variância $\sigma^2 = Var(X)$, a distribuição amostral de N amostras de tamanho n pode fornecer as seguintes informações sobre o estimador:

Tendência central: A tendência central das estimativas obtidas a partir do estimador, geralmente representada pela média da distribuição amostral, é expressa por

$$\mu_{\overline{X}} = \frac{\sum_{i=1}^{N} \overline{X}_i}{N}, \tag{9.6}$$

onde $\overline{X}_1, \overline{X}_2, \cdots, \overline{X}_N$ são as médias das N amostras. Se assumirmos que a média das

médias amostrais de um estimador não tendencioso é igual à média da população μ, então o estimador tende a se concentrar em relação ao verdadeiro valor do parâmetro populacional μ.

Variabilidade: A variabilidade do estimador é representada pela dispersão dos valores na distribuição amostral, geralmente medidos pelo desvio padrão ou variância. Seja $\overline{X}$ a média de uma amostra e considere que X_{ij} de cada observação j na amostra X_i tenha a mesma variabilidade em relação à média da amostra $\overline{X}_i$. Se essa variabilidade é a mesma que a variabilidade da população inteira $Var(X)$, podemos calcular a variabilidade da média amostral usando a fórmula:

$$Var(\overline{X}) = Var(\frac{\overline{X}_1 + \overline{X}_2 + \cdots + \overline{X}_N}{N}).$$

Ao expandir esta expressão, temos

$$\begin{aligned}\frac{\sum_{i=1}^{N} Var(\overline{X}_i)}{N} &= \frac{\sum_{i=1}^{N} \frac{\sum_{j=1}^{n} Var(X_{ij})}{n^2}}{N} = \frac{\sum_{i=1}^{N} \frac{nVar(X)}{n^2}}{N} \\ &= \frac{N\frac{Var(X)}{n}}{N} = \frac{Var(X)}{n} = \frac{\sigma^2}{n}, \end{aligned} \tag{9.7}$$

onde n é o tamanho da amostra e σ^2, a variância populacional. Eq. 9.7 revela que a variabilidade de uma distribuição amostral é inversamente proporcional ao tamanho da amostra n. Uma menor variabilidade indica que o estimador é mais consistente e preciso. Podemos afirmar que, à medida que o tamanho da amostra n aumenta, a variabilidade diminui e, consequentemente, a precisão do estimador aumenta. Em outras palavras, com amostras maiores, o estimador tende a se concentrar mais próximo do verdadeiro valor do parâmetro populacional. Isso é porque uma amostra maior fornece mais informações sobre a população, resultando em estimativas mais confiáveis e menos susceptíveis a flutuações aleatórias. A Figura 9.2 mostra a distribuição de probabilidade populacional assimétrica dos pesos das maçãs (gráfico em cinza) e duas distribuições amostrais. Estas são compostas por N=500.000 amostras de tamanho $n = 5$ (gráfico em vermelho) e $n = 20$ (gráfico em azul). Observe que uma amostra de tamanho maior (gráfico em azul) tem uma dispersão menor do que uma de tamanho menor (gráfico em vermelho).

Erro padrão: O erro padrão (*standard error*, em inglês) de um estimador é definido como o desvio padrão $\sigma_{\overline{X}}$ da distribuição das estimativas amostrais do estimador. Ele é uma medida da variabilidade das médias amostrais em torno da verdadeira média populacional μ. Quanto menor o erro padrão, mais precisas são as estimativas da média populacional μ feitas com base em amostras da população. O cálculo do erro

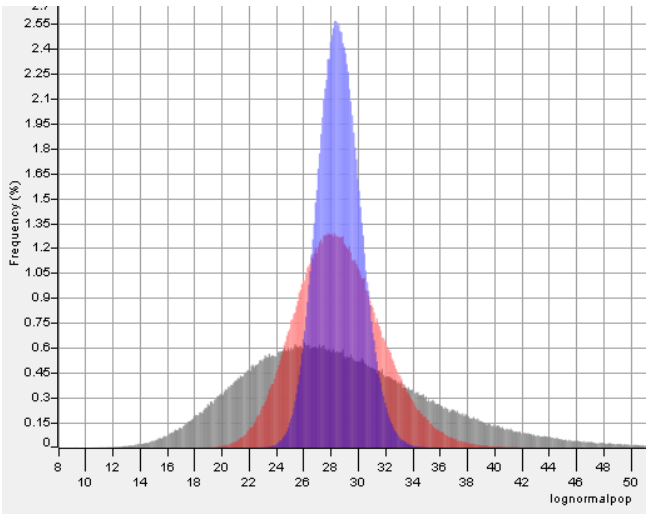


Figura 9.2: Distribuição populacional dos pesos de um conjunto de maçãs, com μ=100 e σ=15 (gráfico em cinza), juntamente com duas distribuições amostrais geradas a partir deste conjunto. As amostras têm tamanhos de n=5 (gráfico em vermelho) e n=20 (gráfico em azul), respectivamente, com 500.000 amostras em cada caso. (Fonte)

padrão $SE(\overline{X})$ envolve o desvio padrão σ da população e o tamanho da amostra n, conforme mostrado pela seguinte equação

$$SE(\overline{X}) = \sigma_{\overline{X}} = \sqrt{Var(\overline{X})} = \frac{\sigma}{\sqrt{n}}. \tag{9.8}$$

Isso confirma a tendência de diminuição da dispersão das médias amostrais em torno da média populacional com o aumento do tamanho das amostras. O Teorema Central do Limite (Seção 8.4) fundamenta essa observação.

A Figura 9.3 ilustra graficamente a diferença entre o desvio padrão de uma distribuição populacional e o erro padrão do estimador.

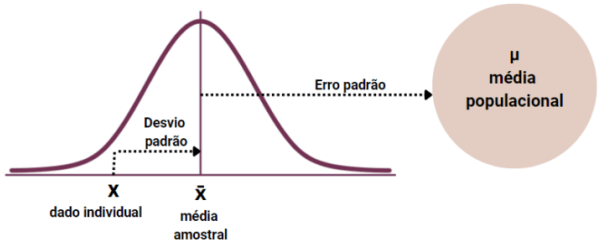


Figura 9.3: Desvio padrão e erro padrão. (Fonte)

O erro padrão foca na precisão da estimativa de um parâmetro populacional usando estatísticas amostrais, enquanto o desvio padrão descreve a variabilidade dos dados dentro de um conjunto. Eles servem a propósitos distintos na análise estatística.

Observe que, como a variância populacional σ^2 é geralmente desconhecida, a prática comum é substituí-la pela variância amostral s^2, calculada a partir dos dados da

amostra, conforme a definição na Equação 9.3.

Finalmente, é importante ressaltar as relações fundamentais entre as propriedades das distribuições amostrais assintoticamente normais, a exemplo da distribuição de médias amostrais, e as características da população de origem, conforme postulado pelo Teorema Central do Limite. Esse teorema afirma que, à medida que o tamanho da amostra n aumenta, a distribuição das médias amostrais se aproxima de uma distribuição normal, independentemente da forma da distribuição original da população, desde que a variância populacional seja finita. É importante distinguir que o que determina essa aproximação à normalidade não é o número N de amostras coletadas, mas sim o tamanho de cada amostra n. À medida que n cresce, a variabilidade das médias amostrais, expressa pelo erro padrão, diminui, estreitando a distribuição amostral e tornando-a mais simétrica e próxima da normal. Portanto, embora mais amostras possam melhorar a precisão empírica de uma análise, é o aumento do tamanho da amostra que tem impacto direto sobre a forma e a dispersão da distribuição amostral.

Em geral, para muitas distribuições populacionais, um tamanho de amostra de n$\geq$30 é frequentemente considerado suficiente para que a distribuição das médias amostrais seja razoavelmente bem aproximada por uma distribuição normal. No entanto, se a distribuição populacional for extremamente não normal ou tiver caudas pesadas, tamanhos de amostra maiores podem ser necessários para alcançar uma boa aproximação. Isso é aplicável tanto a distribuições amostrais com reposição quanto sem reposição, desde que o tamanho da amostra seja suficientemente grande em relação ao tamanho da população. Quando as amostras são extraídas sem reposição, a variabilidade entre elas pode ser reduzida devido à ausência de repetição de unidades, o que pode limitar a sua aplicação. Para nossos propósitos de análise visual, assumiremos que a representatividade da população não é crítica e que a duplicação de unidades não apresenta problemas, optando assim por utilizar distribuições amostrais com reposição na maioria das análises.

9.2.1 Distribuições de Proporções Amostrais

O Teorema Central do Limite também se aplica à proporção amostral como **estimador da proporção populacional**, de modo que, para tamanhos de amostra suficientemente grandes, sua distribuição amostral é aproximadamente normal. Isso é especialmente útil em estatísticas inferenciais, pois permite que, à medida que o tamanho da amostra n aumenta, a distribuição das proporções amostrais se aproxime de uma distribuição normal, independentemente da distribuição da população original. Essa propriedade é extremamente útil em contextos como pesquisas de opinião, estudos de mercado e saúde pública, onde frequentemente estamos interessados em estimar proporções (por exemplo, a proporção de pessoas que apoiam uma determinada política ou que possuem uma condição de saúde). O exemplo apresentado na Seção 8.6.1 é uma aplicação do Teorema Central do Limite na **estimativa da proporção populacional** p de "sair cara nos lançamentos de uma moeda não viciada" a partir da distribuição da proporção amostral $\hat{p}$. Para uma amostra de n variáveis aleatórias com distribuição

de Bernoulli, Y_1, Y_2 $\cdots$, Y_n, onde a esperança de cada variável é $E(Y_i) = \mu_{Y_i} = p$ e a variância de cada variável é dada pela Eq. 8.34, um estimador não viciado $\hat{p}$, usando essas n variáveis aleatórias, é dado por

$$E(\hat{p}) = E(\sum_{i=1}^{n} \frac{Y_i}{n}) = p \quad Var(\hat{p}) = Var(\sum_{i=1}^{n} \frac{Y_i}{n}) = \frac{p(1-p)}{n}$$

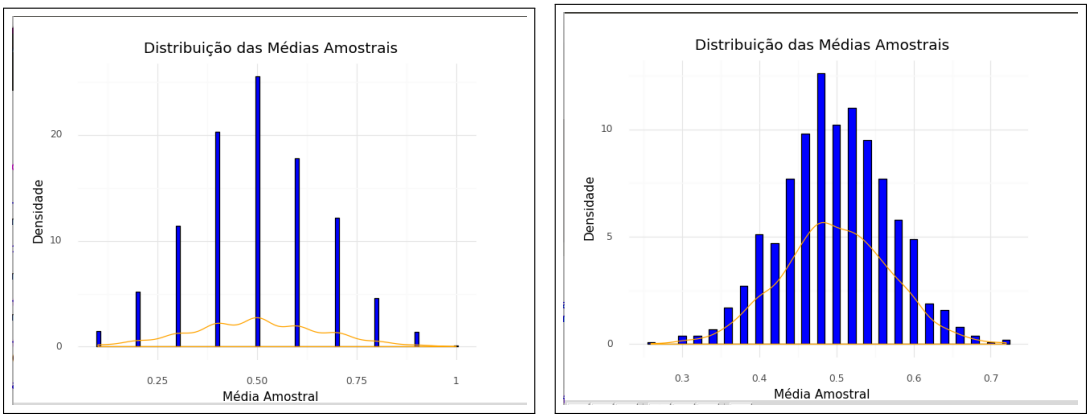
Quando n é grande o suficiente para que a aproximação normal seja válida para a distribuição de proporções amostrais, é vantajoso recorrer a essa distribuição para calcular probabilidades relacionadas a uma proporção amostral. Isso simplifica significativamente a análise estatística e a interpretação dos resultados, devido à familiaridade e facilidade de cálculo associadas à distribuição normal.

Pode-se plotar a distribuição de frequência de ocorrência das proporções amostrais usando geom_histogram disponível na biblioteca ggplot2 (R) ou plotnine (Python). Para ilustrar, vamos plotar as médias amostrais mean_samples geradas no exemplo da Seção 8.6.1 como mostrado na Figura 9.4a. Adicionalmente, vamos mapear a variável y para a estatística de densidade calculada pelo ggplot e plotar a curva desta distribuição de densidade de probabilidade em laranja para visualizar o formato da distribuição. Repetimos o processo para $n = 50$, $n = 200$ e $n = 500$ para ilustrar comparativamente na Figura 9.4 como, à medida que n cresce, os erros padrão diminuem e a curva de distribuição aproxima da curva de distribuição normal.

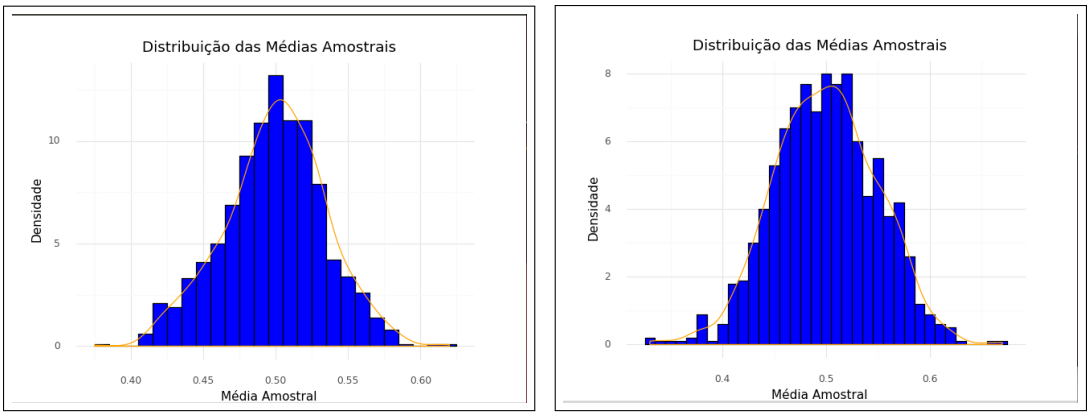
R: Embora possa usar o vetor mean_samples diretamente sem convertê-lo em um data.frame, é uma boa prática criar um data.frame para manter consistência e facilitar a manipulação dos dados posteriormente, se necessário.

```
# Carregando a biblioteca ggplot2
library(ggplot2)


# Criando um data frame com os dados das médias amostrais
df <- data.frame(mean_samples)


# Plotando a distribuição das médias amostrais
ggplot(df, aes(x=mean_samples)) +
  geom_histogram(aes(y=..density..), binwidth=0.01, fill="blue", color="black") +
  labs(x="Média Amostral", y="Densidade", title="Distribuição das Médias Amostrais") +
  geom_density(color="orange") +
  theme_minimal()
```

Python: As médias amostrais mean_samples foram processadas usando os arranjos do pacote numpy, que são otimizados para operações matemáticas. Para visualizar esses dados, usamos pd.DataFrame() para convertê-los em uma estrutura tabular.



(a) n=10, $\mu_{\overline{X}}$=0.4956, $SE(\overline{X})$=0.004897 (b) n=50, $\mu_{\overline{X}}$=0.5011, $SE(\overline{X})$=0.06727



(c) n=200, $\mu_{\overline{X}}$=0.5002, $SE(\overline{X})$=0.0011025 (d) n=500, $\mu_{\overline{X}}$=0.5009, $SE(\overline{X})$=0.00071

Figura 9.4: Distribuição amostral resultante de 1000 amostras, cada uma composta por n lançamentos. As amostras de $n = 10$ lançamentos são as do exemplo dado na Seção 8.6.1. Note como a distruibução representada pela curva em laranja tende a uma distribuição normal quando n cresce.

```
# Carreegando os pacotes
import pandas as pd
from plotnine import *

# Criando um DataFrame com os dados das médias amostrais
df = pd.DataFrame({'mean_samples': mean_samples})
# Plotando a distribuição das médias amostrais
(ggplot(df, aes(x='mean_samples')) +
 geom_histogram(aes(y='stat(density)'), binwidth=0.01, fill='blue', color='black') +
 labs(x='Média Amostral', y='Densidade', title='Distribuição das Médias Amostrais') +
 geom_density(color='orange') +
 theme_minimal()
)
```

No exemplo da Seção 8.6.1, cada amostra de tamanho n é gerada aleatoriamente com base numa

proporção pré-definida, considerando uma população de tamanho infinitamente grande. Quando a população é conhecida, amostramos as n observações das N amostras, **com reposição**, diretamente a partir dessa população. Isso pode ser simulado em R e em Python, como demonstram os seguintes códigos que geram uma distribuição amostral de 33 amostras de tamanho 50, com reposição, a partir de uma população de 2400 bolinhas vermelhas e azuis previamente definidas. A Figura 9.5 ilustra uma distribuição amostral gerada.

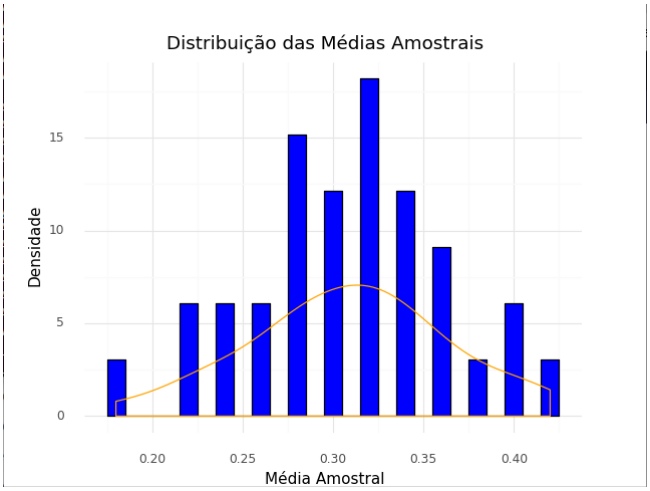


Figura 9.5: Distribuição amostral de $N = 33$ amostras, cada uma com tamanho $n = 50$, obtidas de uma população de 2400 bolinhas azuis e vermelhas, com uma proporção de 70% para azuis e 30% para vermelhas.

R: São usadas as bibliotecas rsample e ggplot2. A biblioteca rsample oferece uma variedade de métodos para realizar amostragem e reamostragem.

```
library(rsample)
library(ggplot2)

# Definição da população
set.seed(123)  # Define uma semente para reproducibilidade

# Especificar proporção desejada de bolinhas vermelhas e azuis
prop_vermelhas <- 0.3  # Por exemplo, 30% vermelhas
prop_azuis <- 1 - prop_vermelhas  # O restante, 70%, será azul

# Criar vetor com 2400 bolinhas, com a proporção especificada de vermelhas e azuis
populacao <- sample(c("Vermelha", "Azul"),
size = 2400,
replace = TRUE,
prob = c(prop_vermelhas, prop_azuis))
```

```
# Geração de 33 amostras de tamanho 50
amostras <- rep_sample_n(populacao, size = 50, reps=33)

# Calcular as médias amostrais
medias_amostrais <- sapply(amostras, function(x) mean(x$populacao))

# Criar um data frame com as médias amostrais
df <- data.frame(mean_samples = medias_amostrais)

# Plotando a distribuição das méidas amostrais
ggplot(df, aes(x = mean_samples)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.01, fill = "blue",
  color = "black") +
  geom_density(color = "orange") +
  labs(x = "Média Amostral", y = "Densidade",
  title = "Distribuição das Médias Amostrais") +
  theme_minimal()
```

Python: São usadas as bibliotecas numpy e pandas e plotnine.

```
import numpy as np
import pandas as pd
from plotnine import *

# Definição da população
np.random.seed(123)  # Define uma semente para reproducibilidade

# Especificar proporção desejada de bolinhas vermelhas e azuis
prop_vermelhas = 0.3  # Por exemplo, 30% vermelhas
prop_azuis = 1 - prop_vermelhas  # O restante, 70%, será azul

# Criar vetor com 2400 bolinhas, com a proporção especificada de vermelhas e azuis
populacao = np.random.choice(["Vermelha", "Azul"], size=2400, replace=True,
p=[prop_vermelhas, prop_azuis])

# Geração de 33 amostras de tamanho 50
```

```
amostras = [np.random.choice(populacao, size=50, replace=False) for _ in range(33)]

# Calcular as médias amostrais
medias_amostrais = [np.mean(amostra == "Vermelha") for amostra in amostras]

# Criar um DataFrame com as médias amostrais
df = pd.DataFrame({"mean_samples": medias_amostrais})

# Plotando a distribuição das méidas amostrais
(ggplot(df, aes(x="mean_samples")) +
 geom_histogram(aes(y="..density.."), binwidth=0.01, fill="blue", color="black") +
 geom_density(color="orange") +
 labs(x="Média Amostral", y="Densidade",
 title="Distribuição das Médias Amostrais") +
 theme_minimal())
```

9.2.2 Distribuições Teóricas

As distribuições amostrais são fundamentais na inferência estatística, pois descrevem como uma estatística calculada a partir de uma amostra se comporta quando diferentes amostras são extraídas da mesma população. Em muitos casos, especialmente quando o tamanho da amostra é suficientemente grande ou o estimador tem comportamento assintoticamente normal, o Teorema Central do Limite garante que a distribuição amostral dessa estatística se aproxima de uma distribuição normal. No entanto, as distribuições amostrais não são necessariamente normais. Para algumas estatísticas, especialmente em contextos com amostras pequenas ou estimadores complexos, a distribuição amostral pode ser assimétrica ou seguir outras formas específicas. Nesses casos, uma das seguintes distribuições teóricas clássicas, ilustradas na Figura 9.6, pode fornecer uma aproximação adequada do comportamento dos dados observados, permitindo a aplicação rigorosa de técnicas estatísticas baseadas em modelos analíticos bem estabelecidos:

Distribuição Student (t): é definida em termos de distribuições amostrais, frequentemente usada para inferir sobre a média populacional a partir de uma amostra. Graficamente, é semelhante à normal, mas tem caudas mais longas, refletindo maior incerteza associada a amostras pequenas. É utilizada quando a amostra é pequena ($n \leq 30$) e a variância da população é desconhecida. A distribuição t é definida por um parâmetro chamado "graus de liberdade", que está relacionado ao tamanho da amostra. À medida que o tamanho da amostra aumenta, a distribuição t se aproxima

da distribuição normal. A convergência da distribuição t para a distribuição normal à medida que o tamanho da amostra aumenta está diretamente relacionada ao Teorema Central do Limite.

Distribuição Qui-quadrado: é uma distribuição assimétrica definida como a soma dos quadrados de k variáveis aleatórias Z_i independentes que seguem uma distribuição normal padrão $N(Z_i; 0, 1)$. É usada principalmente em testes de hipóteses sobre variâncias e em análises de tabelas de contingência. O único parâmetro que define a distribuição qui-quadrado é o número de graus de liberdade d, que é igual ao número de variáveis normais independentes que foram somadas e elevadas ao quadrado. Assim, uma distribuição qui-quadrado com $d = k$ graus de liberdade é denotada como $\chi^2(k)$. Graficamente, a distribuição qui-quadrado é assimétrica e tem uma cauda longa à direita. À medida que o número de graus de liberdade aumenta, a distribuição se aproxima de uma distribuição normal, mas é sempre assimétrica para graus de liberdade baixos.

Distribuição F: é definida pela razão de duas variáveis aleatórias, X_1^2 e X_2^2, que seguem distribuições qui-quadrado, cada uma dividida por seus respectivos graus de liberdade d_1 e d_2.

$$F = \frac{X_1^2/d_1}{X_2^2/d_2}. \tag{9.9}$$

É uma distribuição contínua utilizada principalmente em testes de hipóteses para comparar variâncias de duas populações. É fundamental em análises de variância (ANOVA) e em modelos de regressão, onde auxilia na avaliação da significância estatística dos resultados. Graficamente, esta distribuição é assimétrica e sempre tem caudas à direita. À medida que d_1 e d_2 aumentam, a distribuição se aproxima de uma distribuição normal, mas permanece assimétrica.

De forma mais geral, a estatística se apoia em diversas distribuições teóricas, que são modelos matemáticos idealizados usados tanto para representar o comportamento de variáveis populacionais, como características observadas diretamente nos indivíduos, quanto para descrever o comportamento de estatísticas descritivas amostrais, como médias, proporções e variâncias calculadas a partir das amostras. As funções de densidade de probabilidade (ver Seção 8.5.2) representam distribuições contínuas, enquanto as funções de massa de probabilidade (ver Seção 8.5.1) representam distribuições discretas.

Os recursos gráficos desempenham um papel importante na identificação e na verificação de qual distribuição teórica se ajusta melhor a uma distribuição de interesse. Eles ajudam verificar suposições antes de realizar análises estatísticas. Uma ferramenta comum é o **QQ-Plot** (gráfico quantil-quantil) (Figura 6.11), que compara os quantis de uma distribuição amostral com os quantis de uma distribuição teórica (como a normal ou a t). Se os pontos no QQ-Plot se alinharem aproximadamente

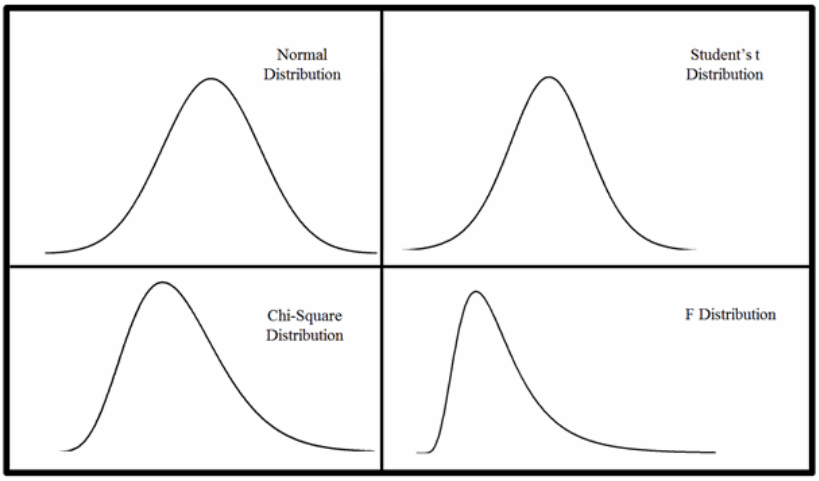


Figura 9.6: Modelos matemáticos de distribuições.

em uma linha reta, isso sugere que a amostra segue a distribuição teórica especificada. Se os pontos desviam significativamente dessa linha, isso sugere que os dados não satisfazem a normalidade. Essa visualização é extremamente útil para verificar suposições antes de realizar análises (estatísticas) de dados.

9.2.3 Normalização das Distribuições

A normalização de distribuições é uma prática comum em estatística devido à sua utilidade na comparação de conjuntos de dados, cálculo de probabilidades, modelagem estatística e simplificação de métodos analíticos. Ela se refere ao processo de ajustar a função de densidade de probabilidade para que a área total sob a curva seja igual a 1, garantindo que a distribuição represente uma probabilidade válida. Este processo varia com o tipo de distribuição. Para as distribuições teóricas, citadas na Seção 9.2.2, aplica-se os seguintes procedimentos:

Distribuição Normal: A normalização é feita transformando a distribuição original $N(X;\mu,\sigma^2)$ em uma **distribuição padronizada** $N(X;0,1)$, usando a fórmula:

$$Z = \frac{X-\mu}{\sigma} \tag{9.10}$$

Distribuição Student: A normalização na distribuição t é feita em relação aos graus de liberdade ($\nu = (n-1)$). Para uma variável T que segue a distribuição normalizada t, a normalização segue:

$$T = \frac{\bar{X}-\mu}{s/\sqrt{n}}, \tag{9.11}$$

onde $\bar{X}$ é a média da amostra, s é o desvio padrão da amostra e n é o tamanho da amostra.

Note que não faz sentido de falar em "normalização" da distribuição qui-quadrado e da distribuição F da mesma forma que fazemos com as distribuições z e t. A distribuição qui-quadrado é

uma distribuição que surge a partir da soma dos quadrados de variáveis aleatórias normais padrão independentes. Ela já é uma distribuição de probabilidade bem definida, caracterizada apenas pelo número de graus de liberdade k. A distribuição F é, por sua vez, caracterizada por dois conjuntos de graus de liberdade (d_1 e d_2). Esses graus de liberdade estão relacionados às amostras que estão sendo comparadas e não necessitam de ajustes ou normalizações adicionais para serem utilizados em testes estatísticos. A área total sob a curva de ambas as distribuições é sempre igual a 1.

Após as devidas normalizações, essas distribuições amostrais são, de fato, funções de probabilidade de variáveis aleatórias, como destacado na Seção 9.2.2. A partir dessas distribuições amostrais, podemos inferir a frequência de ocorrência $f(x)$ de um valor x de uma variável aleatória X como também a probabilidade de ocorrência $p(c \leq X \leq d)$ de valores pertencentes a um intervalo $[c, d] \in X$ (Eq. 8.9). Antes da era digital, estatísticos e pesquisadores frequentemente recorriam a tabelas de consulta para obter probabilidades e valores críticos associados a essas distribuições. Essas **tabelas de distribuição cumulativa** (tabela z, tabela t, tabela qui-quadrado e tabela F) eram compiladas a partir de cálculos matemáticos complexos e forneciam valores que permitiam a realização de testes estatísticos e a construção de intervalos de confiança.

Embora úteis, essas tabelas tinham limitações, como a necessidade de conhecimento prévio sobre a distribuição em questão e a dificuldade de encontrar valores para casos não tabulados. Hoje em dia, ferramentas estatísticas como R e Python oferecem pacotes que facilitam o acesso a essas distribuições. Esses pacotes permitem que os usuários realizem cálculos de probabilidades, testes de hipóteses e análises de dados de forma rápida e precisa. Com funções integradas, os estatísticos podem facilmente gerar valores, gráficos e relatórios, eliminando a necessidade de tabelas de consulta. Por exemplo, para calcular $P(x \leq d)$, em R, utilizamos as funções `pnorm()`, `pt()`, `pchisq()` e `pf()` para obter a probabilidade cumulativa até d nas distribuições padronizadas normal z, t, qui-quadrado e F, respectivamente. Em Python, as funções `norm.cdf()`, `t.cdf()`, `chi2.cdf()` e `f.cdf()` da biblioteca `scipy.stats` permitem calcular a probabilidade cumulativa para as respectivas distribuições.

9.2.4 Distribuições *Bootstrap*

As distribuições amostrais nos permitem compreender o comportamento de um estimador ao longo de diferentes amostras e quantificar sua variabilidade. Com base nelas, podemos construir intervalos de confiança para inferir o valor de um parâmetro populacional com um determinado nível de confiança, tornando a estimativa mais informativa e estatisticamente fundamentada. No entanto, a inferência estatística paramétrica[1], fundamentada em distribuições amostrais tradicionais, pode enfrentar desafios significativos, pois essas distribuições são derivadas matematicamente com base em suposições sobre a distribuição da população subjacente e em teoremas como o Teorema Central do Limite. Es-

[1] Técnicas paramétricas fazem suposições sobre a forma da distribuição dos dados (por exemplo: normal, t, exponencial), e muitas vezes exigem conhecimento (ou estimação) de parâmetros como média ou variância.

sas derivações frequentemente dependem do conhecimento ou da suposição da forma da distribuição populacional, o que dificulta a realização de inferências precisas sobre parâmetros e estatísticas de interesse. Além disso, muitas técnicas estatísticas tradicionais dependem de suposições fortes, como normalidade dos dados e homogeneidade da variância, que nem sempre podem ser verificadas ou satisfeitas na prática. Outro obstáculo comum é o tamanho reduzido da amostra, que pode resultar em estimativas instáveis ou pouco confiáveis da distribuição amostral.

Nesse contexto, as **distribuições *bootstrap*** surgem como uma alternativa. Elas podem ser entendidas como distribuições amostrais obtidas por reamostragem com reposição a partir da própria amostra observada. Isso significa que depois que um elemento é selecionado, ele é "reposto" no conjunto original antes da próxima seleção. . Em particular, não exige suposições teóricas sobre a distribuição dos dados, é altamente flexível e aplicável a uma ampla variedade de problemas estatísticos, independentemente da forma da distribuição original ou do tamanho da amostra. A principal vantagem das distribuições *bootstrap* é que fornecem uma aproximação empírica da distribuição amostral da estatística de interesse, permitindo a construção de intervalos de confiança e testes de hipóteses sem a necessidade de distribuições teóricas pré-definidas. Essa abordagem é viabilizada por meio da técnica de reamostragem *bootstrap* (*bootstrap resampling*, em inglês), na qual múltiplas amostras são geradas a partir da amostra original, com reposição, simulando o processo de amostragem repetida.

Introduzida por Bradley Efron em 1979, a técnica *bootstrap*[2] estima a distribuição de uma estatística de interesse a partir de uma única amostra de dados. No processo de **reamostragem *bootstrap***, múltiplas amostras de dados (chamadas de amostras *bootstrap*) são geradas a partir de uma amostra original por meio de **reamostragem com reposição**. Isso significa que observações são selecionadas aleatoriamente da amostra original e, após cada seleção, devolvidas à amostra para que possam ser selecionadas novamente. Isso garante que as amostras *bootstrap* tenham o mesmo tamanho que a amostra original e que tenham variações nas amostras, possibilitando estimar a distribuição da estatística de interesse.

Ao calcular a estatística de interesse em cada amostra *bootstrap*, obtemos uma série de estimativas. Reunindo essas estimativas, formamos uma distribuição de frequência que descreve a frequência com que cada valor ocorre entre os estimadores em N amostras de tamanho n retiradas de uma amostra original da população. Essa distribuição de frequência é uma aproximação empírica da distribuição amostral da estatística de interesse. A forma exata da distribuição *bootstrap* pode variar dependendo da natureza dos dados e da estatística sendo calculada.

Em muitos casos, especialmente quando o tamanho da amostra é suficientemente grande e o estimador é assintoticamente normal, a distribuição *bootstrap* das estatísticas estimadas pode se aproximar de uma distribuição normal. Embora o Teorema Central do Limite se aplique estritamente a amos-

[2]Não há uma tradução direta para *bootstrap* em português no contexto de estatística. O termo *bootstrap* se refere à ideia de "puxar-se pelos próprios cadarços", ou seja, "ter sucesso apenas pelos próprios esforços ou habilidades".

tras independentes da população, o processo de *bootstrap* simula empiricamente a variabilidade que se esperaria nesse cenário, por meio da reamostragem com reposição da amostra observada. Nessa condição, o *bootstrap* pode ser utilizado para estimar o erro padrão, construir intervalos de confiança baseados na normalidade e realizar testes estatísticos padronizados, aproveitando resultados teóricos bem estabelecidos. No entanto, é essencial destacar que a validade inferencial do bootstrap depende da qualidade da amostra original: é necessário que ela seja uma representação razoável da população subjacente, pois todas as reamostras são baseadas unicamente nessa amostra inicial.

Vamos exemplificar como calcular distribuições *bootstrap* a partir de uma amostra original obtida de uma população de 2400 bolinhas azuis e vermelhas, como apresentado na Seção 9.2.1, utilizando tanto R quanto Python. A Figura 9.7 exibe uma distribuição *bootstrap* derivada de uma amostra original de tamanho $n = 50$, com reposição. Ao compará-la com a distribuição amostral ilustrada na Figura 9.5, é bem significativa a semelhança entre as duas distribuições.

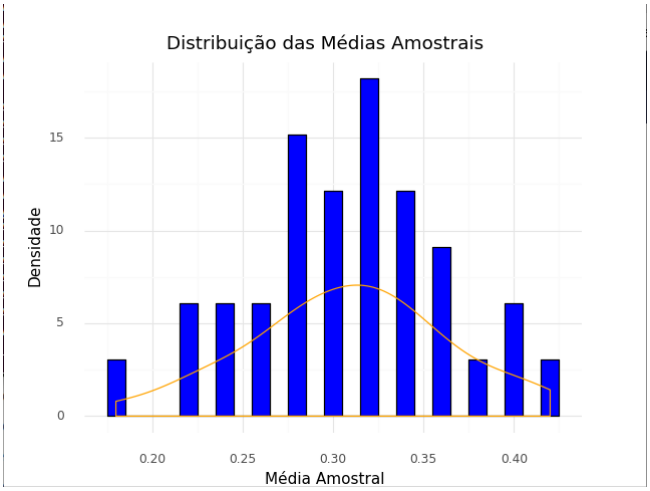


Figura 9.7: Distribuição *bootstrap* de $N = 33$ amostras, cada uma com tamanho $n = 50$, obtidas de uma amostra original de tamanho n=50 extraída a partir de uma população com uma proporção de 70% para azuis e 30% para vermelhas.

R:

```
library(rsample)
library(ggplot2)

# Definição da população
set.seed(123)  # Define uma semente para reproducibilidade

# Especificar proporção desejada de bolinhas vermelhas e azuis
prop_vermelhas <- 0.3  # Por exemplo, 30% vermelhas
prop_azuis <- 1 - prop_vermelhas  # O restante, 70%, será azul
```

```
# Criar vetor com 2400 bolinhas, com a proporção especificada de vermelhas e azuis
populacao <- sample(c("Vermelha", "Azul"),
size = 2400,
replace = TRUE,
prob = c(prop_vermelhas, prop_azuis))

# Geração de 1 amostra original de tamanho 50
amostra_original <- rep_sample_n(populacao, size = 50, reps=1)

# Geração de 33 amostras de tamanho 50 a partir da amostra original
amostras <- rep_sample_n(amostra_original, size = 50, reps=33, replace=TRUE)

# Calcular as médias amostrais
medias_amostrais <- sapply(amostras, function(x) mean(x))

# Criar um data frame com as médias amostrais
df <- data.frame(mean_samples = medias_amostrais)

# Plotando a distribuição das méidas amostrais
ggplot(df, aes(x = mean_samples)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.01, fill = "blue", color = "black
  geom_density(color = "orange") +
  labs(x = "Média Amostral", y = "Densidade", title = "Distribuição das Médias Amostr
  theme_minimal()
```

Python:

```
import numpy as np
import pandas as pd
from plotnine import *

# Definição da população
np.random.seed(123)  # Define uma semente para reproducibilidade

# Especificar proporção desejada de bolinhas vermelhas e azuis
prop_vermelhas = 0.3  # Por exemplo, 30% vermelhas
prop_azuis = 1 - prop_vermelhas  # O restante, 70%, será azul
```

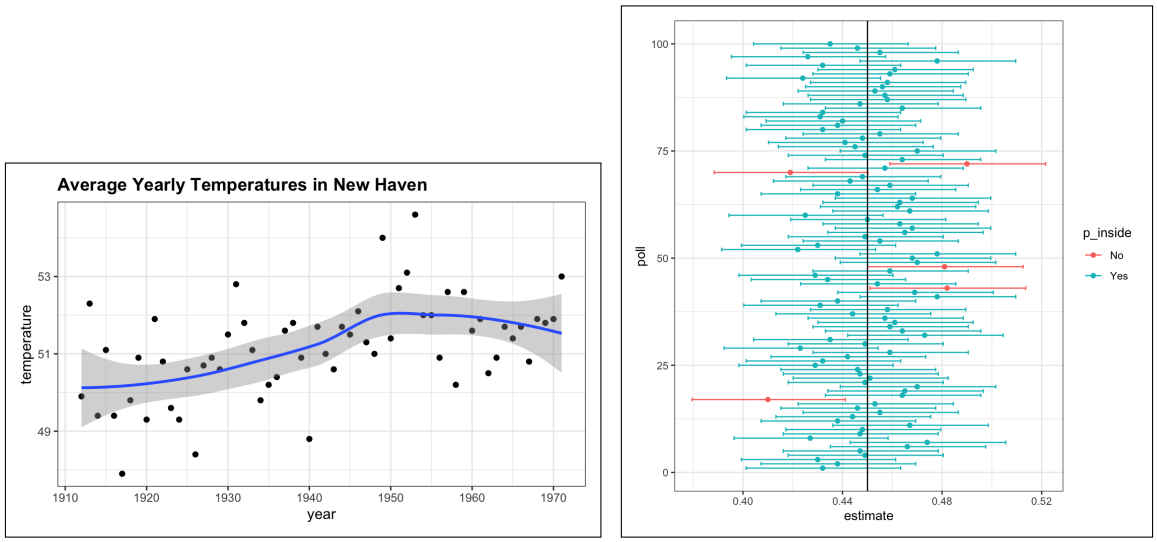
```
# Criar vetor com 2400 bolinhas, com a proporção especificada de vermelhas e azuis
populacao = np.random.choice(["Vermelha", "Azul"], size=2400, replace=True,
p=[prop_vermelhas, prop_azuis])

# Geração de 1 amostra original de tamanho 50
amostra_original = [np.random.choice(populacao, size=50,
replace=False)]

# Geração de 33 amostras de tamanho 50 a partir da amostra original
amostras = [np.random.choice(np.array(amostra_original).ravel(), size=50,
replace=True) for _ in range(33)]

# Calcular as médias amostrais
medias_amostrais = [np.mean(amostra == "Vermelha") for amostra in amostras]

# Criar um DataFrame com as médias amostrais
df = pd.DataFrame({"mean_samples": medias_amostrais})

# Plotando a distribuição das méidas amostrais
(ggplot(df, aes(x="mean_samples")) +
 geom_histogram(aes(y="..density.."), binwidth=0.01, fill="blue", color="black") +
 geom_density(color="orange") +
 labs(x="Média Amostral", y="Densidade", title="Distribuição das Médias Amostrais") +
 theme_minimal())
```

9.3 Estimativa Intervalar

Ao realizar análises estatísticas e inferências sobre uma população com base em uma amostra, é essencial compreender a incerteza associada às estimativas derivadas dessas amostras. Na Seção 9.1, discutimos como, ao selecionar uma amostra da população, calculamos uma estimativa pontual para tentar estimar o valor de um parâmetro populacional desconhecido. No entanto, é importante reconhecer que essa estimativa pontual representa apenas uma observação entre um conjunto muito grande de estimativas que poderiam ser obtidas de diferentes amostras da mesma população. Na Seção 9.2, exploramos a noção de distribuição amostral, que é uma distribuição que engloba as possíveis estimativas pontuais que poderiam ser calculadas a partir de diferentes amostras de uma população. Essa distribuição amostral possui propriedades fundamentais, como a média (amostral), que corresponde

ao parâmetro populacional que estamos tentando estimar (Figura 9.1), e o desvio padrão, que reflete a variabilidade das estimativas pontuais em torno da verdadeira média populacional.

A partir das distribuições amostrais, podemos derivar "pares" de média e desvio padrão amostral, ou percentis, para cada uma das N amostra. Esses pares fornecem informações sobre a variabilidade de um estimador e nos permitem quantificar a incerteza em torno de uma estimativa pontual. Isso é ilustrado na Figura 9.8, onde podemos visualizar a dispersão das médias amostrais em torno da verdadeira média populacional, bem como a dispersão dos desvios padrão, ou percentis, amostrais. Esses gráficos ajudam a entender a precisão da estimativa e a avaliar a confiabilidade da inferência estatística feita a partir dos dados amostrais por um estimador.



(a) Gráfico de faixa (b) Gráfico de intervalos de confiança

Figura 9.8: Gráficos de distribuição das variações de um estimador ao calcular uma estatística de interesse. (Fonte: [70])

Uma **estimativa intervalar** é uma estimativa baseada em N amostras que fornece uma faixa de valores plausíveis onde o parâmetro populacional provavelmente estará contido com uma certa incerteza como mostrado na Figura 9.8. Essa faixa é denominada **intervalo de confiança** (CI, do inglês *Confidence Interval*), e a incerteza associada a ela é chamada **nível de confiança** ou **coeficiente de confiança**, representado por α. O nível de confiança é expresso como uma porcentagem, geralmente denotada como 95%, 90% ou outra porcentagem. Por exemplo, um intervalo de confiança de 95% para a média populacional indica que, se repetíssemos o processo de amostragem muitas vezes, aproximadamente 95% dos intervalos de confiança resultantes conteriam o verdadeiro valor da média populacional da estatística de interesse. A diferença entre os extremos de um intervalo de confiança é denominada **amplitude**.

O procedimento para construir um intervalo de confiança segue um conjunto semelhante de etapas que visam estabelecer uma faixa de variação para as estimativas pontuais em qualquer distribuição amostral:

Coleta de dados: Seleção de amostras representativas da população de interesse. As amostras devem ser selecionadas de forma aleatória e representar adequadamente a população em estudo.

Escolha do parâmetro: Determinação do parâmetro populacional a ser estimado, que pode ser a média, a proporção, a variância, a mediana ou qualquer outra estatística de interesse.

Escolha do estimador: Seleção do método de estimação mais adequado para o parâmetro em análise. Isso pode incluir a utilização de estimadores pontuais específicos, como a média amostral, proporção amostral, mediana amostral, entre outros.

Cálculo do estimador pontual: Cálculo do valor do estimador pontual utilizando os dados da amostra.

Determinação da variabilidade do estimador: Estimativa da variabilidade do estimador para levar em conta a incerteza associada à estimativa. Isso pode ser feito usando fórmulas específicas, técnicas de *bootstrap* ou outras abordagens estatísticas, dependendo das características dos dados e das suposições feitas sobre a distribuição subjacente.

Escolha do nível de confiança: Decisão por quão confiantes queremos estar de que nosso intervalo de confiança contenha o verdadeiro valor do parâmetro populacional. Geralmente, expressamos isso como uma porcentagem, como 95% ou 99%.

Cálculo do intervalo de confiança: Uso do estimador pontual e sua variabilidade estimada para cálculo dos limites do intervalo de confiança. Isso geralmente envolve adicionar e subtrair uma margem de erro ao redor do estimador pontual.

Interpretação do intervalo de confiança: O intervalo de confiança é uma faixa de valores que provavelmente inclui o verdadeiro parâmetro populacional. Ao analisar a relação entre o nível de confiança e a largura, ou a **amplitude**, do intervalo de confiança, é importante considerar não apenas o nível de confiança, mas também outros fatores, como o tamanho da amostra e a variabilidade dos dados.

9.3.1 Distribuição Amostral Subjacente

É importante ter uma noção da distribuição amostral subjacente ao estimar um intervalo de confiança. Esta distribuição influencia diretamente a forma como calculamos o intervalo de confiança e como interpretamos seus resultados. Por exemplo, se a distribuição segue uma distribuição normal, podemos usar os métodos baseados na distribuição normal, como a distribuição Z e t, para calcular o intervalo de confiança. Caso contrário, pode ser necessário recorrer a métodos não paramétricos[3] ou técnicas

[3] **Técnicas não paramétricas** não fazem suposições rígidas sobre a distribuição dos dados. Em vez disso, elas usam diretamente os dados observados para inferir propriedades da população.

de reamostragem, como a distribuição *bootstrap*, para estimar o intervalo de confiança diretamente a partir dos dados observados.

A técnica de *bootstrap* nos permite criar múltiplas amostras *bootstrap* a partir de uma única amostra original através de reamostragem com reposição (Seção 9.2.4. Ao criar essas amostras, podemos calcular a estatística de interesse para cada uma delas. A distribuição das estatísticas *bootstrap* fornece uma estimativa da distribuição amostral da estatística de interesse, mesmo que a distribuição original dos dados não seja conhecida ou não seja normal. Com base nessa distribuição amostral estimada, podemos calcular intervalos de confiança de maneira robusta, sem depender de suposições sobre a distribuição dos dados.

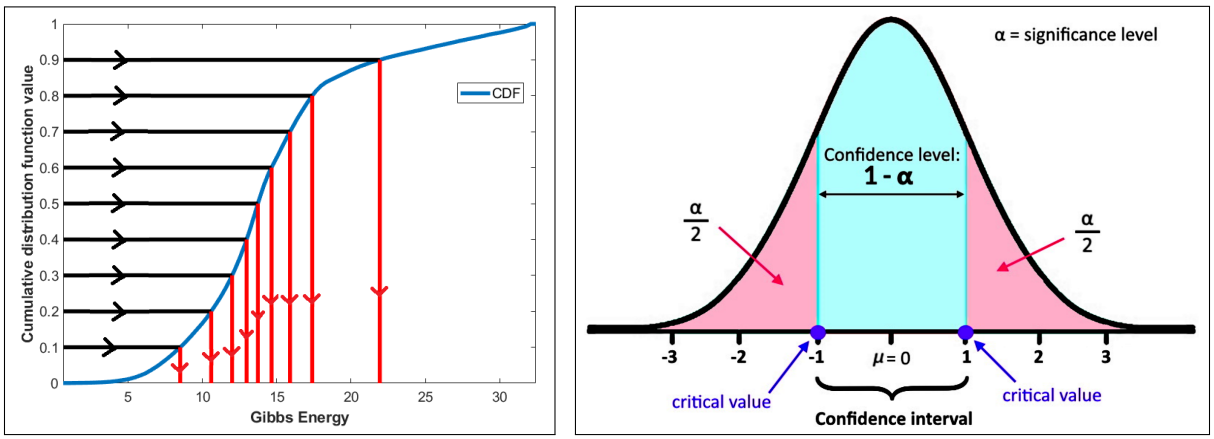
9.3.2 Cálculo de Intervalos de Confiança

Para determinar os limites do intervalo de confiança correspondente a um nível de confiança de $(1 - \alpha) \times 100\%$, onde α representa o nível de significância, que corresponde à probabilidade de rejeitar a hipótese nula quando ela é verdadeira, dois métodos comuns são frequentemente empregados[36]:

Método do percentil: Os extremos do intervalo de confiança são derivados dos percentis $\frac{\alpha}{2} \times 100$ e $(1 - \frac{\alpha}{2}) \times 100$ da distribuição amostral da estatística de interesse. Esses percentis representam os pontos na distribuição em que a probabilidade acumulada é igual a $\frac{\alpha}{2} \times 100$ e $(1 - \frac{\alpha}{2}) \times 100$, respectivamente. Eles são determinados a partir da função de distribuição cumulativa, conforme detalhado na Seção 6.1.1. Embora simples e intuitivo, este método pode exigir esforço computacional considerável em conjuntos de dados volumosos.

Método do erro padrão: Neste método, os extremos do intervalo de confiança são determinados utilizando o erro padrão do estimador da estatística de interesse e o valor associado a $\frac{1-\alpha}{2} \times 100$, geralmente obtido a partir de tabelas de probabilidades cumulativas padronizadas, dependendo da distribuição da estatística de interesse e do tamanho da amostra. O erro padrão representa a variabilidade esperada da estatística de interesse na amostra e é calculado com base em propriedades teóricas das distribuições estatísticas. Este método é frequentemente usado em situações onde os conjuntos de dados são extensos ou a distribuição da estatística de interesse é conhecida, sendo uma alternativa ao método do percentil quando a distribuição amostral não é facilmente determinada.

Por exemplo, os intervalos de confiança para uma média populacional μ podem ser definidos como intervalos simétricos em torno da média amostral $\overline{X}$, assumindo uma distribuição normal aproximada. A largura desses intervalos é determinada pelo erro padrão da média $SE(\overline{X})$, multiplicado pelo valor $z_{\frac{1-\alpha}{2}}$ da distribuição normal padrão $N(Z; 0, 1)$ associado ao nível de confiança desejado $(1 - \alpha) \times 100$,



(a) Método do Percentil (b) Método do Erro Padrão

Figura 9.9: Cálculo dos extremos de um intervalo de confiança: (a) Método do percentil (Fonte: Cloudfront) e (b) Método do erro padrão. (Fonte: Researchgate)

onde $P(\mid Z \mid < z_{\frac{1-\alpha}{2}}) = 1 - \alpha$

$$CI(\mu, 1-\alpha) = \overline{X} \pm (z_{\frac{1-\alpha}{2}} \times SE(\overline{X})) = \overline{X} \pm (z_{\frac{1-\alpha}{2}} \times \frac{\sigma}{\sqrt{n}}). \tag{9.12}$$

Na Figura 9.10, podemos observar que, no método do erro padrão, diferentes níveis de confiança são representados por intervalos específicos. Por exemplo, p=68.26% é refletido no intervalo de confiança $[-SE(\overline{X}), +SE(\overline{X})]$, enquanto p=95.4% é representado pelo intervalo $[-2\times SE(\overline{X}), +2\times SE(\overline{X})]$. Esses intervalos são relativos à distribuição normal $N(X; \mu, \alpha)$. Na distribuição normal padrão $N(Z; 0, 1)$, os valores associados são $z_{0.341} = 1.0$ e $z_{0.477} = 2.0$ ao consultar a tabela z de probabilidades acumuladas [8]. Esses valores correspondem precisamente aos multiplicadores dos desvios-padrão da distribuição amostral, ou erros padrão, como apresentado na Eq. 9.12.

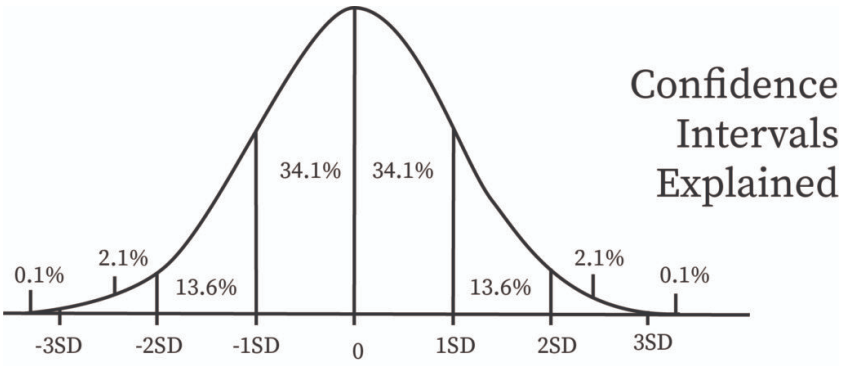


Figura 9.10: Intervalos de confiança para níveis de confiança $p = 68,26\%$ e $p = 95.4\%$: $[-SE(\overline{X}), +SE(\overline{X})]$ e $[-2\times SE(\overline{X}), +2\times SE(\overline{X})]$, respectivamente. (Fonte: `https://www.bartleby.com/subject/math/statistics/concepts/confidence-intervals`)

Quando a variância é desconhecida, podemos construir intervalos de confiança para μ usando o modelo t-Student. Através da tabela t-Student de probabilidades acumuladas com $(n-1)$ graus de

liberdade [90], podemos obter o valor $t_{\frac{1-\alpha}{2}}$ tal que

$$(1-\alpha) = P(-t_{\frac{1-\alpha}{2}} < \frac{\overline{X}-\mu}{\frac{s}{\sqrt{n}}} < t_{\frac{1-\alpha}{2}}).$$

Logo, o intervalo de confiança $CI(\mu, 1-\alpha)$, com variância desconhecida, pode ser dado por

$$CI(\mu, 1-\alpha) = \overline{X} \pm (t_{\frac{1-\alpha}{2}} \times SE(\overline{X})) = \overline{X} \pm (t_{\frac{1-\alpha}{2}} \times \frac{s}{\sqrt{n}}). \tag{9.13}$$

Em termos práticos, os intervalos de confiança são muito utilizados pelos pesquisadores e tomadores de decisão, pois permitem avaliar a robustez das conclusões estatísticas e tomar decisões informadas com base nos resultados analíticos. Por exemplo, é possível calcular intervalos de confiança para uma proporção populacional usando o estimador $\hat{p}$. Apesar de a proporção amostral $\hat{p}$ seguir uma distribuição binomial, o Teorema Central do Limite nos assegura que, **para um tamanho n de amostra suficientemente grande, podemos aproximar** a distribuição das proporções amostrais $\hat{p}$ por **uma distribuição normal**, com a mesma média populacional p e variância $\frac{p(1-p)}{n}$ (Seção 8.5.2).

Quando lidamos com uma única amostra de tamanho n, é comum aproximarmos a proporção p da população pela estimativa pontual do estimador $\hat{p}$, seguindo uma **abordagem otimista** [52], como:

$$\begin{aligned} \hat{p} &= \frac{\sum_{i=1}^{n} X_i}{n}, \text{ onde } X_i \in \{0,1\} \\ Var(\hat{p}) &= \frac{\hat{p}(1-\hat{p})}{n}. \end{aligned} \tag{9.14}$$

Essa estratégia nos permite estabelecer um intervalo de confiança onde é provável que o verdadeiro valor do parâmetro p esteja contido, com um determinado nível de confiança $1-\alpha$. É uma abordagem "otimista" por fornecer uma precisão aparente maior baseada na amostra específica. A informação da amostra ($\hat{p}$) para estimar a variância da distribuição amostral é aplicada. Se a amostra por acaso tiver uma proporção extrema (muito perto de 0 ou 1), o valor de $\hat{p}(1-\hat{p})$ será menor, resultando em um intervalo de confiança mais estreito.

Uma segunda **abordagem conservativa** é considerar que $p(1-p) \leq \frac{1}{4}$, ou seja,

$$\begin{aligned} \hat{p} &= \frac{\sum_{i=1}^{n} X_i}{n}, \text{ onde } X_i \in \{0,1\} \\ \text{Var}(\hat{p}) &= \frac{\hat{p}(1-\hat{p})}{n} \leq \frac{\frac{1}{4}}{n}. \end{aligned} \tag{9.15}$$

Ao aproximarmos a distribuição binormal pela distribuição normal, podemos obter o valor de $z_{\frac{1-\alpha}{2}}$ na tabela z de probabilidades acumuladas [8], por meio do nível de confiança $1-\alpha$. Substituindo $z_{\frac{1-\alpha}{2}}$ na Equação 9.12, juntamente com os valores de $\mu_{\hat{p}}$ e $\sqrt{\text{Var}(\hat{p})}$, obtemos um intervalo de confiança com uma probabilidade de α de conter a verdadeira proporção populacional. É uma abordagem "conservativa"

porque superestima a variabilidade, levando a um intervalo mais seguro (maior probabilidade de conter p). Estamos usando o valor máximo possível para a variância da proporção ($p(1-p)$), que é $\frac{1}{4}$, para calcular o intervalo de confiança. Isso resulta em um intervalo de confiança mais amplo do que a abordagem "otimista", garantindo uma probabilidade de cobertura do verdadeiro parâmetro p pelo menos igual ao nível de confiança desejado, independentemente do valor real de p.

Para evitar a consulta à tabela z de probabilidades acumuladas [8] na obtenção dos valores $z_{\frac{p}{2}}$, uma terceira abordagem consiste em utilizar funções disponíveis em R e Python que automatizam esses cálculos. Por exemplo, usando uma abordagem otimista, podemos estimar o intervalo de confiança para a probabilidade de "cara" em uma série de lançamentos de moeda com um nível de confiança de 95% em conter a verdadeira proporção populacional. Isso é calculado com base em uma única amostra sample de tamanho n do exemplo apresentado na Seção 8.6.1. Segue-se uma implementação do procedimento em R e Python:

R: É necessário carregar a biblioteca stats para ter acesso às funções estatísticas avançadas, como qnorm.

```
# Carregar a biblioteca stats
library (stats)

# Definição do tamanho da amostra
n <- 50         # Número de lançamentos em cada observação

# Gerar uma observação de n lançamentos da moeda (Cara = 1, Coroa = 0)
sample <- rbinom(n, 1, 0.5)

# Nível de confiança
confianca <- 0.95

# Calculando a média
media_amostral <- mean(sample)
desvio_padrao_amostral <- sd(sample)

# Calcular o intervalo de confiança usando a distribuição normal (Z)
margem_de_erro <- qnorm(1 - (1 - confianca) / 2) *
    desvio_padrao_amostral / sqrt(length(sample))

intervalo_confianca <- c(media_amostral - margem_de_erro,
```

```
    media_amostral + margem_de_erro)

print("Média amostral:", media_amostral)
print("Desvio padrão amostral:", desvio_padrao_amostral)
print("Intervalo de confiança:", intervalo_confianca)
```

Python: Recorremos às funções estatísticas do módulo scipy.stats do pacote scipy.

```
# Carregar a biblioteca scipy.stats
import numpy as np
from scipy import stats

# Definição do tamanho de cada amostra
n=50          # Número de lançamentos em cada observação

# Gerar uma observação de n lançamentos da moeda (Cara = 1, Coroa = 0)
sample = np.random.binomial(1,0.5,size=n)

# Nível de confiança
confianca = 0.95

# Calculando a média e o desvio padrao amostral
media_amostral = np.mean(sample)
desvio_padrao_amostral = np.std(sample, ddof=1)
    # Use ddof=1 para calcular o desvio padrão amostral

# Calcular o intervalo de confiança usando a distribuição normal (Z)
intervalo_confianca = stats.norm.interval(confianca, loc=media_amostral,
    scale=desvio_padrao_amostral/np.sqrt(len(sample)))

print("Média amostral:", media_amostral)
print("Desvio padrão amostral:", desvio_padrao_amostral)
print("Intervalo de confiança:", intervalo_confianca)
```

Os resultados numéricos destacam que, com um nível de confiança de 95%, o valor real da proporção de "caras", idealmente 0,5 para uma moeda justa, está contido nos intervalos de confiança de 95% correspondentes, quando os tamanhos da amostra são apropriadamente selecionados. Além disso, observa-se que o intervalo diminui de amplitude à medida que o tamanho da amostra n aumenta:

n=10: Intervalo de Confiança = $(-0.061328531272007214, 0.46132853127200724)$.

n=50: Intervalo de Confiança = $(0.380114615309879, 0.659885384690121)$.

n=200: Intervalo de Confiança = $(0.43053091844331537, 0.5694690815566846)$.

n=500: Intervalo de Confiança = $(0.4601313848582645, 0.5478686151417356)$.

De forma análoga, pode-se computar o percentil associado a uma determinada probabilidade acumulada para a distribuição t-Student usando a função `qt()` da biblioteca `stats` em R. E em Python, podemos usar a função `ppf()` do pacote `scipy.stats`.

9.3.3 Interpretação de Intervalos de Confiança

Para interpretar corretamente os intervalos de confiança, deve-se atentar para diversos aspectos, sempre contextualizando sua interpretação com a natureza específica do problema em questão. Entender como a precisão e a incerteza influenciam as decisões práticas é essencial nesse processo [36].

É importante ressaltar que um intervalo de confiança não oferece uma estimativa precisa do parâmetro populacional, mas sim uma faixa de valores que provavelmente contém o verdadeiro parâmetro com uma determinada probabilidade de confiança. Evite o equívoco comum de interpretar o intervalo de confiança como a probabilidade de que o parâmetro esteja dentro de um intervalo específico. Em vez disso, reconheça que o parâmetro é fixo e o intervalo de confiança é uma medida da incerteza associada à estimativa do parâmetro, fornecendo uma indicação da variabilidade que se espera encontrar em torno da estimativa pontual.

A escolha do nível de confiança é crucial, pois determina a probabilidade de que o intervalo de confiança contenha o verdadeiro parâmetro. Embora um nível de confiança comum seja 95%, diferentes escolhas são possíveis, cada uma com suas implicações. Além disso, o tamanho da amostra desempenha um papel fundamental na precisão do intervalo de confiança. À medida que o tamanho da amostra representativa aumenta, o intervalo de confiança estimado tende a se tornar mais estreito, resultando em uma estimativa mais precisa da estatística populacional de interesse dentro desse intervalo.

É também essencial garantir que a distribuição da amostra seja apropriada para a construção do intervalo de confiança, levando em consideração casos de assimetria ou heterogeneidade nos dados.

9.4 Considerações Finais

Enquanto nos capítulos anteriores tínhamos acesso a todos os dados de uma população e realizávamos estatísticas resumidas sobre suas características, abordamos neste capítulo técnicas que nos permitem estimar parâmetros populacionais com base em amostras, que são subconjuntos consideravelmente menores da população.

Na Seção 9.1, discutimos a estimativa pontual, que busca prever um único valor representativo de um parâmetro específico da população, como a média populacional derivada da média da amostra. Embora existam várias técnicas para inferência estatística que não dependem das distribuições subjacentes da amostra, as abordagens clássicas, como testes estatísticos z e t, pressupõem uma distribuição normal subjacente da amostra.

Na Seção 9.2, abordamos noções sobre distribuições amostrais e sua natureza aleatória, incluindo as distribuições teóricas amplamente usadas como referências. É uma prática comum e válida avaliar a normalidade de uma distribuição antes de aplicar as inferências estatísticas. Apesar de existirem testes eficazes para realizar essa avaliação, como o teste de Shapiro-Wilk ou o teste de Kolmogorov-Smirnov, apresentamos uma técnica gráfica alternativa eficaz, o gráfico quantil-quantil, para avaliar visualmente a normalidade dos dados.

Na Seção 9.3, exploramos os intervalos de confiança, os quais delimitam uma faixa de valores plausíveis para o parâmetro populacional, refletindo a incerteza em torno da estimativa pontual. Apresentamos duas abordagens para calcular esses intervalos: uma paramétrica (teste Z e T) adequada para amostras que seguem uma distribuição normal, e uma não paramétrica (*bootstrap*), útil quando as amostras não atendem aos critérios de normalidade.

Além disso, introduzimos a técnica de *bootstrap*, que se tornou uma alternativa robusta e não tendenciosa para fazer inferências estatísticas, como intervalos de confiança (*bootstrap* de percentil). Com o avanço da capacidade computacional, o *bootstrap* é especialmente útil quando os dados não seguem uma distribuição normal ou quando o tamanho da amostra é pequeno. No entanto, é importante reconhecer que o *bootstrap* possui suas próprias limitações e considerações. Por exemplo, é necessário escolher um número apropriado de reamostragens e definir claramente a estatística de interesse para obter resultados confiáveis.

Ao longo do capítulo, exploramos os recursos gráficos que proporcionam uma compreensão mais profunda do comportamento dos dados subjacentes aos resultados de uma estatística de inferência. A visualização dos dados desempenha um papel importante na compreensão das características subjacentes e na identificação das distribuições dos dados, auxiliando na validação de pressupostos estatísticos e na escolha do método mais apropriado para realizar inferências sobre a população [36].

9.5 Exercícios

1. Faça os exercícios de aprendizado LC7.8 – LC7.24 relacionados à **amostragem**, conforme proposto em [36]. Verifique suas respostas no apêndice D do mesmo livro.

2. Faça os exercícios de aprendizado LC8.1 – LC8.5 relacionados aos **intervalos de confiança**, conforme proposto em [36]. Verifique suas respostas no apêndice D do mesmo livro.

3. Identifique as perguntas, os parâmetros e as amostras nos exemplos apresentados no Capítulo 8 em [36].

4. Segue a manchete da revista Exame: "Nunes tem 26,9%, Marçal, 26,8%, e Boulos, 20,1% em SP, diz pesquisa Futura." Adicionalmente, a matéria informa "De acordo com a margem de erro, que é de 3,1 pontos percentuais para mais ou para menos, Nunes e Marçal estão empatados tecnicamente na liderança. Boulos aparece em terceiro." Com base nas informações apresentadas, responda às seguintes questões:

   (a) Como você interpreta as proporções e as margens de erro em relação à população de eleitores?

   (b) Que tipo de estimativa está sendo feita: pontual ou intervalar? Justifique sua resposta.

   (c) É possível determinar o nível de confiança das intenções de voto? Justifique sua posição.

5. O conjunto de dados *airquality* está disponível no reepositório R e contém dados sobre a qualidade do ar em Nova York durante os meses de maio a setembro de 1973. Este conjunto de dados inclui informações sobre a temperatura, radiações solares, velocidade do vento e concentração de poluentes no ar, o ozônio (Ozone), que pode gerar várias perguntas e análises interessantes, como a análise de mudanças no clima, qualidade do ar e impactos na saúde. Resolva as seguintes questões:

   (a) Utilize as funções apropriadas de R/Python para determinar analiticamente, para cada um dos quatro parâmetros, a média amostral e o intervalo de confiança com um nível de confiança de 95%.

   (b) Com uso de ggplot, plote a distribuição dos valores de cada parâmetro para avaliar o grau de normalidade dos dados.

   (c) Determine a estimativa pontual da média de cada parâmetro na amostra do período estudado.

   (d) Aplique a técnica *bootstrap* para construir, para cada parâmetro, a distribuição das médias amostrais de 100 amostras contendo 50 observações cada. Determine a média das médias amostrais de cada parâmetro

   (e) Com uso de ggplot, plote os intervalos de probabilidade que compreendem 2.5% a 97.5% dos dados para 100 amostras *bootstrap* de cada parâmetro. Qual é a quantidade desses intervalos que contêm a média das médias amostrais calculada no item 5d?

   (f) O que a média das médias amostrais nos diz sobre a temperatura, radiações solares, velocidade do vento e concentração de ozônio durante o período estudado?

(g) Para cada parâmetro, compare a largura e a interpretação dos intervalos de probabilidade construídos no item 5e com a largura e a interpretação do intervalo de confiança calculado no item 5a. O que essa comparação revela?

Capítulo 10

Estatística de Inferência: Testes de Hipóteses

No Capítulo 9, exploramos os conceitos de estimativa pontual e intervalar, que são fundamentais para a inferência estatística. Ambas as metodologias nos permitem fazer afirmações sobre características populacionais a partir de amostras, mas diferem significativamente em como abordam a incerteza e a interpretação dos resultados. A **estimativa pontual** se refere a um único valor calculado a partir dos dados de uma amostra, que serve como a melhor suposição para o parâmetro populacional de interesse. Embora essa estimativa seja útil, ela não fornece informação sobre a precisão ou a confiabilidade dessa suposição. Assim, a estimativa pontual carece de um contexto que permita entender o grau de incerteza associado a ela. Por outro lado, a **estimativa intervalar** inclui explicitamente o grau de incerteza na estimativa feita. Um intervalo de confiança, por exemplo, não apenas fornece um intervalo em que o parâmetro populacional é provável de se encontrar, mas também quantifica a incerteza em torno do intervalo conter o parâmetro populacional.

Entretanto, tanto as estimativas pontuais quanto os intervalos de confiança apresentam limitações em suas inferências. As conclusões derivadas dessas estimativas não permitem avaliar a plausibilidade de uma suposição em uma população, com base nas evidências fornecidas pelos dados amostrais. Neste capítulo, avançaremos ao introduzir os **testes de hipóteses**, uma abordagem estatística que nos permite **verificar suposições** sobre uma população utilizando dados amostrais.

O processo de teste de hipóteses começa com a formulação de duas proposições complementares, como será detalhado na Seção 10.1: a hipótese nula (H_0) e a hipótese alternativa (H_1 ou H_a). Essas hipóteses representam, respectivamente, a suposição a ser testada e a possibilidade de que exista um efeito ou diferença significativa. Para decidir entre essas duas hipóteses, utilizamos uma estatística amostral, ou seja, uma medida calculada a partir dos dados observados em uma amostra. Na Seção **??**, mostramos como essas estatísticas podem ser transformadas em estatísticas de teste, cujas distribuições nos permitem avaliar, de forma padronizada, a plausibilidade da hipótese nula. A escolha do teste

estatístico apropriado depende do tipo de dados e do contexto do estudo. Testes como o teste z (normal) e o teste t de Student são abordados ao longo do texto, cada um com suas respectivas aplicações.

Na Seção 10.3, introduzimos o conceito de nível de significância (α), que representa a probabilidade máxima tolerada de rejeitar a hipótese nula quando ela é verdadeira. Essa definição está diretamente ligada à distribuição da estatística de teste sob H_0, e estabelece o critério para rejeição ou não da hipótese nula com base em valores extremos observados. Além disso, a Seção 10.4 explora como a definição da hipótese alternativa influencia o tipo de teste a ser conduzido (unicaudal ou bicaudal), o que também afeta a interpretação dos resultados. Ao realizar testes de hipóteses, deve-se sempre considerar os erros estatísticos, conforme discutido na Seção 10.5. O erro Tipo I (de probabilidade α) ocorre quando rejeitamos a hipótese nula, embora ela seja verdadeira, ou seja, cometemos um falso positivo ao **afirmar** a existência de uma diferença em relação ao cenário inicial, quando na verdade essa diferença não existe. Por sua vez, o erro Tipo II (de probabilidade β) acontece quando deixamos de rejeitar a hipótese nula, mesmo ela sendo falsa, resultando em um falso negativo, isto é, na **negação** da existência de uma diferença em relação ao cenário inicial, embora tal diferença de fato exista.

Na Seção 10.6, apresentamos o conceito de p-valor, que corresponde à probabilidade de obter um resultado tão extremo quanto o observado, assumindo que H_0 seja verdadeira. Esse valor fornece uma medida da força da evidência contra a hipótese nula, sendo amplamente utilizado para apoiar decisões estatísticas. Os resultados dos testes são sempre interpretados à luz das distribuições amostrais e devem ser contextualizados de forma cuidadosa. A Seção 10.7 destaca a importância de uma interpretação criteriosa para evitar inferências incorretas ou conclusões precipitadas. Por fim, a Seção 10.8 sintetiza as abordagens apresentadas, diferenciando os procedimentos paramétricos (que assumem distribuição conhecida para os dados) dos não paramétricos (que fazem menos suposições), orientando a escolha do teste mais adequado a cada situação.

10.1 Hipóteses

Em um teste de hipóteses, formulamos duas proposições opostas: a hipótese nula (H_0) e a hipótese alternativa (H_1 ou H_a). A **hipótese nula** representa a suposição inicial de que não há efeito, diferença ou mudança significativa em relação ao que se conhece ou se espera como comportamento padrão de uma população. Ela funciona como um ponto de partida neutro, que assumimos ser verdadeiro até que as evidências indiquem o contrário. Já a **hipótese alternativa** expressa a possibilidade de que existe uma diferença real ou um efeito significativo, muitas vezes causado por algum fator específico ou intervenção. Nesse caso, assumimos que os dados observados vêm de uma população diferente daquela descrita por H_0, com parâmetros distintos.

Vamos transpor esse conceito para um cenário prático. Suponha que o tempo médio de reação

de motoristas ao volante seja conhecido como 1,2 segundos. Um novo suplemento alimentar (uma intervenção) é testado com a hipótese de que ele reduz esse tempo de reação. Temos, então, duas proposições opostas:

H_0**:** O suplemento não altera o tempo de reação, pois a média continua 1,2 s.

H_1**:** O suplemento reduz o tempo de reação, pois a média é menor que 1,2 s.

É fundamental compreender a distribuição assumida sob a hipótese nula, pois ela serve de base para avaliarmos a evidência fornecida pelos dados. Essa distribuição pode ser caracterizada de duas formas complementares: como uma distribuição populacional e como uma distribuição amostral. A **distribuição populacional** descreve o comportamento da variável de interesse em toda a população, assumindo que H_0 é verdadeira. Ela pode assumir diferentes formas, como normal, binomial ou outras, dependendo da natureza da variável e do contexto do teste, como mostrada na Seção 6.1.1. Já a **distribuição amostral** se refere à distribuição de uma estatística (como média ou proporção) calculada a partir de amostras extraídas dessa população, também sob a suposição de que H_0 é verdadeira, como discutido na Seção 9.2. É essa distribuição amostral que utilizamos para calcular probabilidades e definir regiões críticas nos testes de hipóteses.

Em análises que envolvem médias ou proporções amostrais, a distribuição dessas estatísticas tende a se aproximar de uma distribuição normal à medida que o tamanho da amostra aumenta, que é um resultado garantido pelo Teorema Central do Limite. Isso ocorre independentemente da forma da distribuição populacional original, desde que certas condições sejam atendidas, como tamanho amostral suficientemente grande e independência entre observações. Esse comportamento assintoticamente normal das estatísticas amostrais é essencial para a realização de testes de hipóteses, pois nos permite aplicar métodos baseados na normalidade, mesmo quando a população original não segue uma distribuição normal.

A Figura 10.1 ilustra esses conceitos ao mostrar duas distribuições amostrais distintas: uma representando o comportamento esperado sob H_0 e outra sob H_1, refletindo dois cenários populacionais distintos.

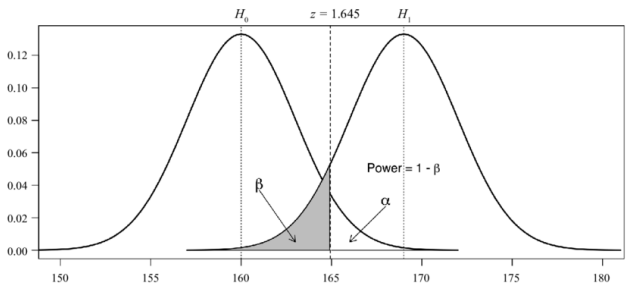


Figura 10.1: Distribuições amostrais para duas hipóteses de testes: nula (H_0) e alternativa (H_1). (Fonte)

10.2 Distribuições Amostrais sob Hipóteses

Para conduzir um teste de hipóteses, começamos coletando uma nova amostra e, a partir dela, calculamos uma estatística para teste, que é uma medida resumida dos dados que será usada para avaliar a validade da hipótese nula (H_0). Essa estatística é então comparada com a distribuição das estatísticas esperadas sob a suposição de que H_0 é verdadeira. A pergunta central que guia essa análise é:

> Essa observação é plausível dentro do mundo descrito por H_0?

Se o valor observado estiver muito distante do valor esperado sob H_0, como o valor $z = 1{,}645$ mostrado na Figura 10.1, consideramos improvável que ele tenha sido gerado por essa distribuição amostral. Nessa situação, rejeitamos H_0 e passamos a considerar a hipótese alternativa (H_1) como uma explicação mais plausível para os dados observados.

No exemplo do suplemento alimentar da Seção 10.1, podemos coletar uma amostra dos tempos de reação após o consumo do suplemento e calculamos a média amostral. Essa média é comparada com a distribuição amostral esperada sob a hipótese nula, que afirma que o suplemento não tem efeito. Se o valor observado cair em uma região de cauda da distribuição, ou seja, em uma faixa onde valores são muito raros se H_0 for verdadeira, rejeitamos H_0 e aceitamos a hipótese alternativa de que o suplemento reduz o tempo de reação.

Isso demonstra o quão cruciais as distribuições amostrais são na condução de testes de hipóteses, pois elas nos permitem determinar a probabilidade de obter nossos resultados por puro acaso. A questão fundamental que surge é: como se obtém uma distribuição amostral adequada para um teste de hipóteses?

10.2.1 Distribuição da Estatística de Teste

As distribuições amostrais variam em forma, dispersão e posição central, dependendo da média e desvio padrão da população, bem como do tamanho da amostra. Essas variações dificultam a comparação direta entre diferentes distribuições e tornam complexo o cálculo de probabilidades. Para resolver isso, é comum transformar distribuições amostrais em distribuições teóricas com propriedades conhecidas, como a distribuição normal padrão, que tem média 0 e variância 1. A adoção de distribuições teóricas facilita o cálculo de probabilidades e torna mais simples a comparação entre diferentes contextos e testes. Os valores obtidos após aplicar uma transformação apropriada sobre uma estatística amostral, muitas vezes uma padronização, são chamados de **estatísticas de teste**. Ao transformarmos todas as estatísticas amostrais de uma distribuição amostral, obtemos uma **distribuição da estatística de teste**.

A distribuição da estatística de teste é, de fato, uma transformação ou padronização da distribuição amostral, que é criada especificamente para avaliar a hipótese nula. A distribuição teórica dessa estatística de teste sob a hipótese nula H_0 é conhecida e serve como referência para a decisão estatística.

Por exemplo, ao compararmos o valor calculado da estatística de teste T com essa distribuição sob H_0, conseguimos avaliar a plausibilidade da hipótese nula sem depender diretamente da distribuição específica da estatística amostral original. Essa estratégia de comparação, baseada em distribuições conhecidas, constitui a base da inferência estatística: ela nos permite tirar conclusões sobre uma população com base nos dados obtidos de uma amostra.

Em testes que comparam a média ou proporção de uma única amostra grande (geralmente com $n \geq 30$) com um valor populacional conhecido, costuma-se usar o teste Z, pois a distribuição amostral da estatística tende a ser aproximadamente normal nesse cenário. No entanto, em situações mais complexas, como na comparação entre duas populações (ex.: diferença entre duas médias) ou na análise de frequências observadas versus esperadas (ex.: tabelas de contingência), o cálculo da estatística de teste se torna mais elaborado. Isso ocorre porque é necessário combinar variâncias, ajustar graus de liberdade e modelar expectativas com base em distribuições mais sofisticadas. Por essa razão, diferentes situações exigem distintas distribuições teóricas, apresentadas na Seção 9.2.2, e fórmulas específicas para o cálculo da estatística de teste, cada uma associada a uma distribuição apropriada:

Teste Z: Aplicado para comparar a média de uma amostra com a média de uma população quando o tamanho da amostra é grande (geralmente n $>$ 30) e a variância populacional é conhecida. Este teste se baseia na distribuição normal para determinar a probabilidade de observar uma estatística tão extrema quanto a calculada, sob a hipótese nula. A fórmula para calcular a **estatística de teste Z** é:

$$Z = \frac{\overline{X} - \mu_0}{\frac{\sigma}{\sqrt{n}}}$$

onde $\overline{X}$ é a média da amostra, μ é a média da distribuição amostral sob H_0, s é o desvio padrão amostral e n é o tamanho da amostra.

Teste t de Student: Utilizado para comparar as médias de duas populações, especialmente quando o tamanho da amostra é pequeno e a distribuição da população não é conhecida. O teste t calcula a diferença entre as médias amostrais, levando em consideração a variabilidade das amostras. A **estatística de teste t** é calculada usando a seguinte fórmula:

$$t = \frac{\overline{X}_1 - \overline{X}_2}{s_{\text{pooled}} \cdot \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

onde $\overline{X}_1$ e $\overline{X}_2$ representam as médias amostrais das duas populações que estão sendo comparadas. O desvio padrão combinado das duas amostras, s_{pooled}, é calculado da seguinte forma:

$$s_{\text{pooled}} = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}}$$

Neste caso, s_1 e s_2 são os desvios padrões das duas amostras, enquanto n_1 e n_2 são os tamanhos das amostras.

Além disso, o teste t de Student pode ser aplicado a uma única amostra, especialmente quando o tamanho da amostra é pequeno. Nessa situação, a fórmula se simplifica para

$$t = \frac{\overline{X} - \mu_0}{\frac{s}{\sqrt{n}}}$$

onde $\overline{X}$ é a média da amostra, μ_0 é a média da distribuição amostral sob a hipótese nula, s é o desvio padrão da amostra e n é o tamanho da amostra. Este método permite avaliar se a média amostral é significativamente diferente da média populacional.

Teste Qui-quadrado: O teste Qui-quadrado é um teste estatístico não paramétrico usado para analisar dados categóricos organizados numa tabela de contigência de m linhas e n colunas. Existem duas aplicações principais:

- Teste Qui-quadrado de Independência: Avalia se existe uma associação estatisticamente significativa entre duas variáveis categóricas. Em outras palavras, verifica se as duas variáveis são independentes ou se há uma relação entre elas. Este é o foco da questão a seguir.
- Teste Qui-quadrado de Bondade do Ajuste (em inglês, *Goodness-of-Fit Test*): Verifica se a distribuição observada de uma única variável categórica difere significativamente de uma distribuição esperada.
- Teste Qui-quadrado para uma Única Variância Populacional (ou Teste Qui-quadrado de Variância Única): Determinar se a variância de uma única população de **dados contínuos** é significativamente diferente de um valor populacional conhecido (hipotético).

O teste mais conhecido é o de independência. A estatística de teste é a **estatística qui-quadrado** (χ^2), que mede a diferença entre as frequências observadas e as frequências esperadas em cada célula da tabela de contigência, sob a hipótese nula de que não existe associação entre as variáveis. A fórmula para calcular a **estatística de teste qui-quadrado** é expressa da seguinte forma:

$$\chi^2 = \sum \frac{(O_{ij} - E_{ij})^2}{E_{ij}},$$

onde O_{ij} representa a frequência observada e E_{ij} representa a frequência esperada computada com a seguinte equação

$$E_{ij} = \frac{Totaldalinha \times Totaldacoluna}{Totalgeral}.$$

Este cálculo permite avaliar a magnitude da discrepância entre as observações e as expectativas. Para determinar a distribuição Qui-quadrado subjacente, é ainda necessário calcular o número de graus de liberdade

$$gl = (m-1) \times (n-1).$$

ANOVA (Análise de Variância): Utilizada para comparar as médias de três ou mais grupos simultaneamente. A ANOVA avalia se existem diferenças significativas entre as médias dos grupos, analisando a variação dentro e entre os grupos. A **estatística de teste da ANOVA**, F de Snedecor, é calculada pela razão entre a variância entre os grupos e a variância dentro dos grupos, expressa da seguinte forma:

$$F = \frac{\text{Variância entre os grupos}}{\text{Variância dentro dos grupos}}.$$

Essa estatística permite identificar se pelo menos uma média de grupo difere significativamente das demais. Um F grande sugere que a variância (explicada) entre os grupos é muito maior que a variância (inexplicada ou erro aleatório) dentro dos grupos.

A necessidade de diferentes distribuições teóricas, Z, t, Qui-quadrado e ANOVA, para lidar com diversos cenários de teste estatístico é fundamental para a validade da inferência estatística. Isso ocorre porque cada cenário envolve características distintas dos dados e das questões de pesquisa que influenciam a distribuição da estatística de teste sob a hipótese nula. Os cenários podem variar quanto a

tipo de dados e nível de medida : Distinguem-se essencialmente

variáveis categóricas (nominais/ordinais): Quando lidamos com contagens ou frequências em categorias (como nível educacional vs. tipo de mídia), as diferenças são avaliadas pela distribuição Qui-quadrado (χ^2). Esta distribuição é naturalmente adequada para testar a independência ou a bondade do ajuste de frequências observadas versus esperadas.

variáveis contínuas (intervalares/racionais): Ao comparar médias ou proporções de dados contínuos, precisamos de distribuições que modelam essas quantidades.

parâmetro de interesse : Destacam-se

médias: Quando a variância populacional é conhecida (ou a amostra é muito grande, permitindo usar a variância amostral como uma boa estimativa da populacional), a estatística padronizada segue uma distribuição Z (Normal Padrão). E quando

a variância populacional é desconhecida e precisa ser estimada pela amostra (o caso mais comum), a incerteza adicional é contabilizada pela distribuição t de Student. A distribuição t tem "caudas" mais pesadas que a Z, refletindo essa incerteza extra, e se aproxima da Z à medida que o tamanho da amostra aumenta. Para comparar múltiplas médias (três ou mais grupos), utilizamos a distribuição F. O Teste F na ANOVA compara a variância entre os grupos com a variância dentro dos grupos. Se a variância entre os grupos é significativamente maior, inferimos que as médias dos grupos são diferentes.

proporções: Para proporções, especialmente com grandes amostras, a distribuição Z é frequentemente usada devido à aproximação da distribuição binomial pela normal.

tamanho da amostra: Como mencionado, o tamanho da amostra influencia a escolha entre o Teste Z e o Teste t para médias, devido à confiabilidade da estimativa da variância populacional.

pressupostos sobre a população: Cada distribuição teórica tem pressupostos subjacentes sobre a distribuição da população. Por exemplo, muitos testes t e F pressupõem normalidade dos dados (ou que o tamanho da amostra é grande o suficiente para que o Teorema do Limite Central se aplique). O Qui-quadrado não exige normalidade, mas requer contagens mínimas esperadas.

10.2.2 Distribuições *Bootstrap*

Tradicionalmente, para compreender o comportamento de uma estatística (como a média, mediana, variância, entre outras), seria necessário coletar várias amostras independentes da população. No entanto, essa abordagem é frequentemente inviável na prática. Como discutido na Seção 9.2.4, o método do *bootstrap* contorna essa limitação ao gerar, por meio computacional, múltiplas amostras artificiais a partir de uma única amostra observada, simulando assim uma distribuição amostral.

A principal vantagem do *bootstrap* é que ele não exige suposições rígidas sobre a distribuição da população de origem, podendo ser aplicado mesmo quando se dispõe de apenas uma amostra. O procedimento clássico do *bootstrap* consiste em gerar diversas **reamostragens com reposição** da amostra original [36]. Para cada reamostragem, calcula-se a estatística de interesse, e o conjunto dessas estatísticas compõe a distribuição bootstrap, conforme explorado na Seção 9.2.4. Essa mesma distribuição *bootstrap* pode ser usada para testes de hipóteses, mas com uma modificação crucial: antes da reamostragem, a amostra original deve ser ajustada para refletir o cenário em que a hipótese nula (H_0) é assumida como verdadeira. Isso garante que a distribuição *bootstrap* gerada represente a variabilidade esperada da estatística se a H_0 fosse de fato verdadeira, permitindo um cálculo válido

do p-valor empírico.

Outra abordagem relacionada é a **técnica de permutação**, usada principalmente em testes de hipóteses para avaliar se há diferença estatisticamente significativa entre dois ou mais grupos. Essa técnica parte da premissa de que, sob a hipótese nula, as observações dos diferentes grupos são intercambiáveis. O processo começa com o cálculo da estatística de interesse nos dados originais (por exemplo, a diferença entre médias). Em seguida, os rótulos dos grupos são embaralhados aleatoriamente — sem reposição — e, a cada permutação, a estatística é recalculada. A repetição desse processo gera uma distribuição amostral empírica da estatística sob a hipótese nula, permitindo avaliar se a estatística observada é extrema em comparação ao que seria esperado por acaso.

Ao contrário do *bootstrap* clássico, a permutação não utiliza reposição: os dados são apenas rearranjados entre os grupos. Isso torna a técnica especialmente útil para testes não paramétricos, baseando-se diretamente na estrutura observada dos dados sem depender de distribuições teóricas.

10.3 Nível de Significância

A decisão de rejeitar a hipótese nula H_0 é baseada no **nível de significância** α, que representa a probabilidade máxima aceitável de cometer um erro de rejeitar H_0 quando ela é verdadeira. O **valor crítico** é um ponto de corte na distribuição da estatística considerada sob H_0. Ele pode ser definido:

- na escala da estatística de teste (como z, t, χ^2, F, etc.), ou
- diretamente na escala da estatística amostral (como $\bar{x}$, p, etc.), se sua distribuição for conhecida.

Esse valor delimita a(s) **região(ões) crítica(s)**, que são áreas extremas da distribuição cuja probabilidade acumulada é α, como ilustra a Figura 10.1. Tais regiões correspondem aos resultados considerados suficientemente improváveis sob H_0. Se a estatística calculada a partir da amostra cair nessas regiões, rejeitamos H_0. Em distribuições simétricas, como a normal padrão, os limiares de rejeição são simétricos, como $-z_c$ e z_c.

Antes da era digital, era comum consultar valores críticos em tabelas de probabilidades acumuladas após transformar as estatísticas em estatísticas de teste. A escolha da distribuição usada (normal, t, qui-quadrado ou F) dependia do teste e do contexto. Hoje, com ferramentas como R ou Python, podemos obter valores críticos de forma muito mais eficiente. Funções como `stats.norm.ppf()` (Python) ou `qnorm()` (R) retornam diretamente o valor crítico com base em α e na distribuição desejada. Essas ferramentas também automatizam etapas como a conversão em estatísticas de teste, que ainda é necessária do ponto de vista estatístico, mas agora é feita internamente.

Por exemplo, para um teste unilateral com $\alpha = 0{,}05$, obtermos através da tabela z de probabili-

dades acumuladas o valor crítico na distribuição normal padrão:

$$z_c = 1{,}645.$$

Se conhecermos o desvio padrão populacional σ, ou o desvio padrão amostral s (em amostras grandes), podemos, por meio de uma transformação inversa, encontrar o valor crítico correspondente da média amostral pela função que relaciona estatísticas amostrais com estatísticas de teste:

$$\bar{x}_c = z_c \times \frac{\sigma}{\sqrt{n}} + \mu_0 = 1{,}645.$$

Alternativamente, se soubermos a distribuição amostral sob H_0, podemos obter diretamente o valor crítico da estatística amostral sem passar pela estatística de teste. Basta encontrar o percentil $(1-\alpha)$ da distribuição de $\bar{x}$:

$$P(\bar{x} \leq \bar{x}_c) = 1 - \alpha.$$

Apesar dessa alternativa direta, a aplicação de estatística de teste não é descartável. Ela é uma ferramenta de generalização, que permite consultar tabelas de propriedades acumuladas (como antes da era digital), comparar testes diferentes com uma mesma referência de distribuição e reduzir os valores em uma forma comum mesmo com diferentes unidades e escalas. Hoje, seu papel continua central, mas sua execução é automatizada por *softwares*.

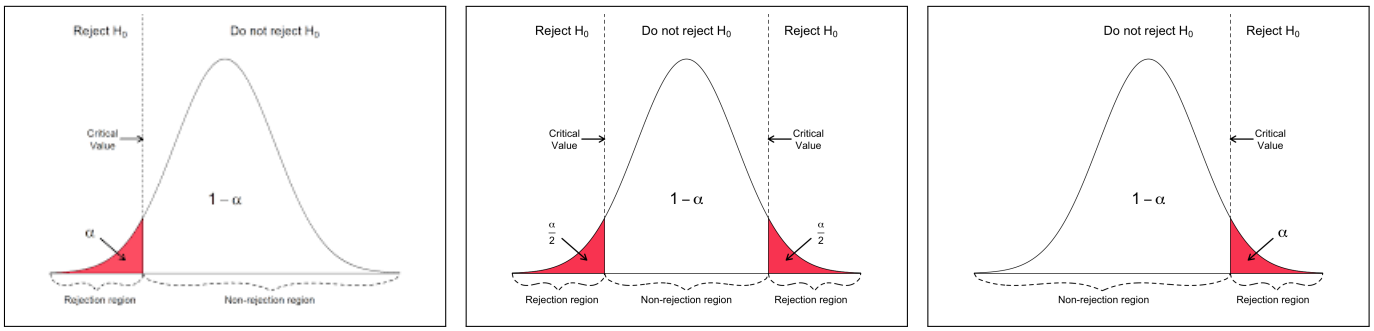
10.4 Tipos de Testes de Hipóteses

Ao introduzirmos os conceitos de estatística de teste, nível de significância e valor crítico, focamos inicialmente apenas na hipótese nula (H_0), pois a decisão de rejeitar ou não H_0 baseia-se na probabilidade de se observar os dados amostrais, ou resultados ainda mais extremos, sob a suposição de que H_0 é verdadeira. Porém, a formulação da hipótese alternativa (H_1 ou H_a), que representa aquilo que o pesquisador acredita ser verdadeiro caso H_0 não se sustente, influencia diretamente o tipo de teste a ser realizado. Na Figura 10.2, podemos observar as diferentes direções possíveis para H_1 em relação à hipótese nula, e, consequentemente, a localização da região crítica na distribuição da estatística de teste. Essas diferentes direções correspondem a três tipos de testes apresentados na Figura 10.2: unilateral à esquerda, bilateral e unilateral à direita.

Um **teste bilateral** é usado quando queremos verificar se o parâmetro de interesse é diferente do valor da hipótese nula θ_0, ou seja, pode ser maior ou menor do que esse valor. A hipótese alternativa é do tipo:

$$H_1 : \theta \neq \theta_0$$

Nesse caso, a região de rejeição de H_0 está dividida nas duas caudas da distribuição, cada uma com



(a) unilateral esquerda (b) bilateral (c) bilateral

Figura 10.2: Evidências contra a hipótese nula obtidas com base em valores críticos derivados do nível de significância α e de uma distribuição amostral sob a hipótese nula H_0. (Fonte)

área $\frac{\alpha}{2}$, como mostrado na Figura 10.2b. Pois, estamos interessados em qualquer desvio significativo, seja para cima (positivo) ou para baixo (negativo).

Um **teste unilateral** é usado quando queremos verificar se o parâmetro é apenas maior ou apenas menor que o valor especificado. A hipótese alternativa pode ter duas formas:

unilateral à esquerda:

$$H_1 : \theta < \theta_0$$

Neste caso, a região crítica está na cauda esquerda (Figura 10.2a), pois estamos interessados em apenas os valores extremos menores do que o valor da hipótese nula.

unilateral à direita:

$$H_1 : \theta > \theta_0$$

Aqui, a região crítica está na cauda direita da distribuição (Figura 10.2c), uma vez que estamos interessados em apenas os valores extremos maiores do que o valor da hipótese nula.

Sintetizando, a escolha entre os tipos de teste, bilateral ou unilateral, depende do problema de pesquisa: se há interesse em detectar qualquer diferença (para mais ou para menos), usa-se o bilateral; se há uma direção esperada ou relevante (aumento ou redução), opta-se pelo teste unilateral. No exemplo do teste do efeito de um suplemento alimentar no tempo médio de reação de motorista ao volante (Seção 10.1), a hipótese alternativa é a redução do tempo em relação ao valor da hipótese nula. Portanto, o tipo de teste a ser conduzido é do tipo unilateral à esquerda.

A tomada de decisão em um teste de hipóteses é realizada comparando a estatística obtida a partir de uma amostra de teste com o valor crítico x_c. Para ilustrar esse processo, suponha que desejamos avaliar se uma moeda é tendenciosa, ou seja, se a proporção de caras obtidas difere da proporção esperada de 0,5 sob a hipótese nula. Como estamos interessados em qualquer desvio (para mais ou para menos), devemos conduzir um teste bilateral.

Após 50 lançamentos, obtivemos uma proporção de 0.65 de caras (nossa estatística amostral de teste). Para avaliar a evidência estatística contra a hipótese nula H_0, utilizando um nível de significância $\alpha = 0.05$, recorremos aos valores críticos previamente calculados na Seção 9.3.2: $z_{0.025} = 0.38011$ e $z_{0.975} = 0.65989$. Esses valores definem os limites da região de aceitação do teste bilateral. Valores fora desse intervalo compõem a região de rejeição. Se a estatística amostral estiver dentro da região de aceitação, concluímos que não há evidência estatística suficiente para rejeitar H_0 em favor da hipótese alternativa H_1. Por outro lado, se a estatística estiver na região de rejeição, então há evidência suficiente para rejeitar H_0.

No exemplo, a proporção observada (0.65) está dentro da região de aceitação, ou seja, entre os valores críticos. Assim, afirmamos que os dados não fornecem evidência estatística suficiente para rejeitar H_0. Em outras palavras, com base nos dados observados e no nível de significância de 5%, não podemos concluir que a moeda está viciada.

10.5 Erros em Testes de Hipóteses

Dado um nível de significância α, testes de hipóteses consistem essencialmente em decidir se se deve rejeitar ou não a hipótese nula H_0, observando a posição de uma estatística amostral, ou estatística de teste, dentro de uma distribuição construída sob a suposição de que H_0 é verdadeira. É importante destacar que, quando um valor observado a partir da amostra recai em uma região crítica, definida com base na hipótese alternativa H_1 como vimos na Seção 10.4, isso não implica automaticamente que a hipótese nula H_0 seja falsa. Tal ocorrência apenas indica que os dados fornecem evidência estatística suficiente para rejeitar H_0, mas não garantem que H_1 seja verdadeira, nem que H_0 seja falsa com certeza.

Como vimos na Seção 10.3, a definição das regiões de decisão pode ser feita com base nas distribuições amostrais sob H_0 e H_1, utilizando intervalos de percentis para delimitar regiões críticas (de rejeição) e de não rejeição. As decisões nesse contexto estão sujeitas à incerteza amostral e, portanto, ao risco de cometer erros. Por essa razão, além de decidir entre H_0 e H_1, busca-se garantir que essa decisão seja estatisticamente fundamentada. Partindo do pressuposto de que a ausência de evidência contra H_0 não é o mesmo que evidência a seu favor, procura-se estimar ou controlar as probabilidades associadas aos possíveis erros cometidos nessa decisão.

É inteiramente possível que H_0 seja verdadeira mesmo quando os valores observados recaem na região de rejeição. Nessa situação, comete-se um **erro do tipo I**, que consiste em rejeitar H_0 mesmo sendo ela verdadeira, como representado pela região azul hachurada na Figura 10.3. De forma análoga, é possível não rejeitar H_0 mesmo que ela seja falsa, caso o valor da estatística de teste caia dentro da região de não rejeição. Essa decisão incorreta caracteriza um **erro do tipo II**, que ocorre quando se mantém uma hipótese nula que, na verdade, deveria ser rejeitada. A probabilidade de cometer um

erro do tipo II é denotada por β. Esses erros são formalmente definidos com base na sobreposição das distribuições amostrais sob H_0 e H_1. A construção dessas distribuições permite estimar as probabilidades associadas: α representa a probabilidade de erro do tipo I (rejeitar H_0 quando ela é verdadeira), enquanto β representa a probabilidade de erro do tipo II (não rejeitar H_0 quando ela é falsa).

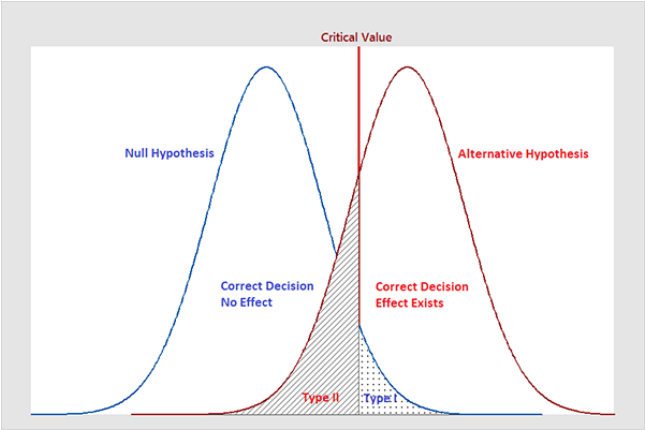


Figura 10.3: Tipos de erros numa tomada de decisão de aceitação (região esquerda do valor crítico) e de rejeição (região direita do valor crítico) de uma hipótese nula H_0: erro do tipo I (falso positivo) e erro do tipo II (falso negativo). (Fonte)

A formulação clássica dos erros do tipo I e do tipo II, baseada no uso de estatísticas de teste e em suas distribuições sob as hipóteses H_0 e H_a, é conceitualmente equivalente às abordagens mais gerais baseadas em distribuições amostrais. Em termos práticos, comete-se um **erro do tipo I** quando a estatística de teste recai na região crítica mesmo que H_0 seja verdadeira, e um **erro do tipo II** quando ela permanece na região de não rejeição apesar de H_0 ser falsa. Embora regiões críticas possam ser definidas por intervalos de percentis extraídos de distribuições amostrais mais amplas, a abordagem tradicional com estatísticas de teste segue sendo fundamental. Isso porque estatísticas como Z, t, χ^2 e F, entre outras, são derivadas de modelos teóricos com distribuições bem estabelecidas sob H_0, o que permite padronizar e sistematizar o processo decisório.

Após compreendermos os erros do tipo I e do tipo II, é importante reconhecer que toda decisão estatística envolve incerteza, e que essa incerteza pode ser quantificada por meio de quatro probabilidades centrais, α, 1 - α, β, 1 - β. Essas probabilidades avaliam o risco de tomar uma decisão incorreta ou correta, considerando se a hipótese nula H_0 é verdadeira ou falsa. Para organizar essas possibilidades, considere a seguinte tabela:

Tabela 10.1: Graus de incerteza em tomadas de decisão

Situação real	Rejetar H_0	Não rejeitar H_0
H_0 verdadeira	erro do tipo I (α)	acerto (1 - α)
H_1 verdadeira	acerto (1 - β)	erro do tipo II (β)

A Tabela 10.1 resume as quatro principais probabilidades que envolvem uma inferência:

item[$(1-\alpha)$:] **Nível de confiança** (em inglês, *confidence level*). Representa a proba-

bilidade de não rejeitar a hipótese nula (H_0) quando ela é verdadeira, ou seja, tomar a decisão correta.

Nível de significância (em inglês, *significance level*). Corresponde à probabilidade de rejeitar a hipótese nula (H_0) quando, na verdade, ela é verdadeira (erro do tipo I).

($1-\beta$): Poder do teste (em inglês, *power of test*). Indica a capacidade ou eficácia do teste de detectar corretamente uma situação em que a hipótese nula (H_0) é falsa. É a probabilidade de rejeitar H_0 quando ela realmente é falsa.

β: Probabilidade de erro do tipo II (em inglês, *Type II error probability*). Representa a chance de não rejeitar a hipótese nula (H_0) quando ela é falsa, incorrendo em um erro do tipo II.

Essas quatro probabilidades expressam, de forma quantitativa, o grau de incerteza que enfrentamos ao tomar decisões com base em dados amostrais. Compreender essas relações é essencial para avaliar a qualidade de um teste estatístico e escolher adequadamente o nível de significância α de regiões de rejeição. Na Seção 10.3, vimos que $P(\bar{X} > \bar{x}_c \mid H_0) = \alpha$, ou seja o valor crítico $\bar{x}_c$ define o limite da região de rejeição para a hipótese nula H_0. Sob a distribuição de $\bar{X}$ assumindo que H_0 é verdadeira, a probabilidade de que $\bar{X}$ assuma um valor menor ou igual a $\bar{x}_c$ é dada por $P(\bar{X} \leq \bar{x}_c \mid H_0) = 1-\alpha$. Por outro lado, considerando a distribuição de $\bar{X}$ sob a hipótese alternativa H_1, temos que a probabilidade de que $\bar{X}$ ultrapasse o valor crítico é $P(\bar{X} \geq \bar{x}_c \mid H_1) = 1-\beta$ e a probabilidade de que $\bar{X}$ fique abaixo do valor crítico é $P(\bar{X} < \bar{x}_c \mid H_1) = \beta$. Figura 10.1 ilustra graficamente essas divisões das distribuições sob H_0 e H_1, evidenciando as quatro regiões e suas respectivas probabilidades de ocorrência.

Um teste ideal seria aquele que tem um valor crítico com baixo α e baixo β, minimizando ambos os tipos de erro. No entanto, essas probabilidades não são independentes entre si. De fato, existe um compromisso (em inglês, *trade-off*): reduzir o nível de significância α (diminuindo a chance de erro tipo I) geralmente aumenta β (chance de erro tipo II), a menos que se aumente o tamanho da amostra ou se melhore a precisão do experimento. Da mesma forma, aumentar o poder do teste exige planejamento amostral adequado e boa sensibilidade da estatística empregada.

No exemplo dado na Seção 10.3, não dispomos de evidências estatísticas suficientes para rejeitar a hipótese nula H_0. Entretanto, sabemos que essa hipótese não é verdadeira – a proporção de ocorrência de caras nos lançamentos deveria ser 0.5. Podemos dizer que ocorreu um erro tipo II ao não rejeitar a hipótese nula com um teste realizado com nível de significância de 5%. Aumentando o tamanho da amostra, a variabilidade da distribuição diminui, pois a separação entre as distribuições sob H_0 e H_1 aumentaria como mostra o exemplo na Seção 9.3.2. Isso reduz o erro tipo II e proporciona evidências estatísticas suficientes para rejeitar H_0 mantendo o mesmo nível de significância.

10.6 P-valor

No procedimento clássico de teste de hipóteses, geralmente começamos fixando um nível de significância α, que representa a probabilidade máxima tolerada de cometer um erro do tipo I, ou seja, rejeitar a hipótese nula H_0 quando ela é verdadeira. Com base em α, definem-se regiões críticas para decidir se H_0 deve ser rejeitada. Uma abordagem alternativa é não fixar previamente o valor de α, permitindo que o usuário do teste escolha esse nível com base na evidência estatística observada. Nesse contexto, é comum calcular o **p-valor** (em inglês, *p-value*), denotado por α^*. O p-valor representa a probabilidade de observar uma estatística amostral $\overline{x}_{obs}$ tão extrema quanto a obtida (ou mais extrema), assumindo que H_0 seja verdadeira.

A definição do p-valor em relação à $\overline{x}_{obs}$ varia conforme a hipótese alternativa:

Teste lateral à esquerda ($H_a : \mu < \mu_0$):

$$\alpha^* = P(\overline{X} < \overline{x}_{obs} \mid H_0 \text{ verdadeira}). \tag{10.1}$$

Teste bilateral ($H_a : \mu \neq \mu_0$) , se a distribuição amostral, ou a distribuição da estatística de teste for simétrica em torno de μ_0

$$\alpha^* = 2 \times P(\overline{X} > \overline{x}_{obs} \mid H_0 \text{ verdadeira}), \text{ quando } \overline{x}_{obs} > \mu_0. \tag{10.2}$$

Para uma formulação mais geral,

$$\alpha^* = 2 \times P(\mid \overline{X} - \mu_0 \mid > \mid \overline{x}_{obs} - \mu_0 \mid\mid H_0 \text{ verdadeira}). \tag{10.3}$$

Teste lateral à direita ($H_a : \mu > \mu_0$):

$$\alpha^* = P(\overline{X} > \overline{x}_{obs} \mid H_0 \text{ verdadeira}). \tag{10.4}$$

O p-valor (α^*) é, portanto, uma medida contínua da evidência contra a hipótese nula H_0. Quanto menor o p-valor, mais forte é a evidência contra H_0 e, portanto, a favor de sua rejeição. Em aplicações práticas, é comum comparar o p-valor com um nível de significância pré-estabelecido, como 0.05 ou 0.01, para tomar uma decisão. Se o p-valor for menor que α, consideramos que há evidência estatística significativa contra H_0 e, portanto, a rejeitamos. Caso contrário, não rejeitamos H_0, pois os dados não oferecem evidência suficiente.

A obtenção do p-valor depende do tipo de teste e da distribuição da estatística. As formas mais comuns de cálculo incluem:

- consulta a tabelas estatísticas (como a normal, t de Student ou qui-quadrado);

- uso de *softwares* estatísticos (como R, Python ou SPSS), que utilizam a estatística de teste para calcular o p-valor; e
- uso de percentis amostrais por meio de técnicas computacionais, como *bootstrap*, em abordagens mais modernas, especialmente com simulações.

Em essência, o p-valor resume numericamente a compatibilidade dos dados com H_0 e serve como uma ferramenta objetiva para guiar decisões em testes de hipóteses. Weiss [99] propõe diretrizes interpretativas que classificam a força da evidência com base no p-valor, auxiliando na comunicação dos resultados.

p-valor	Evidências contra H_0
$p > 0.1$	evidência fraca
$0.05 < p \leq 0.1$	evidência moderada
$0.01 < p \leq 0.05$	evidência forte
$p \leq 0.01$	evidência muito forte

Tabela 10.2: Grau de evidência contra a hipótese nula com base no p-valor.

10.7 Interpretação de Resultados dos Testes de Hipóteses

Interpretar os resultados de um teste de hipóteses não se resume apenas a olhar o p-valor ou a estatística do teste. É muito importante entender o contexto do problema e pensar nas consequências práticas dos resultados obtidos. Quando rejeitamos a hipótese nula, isso não quer dizer que a hipótese alternativa é, com certeza, verdadeira. Significa apenas que os dados fornecem evidências suficientes para dizer que a hipótese nula provavelmente não é válida.

Também precisamos lembrar que todo teste de hipóteses está sujeito a dois tipos de erro:

Erro tipo I: rejeitar a hipótese nula quando ela, na verdade, é verdadeira.

Erro tipo II: não rejeitar a hipótese nula quando ela, na verdade, é falsa.

Esses erros podem ter impactos importantes dependendo da situação. Por isso, ao interpretar os resultados, devemos sempre considerar o risco desses erros e o quão confiáveis são as conclusões.

Outro ponto importante são os intervalos de confiança, que ajudam a entender melhor a incerteza envolvida nas estimativas, como a média de um grupo. Eles complementam as informações do p-valor e fornecem uma visão mais completa dos resultados.

Por fim, ao apresentar os resultados de um teste de hipóteses, é essencial explicar de forma clara o que eles significam dentro do contexto do problema estudado e deixar claro que sempre existe algum grau de incerteza envolvido.

10.8 Procedimentos de Testes de Hipóteses

Em testes de hipóteses, quando a distribuição da estatística de teste é conhecida e o contexto da pesquisa bem definido, seguimos geralmente os seguintes passos:

1. **Formulação das hipóteses:** Estabelecem-se a hipótese nula (H_0), que tipicamente assume a inexistência de um efeito ou diferença, e a hipótese alternativa (H_a), que representa a afirmação a ser investigada.
2. **Definição do nível de significância (α):** Define-se o **nível de significância**, denotado por α (Figura 10.1), que representa a probabilidade máxima de rejeitar erroneamente H_0.
3. **Seleção da estatística de teste apropriada:** Escolhe-se a estatística de teste adequada à natureza dos dados e às hipóteses em avaliação.
4. **Coleta e cálculo:** Coletam-se os dados amostrais e calcula-se o valor da estatística de teste correspondente.
5. **Decisão estatística:** A decisão sobre H_0 é tomada comparando a estatística de teste com o **valor crítico** da sua distribuição ou avaliando o **p-valor**. Se a estatística de teste estiver na região crítica (além do valor crítico) ou se o p-valor for menor que α, rejeita-se H_0. Caso contrário, falha-se em rejeitar H_0.
6. **Interpretação no contexto:** Os resultados estatísticos são interpretados à luz do problema de pesquisa, buscando entender suas implicações práticas.

A escolha do teste estatístico é uma etapa crucial no processo de teste de hipóteses, pois depende da natureza dos dados e dos objetivos do estudo. Como discutido na Seção 9.2.2, muitos testes estatísticos tradicionais assumem que os dados seguem uma distribuição normal. Uma maneira eficaz de avaliar visualmente essa suposição é por meio do gráfico quantil-quantil (Q-Q plot), ilustrado na Figura 6.11. Embora alguns testes, como o teste qui-quadrado, não exijam normalidade, eles se aplicam apenas a variáveis categóricas e têm limitações quanto ao tipo de dado analisado.

Quando trabalhamos com dados numéricos que não satisfazem o pressuposto de normalidade, podemos recorrer a métodos não-paramétricos, que são mais flexíveis por não dependerem de uma distribuição específica. Entre esses métodos, destacam-se o *bootstrap* de reamostragem e o teste de permutação (em inglês, *permutation test*), conforme abordado na Seção 10.2.2. O método *bootstrap* consiste em gerar diversas amostras aleatórias com reposição a partir dos dados originais para estimar a distribuição de uma estatística de interesse (como a média ou a mediana). Essa abordagem permite construir intervalos de confiança e distribuições amostrais para as hipóteses nula e alternativa sem a necessidade de atender aos pressupostos rígidos dos testes paramétricos.

Por fim, a simulação de Monte Carlo pode ser utilizada para complementar essas técnicas, gerando distribuições amostrais por meio de simulações repetidas. Ao integrar métodos não paramétricos com simulações de Monte Carlo, é possível realizar testes de hipóteses de forma mais robusta e confiável, especialmente em situações onde os pressupostos dos testes paramétricos não são atendidos.

10.9 Considerações Finais

Neste capítulo, aprofundamos o estudo dos testes de hipóteses, uma técnica fundamental da inferência estatística que permite avaliar a plausibilidade de suposições sobre parâmetros populacionais com base em dados amostrais. Enquanto o Capítulo 9 tratou da estimação de parâmetros por meio de distribuições amostrais, aqui nos concentramos na tomada de decisão estatística entre uma hipótese nula (H_0) e uma hipótese alternativa (H_1), com base em evidências extraídas da amostra.

Apresentamos duas abordagens principais para a aceitação ou rejeição de hipóteses: valores críticos e p-valor. Embora os **valores críticos** sejam uma abordagem tradicional e conceitualmente direta, o **p-valor** tem ganhado preferência por oferecer maior flexibilidade, clareza interpretativa e facilidade de comparação entre estudos. Discutimos também o papel da **estatística de teste** e como sua distribuição sob a hipótese nula permite estabelecer critérios objetivos de decisão com base no **nível de significância** (α). A definição adequada desse nível é essencial, pois está diretamente relacionada à probabilidade de cometer um **erro Tipo I** (rejeitar H_0 quando ela é verdadeira). Também abordamos o **erro Tipo II** (β) e a importância de equilibrar o poder do teste com o controle dos erros estatísticos. A formulação da hipótese alternativa determina se o teste será unilateral (à direita ou à esquerda) ou bilateral, o que impacta diretamente a região crítica e, portanto, a chance de erro Tipo I. Testes bilaterais investigam desvios significativos em ambas as direções em relação ao valor esperado sob H_0, dividindo α entre as duas caudas da distribuição. Já os testes unilaterais se concentram em detectar desvios em apenas uma direção, alocando todo o α em uma única extremidade.

Com o avanço da capacidade computacional, métodos como o ***bootstrap*** e **simulações de Monte Carlo** tornaram-se ferramentas indispensáveis na análise estatística moderna. Introduzimos dois métodos de *bootstrap*, **reamostragem** e **percentil**, e mostramos como esses recursos permitem estimar valores críticos e **intervalos de confiança** sem depender de pressupostos rígidos sobre a distribuição dos dados. Além disso, apresentamos o **teste de permutação**, uma aplicação do *bootstrap* em testes não paramétricos de hipótese. Essa flexibilidade amplia o escopo de aplicação dos testes de hipóteses, especialmente em contextos com pequenos tamanhos amostrais ou distribuições desconhecidas. A utilização de ferramentas computacionais, como *softwares* estatísticos (R, Python, etc.), viabilizou a implementação desses métodos de forma rápida e precisa. Hoje, é possível gerar distribuições amostrais e calcular estatísticas complexas em tempo real, eliminando a necessidade de tabelas predefinidas. Isso não apenas agiliza o processo analítico, mas também aumenta sua precisão, permitindo análises mais

robustas e replicáveis.

Por fim, destacamos que a interpretação dos resultados é tão importante quanto os cálculos realizados. Uma avaliação cuidadosa, que leve em conta o contexto, os pressupostos e os possíveis erros estatísticos, é essencial para que as conclusões extraídas dos testes de hipóteses sejam válidas e informativas. A aplicação consciente dessas técnicas contribui significativamente para a qualidade da inferência estatística em pesquisas científicas e decisões baseadas em dados.

10.10 Exercícios

1. Faça os exercícios de aprendizado LC9.1 – LC9.15 relacionados aos **testes de hipótese**, conforme proposto em [36]. Verifique suas respostas no apêndice D do mesmo livro.

2. Explique com suas próprias palavras os textos nas Seções 9.6.2 e 9.6.3 em [36].

3. Identifique as perguntas, as estatísticas do teste e as amostras nos exemplos apresentados no Capítulo 9 em [36].

4. Visualize, com base nas distribuições das estratégias stick e switch, as quatro probabilidades: α (taxa de erro tipo I), $(1-\alpha)$ (níveis de confiança), β (taxa de erro tipo II) e $(1-\beta)$ (poder do teste) para a hipótese de que "é mais benéfico manter a porta inicialmente escolhida" sob um nível de significância de 0.05 no contexto do problema de Monty Hall, discutido no item 3 da Seção8.8.

5. Em 1974 foi publicado no *Journal of Applied Psychology* um estudo, usando os dados promotions disponíveis em [13], para responder a pergunta: "Será que seu gênero afetará suas chances de ser promovido?". Na Seção 9.1 em [36], os autores formularam a pergunta como duas hipóteses H_0 e H_1 e aplicaram a técnica de *bootstrap* (teste de permutação) para respondê-la. Plote o gráfico de distribuição das diferenças de proproções amostrais, hachurando a área correspondente ao p-valor e ao nível de significância 0.05%.

6. Em um episódio do programa americano *Mythbusters*, os apresentadores realizaram um experimento para investigar a pergunta "Se você vê outra pessoa bocejar, é mais provável que você boceje também?". Os dados mythbuster_yawn estão disponíveis no [13]. Na Seção 8.6 em [36], os autores mostram como utilizar a técnica do percentil para estimar, com um nível de confiança de 95%, o intervalo de confiança do parâmetro populacional P, que representa a diferença entre a proporção de pessoas que bocejam ao ver outra pessoa bocejar e a proporção de pessoas que bocejam sem observar outra pessoa bocejar. Aplique o teste de permutação para respondê-la. Plote o gráfico de distribuição das diferenças de proproções amostrais, hachurando a área correspondente ao p-valor e ao nível de significância 0.05%.

Capítulo 11

Estatística de Inferência: Regressão

No Capítulo 9, discutimos os conceitos de estimativa pontual e intervalar, fundamentais para a inferência estatística. Essas técnicas permitem fazer afirmações sobre parâmetros populacionais com base em dados amostrais, tratando explicitamente a incerteza associada ao processo de amostragem. A estimativa pontual fornece um valor único como aproximação do parâmetro populacional, enquanto a estimativa intervalar oferece um intervalo de confiança que expressa a margem de erro esperada. No Capítulo 10, aprofundamos o estudo dos testes de hipóteses, que complementam as estimativas ao fornecer um arcabouço para avaliar suposições sobre parâmetros populacionais. Por meio desses testes, podemos verificar, com base nos dados amostrais, se há evidências estatísticas suficientes para rejeitar ou não uma hipótese previamente formulada.

Neste capítulo, damos mais um passo na estatística inferencial ao introduzir a **regressão**, uma técnica que permite investigar **relações entre variáveis**. Enquanto as estimativas e os testes de hipóteses se concentram em características individuais de uma população, a regressão busca modelar como uma variável dependente (ou resposta) se comporta em função de uma ou mais variáveis independentes (ou explicativas). Do ponto de vista matemático, a regressão é um conjunto de técnicas estatísticas que estimam uma **função matemática** capaz de descrever a relação entre essas variáveis. O modelo resultante, geralmente apresentado na forma de uma equação algébrica, permite não apenas inferir a intensidade e o sentido populacional da associação entre as variáveis envolvidas a partir de dados amostrais (**regressão explicativa**), mas também prever valores futuros a partir desses dados (**regressão preditiva**).

Abordaremos a regressão explicativa com base em dados amostrais, começando pelos fundamentos na Seção 11.1 que sustentam essa técnica, incluindo a definição e os princípios básicos da regressão. Em seguida, exploraremos na Seção 11.2 o pré-processamento, ou a análise exploratória de dados, que são cruciais para preparar o conjunto de dados e identificar padrões antes da modelagem. Passaremos então à Seção 11.3, onde discutiremos a modelagem da regressão linear simples e multivariada, detalhando suas aplicações e diferenças. Por fim, apresentaremos as métricas para a avaliação do modelo na Seção

11.4, definindo o coeficiente de determinação R^2 e o índice de variância explicada (IVF), que nos permitem quantificar a eficácia do modelo em explicar a variação dos dados.

11.1 Fundamentos da Regressão

A **regressão** é um método estatístico desenvolvido pelo geneticista Francis Galton durante suas pesquisas sobre variação e hereditariedade de características humanas entre as décadas de 1860 e 1890 [70]. Essa técnica visa a estimar a relação entre uma **variável dependente** y (ou **resposta/resultado**) e uma ou mais **variáveis independentes** x_i (ou **explicativas**). Galton ampliou o conceito de **coeficiente de correlação**, que quantifica, através de valores entre -1 e 1, a força e a direção da associação linear entre duas variáveis numéricas. Essa abordagem é detalhada na Seção 6.2. Galton propôs representar a relação entre duas variáveis por meio de uma **função matemática** f, que descreve como as variáveis independentes influenciam a variável dependente, conforme a seguinte equação:

$$y = f(x_1, x_2, \cdots, x_n) = a_1 g_1(x_1) + a_2 g_2(x_2) + \cdots + a_n g_n(x_n). \tag{11.1}$$

Os parâmetros a_i que descrevem a relação entre a variável dependente y (resultado) e as variáveis independentes (explicativas) são denominados **coeficientes de regressão**. As variáveis independentes x_i podem ser contínuas ou categóricas.

Assim, a **regressão** consiste na **estimativa dos coeficientes de uma função** ou de um modelo matemático, que melhor se ajusta a um conjunto de observações, com o objetivo de identificar e descrever as relações subjacentes entre as variáveis nos **dados amostrais**. Para atender a diferentes tipos de dados e cenários analíticos, são aplicadas diversas funções, resultando em diferentes abordagens de regressão:

Regressão Linear: Este é o tipo mais comum de regressão, utilizado para modelar a relação entre variáveis através de uma linha reta (no caso de uma única variável explicativa) ou um plano (no caso de duas variáveis explicativas), com $g_i(x_i) = x_i$. É especialmente adequada quando essa relação é aproximadamente linear. Nesse contexto, o **intercepto** representa o valor da variável dependente quando todas as variáveis independentes são zero. Já a **inclinação** indica a variação média na variável dependente para cada aumento de uma unidade na variável independente.

Regressão Não-Linear: Neste tipo de regressão, a relação entre as variáveis é modelada por meio de uma função g_i não linear. Essa abordagem é especialmente útil quando a interação entre as variáveis explicativas e variável de resultado não pode ser adequadamente representada por uma linha reta.

Regressão Polinomial: Esta é uma forma específica de regressão não-linear que se dis-

tingue por modelar a relação entre as variáveis através de um polinômio de grau superior a 1, ou seja, $g_i(x_i) = x_i^{m_i}$, com $m_i >= 1$. Ao contrário de outros métodos de regressão não-linear, que podem utilizar funções exponenciais, logarítmicas ou senoidais, a regressão polinomial permite capturar padrões complexos nos dados por meio de termos polinomiais, oferecendo maior flexibilidade na modelagem de relações não-lineares.

Regressão Logística: Esse tipo de regressão é utilizado quando a variável dependente é categórica, em vez de contínua. É frequentemente empregado para modelar a probabilidade de um evento binário, como a previsão da inadimplência de um cliente, com base em características independentes como renda e idade. Essa abordagem permite entender melhor como diferentes fatores influenciam a probabilidade de ocorrência de um evento específico.

Os coeficientes de regressão são estimados com base nos valores observados de uma amostra de dados. Os valores produzidos pelo modelo de regressão são chamados de **valores ajustados**, enquanto a diferença entre os valores observados e os ajustados é conhecida como **resíduos**. Como será abordado na Seção 11.3, quando a variável dependente é contínua, uma abordagem clássica para estimar esses coeficientes consiste em formular o problema como uma minimização da soma dos quadrados dos resíduos. Em outras palavras, busca-se encontrar os coeficientes que minimizam a soma dos quadrados das diferenças entre os valores observados e aqueles previstos pelo modelo. Já para a regressão logística, uma técnica comum para estimar os parâmetros é a Máxima Verossimilhança. Essa abordagem busca encontrar os valores dos parâmetros do modelo que maximizem a probabilidade de ter observado os dados que foram coletados. A regressão logística não será aprofundada neste manuscrito.

É importante destacar que, embora a regressão seja uma técnica bem fundamentada, ao aplicamo-na para inferir relações entre características de uma população a partir de dados amostrais, somos confrontados com dois níveis de incerteza. Primeiro, a incerteza inerente à própria natureza dos dados observados, pois os dados que coletamos são, por natureza, variáveis aleatórias. Eles representam apenas uma amostra de um universo maior e, mesmo que a relação "verdadeira" exista na população, nossas medições estão sujeitas a variabilidade aleatória e erros de observação. E segundo, a incerteza residual do modelo, pois mesmo o "melhor" modelo ajustado não explica perfeitamente toda a variabilidade da variável dependente. Sempre haverá resíduos, indicando que o modelo capturou uma tendência, mas não a relação determinística exata, pois fatores não incluídos ou a aleatoriedade intrínseca do fenômeno contribuem para a variação.

Por isso, a validação do modelo é essencial, e deve incluir técnicas como a análise de resíduos e o uso de métricas de desempenho, entre elas o coeficiente de determinação R^2, que mede a proporção da variabilidade dos dados explicada pelo modelo, como veremos na Seção 11.4. Além disso, a inferência estatística sobre os coeficientes é igualmente importante. Realizada por meio de testes de significância

e intervalos de confiança, ela fornece uma estimativa do grau de incerteza associado à relação entre as variáveis, o que pode subsidiar a tomada de decisões, conforme veremos no Capítulo 12. Neste capítulo, nosso foco principal será a construção de modelos que descrevam as relações de forma **determinística** a partir de dados amostrais.

11.2 Pré-processamento

O pré-processamento, também conhecido como análise exploratória de dados (em inglês, *Exploratory Data Analysis* – EDA), é uma etapa crucial antes de realizar uma regressão. Essa fase permite uma compreensão aprofundada das distribuições das variáveis envolvidas, facilitando a identificação de padrões e anomalias que podem influenciar os resultados. No Capítulo 7, abordamos como a criação de visualizações de dados não apenas torna as informações mais acessíveis, mas também revela *insights* que podem se perder em tabelas numéricas. Essas visualizações comunicam de forma clara e eficaz as características dos dados, auxiliando na escolha do tipo de modelo de regressão.

Ao examinar as distribuições por meio de **histogramas** ou **gráficos de caixa** (em inglês, *boxplots*) com cinco estatísticas descritivas, é possível verificar se as variáveis explicativas e a variável de resposta seguem uma distribuição normal ou apresentam assimetrias, pois assimetrias podem indicar que a relação entre as variáveis não é linear. Figura 11.1 ilustra os gráficos de caixa correspondentes às distribuições da variável dependente, que são as alturas dos filhos de Galton, para cada estrato definido pela altura da variável independente, que são as alturas dos pais de Galton.

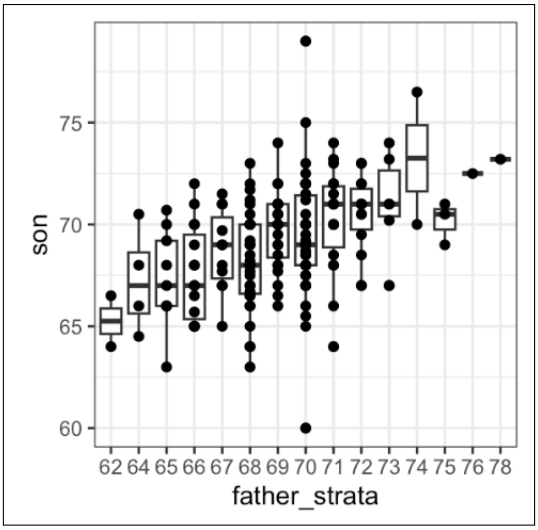


Figura 11.1: Gráficos de caixa: exploração da distribuição dos valores de uma variável dependente (childHeight) em relação a cada valor de uma variável independente (father) do conjunto de dados GaltonFamilies. (Fonte: [70])

Além disso, a análise exploratória não apenas identifica potenciais relacões entre as variáveis, mas também capacita o analista a escolher quais variáveis independentes incluir no modelo. A visualização desses relacionamentos, através de **matriz de gráficos de dispersão**, torna mais clara a compreensão

das interações, além de ajudar a evitar a inclusão de variáveis irrelevantes que poderiam comprometer a qualidade do modelo. A Figura 11.2 ilustra uma matriz de gráficos de dispersão das características father, mother e childHeight dos dados GaltonFamilies coletados pelo genecista Galton [96]. Nessa matriz são mostradas as relações de todas as possíveis combinações das três características dos dados.

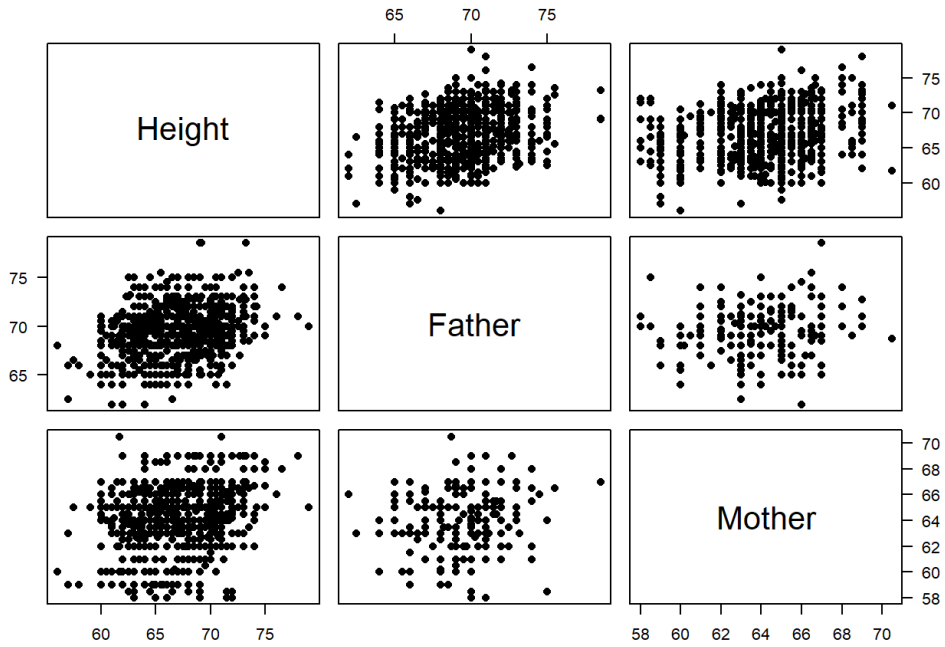


Figura 11.2: Matriz de gráficos de dispersão: exploração simultânea das relações entre três ou mais variáveis, par a par. (Fonte: Github)

Outro aspecto relevante da análise exploratória é a detecção de *outliers* e valores ausentes. Os *outliers* podem distorcer os resultados da regressão, levando a estimativas imprecisas e interpretações errôneas. Por isso, é importante identificá-los e decidir como tratá-los para garantir a robustez do modelo. Da mesma forma, o manejo adequado de valores ausentes é essencial, pois sua presença pode comprometer a integridade da análise e reduzir a representatividade dos dados. Os **gráficos de caixa** são ferramentas eficazes para detectar valores extremos,. Já um gráfico de calor (em inglês, *heatmap*), que utiliza diferentes cores para representar a variação de valores (cores mais escuras para valores altos e mais claras para valores baixos) em uma matriz de dados, pode facilitar a identificação de padrões e tendências, como ilustra a Figura 11.3.

Além de analisar os dados brutos, o cálculo de **estatísticas resumidas**, como médias, medianas, desvios padrão e quantis, fornece uma visão geral importante que pode orientar as decisões durante o pré-processamento. A partir dessas estatísticas, é possível identificar tendências gerais e áreas que demandam atenção, além de guiar a escolha de técnicas de normalização ou transformação, se necessário.

Para compreender a natureza e a intensidade das relações entre as variáveis e orientar as decisões

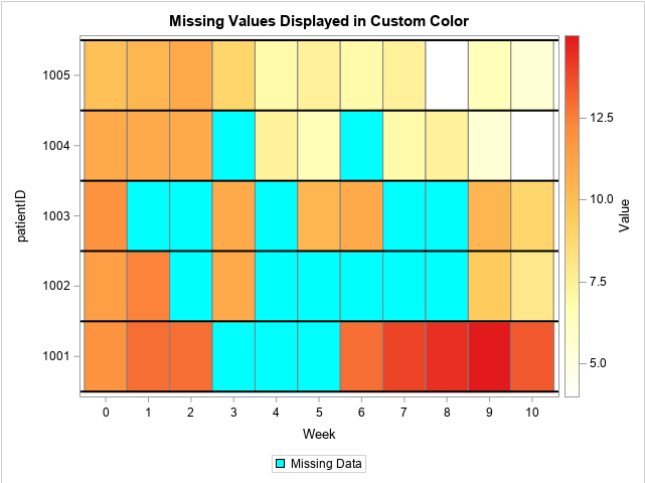


Figura 11.3: Gráfico de calor para visualizar a presença, com diferentes cores, e ausência (em branco) de um valor nas observações. (Fonte: Blogs.SAS)

de modelagem por regressão, pode ajudar também a visualização das estatísticas de relação, como a covariância e a correlação, entre duas variáveis de interesse. A **covariância**, quantificada pela Equação 6.9, mensura a variação conjunta de duas variáveis, indicando se elas tendem a aumentar ou diminuir juntas, ou se não apresentam uma relação aparente. Por outro lado, o **coeficiente de correlação de Pearson**, conforme definido pela Equação 6.10, é uma medida padronizada que varia de -1 a 1. Uma correlação de 1 indica uma relação linear perfeita positiva, -1 indica uma relação linear perfeita negativa e 0 indica ausência de relação linear. A Figura 11.4 ilustra os padrões de gráficos de dispersão associados aos diferentes coeficientes de correlação de Pearson. Conforme discutido na Seção 6.2, a correlação é mais fácil de interpretar e comparar do que a covariância, pois é independente das unidades de medida das variáveis e fornece uma medida mais clara da força e direção da relação entre elas.

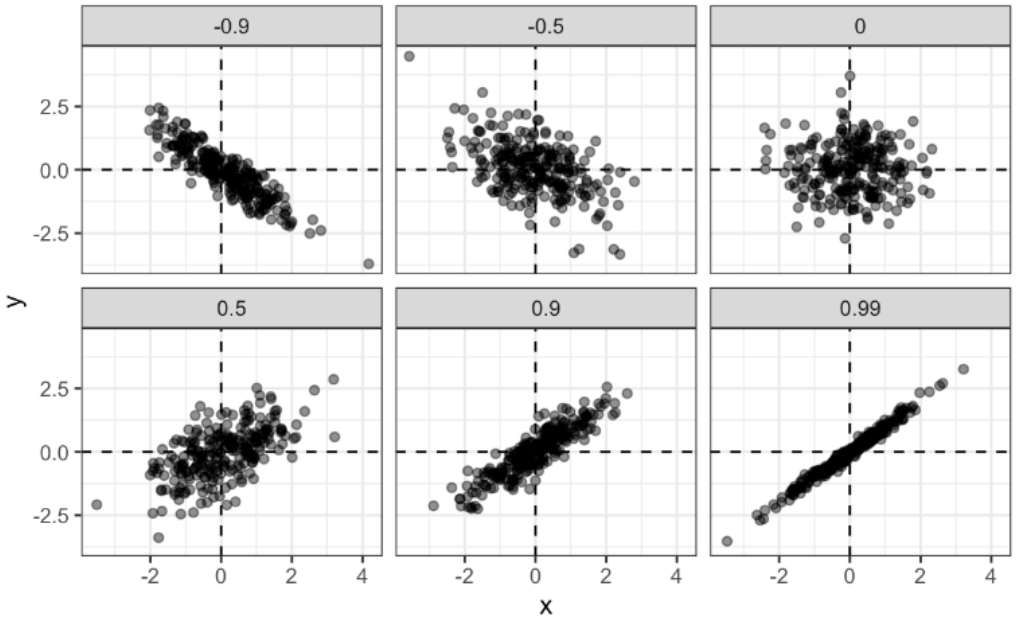


Figura 11.4: Coeficientes de correlação: padrões de gráficos de dispersão. (Fonte: [70])

Podemos resumir os coeficientes de correlação entre todas as variáveis, par a par, em um mapa de calor (ou *heatmap*), onde cada célula representa o coeficiente de correlação entre as duas variáveis que se cruzam nessa célula. Um coeficiente ρ próximo de 1 indica que as variáveis estão fortemente relacionadas. A Figura 11.5 ilustra um mapa de calor com coeficientes de correlação entre as quatro variáveis dos romances *bestsellers* na Amazon.

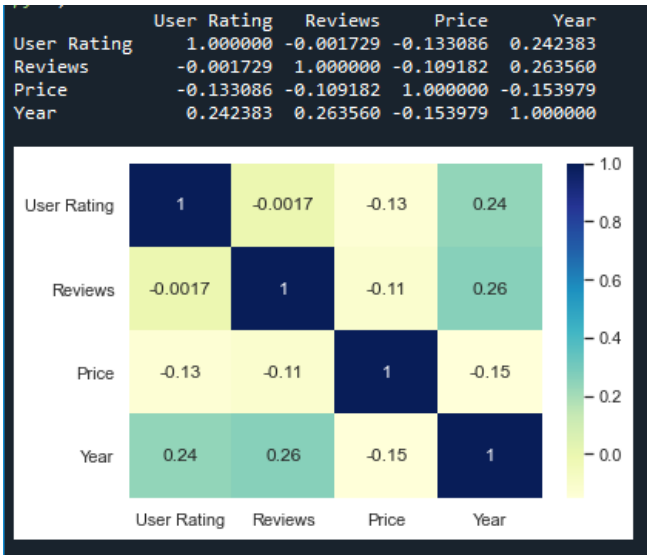


Figura 11.5: Mapa de calor com coeficientes de correlação entre as características avaliações dos usuários, resenha, preço e ano dos romances *bestsellers* na Amazon. Fonte: Geeksforgeeks

Se as variáveis altamente correlacionadas são independentes em um modelo de regressão, pode-se resultar em multicolinearidade. A **multicolinearidade** ocorre quando duas ou mais variáveis independentes em um modelo de regressão estão altamente correlacionadas entre si, tornando difícil separar os efeitos individuais de cada variável sobre a variável dependente, o que pode levar a estimativas imprecisas dos coeficientes de regressão. Por outro lado, variáveis independentes altamente correlacionadas com a variável dependente são candidatas promissoras para inclusão no modelo, pois têm o potencial de explicar uma quantidade significativa da variação na variável dependente. Por exemplo, a Figura 11.6 revela que o comprimento da pétala está altamente correlacionado com a largura da pétala e o comprimento do sépala, enquanto o comprimento do sépala está negativamente correlacionado com as outras três variáveis; portanto, o comprimento da pétala e o comprimento do sépala são candidatos para o modelo de regressão.

Na Seção 6.5, destacamos que o coeficiente de correlação de Pearson mede apenas a **relação linear** entre as variáveis e pode não capturar relações não-lineares. Os dados de Anscombe, ilustrados na Figura 1.1, são um exemplo clássico que demonstra essa limitação. Apesar de apresentarem características estatísticas idênticas – mesma média, variância, coeficiente de correlação de Pearson e linha de regressão linear – os quatro conjuntos de 11 pares de valores são visualmente distintos. Enquanto um conjunto mostra uma relação linear clara, outro apresenta uma relação não linear ou nenhuma relação aparente. Isso evidencia que confiar apenas no coeficiente de correlação de Pearson para ava-

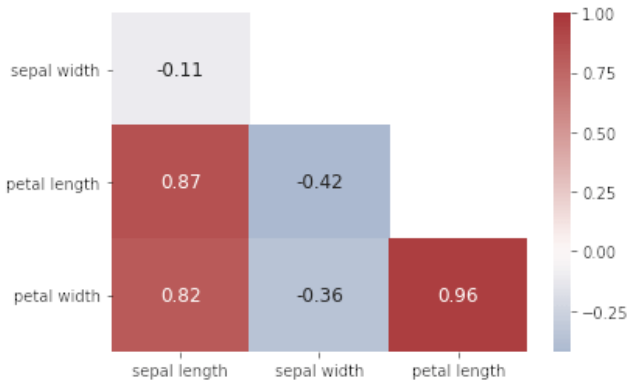


Figura 11.6: Mapa de calor com coeficientes de correlação explícitos em cada célula revela a multicolinearidade.

liar a relação entre variáveis pode levar a conclusões equivocadas, já que ele não detecta relações não lineares. Os dados de Anscombe são frequentemente utilizados para enfatizar a importância da visualização dos dados e para demonstrar que é essencial explorar graficamente os padrões nos dados antes de tirar conclusões sobre a relação entre as variáveis e na seleção das variáveis para regressão.

11.3 Modelagem

Nesta seção, exploraremos dois dos métodos mais comuns para representar algebricamente as relações lineares entre variáveis: a regressão linear simples e a regressão linear multivariada. A **regressão linear simples** analisa a relação entre uma variável dependente e uma única variável independente, oferecendo *insights* sobre como as mudanças na variável independente impactam a variável dependente. Em contrapartida, a **regressão linear multivariada** amplia essa análise ao considerar simultaneamente múltiplas variáveis independentes, permitindo uma compreensão mais profunda e complexa das interações entre os fatores.

A técnica clássica para estimar modelos de regressão linear é conhecida como **Regressão Linear por Mínimos Quadrados Ordinários** (em inglês, *Ordinary Least Squares* – OLS). Nesse método, os coeficientes da função linear são determinados de forma a minimizar a soma dos quadrados das diferenças entre os valores observados e os valores previstos pelo modelo. Entre as principais propriedades favoráveis do OLS, destacam-se:

- É uma técnica simples e eficiente quando os pressupostos do modelo linear clássico são satisfeitos.
- Fornece interpretações claras para os coeficientes estimados: cada coeficiente a_j representa o efeito marginal da variável explicativa x_j, mantendo as demais variáveis constantes.
- Permite realizar inferência estatística sobre os coeficientes (como testes de hipótese e intervalos de confiança), conforme será explorado no Capítulo 12.

- Pode ser implementado de forma elegante por meio de álgebra matricial. A solução para a Eq. 11.1 é expressa como:

$$a = (x^T x)^{-1} x^T y,$$

onde x^T denota a transposta do vetor x, composta pelas variáveis explicativas.

Porém, para que o ajuste calculado seja estatisticamente válido e confiável, especialmente quando queremos **inferir sobre a população com base em uma amostra**, alguns pressupostos essenciais precisam ser respeitados, como veremos no Capítulo 12:

- As variáveis explicativas não devem ser altamente correlacionadas entre si (ausência de multicolinearidade).
- Os resíduos do modelo devem apresentar **homocedasticidade**, ou seja, variância constante ao longo das observações.
- O OLS não lida bem com *outliers* ou com relações não lineares entre as variáveis, a menos que sejam aplicadas transformações apropriadas.

Esses pressupostos garantem que os coeficientes estimados tenham boas propriedades estatísticas, como:

- serem não viesados (em média, acertam o valor verdadeiro),
- terem variância mínima entre os estimadores lineares (eficiência), e
- permitirem a construção de testes de hipótese e intervalos de confiança confiáveis.

Neste capítulo nós nos ocupamos com o modelo determinístico de regressão em que a Eq. 11.1 descreve uma relação exata e previsível entre x_i e y. Para cada vetor de X, há um único valor de y. Não há aleatoriedade ou incerteza no relacionamento.

11.3.1 Regressão Linear Simples

A **regressão linear simples** oferece uma abordagem interpretável para modelar a relação entre variáveis, assumindo uma relação linear determinística entre a variável dependente y e uma variável independente x expressa por meio de uma equação linear do tipo

$$y = b_1 x + b_0, \tag{11.2}$$

onde b_1 é o coeficiente angular (inclinação da linha), e b_0 é o coeficiente linear (intercepto da linha). Enquanto o coeficiente de correlação quantifica a força e a direção da associação linear entre duas

variáveis, o coeficiente angular de regressão o complementa, descrevendo **como** a variável dependente muda em relação à variável independente.

Ao aplicar o **método dos mínimos quadrados** para derivar a regressão linear, é minimizada a soma dos quadrados das diferenças entre os valores observados e os valores ajustados, também conhecidos como **resíduos quadrados**. Esses resíduos podem ser interpretadas como as distâncias verticais entre os pontos de dados e a **linha de regressão**, como ilustra a Figura 11.7. O resultado do processo de minimização é a determinação dos coeficientes b_1 e b_0, que otimizam o ajuste aos dados, definindo a **linha de melhor ajuste** (em inglês, *best-fitting line*).

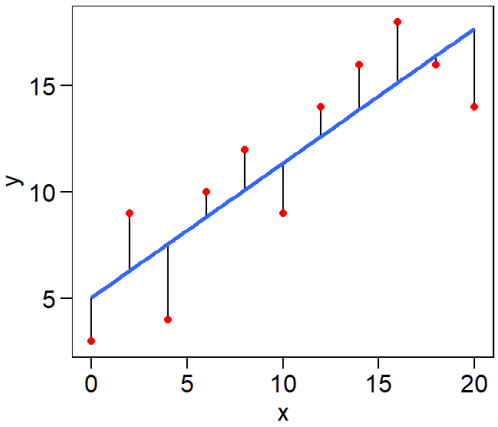


Figura 11.7: Resíduos numa regressão linear: distâncias entre os valores observados (em vermelho) e os valores ajustados pela regressão linear (em azul). (Fonte: Github)

Ao denotarmos os valores observados por y_i e os valores independentes por x_i, a minimização da soma q dos quadrados das distâncias d_i para m pares ordenados (x_i, y_i) pode ser expressa por:

$$q = \sum_{i=1}^{m} d_i^2 = \sum_{i=1}^{m} (y_i - b_0 - b_1 x_i)^2. \tag{11.3}$$

Ao expandirmos a expressão $(y_i - \beta_0 - \beta_1 x_i)^2$ e, em seguida, somar todos os termos, obtemos uma expressão quadrática em β_0 e β_1, e então derivamos essa expressão em relação a β_0 e β_1, respectivamente. Igualando as derivadas a zero, condição necessária para máximos e mínimos locais, chegamos ao seguinte sistema de equações, denominado **equações normais** do problema:

$$\begin{cases} m b_0 + (\sum_{i=1}^{m} x_i) b_1 = \sum_{i=1}^{m} y_i \\ (\sum_{i=1}^{m} x_i) b_0 + (\sum_{i=1}^{m} x_i^2) b_1 = \sum_{i=1}^{m} x_i y_i \end{cases} \tag{11.4}$$

As duas icógnitas β_0 e β_1 correspondem exatamente aos coeficientes da reta procurada.

Vasilescu apresenta o uso da função `lm()` em R para realizar uma regressão linear simples entre as variáveis `childHeight` e `gender` do conjunto de dados `GaltonFamilies`. Ele utiliza ainda a função `summary()` para gerar um resumo estatístico do modelo resultante, como mostra o seguinte trecho de códigos:

```
m0 = lm(childHeight ~ gender, data=GaltonFamilies)
summary(m0)
```

Ferreti, por sua vez, usa a função OLS() do pacote statsmodels.api em Python para obter resultados equivalentes:

```
m0 = sm.OLS.from_formula("childHeight ~ gender",data=GaltonFamilies)
result = m0.fit()
result.summary()
```

Além disso, Ferreti mostra o pré-processamento dos dados de GaltonFamilies para minimizar os impactos negativos de dados ausentes e *outliers*.

Uma alternativa para o cálculo dos coeficientes b_0 e b_1 da regressão linear é simplificar o processo utilizando estatísticas de média e desvio padrão, quando as variáveis independentes e dependentes apresentam distribuições normais bivariadas $N(X, Y; \mu_{X,Y}, \sigma_{X,Y})$ e uma relação linear exata entre elas. Neste caso, os coeficientes b_0 e b_1 podem ser diretamente estimados utilizando as médias, μ_X e μ_Y, os desvios-padrão, σ_X e σ_Y, e o coeficiente de correlação ρ entre as duas variáveis, de acordo com a seguinte relação [70]:

$$\begin{aligned} \frac{Y-\mu_Y}{\sigma_Y} &= \rho\frac{X-\mu_X}{\sigma_X} \\ Y &= \mu_Y + \rho(\frac{X-\mu_X}{\sigma_X})\sigma_Y. \end{aligned} \tag{11.5}$$

Essa abordagem simplifica significativamente os cálculos, tornando-os menos sensíveis a erros e mais eficientes computacionalmente. No entanto, na prática, raramente os dados seguem os pressupostos estritos que discutiremos no Capítulo 12, e, portanto, é preferível utilizar métodos tradicionais de estimativa de coeficientes de regressão que considerem todos os pontos disponíveis, como o método dos mínimos quadrados ordinários.

11.3.2 Regressão Linear Multivariada

A regressão linear múltipla, ou multivariada, tem como objetivo explicar a relação entre uma variável de resultado y e várias variáveis explicativas (x_1, x_2, $\cdots$, x_m). O foco é encontrar a função algébrica que melhor descreve essa relação linear, permitindo fazer associações entre os valores de y e as características dos x. Para identificar as variáveis que podem influenciar a variável de resultado, recomenda-se aplicar as técnicas de Análise Exploratória de Dados (EDA) apresentadas na Seção 11.2. Quanto maior o conhecimento prévio sobre o problema em estudo e a natureza das variáveis envolvidas, mais precisa será a regressão. Dependendo das características das variáveis e das hipóteses de pesquisa, podem-se distinguir dois tipos de modelos de regressão [36]:

Modelo de inclinações paralelas: Este modelo assume que os principais efeitos sobre a variável dependente resultam das variáveis explicativas, desconsiderando as interações entre elas. A relação entre a variável de resposta y e cada variável explicativa x_i é modelada como tendo um coeficiente de inclinação β_i separado, mas um intercepto b_0 comum:

$$y = b_0 + b_1 x_1 + b_2 x_2 + ... + b_m x_m \tag{11.6}$$

Os comandos para gerar um modelo linear com m variáveis independentes são $lm(y \sim x_1 + x_2 + \cdots + x_m, data = dataframe)$ em R e $sm.OLS.from_formula("y \sim x_1 + x_2 + \cdots + x_m", data = dataframe)$ em Python.

Modelo de interação: Este modelo permite considerar a "interação" entre as variáveis explicativas, além dos efeitos individuais de cada variável x_i. A relação entre a variável de resposta y e cada variável explicativa x_i é modelada como tendo um termo de interação adicional ao modelo de inclinações paralelas para levar em conta o efeito combinado de j variáveis explicativas $x_i \cdots x_i + j$ na variável de resposta, assumindo o seguinte aspecto:

$$y = b_0 + b_1 x_1 + b_2 x_2 + ... + b_n x_n + b_{12} x_1 x_2 + ... + b_{n-1,n} x_{n-1} x_n + ... \tag{11.7}$$

Para implementá-lo, utilizamos o comando $lm(y \sim x_1 * x_2 * \cdots * x_m, data = dataframe)$ em R. De forma equivalente, em Python, usamos $sm.OLS.from\_formula("y \sim x_1 * x_2 * \cdots * x_m", data = dataframe)$.

Ao decidir entre um modelo de interação ou um modelo de inclinações paralelas para uma aplicação específica, o princípio da Navalha de Occam serve como guia crucial. Esse princípio sugere a escolha do modelo mais simples que explique adequadamente os dados [36]. No contexto da regressão, a simplicidade se traduz em menor número de parâmetros. Comparando os coeficientes de determinação R^2 dos dois modelos, a serem explicados na Seção 11.4, podemos determinar se a complexidade adicional do modelo de interação, com seu termo de interação e, consequentemente, mais parâmetros, compensa em termos de capacidade explicativa. Se o modelo de interação apresentar um ganho substancial em R^2 em relação ao modelo de inclinações paralelas, a complexidade adicional pode ser justificada. No entanto, se a diferença em R^2 for mínima, a Navalha de Occam nos direcionaria para o modelo de inclinações paralelas, mais simples e parcimonioso.

Quando as duas variáveis Y e X seguem uma **distribuição bivariada normal**, ou seja elas estão correlacionadas de maneira que seus gráficos de dispersão (em inglês, *scatterplots*) apresentam uma forma oval, o coeficiente de correlação e o coeficiente angular de regressão são uma boa representação da relação. Porém, em muitos conjuntos de dados do mundo real, essa condição não se mantém. Além

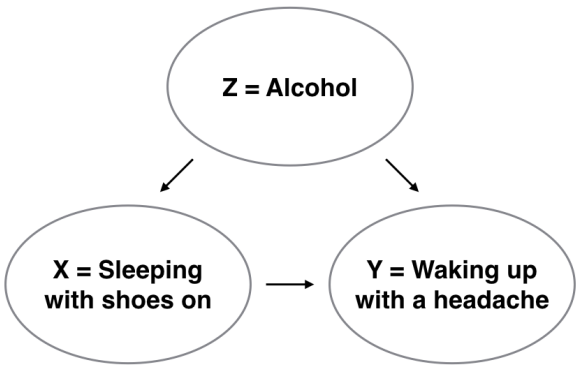


Figura 11.8: Associação espúria das varíaveis X e Y pela distorção introduzida pela terceira variável Z (Fonte: [36]).

disso, a interpretação dos coeficientes de regressão exige uma análise meticulosa, que deve considerar não apenas as variáveis explicativas, mas também suas interações. É crucial atentar para as **variáveis confusoras** (em inglês, *confounding variables*), que são fatores externos capazes de influenciar tanto a variável dependente quanto a independente. A presença dessas variáveis pode distorcer a interpretação dos resultados, levando a conclusões errôneas. Para entender a relevância das variáveis confusoras, é essencial reconhecer como elas se manifestam: podem criar associações espúrias, sugerindo relações entre as variáveis que, na verdade, são mediadas por fatores não considerados. Ismay exemplifica isso ao abordar a aparente associação espúria entre "dormir com sapatos" e "dores de cabeça", sem levar em conta o impacto do "consumo excessivo de álcool", como mostra a Figura 11.8. Nesse caso, o consumo de álcool atua como uma variável confusora, distorcendo a percepção da relação entre as duas variáveis e gerando associações enganosas [36].

Portanto, estabelecer relações de causalidade é um desafio significativo, frequentemente exigindo experimentos bem planejados ou a aplicação de métodos que controlam os efeitos das variáveis confusoras. A realização de experimentos bem planejados e a A aplicação de métodos estatísticos que controlam os efeitos das variáveis confusoras buscam minimizar o impacto dessas variáveis, permitindo que os pesquisadores se concentrem na relação direta entre a variável de resultado y e as variáveis explicativas x_i. A identificação de variáveis confusoras requer uma combinação de análise crítica dos dados, entendimento do contexto da pesquisa e utilização de métodos estatísticos apropriados. Não há uma abordagem única e infalível para detectar todas as variáveis de confusão. Contudo, adotar um conjunto articulado de boas práticas metodológicas, desde o planejamento do estudo até a análise dos dados, é fundamental. Essa abordagem visa reduzir vieses e aprimorar a validade das inferências, mitigando os efeitos das variáveis de confusão e contribuindo para uma interpretação mais precisa e confiável dos modelos.

11.4 Métricas para Avaliação do Modelo

As métricas de regressão são fundamentais para avaliar o desempenho de modelos de regressão, pois permitem medir a precisão das previsões em comparação com os valores reais. Entre as métricas mais utilizadas, destacam-se:

Erro Médio Absoluto (em inglês, *mean absolute error* – MAE): Calcula a média dos erros absolutos entre os valores previstos e os valores reais, oferecendo uma visão clara da precisão média das previsões.

Erro Quadrático Médio (MSE) (em inglês, *mean square error* – MSE): Mede a média dos erros ao quadrado, o que penaliza mais fortemente as previsões que se afastam significativamente dos valores reais, sendo útil para identificar discrepâncias maiores.

Raiz do Erro Quadrático Médio (RMSE) (em inglês, *root mean square error* – RMSE): Representa a raiz quadrada do MSE, trazendo a métrica de volta à mesma escala dos dados originais, facilitando assim a interpretação dos resultados.

O **coeficiente de determinação (R^2)** é uma métrica importante para avaliação do poder explicativo de um modelo de regressão. Ele indica a proporção da variação na variável dependente que é explicada pelo modelo. Um valor de R^2 igual a 0 significa que o modelo não explica nenhuma variabilidade dos dados, enquanto um valor de 1, ou um valor alto, não garante que o modelo seja bom para previsão, que as variáveis independentes sejam a causa da variável dependente, ou que o modelo seja o mais adequado. **Ele apenas quantifica a proporção da variância explicada**. A fórmula para calcular R^2 é:

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}},$$

onde SS_{res} é a soma dos quadrados dos resíduos, que mede a variabilidade dos dados em relação ao modelo ajustado e SS_{tot} é a soma total dos quadrados, que mede a variabilidade total dos dados em relação à média. Esses valores podem ser calculados como:

$$SS_{\text{res}} = \sum (y_i - \hat{y}_i)^2$$

$$SS_{\text{tot}} = \sum (y_i - \bar{y})^2$$

sendo y_i os valores observados, $\hat{y}_i$ os valores ajustados pelo modelo, e $\bar{y}$ é a média dos valores observados.

Apesar de ser uma métrica amplamente utilizada, o R^2 possui uma limitação importante: ele tende a aumentar sempre que se adicionam novas variáveis independentes ao modelo, mesmo que essas variáveis não sejam realmente relevantes. Para contornar esse problema, é utilizado o **coeficiente de determinação ajustado** ($R_ajustado^2$), que penaliza a inclusão de variáveis irrelevantes. O $R_ajustado^2$ ajusta o valor de R^2 levando em conta o número de variáveis independentes (k) e o

tamanho da amostra (n), sendo calculado por:

$$R^2_{\text{ajustado}} = 1 - \left(1 - R^2\right) \cdot \frac{n-1}{n-k-1}$$

Esse ajuste faz com que o $R_ajustado^2$ só aumente quando a adição de uma nova variável melhora o modelo mais do que seria esperado pelo acaso. Caso contrário, ele pode até diminuir. Por isso, o $R_ajustado^2$ é geralmente preferido ao R^2 quando se comparam modelos com diferentes números de variáveis, pois fornece uma medida mais realista da qualidade do ajuste, considerando a complexidade do modelo.

Uma outra métrica é o **fator de aumento da variância** (em inglês, *Variance Inflation Factor* – VIF). É uma medida que quantifica o quanto a variância dos coeficientes estimados em um modelo de regressão multivariada é aumentada devido à multicolinearidade entre as variáveis explicativas [2]. Quando há multicolinearidade, as variâncias dos coeficientes estimados se tornam maiores, dificultando a interpretação dos resultados. O VIF é calculado para cada variável explicativa no modelo. Especificamente, o VIF para o coeficiente estimado β_j (denotado como VIF_j) é o fator pelo qual a variância de β_j é "inflacionada" pela correlação com as outras variáveis explicativas. A fórmula para calcular o VIF de uma variável explicativa j é:

$$VIF_j = \frac{1}{1 - R_j^2}, \tag{11.8}$$

onde R_j^2 é o valor de R^2 obtido ao regressar a variável explicativa j nas demais variáveis explicativas do modelo. Um VIF geralmente aceitável é abaixo de 5, e valores acima de 5 ou 10 indica alta multicolinearidade, o que pode ser problemático para a interpretação dos resultados da regressão.

11.5 Considerações Finais

Neste capítulo, exploramos mais uma vertente da estatística inferencial: a construção de modelos matemáticos capazes de descrever a relação entre uma variável dependente e uma ou mais variáveis independentes a partir de dados amostris. Embora existam diferentes tipos de modelos, nosso foco recaiu sobre a regressão linear, especialmente por meio da técnica dos Mínimos Quadrados Ordinários (em inglês *Ordinary Least Squares* — OLS), uma ferramenta matemática consolidada e amplamente utilizada em diversas áreas do conhecimento. A técnica de OLS é simples, eficiente e bastante difundida. Contudo, por lidarmos com variáveis aleatórias, sua aplicação pressupõe o atendimento a determinadas condições estatísticas, como homocedasticidade, normalidade dos resíduos e ausência de autocorrelação. O não atendimento a esses pressupostos compromete a validade estatística do modelo, podendo gerar inferências equivocadas e prejudicar a capacidade preditiva.

Ao longo do capítulo, iniciamos pela fundamentação teórica da regressão, apresentando os princípios

essenciais da modelagem estatística. Destacamos a importância da análise exploratória de dados, etapa crucial para garantir que a modelagem se baseie em informações consistentes e em padrões bem compreendidos. Na sequência, aprofundamos a discussão sobre a regressão linear multivariada, diferenciando modelos com inclinações paralelas daqueles que incluem interações. Introduzimos também o Princípio da Navalha de Occam, que orienta a escolha por modelos mais simples sempre que apresentem desempenho semelhante ao de modelos mais complexos. Ressaltamos, ainda, a necessidade de cautela na interpretação dos coeficientes de regressão. A presença de variáveis confusoras pode distorcer as relações aparentes entre as variáveis independentes e a variável dependente. É essencial lembrar que a regressão, por si só, não estabelece relações causais, mas apenas associações condicionais entre as variáveis.

Outro ponto central deste capítulo foi a ênfase no uso de recursos gráficos durante todo o processo analítico. Os gráficos não apenas facilitam a interpretação dos dados, mas também ajudam a garantir a qualidade e a robustez dos modelos desenvolvidos, tanto para fins explicativos quanto preditivos. Ferramentas visuais como gráficos quantil-quantil, gráficos de dispersão, gráficos de caixa, histogramas e mapas de calor desempenham papéis fundamentais na detecção de padrões, outliers, tendências não lineares e possíveis violações dos pressupostos do modelo. Por exemplo, o gráfico quantil-quantil permite verificar a normalidade dos resíduos, que é um requisito central em muitos modelos de regressão. O alinhamento dos pontos em uma linha reta sugere aderência à distribuição normal. Gráficos de dispersão são indispensáveis para avaliar a relação entre variáveis e detectar padrões inesperados, valores extremos ou tendências não lineares. Gráficos de caixa ajudam na identificação de assimetrias e outliers, enquanto mapas de calor são particularmente úteis na visualização de padrões e correlações em grandes conjuntos de dados. Por fim, histogramas oferecem uma visão clara da distribuição das variáveis, auxiliando na avaliação da adequação dos dados à modelagem proposta.

De forma geral, a integração de análises gráficas fortalece a confiabilidade dos modelos e proporciona maior transparência ao processo analítico. Ao permitir a detecção precoce de problemas, esses recursos contribuem decisivamente para a construção de modelos estatísticos mais robustos, interpretáveis e alinhados aos pressupostos teóricos.

11.6 Exercícios

1. Faça os exercícios de aprendizado LC5.1 a LC5.8 relacionados aos **conceitos básicos de regressão linear simples**, conforme proposto em [36]. Verifique suas respostas no apêndice D do mesmo livro.

2. Faça os exercícios de aprendizado LC6.1 a LC6.3 relacionados à **regressão multivariada**, conforme proposto em [36]. Verifique suas respostas no apêndice D do mesmo livro.

3. Reproduza os gráficos dos modelos de interação e das inclinações paralelas apresentados nas

Figuras 6.3 e 6.8 do livro-texto [36], utilizando os conjuntos de dados disponíveis em evals.rda e MA_schools.rda. Em Python, o comando `read_rda()` do pacote `pyreadr` pode ser usado para ler arquivos com extensão `.rda`, que são arquivos de dados do R. Em seguida, analise e discuta as possíveis causas das variações nas inclinações das retas.

4. Baseando-se na Seção 6.3.4 do livro-texto [36], reproduza os gráficos de dispersão e a linha de regressão apresentados, utilizando o conjunto de dados disponível em Credit. Em seguida, explique o Paradoxo de Simpson com suas próprias palavras, abordando o conceito de variável confusora.

5. O conjunto de dados "mtcars"(*Motor Trend Car Road Tests*) oferece informações detalhadas sobre as características técnicas de 32 modelos de automóveis da década de 1970, incluindo consumo de combustível (mpg), peso do carro (wt), potência do motor (hp), e número de cilindros (cyl), entre outras. Desenvolva um modelo de regressão linear múltipla para explicar o consumo de combustível (mpg) desses veículos. Para isso, utilize a técnica dos Mínimos Quadrados Ordinários (OLS), empregando as variáveis independentes: wt (peso do carro em 1000 lbs), hp (potência bruta em horsepower), qsec (tempo no quarto de milha em segundos) e cyl (número de cilindros: 4, 6 ou 8). Siga as etapas abaixo para construir e avaliar seu modelo:

   (a) **Análise exploratória de dados (EDA)**: Conduza uma EDA completa para identificar *outliers*, explorar a existência de relações não lineares entre as variáveis e verificar possíveis colinearidades.

   (b) **Ajuste do modelo de regressão**: Ajuste um modelo de regressão linear múltipla (de inclinações paralelas) utilizando OLS. Apresente os coeficientes de regressão e o intercepto do modelo.

   (c) **Verificação de pressupostos**: Avalie se o modelo atende aos pressupostos-chave:

      - Normalidade dos resíduos: Utilize um gráfico Quantil-Quantil (Q-Q plot).
      - Homocedasticidade: Analise o gráfico dos resíduos versus valores ajustados.
      - Multicolinearidade: Calcule e interprete o Fator de Aumento da Variância (VIF) para cada variável independente.

   (d) **Avaliação da qualidade do ajuste**: Analise o valor do R^2 (Coeficiente de Determinação) como uma medida da qualidade do ajuste do seu modelo.

   (e) **Discussão e limitações:** Avalie se o modelo desenvolvido é estatisticamente adequado para explicar o consumo de combustível dos carros. Discuta suas principais limitações e o que isso implica para suas conclusões.

Capítulo 12

Modelagem Estatística: Regressão Preditiva

Ao analisarmos um conjunto de dados, geralmente começamos pela **estatística descritiva**, apresentada no Capítulo 6. Essa etapa tem como objetivo organizar, resumir e traduzir os dados em informações úteis, funcionando como uma fotografia da amostra: observamos tendências, identificamos padrões e reconhecemos variações. Tudo restrito ao que foi observado. No entanto, na maioria das vezes, nosso interesse vai além da amostra; queremos tirar conclusões sobre a população da qual ela foi extraída. É nesse ponto que entra a **estatística inferencial**, que, com base na teoria das probabilidades, permite generalizar os resultados e quantificar a incerteza associada a essas inferências. **Estimativas de médias populacionais** e **construção de intervalos de confiança**, discutidos no Capítulo 9, são exemplos clássicos desse processo.

Dando um passo além, abordamos no Capítulo 10 o **teste de hipóteses**, cuja finalidade é avaliar se os dados oferecem evidências suficientes para rejeitar uma hipótese nula em favor de uma alternativa que melhor representa o fenômeno investigado. Esse raciocínio é essencial em situações como comparar tratamentos médicos, avaliar o efeito de campanhas publicitárias ou verificar se duas populações se comportam de maneira semelhante. Muitas vezes, porém, nosso interesse não se limita a verificar se há diferença ou associação, mas em compreender como as variáveis se relacionam e quantificar essa relação. No Capítulo 11, introduzimos o **modelo de regressão**, uma ferramenta que descreve a relação entre uma variável de interesse (dependente) e uma ou mais variáveis explicativas (independentes), além de fornecer métricas que avaliam o quanto essas variáveis ajudam a explicar o comportamento da variável resposta. Até aqui, no entanto, tratamos essa relação de forma essencialmente determinística, focando na tendência central observada nos dados.

Ao longo de todas essas etapas, os recursos gráficos desempenham um papel central na análise estatística. Gráficos como histogramas, diagramas de dispersão, gráficos de caixa e mapas de calor não servem apenas para ilustrar resultados, mas são fundamentais para explorar, compreender e interpretar

os dados. Como vimos no Capítulo 4, nossa percepção é naturalmente sensível a padrões visuais, o que facilita a detecção de tendências, agrupamentos, *outliers* e relações que poderiam passar despercebidas em tabelas numéricas. Assim, a visualização atua como uma ponte entre os dados e nossa compreensão intuitiva dos fenômenos, guiando a formulação de hipóteses, a escolha de modelos e a condução de análises. Em ambientes com grandes volumes de dados, a visualização não é apenas um apoio, mas uma ferramenta essencial para orientar o uso eficiente dos métodos estatísticos e computacionais.

Vamos agora avançar para uma etapa que incorpora explicitamente a incerteza inerente do mundo real na construção de modelos matemáticos. Essa incerteza será modelada pela teoria de probabilidade apresentada no Capítulo 8. Esse é o papel fundamental da **modelagem estatística**. Modelar estatisticamente significa representar, por meio de uma equação matemática, como uma variável depende de outras, reconhecendo que essa relação nunca é perfeita. Sempre há variabilidade, seja por fatores não observados, ruído, erros de medição ou simples aleatoriedade. Por isso, além de captar a tendência geral, os modelos estatísticos incluem um componente probabilístico que expressa essa incerteza de forma explícita. Por exemplo, sabemos que o peso de uma pessoa depende, em parte, da sua altura. Pessoas mais altas tendem a pesar mais, mas há grande variação entre indivíduos com a mesma altura devido a outros fatores como idade, gênero, hábitos de vida ou genética. O modelo estatístico permite quantificar essa relação média e, ao mesmo tempo, reconhecer a variabilidade não explicada.

Ao construir esses modelos, vamos além da descrição e da inferência tradicional. Passamos a contar com ferramentas capazes de fazer previsões, simular cenários e apoiar decisões, sempre levando em conta a incerteza. Nesse contexto surge a **regressão preditiva**, que utiliza padrões dos dados para prever situações futuras ou casos ainda não observados. Ela se distingue, em parte, da regressão "explicativa", cujo foco está em compreender os mecanismos e relações entre variáveis no conjunto de dados analisado. Embora matematicamente semelhantes, diferem no propósito: prever versus relacionar. Um exemplo concreto do poder da [70] a modelagem preditiva vem do mundo esportivo, relatado por Irizarry em [70]. Diante de restrições orçamentárias severas, os gestores do time de beisebol Oakland Athletics (A's) adotaram uma estratégia baseada em análise de dados para identificar jogadores subvalorizados. Utilizando modelos preditivos, substituíram julgamentos subjetivos por critérios estatísticos, o que permitiu competir em alto nível contra equipes com orçamentos muito maiores. Essa abordagem não apenas transformou o desempenho do time, como também revolucionou a gestão esportiva, demonstrando que técnicas estatísticas aplicadas à modelagem são capazes de transformar dados em decisões estratégicas, precisas e competitivas, dentro e fora dos esportes.

Começamos este capítulo com a introdução de modelos matemáticos estatísticos na Seção 12.1. Aqui, expandimos a equação clássica de regressão para incorporar variáveis aleatórias, permitindo uma análise mais robusta das relações entre elas. Em seguida, na Seção 12.2, abordamos os pressupostos para validade de modelos de regressão linear não só para realização de testes de hipótese e construção de intervalos de confiança como também para predição. Enfatizamos a importância de satisfazer pres-

supostos fundamentais, como linearidade, independência dos erros, normalidade e homocedasticidade (conhecidos pela sigla LINE). Além disso, apresentamos gráficos de influência (ou *influential plots*) como ferramentas visuais relevantes para validar essas condições. Prosseguimos para a inferência analítica na Seção 12.3, focando nos métodos tradicionais para estimar parâmetros e testar hipóteses. Em contraste, a inferência computacional, discutida na Seção 12.4, explora técnicas flexíveis como simulação Monte Carlo e *bootstrap* para a análise de dados mais complexos. Por fim, na Seção 12.5, detalhamos as etapas do processo de regressão preditiva. Esta seção sintetiza as fases chave discutidas ao longo do capítulo, desde a formulação da pergunta de pesquisa até a interpretação dos resultados, destacando a natureza iterativa e adaptativa desse processo.

12.1 Modelos Estatísticos

Na Seção 11.3, apresentamos os fundamentos matemáticos utilizados para estimar os coeficientes de uma regressão linear simples a partir de um conjunto de dados observados (x, y), conforme a Eq. 11.2. Já na Seção 11.4, discutimos as métricas que avaliam a qualidade do ajuste desse modelo, permitindo quantificar sua capacidade de relacionar as variáveis independentes e a variável dependente.

É importante destacar que, na prática, raramente temos acesso aos dados de toda a população. Trabalhamos, portanto, com uma amostra, um subconjunto extraído da população, que é utilizada para inferir propriedades dos parâmetros populacionais. O modelo de regressão ajustado a uma amostra pode ser representado por:

$$y = \hat{\beta}_0 + \hat{\beta}_1 \cdot x \tag{12.1}$$

onde y é um valor observado da variável dependente, x é o valor observado da variável independente, $\hat{\beta}_0$ é o intercepto estimado, e $\hat{\beta}_1$ é o coeficiente angular estimado que representa a taxa de variação média de y em relação a x. Esses coeficientes, como discutido na Seção 11.1, são **estimativas amostrais** dos verdadeiros parâmetros populacionais β_0 e β_1.

Naturalmente, os dados observados (x, y) são apenas uma realização particular dentre todas as possíveis amostras que poderiam ser extraídas da população. Isso implica que há variabilidade amostral: se coletarmos uma nova amostra, os valores observados poderão ser diferentes, assim como os coeficientes estimados. Portanto, tanto as variáveis x e y devem ser tratados como **variáveis aleatórias**. Formalizando essa perspectiva probabilística, o modelo de regressão linear simples pode ser expresso como um modelo estatístico populacional:

$$Y = \beta_0 + \beta_1 \cdot X, \tag{12.2}$$

onde X e Y são variáveis aleatórias que representam, respectivamente, uma variável independente e

uma variável dependente da população, e β_0 e β_1 são parâmetros populacionais .

Contudo, a equação (12.2) descreve apenas a parte sistemática da relação entre X e Y. Na prática, parte da variabilidade de Y não é explicada pela linha de regressão. Para representar essa parcela, introduz-se o termo de erro aleatório ε, que corresponde aos resíduos, ou à diferença entre os valores observados e os ajustados como vimos no Capítulo 11. O **modelo estatístico populacional** completo é então:

$$Y = \beta_0 + \beta_1 \cdot X + \varepsilon. \tag{12.3}$$

O termo ε representa fatores não observados e variações aleatórias, sendo essencial para capturar a incerteza do modelo. Para garantir que os estimadores dos parâmetros do modelo ($\hat{\beta}_0$ e $\hat{\beta}_1$) sejam confiáveis, e para assegurar a validade dos testes de hipótese e dos intervalos de confiança, pressupomos que ε possua média zero, variância constante, seja independente dos valores de X e, idealmente, siga uma distribuição normal (especialmente para fins de inferência).

Essa formulação evidencia que o **modelo não é determinístico**, mas sim uma aproximação sujeita à incerteza inerente ao processo amostral. Assim, ao utilizar o modelo para previsão ou inferência, estamos, na verdade, utilizando distribuições de probabilidade associadas às variáveis e aos estimadores, o que é fundamental para a construção de intervalos de confiança e para a realização de testes de hipótese sobre os parâmetros.

A Figura 12.1 ilustra um exemplo prático do ajuste de uma regressão univariável linear simples utilizando o método dos mínimos quadrados ordinários (OLS), implementado com a função do pacote statsmodels de Python. Este exemplo segue a abordagem probabilística discutida, em que os coeficientes são obtidos como estimativas baseadas em uma realização amostral dos dados.

A tabela de resultados do statsmodels.OLS é essencial para avaliar a qualidade e significância de um modelo de regressão. Ela apresenta os coeficientes, a confiabilidade de suas estimativas e permite verificar se as suposições do modelo foram razoavelmente atendidas. As informações estão organizadas em três seções: estatísticas do modelo, coeficientes estimados e estatística dos resíduos. Conforme abordado na Seção 11.3, a análise da estatística dos resíduos é tão importante quanto à análise da estatística do modelo. Os resíduos revelam o quão bem o modelo se ajusta aos dados e se as suposições sobre modelo foram violadas.

A seção de estatísticas do modelo, que fica na parte superior da tabela, fornece informações gerais sobre o ajuste do modelo. Segue-se uma descrição breve de cada métrica:

R-squared (R^2) : É o coeficiente de determinação, uma métrica que mede a proporção da variância na variável dependente que é explicada pelas variáveis independentes presentes no modelo. Seu valor varia de 0 a 1. Por exemplo, um R^2 de 0.75 indica que 75% da variabilidade na variável dependente pode ser explicada pelo modelo. Quanto mais próximo de 1, melhor o ajuste do modelo aos dados observados.

```python
import statsmodels.formula.api as smf
import pandas as pd

# Exemplo com um DataFrame
dados = pd.DataFrame({'y': [1, 2, 3, 4, 5], 'x': [2, 4, 5, 4, 5]})

# Cria o modelo usando a sintaxe de fórmula
modelo_formula = smf.ols(formula='y ~ x', data=dados)

# Ajusta o modelo
resultados_formula = modelo_formula.fit()

# Imprime o resumo estatístico
print(resultados_formula.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.600
Model:                            OLS   Adj. R-squared:                  0.467
Method:                 Least Squares   F-statistic:                     4.500
Date:                Wed, 04 Jun 2025   Prob (F-statistic):              0.124
Time:                        14:10:42   Log-Likelihood:                -6.5368
No. Observations:                   5   AIC:                             17.07
Df Residuals:                       3   BIC:                             16.29
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -1.0000      1.955     -0.511      0.644      -7.222       5.222
x              1.0000      0.471      2.121      0.124      -0.500       2.500
==============================================================================
Omnibus:                        nan   Durbin-Watson:                   1.250
Prob(Omnibus):                  nan   Jarque-Bera (JB):                0.638
Skew:                         0.000   Prob(JB):                        0.727
Kurtosis:                     1.250   Cond. No.                         16.6
==============================================================================
```

Figura 12.1: Tabela de estatísticas de uma regressão linear geradas pela função ols do pacote statsmodels em Python.

Adj. R-squared (R^2 Ajustado): É uma versão do R^2 que considera tanto o número de variáveis independentes no modelo quanto o tamanho da amostra. Sua utilidade reside na comparação de modelos com diferentes números de preditores, pois ele penaliza a inclusão de variáveis que não contribuem significativamente para o poder explicativo do modelo. Por isso, o R-quadrado Ajustado é frequentemente mais confiável que o R-quadrado simples ao comparar a performance de modelos distintos.

F-statistic (Estatística F): É a estatística de teste para o teste F global do modelo. Ele serve para verificar se ao menos uma das variáveis independentes exerce um efeito estatisticamente significativo sobre a variável dependente. Para interpretar, um valor F alto, combinado com um P-valor baixo (o Prob (F-statistic)), indica que o modelo como um todo é estatisticamente significativo. Em outras palavras, as variáveis independentes, quando consideradas em conjunto, conseguem explicar uma parte significativa da variância observada na variável dependente.

Prob (F-statistic) (P-valor da Estatística F): É o P-valor associado à estatística F global do modelo. Sua interpretação é direta: se esse valor for menor que o nível de significância (geralmente 0.05), podemos concluir que o modelo, em sua totalidade, é estatisticamente significativo.

No. Observations (Número de Observações): É o número de linhas (amostras) usadas para treinar o modelo.

Df Residuals (Graus de Liberdade dos Resíduos): Correspondem ao número de observações subtraído do total de parâmetros estimados no modelo, incluindo o intercepto. Essencialmente, eles indicam quantos pontos de dados estão "livres para variar" após o modelo ter sido ajustado. O cálculo é dado por $n - k - 1$, onde n é o número de observações e k é o número de variáveis independentes (excluindo o intercepto). Essa métrica é fundamental para calcular a variância dos resíduos e para a realização de diversos testes estatísticos.

Df Model (Graus de Liberdade do Modelo): Representa o número de variáveis independentes incluídas no modelo, sem contar o intercepto. Essa medida é crucial, pois é utilizada no cálculo da estatística F, que avalia a significância global do modelo.

Covariance Type : Este item informa o tipo de matriz de covariância utilizada no cálculo dos erros padrão. Por padrão, o modelo usa o tipo "*nonrobust*" (não robusto), que pressupõe que os erros são homocedásticos (têm variância constante) e independentes. No entanto, há outras opções, como "HC0" e "HC1", que indicam o uso de erros padrão robustos, capazes de corrigir problemas como a heterocedasticidade (quando a variância dos erros não é constante).

AIC (do inglês *Akaike Information Criterion*) e BIC (do inglês *Bayesian Information Criter*
: São critérios de informação usados para comparar diferentes modelos. Valores menores de AIC e BIC geralmente indicam um modelo melhor, especialmente quando se compara modelos aninhados ou com diferentes números de variáveis. Penalizam modelos com mais parâmetros para evitar *overfitting*.

Log-Likelihood : É o logaritmo da função de verossimilhança máxima do modelo. Usado na construção de AIC e BIC. Um valor maior geralmente indica um modelo melhor.

A seção intermediária da tabela, conhecida como tabela de coeficientes (em inglês, *table of coefficients*), é a parte mais importante. Nela, são encontrados os valores para cada coeficiente preditor e o intercepto (coeficientes de regressão), acompanhados de métricas que indicam o grau de incerteza das estimativas e permitem avaliar a significância estatística dos efeitos:

coef (Coeficiente de regressão): Representa o valor estimado para o coeficiente de cada variável independente. A interpretação desse valor varia conforme a variável: para

o intercepto (const), ele representa o valor esperado da variável dependente quando todas as variáveis independentes são zero. Já para as demais variáveis, o coeficiente indica a mudança média na variável dependente para cada aumento de uma unidade na variável independente correspondente, assumindo que todas as outras variáveis permanecem constantes. Por exemplo, se o coeficiente para a variável "horas de estudo" for 0.5, isso significa que, em média, a nota esperada aumenta em 0.5 ponto para cada hora adicional de estudo, considerando que os demais fatores não se alterem.

std err (Erro Padrão): Representa a precisão da estimativa do coeficiente. É o desvio padrão da distribuição amostral do coeficiente. Um erro padrão pequeno indica que a estimativa do coeficiente é mais precisa. Ele é usado para calcular o estatístico t e o intervalo de confiança.

t (Estatística t): É a estatística de teste utilizada para verificar a hipótese nula de que o coeficiente (coef) é igual a zero, ou seja, que a variável independente não possui efeito significativo na variável dependente. Sua expressão é:

$$t = \frac{coef}{std\ err}. \tag{12.4}$$

Em termos mais simples, a estatística t quantifica quantos erros padrão o coeficiente está distante de zero. Quanto maior o valor absoluto de t, maior é essa distância em relação a zero (em unidades do erro padrão) sob a hipótese nula (H_0). Isso indica que é cada vez menos provável que o valor observado tenha ocorrido por acaso, caso a hipótese nula fosse verdadeira. Estatisticamente, esse cenário se traduz em um p-valor baixo, o que significa que a probabilidade de observar aquele valor (ou um valor mais extremo) sob H_0 é pequena.

P(>t) (P-valor ou Valor P): É a probabilidade de observar uma estatística t tão extrema (ou mais extrema) quanto a calculada, assumindo que a H_0 (coeficiente = 0) é verdadeira. Se P(>t) for menor que o nível de significância (geralmente 0.05), rejeitamos a hipótese nula. Isso significa que o coeficiente é estatisticamente significativo e a variável independente tem um efeito significativo na variável dependente. Se P(>t) for maior que o nível de significância, não rejeitamos a hipótese nula. Isso sugere que não há evidência suficiente para concluir que a variável independente tem um efeito significativo.

(0.025 0.975) (Intervalo de Confiança 95%): É o intervalo dentro do qual se espera que o verdadeiro valor do coeficiente da população caia, com um nível de confiança especificado (neste caso, 95%). Se o intervalo de confiança para um coeficiente não incluir zero, isso é consistente com um P-valor baixo e sugere que o coeficiente é estatisti-

camente significativo. Se o intervalo incluir zero, significa que o verdadeiro valor do coeficiente pode ser zero (ou seja, a variável não tem efeito), o que é consistente com um P-valor alto.

A seção inferior da tabela oferece informações importantes sobre os resíduos (ε), que nos permitem avaliar a validade do modelo em relação às suas suposições. Abordamos essas suposições brevemente na Seção 11.3, e aprofundaremos essa análise na Seção 12.2.

Omnibus / Prob(Omnibus) (Teste Omnibus): É o teste de normalidade dos resíduos. O teste Omnibus D'Angostino's é um teste estatístico que verifica se os resíduos seguem uma distribuição normal. Um P-valor baixo (menor que 0.05) sugere que os resíduos não são normalmente distribuídos.

Jarque-Bera (JB)/Prob(JB) : É o outro teste para normalidade dos resíduos. Similar ao Omnibus. Um P-valor baixo sugere não normalidade.

Skew (Assimetria): Essa estatística mede a assimetria da distribuição dos resíduos. Um valor próximo de 0 indica simetria. Valores positivos indicam uma cauda direita mais longa; valores negativos, uma cauda esquerda mais longa.

Kurtosis (Curtose): Essa estatística mede o "achatamento" da distribuição dos resíduos. Para uma distribuição normal, o valor da Curtose é próximo de 3 (ou próximo de 0, se for reportada a curtose excessiva). Valores maiores indicam que a distribuição tem "caudas" mais pesadas (ou seja, mais valores extremos) e um pico mais nítido, sendo essa característica conhecida como leptocúrtica.

Durbin-Watson : É usada para detectar autocorrelação nos resíduos do modelo, sendo particularmente relevante em análises de séries temporais. Seu valor varia de 0 a 4. Um resultado próximo de 2 indica ausência de autocorrelação, enquanto valores abaixo de 2 sugerem autocorrelação positiva, e valores acima de 2, autocorrelação negativa.

Cond. No. (Número de Condição): É uma métrica que avalia a multicolinearidade entre as variáveis independentes do seu modelo. Valores muito altos, geralmente acima de 10-30 (embora não haja um consenso rigoroso sobre o limite exato), sinalizam um problema de multicolinearidade severa. Isso pode tornar as estimativas dos coeficientes do modelo instáveis e, consequentemente, mais difíceis de interpretar de forma confiável.

12.2 Pressupostso para Validade de Regressão

Ajustar um modelo de regressão a uma amostra de dados, como explorado no Capítulo 11, é apenas o primeiro passo. Avaliar a qualidade do modelo é fundamental para garantir seu ajuste adequado e a validade das inferências derivadas. O **processo de modelagem é iterativo**, exigindo ajustes e reavaliações constantes, sempre buscando o modelo mais simples e eficaz para representar a realidade observada. As condições para inferência são ferramentas essenciais na avaliação de modelos. Nem todo modelo ajustado gera resultados adequados para inferência estatística ou predição. Ignorar certos pressupostos pode comprometer a validade de testes, intervalos de confiança e a capacidade de generalização do modelo.

O conjunto de condições apresentado na Seção 11.3, ausência de multicolinearidade, linearidade e homocedasticidade, oferece uma visão preliminar focada nos aspectos que afetam diretamente a qualidade do ajuste (em inglês, *fit*). Também preocupados com a qualidade do ajuste de um modelo, Ismay et al. [36] propuseram o acrônimo LINE para lembrar os quatro elementos essenciais a serem checados em um modelo linear:

- **L**inearidade da relação entre as variáveis: A relação entre a variável dependente (y_i) e a independente (x_i) é de fato linear?
- **I**ndependência dos valores da resposta (y_i) e dos termos de erro (ϵ_i): Cada observação é independente das outras? Para verificar isso, usamos os resíduos se os dados tiverem uma sequência (tempo, etc.); caso contrário, a independência é geralmente garantida por uma amostragem aleatória.
- **N**ormalidade dos termos de erro: A distribuição dos termos de erro é aproximadamente normal?
- **E**quidade ou constância da variância para y_i: A variância da resposta é sempre a mesma, independentemente do valor previsto ($\hat{y}_i$) ou do valor do regressor (x_i)?

Ismay et al. ainda enfatizaram que a linearidade, normalidade e a constância da variância são verificadas por meio do diagnóstico dos resíduos da regressão linear.

No entanto, quando o objetivo é realizar **inferências estatísticas confiáveis** e construir modelos robustos para **predição**, torna-se necessário considerar um conjunto mais amplo de pressupostos que garantem a aplicação correta do método dos mínimos quadrados. No contexto da regressão linear, destacam-se os seguintes:

Linearidade: A relação entre as variáveis deve ser linear nos parâmetros. Isso significa que o modelo está corretamente especificado e que não há omissão de variáveis relevantes ou inclusão de termos desnecessários.

Independência dos Erros: Os resíduos devem ser independentes entre si, ou seja, o erro associado a uma observação não deve fornecer nenhuma informação sobre o erro de outra observação. Esse pressuposto é especialmente importante em dados temporais ou espaciais, onde pode ocorrer padrões temporais, regionais e sazonais (autocorrelação).

Homocedasticidade: A variância dos erros deve ser constante ao longo de todas as observações. A violação desse pressuposto compromete a validade dos erros padrão, dos testes t, F e dos intervalos de confiança.

Normalidade dos Erros: Para que os testes estatísticos (t e F) e os intervalos de confiança sejam válidos, assume-se que os erros seguem uma distribuição normal, especialmente em amostras pequenas.

Ausência de Multicolinearidade: As variáveis independentes não devem estar altamente correlacionadas, ou seja, uma variável não deve ser uma combinação linear quase exata de outras. Isso afeta a precisão e a estabilidade das estimativas dos coeficientes.

Generalização: O modelo deve ser capaz de extrapolar para novos dados de maneira apropriada, evitando ajustes excessivos aos dados observados (fenômeno conhecido como *overfitting*).

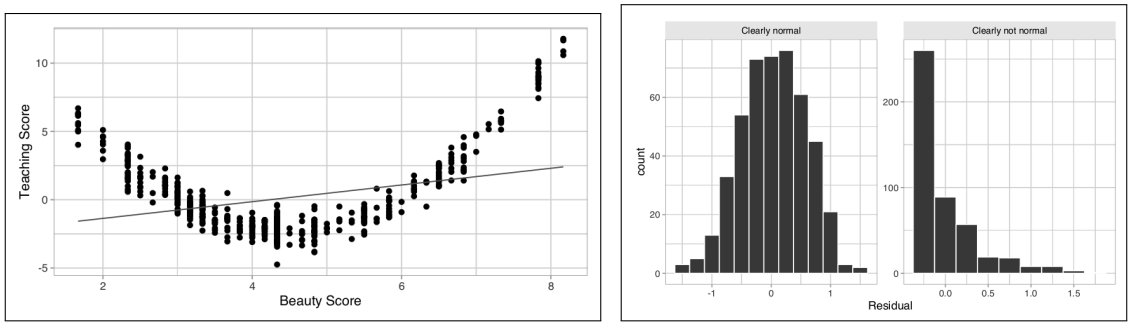
Embora os valores numéricos provenientes da análise, como estatísticas t, valores-p, intervalos de confiança e R^2, forneçam informações importantes, eles não são suficientes, por si só, para garantir a validade do modelo. Avaliar a qualidade do modelo exige uma abordagem mais completa, onde recursos gráficos são fundamentais. Eles nos permitem identificar padrões, desvios e problemas, incluindo o atendimento dos pressupostos do modelo, que nem sempre são óbvios nas tabelas estatísticas.

Histogramas de resíduos, gráficos quantil-quantil (Q-Q plots), matrizes de dispersão e diagramas de influência são ferramentas essenciais para diagnosticar modelos de regressão, auxiliando a:

- identificar violações de linearidade, como ilustra o gráfico de dispersão na Figura 12.2a;
- avaliar a normalidade dos erros, conforme demonstram o histograma na Figura 12.2b, e os gráficos de dispersão, histograma e Q-Q apresentados na Figura 12.3;
- identificar *outliers*, que são claramente visíveis nos gráficos de dispersão, histograma e Q-Q apresentados na Figura 12.3;
- detectar a presença de heterocedasticidade, como ilustram os gráficos de dispersão na Figura 12.4;
- sinalizar indícios de autocorrelação dos resíduos, conforme exemplificado pelos gráficos de dispersão na Figura 12.5;

- identificar multicolinearidade entre variáveis explicativas, como mostram a matriz de dispersão na Figura 12.6 e o mapa de calor das correlações na Figura 12.7; e
- reconhecer pontos de alta influência, como ilustra a Figura 12.8. Esse tópico será abordado em mais detalhes adiante.

Assim, o processo de ajuste de um modelo de regressão deve ser entendido como uma **etapa dentro de um ciclo iterativo**, no qual a análise dos dados, a verificação dos pressupostos e eventuais reespecificações do modelo são fundamentais para assegurar que as conclusões obtidas sejam não apenas estatisticamente válidas, mas também coerentes com a realidade observada.



(a) Linearidade (b) Normalidade dos resíduos

Figura 12.2: Avaliação da linearidade das relações por meio dos gráficos de dispersão de resíduos × valores ajustados, e da normalidade na distribuição dos resíduos por meio de histogramas. Quando a relação entre as variáveis não é linear, os termos não lineares (f(X)) se manifestarão diretamente no gráfico de resíduos. Isso ocorre porque o modelo de regressão linear ($Valor\ ajustado = \beta_0 + \beta_1 X$) não consegue capturar essa curvatura, fazendo com que a parte não linear f(X) seja absorvida pelo novo termo de erro ($\varepsilon\prime = f(X) + \varepsilon$). Consequentemente, os resíduos exibirão um padrão sistemático, revelando a não linearidade não explicada pelo modelo. (Fonte: [36])

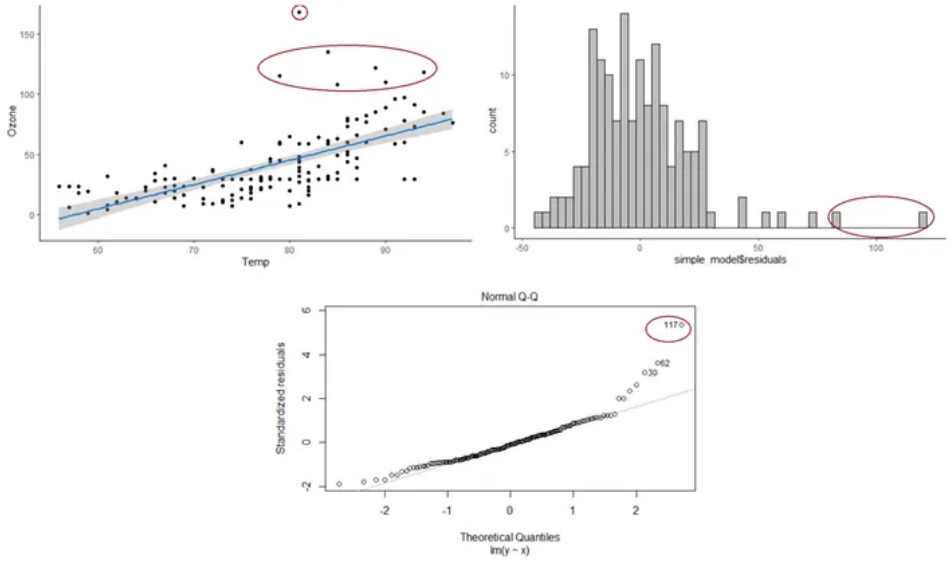
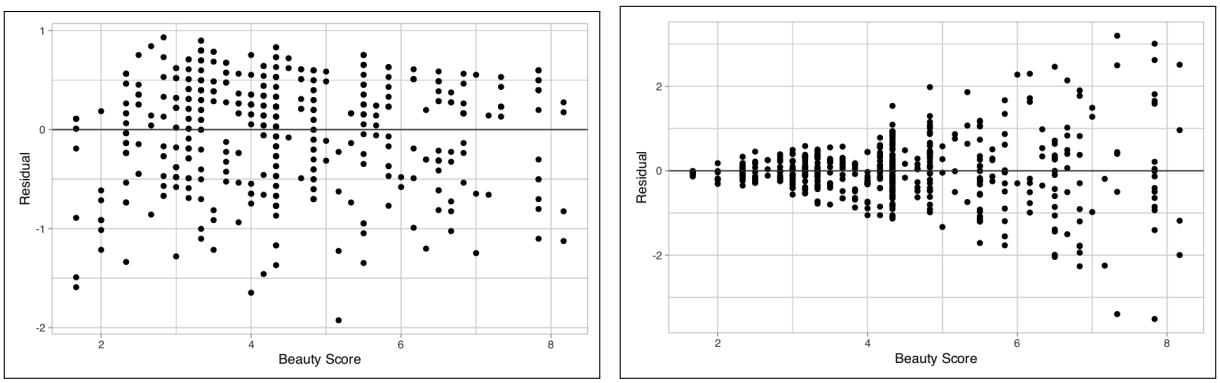


Figura 12.3: Avaliação da normalidade da distribuição de resíduos e identificação dos *outliers* (destacados por elipses vermelhas) através de um gráfico de dispersão (superior direito), histograma (superior esquerdo) e gráfico Q-Q (inferior). (Fonte)



(a) Homocedasticidade (b) Heterocedasticidade

Figura 12.4: Avaliação da variabilidade dos resíduos por meio de gráficos de dispersão, nos quais a linha de regressão foi representada como uma linha horizontal alinhada a $y = 0$: (a) A dispersão de cada resíduo ($e_i = y_i - \hat{y}_i$) em torno da linha é aproximadamente constante ao longo dos valores da variável explicativa x quando a suposição de homocedasticidade é válida, isto é, quando a variabilidade dos erros não depende dos valores de x. (b) A presença de heterocedasticidade é identificada quando os resíduos apresentam variância não constante ao longo de x, ou seja, a dispersão dos resíduos aumenta ou diminui sistematicamente conforme os valores da variável explicativa x variam. Isso se manifesta graficamente como um padrão de funil (abertura ou fechamento) ou outros padrões estruturados na dispersão dos pontos. (Fonte: [36])

Os **gráficos de influência** (em inglês, *influence plots*) ajudam avaliar a robustez de um modelo de regressão. Eles identificam o impacto de pontos individuais nos resultados do modelo, mostrando como a remoção de cada observação pode afetar as estimativas dos coeficientes de regressão e dos resíduos. Cada ponto no gráfico representa uma observação do conjunto de dados, e sua posição indica o nível de influência dessa observação. Esses gráficos não apenas ajudam a detectar pontos que podem distorcer os resultados, mas também fornecem *insights* sobre a confiabilidade das inferências feitas a partir do modelo.

Para criar um gráfico de influência, começamos ajustando o modelo de regressão aos dados. Em seguida, calculamos medidas de influência para cada observação, como o índice de alavancagem (*leverage*), que quantifica o quanto uma observação pode influenciar as estimativas dos parâmetros do modelo. Depois, plotamos essas medidas em relação a uma métrica de interesse, como os resíduos padronizados ou os coeficientes de regressão. Pontos que estão distantes do centro ou que se destacam em relação aos demais podem indicar observações com alta alavancagem ou alta influência sobre o modelo. A Figura 12.8 exemplifica os pontos mais influentes, mostrando aqueles com resíduos padronizados menores que 0,02 na estimativa da relação linear entre a variável dependente childHeight e a variável independente father.

Tanto R quanto Python oferecem suporte para a criação de gráficos de influência em análises de regressão. Em Python, a biblioteca statsmodels é uma excelente ferramenta para isso. Especificamente, a função plot_leverage_resid2() do módulo statsmodels.graphics.regressionplots pode ser usada para gerar um gráfico de influência. Veja um exemplo de como utilizar essa função para plotar o gráfico exibido

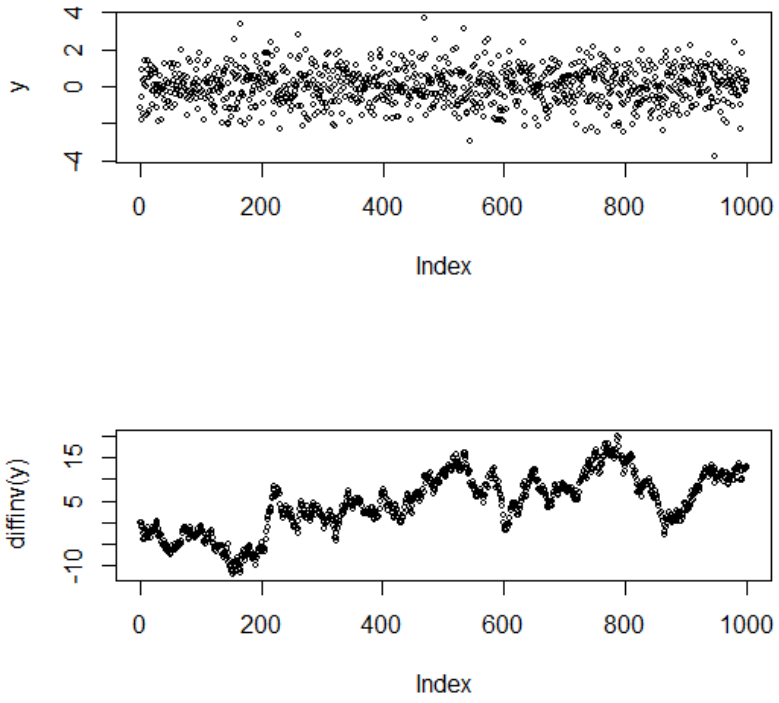


Figura 12.5: Avaliação de autocorrelação dos resíduos através de gráficos de dispersão (correlogramas) para identificar a presença de periodicidade nos resíduos de um modelo estatístico. O objetivo é que os resíduos sejam o mais próximo possível de um "ruído branco" (acima), ou seja, que sejam independentes e sem nenhum padrão sistemático, incluindo periodicidade. Picos significativos em *lags* específicos (abaixo) é um forte indicativo de que há uma periodicidade não capturada pelo modelo original, pois o termo não linear ou periódico $f(X)$, não contemplado pelo modelo linear ($Valor\ ajustado = \beta_0 + \beta_1 X$), manifesta-se através dos resíduos como $\varepsilon\prime = f(X) + \varepsilon$. (Fonte)

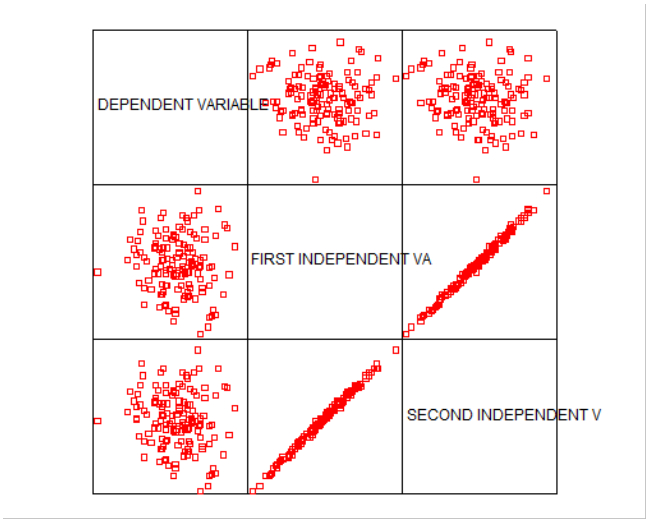


Figura 12.6: Avaliação de multicolinearidade por meio de uma matriz de dispersão que permite analisar a colinearidade entre todas as variáveis, tanto em pares quanto entre uma variável dependente e todas as independentes, podendo sugerir a remoção de variáveis independentes de baixa colinearidade no modelo.

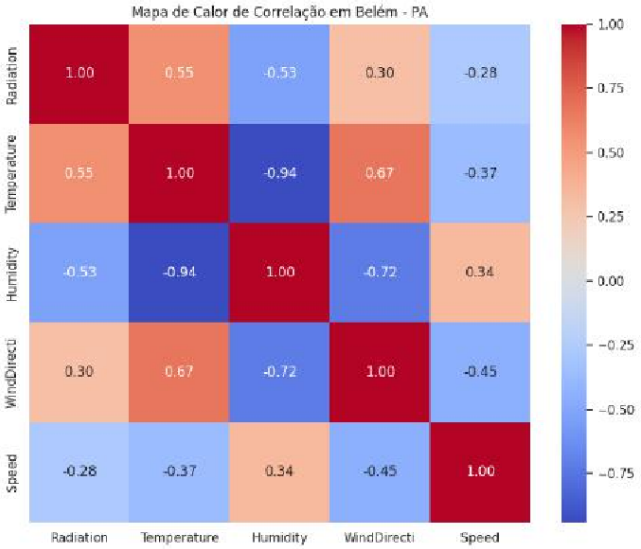


Figura 12.7: Identificação de multicolinearidade por meio de um mapa de calor que exibe visualmente a **matriz de correlação**, revelando os coeficientes de correlação entre as variáveis, par a par.

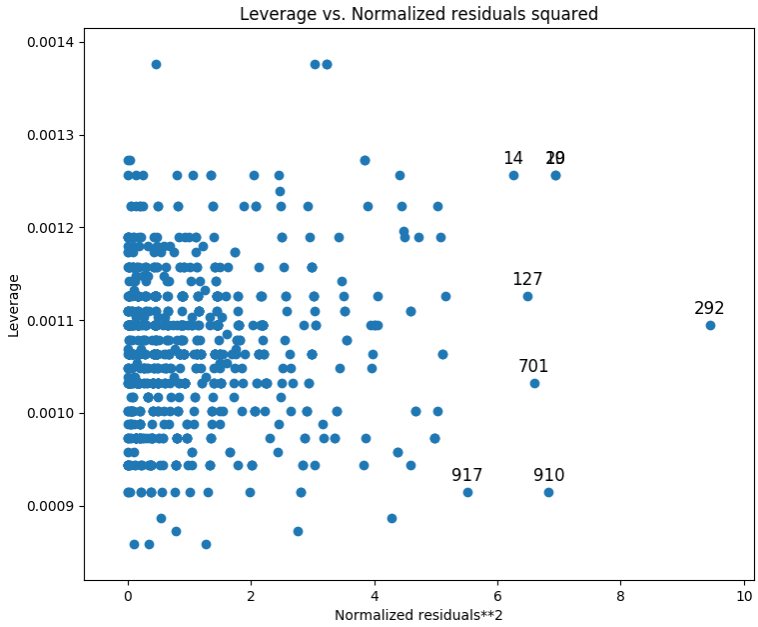


Figura 12.8: Gráfico de influência dos dados de GaltonFamilies na relação entre a variável dependente childHeight e a variável independente father, enfatizando as observações com resíduos menores que 0.02.

na Figura 12.8:

```
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt

df = pd.read_csv('/home/ting/Documents/cursos/ia376/apostila/datasets/GaltonFamilies.csv')
y = df['childHeight']
X = df['father']
```

```
modelo = sm.OLS(y,X).fit()
sm.graphics.plot_leverage_resid2(modelo, alpha=0.02)
```

Em R, pode-se usar a função influence.plot() do pacote car para plotar um gráfico de influência em análises de regressão. Segue-se uma versão em R para plotar o gráfico de influência da Figura 12.8:

```
library(readr)
library(ggplot2)
library(dplyr)
library(tidyr)
library(car)

# Carregar o conjunto de dados
df <- read_csv("/caminho/da/pasta/GaltonFamilies.csv")

# Renomear as colunas para se ajustar à sintaxe do R (opcional)
names(df) <- c("family", "father", "mother", "midparentHeight", "children", "childNum", "gende

# Ajustar o modelo de regressão
modelo <- lm(childHeight ~ father, data = df)

# Gráfico de influência
influencePlot(modelo, id.n=5, main="Gráfico de Influência")
```

12.3 Inferência Analítica

Nesta seção, vamos nos aprofundar na fundamentação matemática da inferência analítica em modelos de regressão amostral. Nosso objetivo é entender como são obtidos os coeficientes de regressão (para cada preditor e o intercepto), juntamente com as métricas que quantificam a incerteza dessas estimativas e permitem avaliar a significância estatística dos efeitos, conforme exemplificado na seção intermediária da tabela da Figura 12.1. Apresentaremos as fórmulas matemáticas desenvolvidas em uma era pré-computacional, quando as extensas simulações computacionais, que exploraremos na Seção 12.4, eram inviáveis. Historicamente, esses métodos de inferência em regressão frequentemente partem do coeficiente de correlação populacional (ρ), que pode ser calculado a partir da Equação 6.9.

Na prática, como normalmente não se dispõe de dados de toda a população, ρ é estimado pelo coeficiente de correlação amostral, denotado por r. Esse coeficiente é calculado com base em uma amostra, utilizando a mesma lógica de ρ, porém ajustado para amostras:

$$r = \frac{\text{Cov}(X, Y)}{s_X s_Y},$$

onde s_X e s_Y são os desvios padrões das variáveis na amostra. Como qualquer estatística amostral, r apresenta variabilidade: ao repetirmos a coleta de amostras, os valores de r obtidos formarão uma distribuição que tende a ser aproximadamente normal quando o tamanho amostral é suficientemente grande. A média dessa distribuição amostral se aproxima de ρ, o verdadeiro coeficiente populacional. A Figura 12.9 ilustra a distribuição de valores amostrais de r entre as variáveis ChildHeight e FatherHeight do conjunto de dados GaltonFamilies, considerando um tamanho de amostra de $n = 25$ [70].

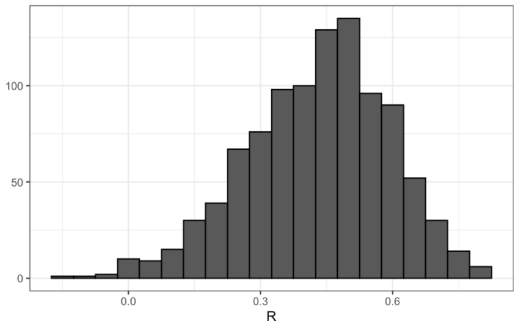


Figura 12.9: Histograma de correlações entre duas variáveis. (Fonte: [70])

A variabilidade nas estimativas de um parâmetro populacional pode ser quantificada por meio do erro padrão, que mede a dispersão das estimativas ao redor do valor verdadeiro. O erro padrão (std err na Figura 12.1) do coeficiente de correlação, por exemplo, pode ser aproximadamente calculado pela seguinte fórmula [70]:

$$SE_r = \sqrt{\frac{1 - r^2}{n - 2}}, \tag{12.5}$$

onde r é o coeficiente de correlação da amostra entre as variáveis da amostra e n é o tamanho da amostra. O denominador $n - 2$ indica que o erro padrão SE_r tende a diminuir à medida que n aumenta, resultando em estimativas de r mais precisas.

O **erro padrão do coeficiente angular da amostra**, denotado como SE_{b_1}, representa a variação esperada de b_1 entre diferentes amostras. Sua fórmula é [36]

$$SE_{b_1} = \frac{\frac{s_y}{s_x}\sqrt{1 - r^2}}{\sqrt{n - 2}} = \frac{s_y}{s_x} SE_r \tag{12.6}$$

onde s_y e s_x são os desvios padrão das variáveis dependente e independente, respectivamente. Já o **erro padrão do intercepto da amostra** (std err), por sua vez, indica quão precisas são as estimativas do intercepto e expressa a incerteza na estimativa de b_0. A fórmula é

$$SE_{b_0} = \sqrt{\frac{\sum_{i=1}^{n} e_i^2}{n - 2}} \cdot \sqrt{\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2}}, \tag{12.7}$$

onde e_i é o resíduo de i-ésima observação, $\bar{x}$ é a média das variáveis explicativas x, e x_i é o valor explicativo de i-ésima observação.

Tanto o intercepto b_0 quanto a inclinação (coeficiente de regressão) b_1 são estimativas sujeitas à incerteza, uma vez que são obtidas a partir de dados amostrais. Para expressar essa incerteza, podemos construir intervalos de confiança. Um intervalo de confiança de 95% para b_i pode ser interpretado de duas formas equivalentes:

1. O intervalo contém os valores que não seriam rejeitados em um teste de hipótese com nível de significância de 5%;

2. Há uma probabilidade de 95% de que o intervalo contenha o verdadeiro valor de β_i.

Isso significa que, se repetirmos o processo de coleta de amostras muitas vezes, em 95% das amostras, o intervalo de confiança incluirá o valor verdadeiro de b_i. Dizemos também que esse intervalo tem um nível de confiança de 95%. Isso nos ajuda a entender a precisão de nossas estimativas e a incerteza associada a elas. Esses intervalos são calculados usando o erro padrão e uma estatística da distribuição b_i. Considerando que o tamanho da amostra seja suficientemente grande para que a distribuição t se aproxime da normal padrão, os intervalos de confiança de 95% para β_0 e β_1 são dados pelas expressões:

$$\beta_1 \pm \text{MOE}_{b_1} = b_1 \pm 1.96 \cdot SE_{b_1} \tag{12.8}$$

$$\beta_0 \pm \text{MOE}_{b_0} = b_0 \pm 1.96 \cdot SE_{b_0} \tag{12.9}$$

onde MOE é a margem de erro (em inglês, *Margin of Error*).

Para testar a hipótese nula de que um coeficiente de regressão populacional β_i é igual a zero, usamos a estatística t, que segue uma distribuição t de Student com $n-2$ graus de liberdade:

$$t = \frac{b_i - 0}{SE_{b_i}} = \frac{b_i}{SE_{b_i}}. \tag{12.10}$$

Essa estatística mede o quão distante está a estimativa amostral do valor hipotético. No contexto do *software* R, assume-se β_i sob a hipótese nula, e a estatística t é usada para calcular o p-valor, permitindo testar a significância do coeficiente. O p-valor é obtido a partir da distribuição t de Student com $n-2$ graus de liberdade, que é a distribuição nula da estatística t quando os pressupostos do modelo linear são atendidos. A tabela de distribuição t de Student é mostrada na Figura 12.10.

12.4 Inferência Computacional

Em contraste com a inferência analítica, que se baseia em fórmulas matemáticas e distribuições teóricas bem definidas, a inferência computacional adota uma abordagem alternativa, fundamentada em al-

t Table

cum. prob	$t_{.50}$	$t_{.75}$	$t_{.80}$	$t_{.85}$	$t_{.90}$	$t_{.95}$	$t_{.975}$	$t_{.99}$	$t_{.995}$	$t_{.999}$	$t_{.9995}$
one-tail	0.50	0.25	0.20	0.15	0.10	0.05	0.025	0.01	0.005	0.001	0.0005
two-tails	1.00	0.50	0.40	0.30	0.20	0.10	0.05	0.02	0.01	0.002	0.001
df											
1	0.000	1.000	1.376	1.963	3.078	6.314	12.71	31.82	63.66	318.31	636.62
2	0.000	0.816	1.061	1.386	1.886	2.920	4.303	6.965	9.925	22.327	31.599
3	0.000	0.765	0.978	1.250	1.638	2.353	3.182	4.541	5.841	10.215	12.924
4	0.000	0.741	0.941	1.190	1.533	2.132	2.776	3.747	4.604	7.173	8.610
5	0.000	0.727	0.920	1.156	1.476	2.015	2.571	3.365	4.032	5.893	6.869
6	0.000	0.718	0.906	1.134	1.440	1.943	2.447	3.143	3.707	5.208	5.959
7	0.000	0.711	0.896	1.119	1.415	1.895	2.365	2.998	3.499	4.785	5.408
8	0.000	0.706	0.889	1.108	1.397	1.860	2.306	2.896	3.355	4.501	5.041
9	0.000	0.703	0.883	1.100	1.383	1.833	2.262	2.821	3.250	4.297	4.781
10	0.000	0.700	0.879	1.093	1.372	1.812	2.228	2.764	3.169	4.144	4.587
11	0.000	0.697	0.876	1.088	1.363	1.796	2.201	2.718	3.106	4.025	4.437
12	0.000	0.695	0.873	1.083	1.356	1.782	2.179	2.681	3.055	3.930	4.318
13	0.000	0.694	0.870	1.079	1.350	1.771	2.160	2.650	3.012	3.852	4.221
14	0.000	0.692	0.868	1.076	1.345	1.761	2.145	2.624	2.977	3.787	4.140
15	0.000	0.691	0.866	1.074	1.341	1.753	2.131	2.602	2.947	3.733	4.073
16	0.000	0.690	0.865	1.071	1.337	1.746	2.120	2.583	2.921	3.686	4.015
17	0.000	0.689	0.863	1.069	1.333	1.740	2.110	2.567	2.898	3.646	3.965
18	0.000	0.688	0.862	1.067	1.330	1.734	2.101	2.552	2.878	3.610	3.922
19	0.000	0.688	0.861	1.066	1.328	1.729	2.093	2.539	2.861	3.579	3.883
20	0.000	0.687	0.860	1.064	1.325	1.725	2.086	2.528	2.845	3.552	3.850
21	0.000	0.686	0.859	1.063	1.323	1.721	2.080	2.518	2.831	3.527	3.819
22	0.000	0.686	0.858	1.061	1.321	1.717	2.074	2.508	2.819	3.505	3.792
23	0.000	0.685	0.858	1.060	1.319	1.714	2.069	2.500	2.807	3.485	3.768
24	0.000	0.685	0.857	1.059	1.318	1.711	2.064	2.492	2.797	3.467	3.745
25	0.000	0.684	0.856	1.058	1.316	1.708	2.060	2.485	2.787	3.450	3.725
26	0.000	0.684	0.856	1.058	1.315	1.706	2.056	2.479	2.779	3.435	3.707
27	0.000	0.684	0.855	1.057	1.314	1.703	2.052	2.473	2.771	3.421	3.690
28	0.000	0.683	0.855	1.056	1.313	1.701	2.048	2.467	2.763	3.408	3.674
29	0.000	0.683	0.854	1.055	1.311	1.699	2.045	2.462	2.756	3.396	3.659
30	0.000	0.683	0.854	1.055	1.310	1.697	2.042	2.457	2.750	3.385	3.646
40	0.000	0.681	0.851	1.050	1.303	1.684	2.021	2.423	2.704	3.307	3.551
60	0.000	0.679	0.848	1.045	1.296	1.671	2.000	2.390	2.660	3.232	3.460
80	0.000	0.678	0.846	1.043	1.292	1.664	1.990	2.374	2.639	3.195	3.416
100	0.000	0.677	0.845	1.042	1.290	1.660	1.984	2.364	2.626	3.174	3.390
1000	0.000	0.675	0.842	1.037	1.282	1.646	1.962	2.330	2.581	3.098	3.300
z	0.000	0.674	0.842	1.036	1.282	1.645	1.960	2.326	2.576	3.090	3.291
	0%	50%	60%	70%	80%	90%	95%	98%	99%	99.8%	99.9%
	Confidence Level										

Figura 12.10: Tabela de distribuição t de Student usada para consultar o p-valor correspondente à estatística de teste.

goritmos e simulações para estimar parâmetros e avaliar incertezas. Essa abordagem é especialmente útil em modelos complexos, como regressão não linear, ou em contextos com grandes volumes de dados, nos quais as soluções analíticas se tornam inviáveis ou excessivamente difíceis. Além disso, conforme destacam Ismay et al. [36], a inferência computacional oferece uma maneira mais intuitiva de compreender conceitos estatísticos fundamentais.

Entre os principais métodos computacionais destacam-se:

Simulação de Monte Carlo (Seção 8.6): utilizada para gerar distribuições amostrais de estimadores e avaliar a variabilidade por meio de simulações repetidas;

Bootstrap (Seção 9.2.4): técnica que permite reamostrar os dados com reposição, es-

timando a distribuição de um estimador sem depender de suposições paramétricas sobre a população original.

No contexto da regressão linear univariada, busca-se estimar o coeficiente de correlação $\hat{\rho}$ e os coeficientes de regressão, $\hat{\beta}_0$ (intercepto) e $\hat{\beta}_1$ (coeficiente angular), a partir de uma amostra com n observações. O procedimento básico consiste em gerar uma distribuição amostral por meio de N simulações de regressões lineares. Cada simulação produz um triplo ($\hat{\rho}_i$,$\hat{\beta}_{0_i}$, $\hat{\beta}_{1_i}$), obtido aplicando o método *bootstrap* à amostra original de tamanho n. O resultado final é um conjunto de N estimativas, cujas distribuições podem ser visualizadas, por exemplo, na forma de histogramas. A Figura 12.11 ilustra um histograma com 1.000 valores do coeficiente angular gerados por esse processo.

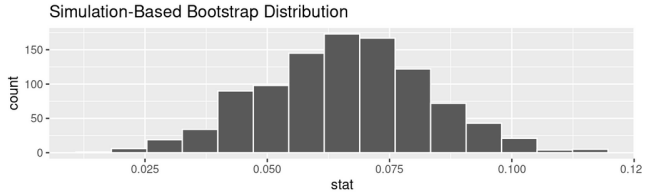


Figura 12.11: Distribuição *bootstrap* de coeficientes angulares obtida com simulações Monte Carlo. (Fonte: [36])

De posse dessas distribuições amostrais, a Seção 9.3.2 apresenta dois métodos para construção de intervalos de confiança a um nível de confiança $(1-\alpha)\times 100\% \in [0,1]$:

Método dos percentis : consiste em determinar os percentis correspondentes a $\frac{\alpha}{2} \times 100\%$ e $(1-\frac{\alpha}{2}) \times \%$ da distribuição empírica gerada pelas simulações. É um método totalmente não paramétrico, que não faz suposições sobre a forma da distribuição dos estimadores.

Método do erro padrão : utiliza a média das estimativas obtidas, seu erro padrão (calculado a partir das simulações) e um valor crítico da distribuição normal ou t (dependendo do contexto) para calcular os limites do intervalo de confiança. Este método aproxima o raciocínio analítico tradicional.

Ismay et al. [36] mostram, por meio de um exemplo com uma amostra de 463 observações, que os resultados dos métodos computacionais (percentis e erro padrão) são bastante semelhantes aos resultados do método analítico clássico. A Figura 12.12 apresenta essa comparação, onde os três métodos geram intervalos praticamente coincidentes.

Para testar a hipótese nula de que não há relação entre as variáveis ($\beta_1 = 0$), aplica-se o teste de permutação, descrito na Seção 10.8, sobre uma amostra original de 463 observações [36]. Esse teste consiste em embaralhar aleatoriamente os valores da variável explicativa x, rompendo qualquer associação com a variável dependente y. Para cada permutação, calcula-se o coeficiente angular, gerando uma distribuição dos coeficientes sob a hipótese nula de ausência de relação entre as variáveis

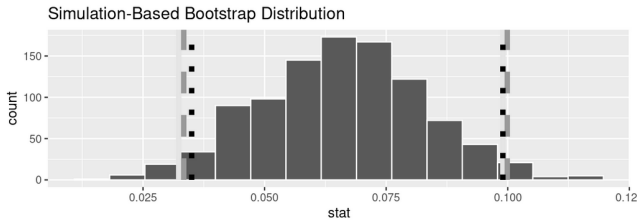


Figura 12.12: Comparação entre os intervalos de confiança calculados pelo método teórico (linha pontilada), método dos percentis (linha cheia) e método de erro padrão (linha tracejada). (Fonte: [36])

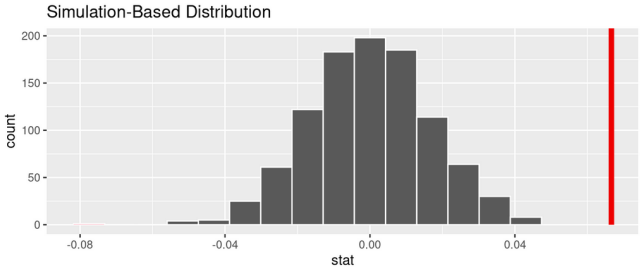


Figura 12.13: Distribuição amostral para um cenário sob hipótese nula e o p-valor da estatística de teste, que é a média da amostra original. (Fonte: [36])

(coeficiente igual a 0). A Figura 12.13 mostra a distribuição de 1.000 coeficientes angulares obtidos por esse procedimento, além do valor do coeficiente observado na amostra real ($\hat{\beta}_1 = 0.067$). O valor-p associado é praticamente zero, o que fornece fortes evidências contra a hipótese nula, indicando que existe uma relação estatisticamente significativa entre as variáveis.

12.5 Procedimento de Construção de um Modelo Estatístico de Regressão

O desenvolvimento de um modelo de regressão preditiva é um processo iterativo, que envolve várias etapas interdependentes, fundamentais para assegurar a validade estatística e a utilidade prática dos resultados. Estas etapas foram discutidas nos Capítulos 12 e **??**. A seguir, apresenta-se uma síntese estruturada do procedimento:

Formulação da Pergunta de Pesquisa: Definir claramente o problema a ser investigado. Isso inclui especificar a variável dependente (ou resposta) que se deseja modelar e as variáveis independentes (ou preditoras) que se acredita influenciarem esse desfecho.

Coleta de Dados: Obter dados adequados e relevantes, seja por meio de experimentos, levantamentos, bases de dados públicas ou privadas, ou outros meios. A qualidade dos dados é determinante para a robustez do modelo.

Análise Exploratória de Dados (EDA) e Pré-Processamento: Explorar os dados

para compreender padrões, tendências, anomalias e relações preliminares. Isso inclui:

- Visualização de distribuições e relações;
- Identificação de *outliers* e valores ausentes;
- Avaliação da linearidade e outras suposições;
- Codificação de variáveis categóricas, normalização e seleção de variáveis relevantes.

Seleção do Modelo: Escolher o tipo de modelo mais adequado à natureza da variável resposta e às características dos dados. Isso pode incluir:

- Regressão linear simples ou múltipla (para variáveis resposta contínuas);
- Modelos não lineares ou polinomiais (quando apropriado);
- Modelos generalizados, como regressão logística (para variável resposta binária), quando a hipótese de regressão linear não é válida.

Estimação dos Parâmetros: Ajustar o modelo aos dados, estimando os coeficientes que melhor descrevem a relação entre as variáveis, geralmente pelo método dos mínimos quadrados ou por métodos computacionais como máxima verossimilhança.

Avaliação e Validação do Modelo: Verificar a qualidade do modelo por meio de:

- Avaliação do ajuste (coeficiente de determinação, resíduos, suposição de homocedasticidade, normalidade, linearidade);
- Avaliação da significância estatística dos coeficientes;
- Medidas de desempenho preditivo (como erro quadrático médio);
- Validação cruzada, *bootstrap* ou divisão treino-teste, para avaliar a generalização do modelo.

Interpretação e Comunicação dos Resultados: Interpretar os coeficientes, analisar os impactos das variáveis preditoras e comunicar os resultados de forma clara, precisa e alinhada ao público-alvo, seja técnico ou leigo. A interpretação deve considerar tanto o contexto estatístico quanto o contexto aplicado do problema.

12.6 Considerações Finais

Neste capítulo, abordamos a construção de modelos estatísticos de regressão preditiva, destacando sua importância na compreensão das relações entre variáveis a partir de dados amostrais. Iniciamos pela formalização dos modelos matemáticos, estendendo a equação determinística para incorporar componentes aleatórios. Essa é uma etapa essencial para representar a variabilidade inerente aos dados e fundamentar a inferência estatística.

Discutimos em detalhe os pressupostos que sustentam a validade dos modelos de regressão linear, como linearidade, independência dos erros, normalidade dos resíduos e homocedasticidade. A verificação desses pressupostos foi enfatizada por meio de técnicas de análise gráfica, utilizando ferramentas como gráficos de resíduos, gráficos de influência e mapas de calor, fundamentais tanto para diagnosticar desvios dos pressupostos quanto para aprimorar a qualidade do modelo.

Na etapa de inferência, exploramos duas abordagens complementares. A inferência analítica, baseada em métodos tradicionais que utilizam distribuições teóricas para estimativas e testes de hipóteses, e a inferência computacional, que emprega técnicas como simulação de Monte Carlo e *bootstrap*. Esta última se mostra especialmente útil quando os pressupostos teóricos não são totalmente atendidos ou quando se trabalha com modelos mais complexos. Pois, o *bootstrap*, por ser uma técnica baseada em reamostragem direta dos dados observados, não exige especificações sobre a distribuição teórica subjacente. Assim, ele constrói a distribuição empírica dos estimadores diretamente da amostra.

Ressaltamos ainda a importância da validação rigorosa dos modelos, etapa fundamental para garantir a capacidade preditiva em novos contextos. Embora métodos computacionais como o *bootstrap* e Monte Carlo sejam valiosos para esse fim, destacamos também o papel crescente das abordagens de Inteligência Artificial e Aprendizado de Máquina, que permitem capturar relações não lineares e padrões em grandes volumes de dados.

Por fim, reconhecemos as limitações dos modelos lineares e incentivamos a exploração de técnicas estatísticas mais avançadas, como regressão polinomial, regressão logística, modelos baseados em árvores (como árvores de decisão e florestas aleatórias) e redes neurais artificiais. A adoção de uma abordagem metodológica diversificada não apenas amplia a capacidade de modelagem preditiva, mas também oferece maior robustez e aplicabilidade dos modelos em diferentes cenários do mundo real.

12.7 Exercícios

1. Faça os exercícios de aprendizado LC10.1 e LC10.2 em [36] relacionados à **modelagem de regressão preditiva**. Verifique suas respostas no apêndice D do mesmo livro.

2. Faça os exercícios de aprendizado LC11.1 e LC11.2 relacionados à **modelagem de regressão** em [36]. Verifique suas respostas no apêndice D do mesmo livro.

3. É possível inferir o volume de uma árvore a partir de suas medidas de altura e/ou circunferência ($=\pi \times$ diâmetro) do tronco? O site apresenta um procedimento para a construção de um modelo de regressão preditiva que busca responder essa questão. O procedimento utiliza o conjunto de dados Tree, onde a variável dependente é o volume da árvore, e as variáveis preditivas são a altura e a circunferência (ou diâmetro) do tronco, em R.

   (a) Qual é a abordagem de inferência adotada pelo autor?

(b) Implemente esse procedimento em Python.

(c) Estime o coeficiente angular da circunferência ou do diâmetro das árvores em relação ao volume da árvore (*Slope1*) pela abordagem alternativa.

(d) Teste a hipótese nula de que não há nenhuma relação entre a circunferência, ou o diâmetro, e o volume da árvore (inclinação angular nula), usando a abordagem de inferência computacional.

4. Há alguma relação entre pesos e alturas numa população de mulheres? Vamos construir um modelo linear simples para prever o peso a partir da altura de uma mulher, utilizando o conjunto de dados women_df, que apresenta a relação entre a altura (*height*) e o peso (*weight*) de 15 mulheres. Siga as seguintes etapas:

   (a) Análise exploratória de dados para identificar as relações presentes nos dados

      i. Faça um gráfico de dispersão da altura vs peso.

      ii. Adicione a linha de regressão ajustada.

   (b) Ajuste do modelo de regressão linear para obter um modelo de regressão linear. Analise os coeficientes de regressão e intercepto.

      i. Ajuste o modelo weight $\sim$ height.

   (c) Verificação dos pressupostos via visualização: Crie os gráficos diagnósticos padrão e verifique linearidade, normalidade e homocedasticidade dos modelos.

      i. Gráfico de dispersão de resíduos e valores ajustados.

      ii. QQ-plot dos resíduos (quantis dos resíduos vs. quantis teóricos da distribuição normal).

      iii. Histograma dos resíduos.

   (d) Avaliação de ajuste e da adequabilidade do modelo para predizer o peso a partir da altura.

      i. Analise o R^2.

      ii. Analise os resíduos.

   (e) Conclusões:

      i. O modelo atende aos pressupostos da regressão linear?

      ii. Há evidência de *outliers* ou influência desproporcional?

      iii. Como você avaliaria a capacidade preditiva do modelo?

Bibliografia

[1] Colaboratory: Frequently asked questions. https://research.google.com/colaboratory/faq.html. Acessado em Março de 2025.

[2] Detecting Multicollinearity Using Variance Inflation Factors. https://online.stat.psu.edu/stat462/node/180/. Acessado em Outubro de 2024.

[3] O que você pode fazer no google drive? https://support.google.com/a/users/answer/9310246?hl=pt-BR. Acessado em Março de 2025.

[4] Hassan Kibirige. A Grammar of Graphics for Python. https://plotnine.org/.

[5] Aditya Sharma. Principal Component Analysis (PCA) in Python Tutorial. https://www.datacamp.com/tutorial/principal-component-analysis-in-python, 2020. Acessado em Março de 2024.

[6] American Museum of Natural History. Optical Illusions and How They Work. https://www.amnh.org/explore/ology/brain/optical-illusions-and-how-they-work. Acessado em Março de 2024.

[7] Andrew Nisbet. Tufte in Matplotlib. https://www.ajnisbet.com/blog/tufte-in-matplotlib, 2020.

[8] Arthur Charpentier. Generating your own normal distribution table. https://www.r-bloggers.com/2013/10/generating-your-own-normal-distribution-table/#google_vignette. Acessado em Março de 2024.

[9] Jacques Bertin. *Graphics and Graphics Information-Processing*. 1981. Original French edition published by Flammarion, Paris, 1977, translated by William J. Berg and Paul Scott.

[10] Matheus Budkewicz. Como instalar o Jupyter Notebook? (Windows e Linux). https://medium.com/horadecodar/como-instalar-o-jupyter-notebook-windows-e-linux-20701fc583c.

[11] Isabel Cafezeiro, Edward Hermann Hausler, Henrique Luiz Cukierman, and Ivan da Costa Marques. Recontando a computabilidade. *Revista Brsileira de História da Ciência*, 3(2):231–251, jul—dez 2010.

[12] G. Casella and R.L. Berger. *Inferência Estatística*. CENGAGE DO BRASIL, 2010.

[13] Chester Ismay. Repositório de dados de Moderndive. `https://github.com/moderndive/moderndive/blob/master/data-raw`. Acessado em Maio de 2024.

[14] Chris Chobris and Daniel Simons. The Invisible Gorilla. `https://www.youtube.com/watch?v=D_m_9N_3u7o`, 2012.

[15] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, June 1970.

[16] Edgar F. Codd. Further normalization of the data base relational model. Technical Report RJ909, IBM, 8 1971.

[17] Wikipedia contributor. Ascombe's quartet. `https://en.wikipedia.org/wiki/Anscombe%27s_quartet`.

[18] Wikipedia contributor. Iris flower data set. `https://en.wikipedia.org/wiki/Iris_flower_data_set`.

[19] Wikipedia contributor. Psicologia cognitiva. `https://pt.wikipedia.org/wiki/Psicologia_cognitiva`.

[20] Kristin A Cook and James J Thomas. Illuminating the path: The research and development agenda for visual analytics. `https://www.osti.gov/biblio/912515`, 5 2005.

[21] The NumPy Steering Council. Numpy. `https://numpy.org/`.

[22] CRAN.R-Project Team. R Data Import/Export. `https://cran.r-project.org/doc/manuals/r-release/R-data.html`. Acessado em Abril de 2024.

[23] Aaron Culich. Jupyter Logo - Design Brief. `https://github.com/jupyter/design/wiki/Jupyter-Logo`.

[24] DataCamp team. prcomp: Principal Components Analysis. `https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/prcomp`.

[25] Datacamp team. ggplot2 Cheat Sheet. `https://www.datacamp.com/cheat-sheet/ggplot2-cheat-sheet`, 2022. Acessado em Março de 2024.

[26] J. E. Gentle, L. Kaufman, and P. J. Rousseuw. Finding groups in data: An introduction to cluster analysis. *Biometrics*, 47(2):788, June 1991.

[27] Garrett Grolemund. Introduction to R Markdown. `https://rmarkdown.rstudio.com/articles_intro.html`, 2014.

[28] Hadley Wickham and Winston Chang and Lionel Henry and Thomas Lin Pedersen and Kohske Takahashi and Claus Wilke and Kara Woo and Hiroaki Yutani and Dewey Dunnington and Teun van den Brand. A quantile-quantile plot. `https://ggplot2.tidyverse.org/reference/geom_qq.html`. Acessado em Março de 2024.

[29] Hadley Wickham and Winston Chang and Lionel Henry and Thomas Lin Pedersen and Kohske Takahashi and Claus Wilke and Kara Woo and Hiroaki Yutani and Dewey Dunnington and Teun van den Brand. Compute empirical cumulative distribution. `https://ggplot2.tidyverse.org/reference/stat_ecdf.html`. Acessado em Março de 2024.

[30] Hadley Wickham and Winston Chang and Lionel Henry and Thomas Lin Pedersen and Kohske Takahashi and Claus Wilke and Kara Woo and Hiroaki Yutani and Dewey Dunnington and Teun van den Brand. Plot basics. `https://ggplot2.tidyverse.org/reference/`. Acessado em Março de 2024.

[31] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, 1st edition, 2001.

[32] J. Han, J. Pei, and H. Ting. Data mining: Concepts and techniques. `https://ebin.pub/data-mining-concepts-and-techniques-4nbsped-0128117605-9780128117606-9780128117613.html`, 2023.

[33] Hassan Kibirige. Plotnine 0.13.4 API Reference. `https://plotnine.org/reference/`. Acessado em Abril de 2024.

[34] Robert F. Hess, Long To, Jiawei Zhou, Guangyu Wang, and Jeremy R. Cooperstock. Stereo vision: The haves and have-nots. *i-Perception*, 6(3):204166951559302, June 2015.

[35] Rashidul Islam and Stephen Brooks. Visualizing uncertainty with simulated chromatic aberration. *Journal of Perceptual Imaging*, 7(0):1–16, September 2024.

[36] Chester Ismay and Albert Y. Kim. *Statistical Inference via Data Science: A ModernDive into R and the Tidyverse: A ModernDive into R and the Tidyverse*. Chapman & Hall/CRC The R Series, 2020.

[37] J. Hathaway and Katie Larson. Python for Data Science. `https://byuidatascience.github.io/python4ds/index.html`.

[38] Nielsen Jakob. Response times: The 3 important limits. `https://www.nngroup.com/articles/response-times-3-important-limits/`, Janeiro 1993.

[39] Jeroen Janssens. Heuristics for Translating Ggplot2 Code. `https://jeroenjanssens.com/heuristics/`, 2019. Acessado em Março de 2024.

[40] John MacFarlane. Pandoc: a universal document converter. `https://pandoc.org/index.html`.

[41] Jose Stotopoli and Leonardo Vils. Relação entre Variáveis - Correlações. `https://storopoli.io/Estatistica/5-Correlacoes.html`, 2021.

[42] João Luís Ferreira Batista. Curso Relâmpago de R. `http://cmq.esalq.usp.br/wiki/doku.php?id=publico:tutoriais:r-relampago:star`.

[43] Jupyter Development Team. nbconvert: Convert Notebooks to other formats. `https://nbconvert.readthedocs.io/en/latest/`.

[44] Robert I. Kabacoff. Quick-R by Datacamp. `https://www.statmethods.net/`.

[45] Kam Tin Seong. Data Profiling, Exploration and Analysis-funModeling Methods. `https://rpubs.com/tskam/R4DSA04-funModeling_methods`. Acessado em Abril de 2024.

[46] Daniel Keim, Jörn Kohlhammer, Geoffrey Ellis, and Florian Mansmann. *Mastering the Information Age Solving Problems with Visual Analytics*. Eurographics Association, 2010.

[47] Kevin Babitz. Introduction to k-Means Clustering with scikit-learn in Python. `https://www.datacamp.com/tutorial/k-means-clustering-python`, 2023.

[48] D.H. Laidlaw, R.M. Kirby, C.D. Jackson, J.S. Davidson, T.S. Miller, M. da Silva, W.H. Warren, and M.J. Tarr. Comparing 2d vector field visualization methods: A user study. *IEEE Transactions on Visualization and Computer Graphics*, 11(01):59–70, January 2005.

[49] LaTex team. The Latex Project. `https://www.latex-project.org/`.

[50] LEONARD P. AYRES. THE WAR WITH GERMANY: A STATISTICAL SUMMARY. `http://www.gwpda.org/docs/statistics/statstc.htm`, 1919.

[51] Lukasz Piwek. Tufte in R. `http://motioninsocial.com/tufte/`, 2017.

[52] Marcos Nascimento Magalhães and Antonio Carlos Pedroso de Lima. *Noções de Probabilidade e Estatística*. edUSP, 7ª ed. edition, 2010.

[53] Mahmoud Harmouch. 17 types of similarity and dissimilarity measures used in data science. `https://towardsdatascience.com/17-types-of-similarity-and-dissimilarity-measures-used-in-data-science-3eb914d2681`, 2021.

[54] Michael Hahsler. dbscan: Density-based Spatial Clustering of Applications with Noise (DBSCAN) . `https://www.rdocumentation.org/packages/dbscan/versions/1.1-12/topics/dbscan`.

[55] Miguel Garcia. Using ggplot in Python: Visualizing Data With plotnine. https://realpython.com/ggplot-python/.

[56] Mirko Stojiljković. NumPy, SciPy, and pandas: Correlation With Python. https://realpython.com/numpy-scipy-pandas-correlation-python/.

[57] Kirill Müller and Hadley Wickham. Tibbles. https://tibble.tidyverse.org/articles/tibble.html.

[58] University of Toronto Libraries team. Data Visualization – Tools & Tutorials. https://mdl.library.utoronto.ca/dataviz/tools-tutorials.

[59] Oz. Psicologia: Resolução de Problemas. https://mundodeoz.wordpress.com/2009/12/27/resolucao-de-problemas/.

[60] Pandas team. DataFrame. https://pandas.pydata.org/docs/reference/frame.html.

[61] Pandas team. Series. https://pandas.pydata.org/docs/reference/series.html.

[62] Pipz Academy team. Guia básico de Markdown. https://docs.pipz.com/central-de-ajuda/learning-center/guia-basico-de-markdown.

[63] Plotly Team. Dendrograms in ggplot2. https://plotly.com/ggplot2/dendrogram/. Acessado em Março de 2024.

[64] Plotnine Team. plotnine.stat_ecdf. https://plotnine.org/reference/stat_ecdf. Acessado em Março de 2024.

[65] Plotnine Team. plotnine.stat_qq. https://plotnine.org/reference/stat_qq. Acessado em Março de 2024.

[66] Posit software. Welcome to Quarto: An open-source scientific and technical publishing system. https://quarto.org/. Acessado em Março de 2025.

[67] Benjamin Pryk. How to Use Jupyter Notebook: A Beginner's Tutorial. https://www.dataquest.io/blog/jupyter-notebook-tutorial/, 2020.

[68] Python team. Python For Beginners. https://www.python.org/about/gettingstarted/.

[69] Rafael A. Irizarry. Introdução à Ciência de Dados - Análise de Dados e Algoritmos de Previsão com R. http://rafalab.dfci.harvard.edu/dslivro/, 2020.

[70] Rafael A. Irizarry. Introduction to Data Science: Data Analysis and Prediction Algorithms with R. https://rafalab.dfci.harvard.edu/dsbook/, 2023.

[71] Rafael A. Irizarry. Introduction to Data Science: Data Wrangling and Visualization with R. https://rafalab.dfci.harvard.edu/dsbook-part-1/, 2025.

[72] Rafael A. Irizarry. Statistics and Prediction Algorithms Through Case Studies. https://rafalab.dfci.harvard.edu/dsbook-part-2/, 2025.

[73] RDocumentation team. clara: Clustering Large Applications. https://www.rdocumentation.org/packages/cluster/versions/1.3-5/topics/clara.

[74] RDocumentation team. hclust: Hierarchical Clustering. https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/hclust.

[75] RDocumentation team. kmeans: K-Means Clustering. https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/kmeans.

[76] Robert McDonnell. As cinco melhores extensões ao ggplot2. https://ibpad.com.br/ciencia-dados/cinco-melhores-extensoes-ao-ggplot2/, 2017. Acessado em Março de 2024.

[77] Jonathan C. Roberts. State of the art: Coordinated & multiple views in exploratory visualization. In *Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV 2007)*. IEEE, July 2007.

[78] Verônica Santana. Tutorial R/RStudio. https://edisciplinas.usp.br/pluginfile.php/4883125/mod_resource/content/1/Tutorial.pdf.

[79] Ben Schneiderman. The Eight Golden Rules of Interface Design. https://www.cs.umd.edu/users/ben/goldenrules.html.

[80] Scikit Learn team. sklearn.cluster.AgglomerativeClustering. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html.

[81] Scikit Learn team. sklearn.cluster.DBSCAN. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html.

[82] Scikit-learn team. sklearn.decomposition.PCA. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.

[83] Seaborn Team. seaborn.pairplot. https://seaborn.pydata.org/generated/seaborn.pairplot.html. Acessado em Março de 2024.

[84] Adam Shafi. How to Use Jupyter Notebooks: The Ultimate Guide. https://www.datacamp.com/tutorial/tutorial-jupyter-notebook, 2020.

[85] Shiny team. Easy web apps for data science without the compromises. `https://shiny.posit.co/`.

[86] B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings 1996 IEEE Symposium on Visual Languages*, VL-96. IEEE Comput. Soc. Press.

[87] STHDA team. Data Handling in Python: File IO: Create, Export, Import and Convert Data. `https://rowannicholls.github.io/python/data/create_export_import_convert.html`.

[88] STHDA team. Import and export data using R. `http://www.sthda.com/english/wiki/import-and-export-data-using-r`.

[89] STHDA team. R Built-in Data Sets. `http://www.sthda.com/english/wiki/r-built-in-data-sets`.

[90] STHDA team. t distribution table. `http://www.sthda.com/english/wiki/t-distribution-table`. Acessado em Abril de 2024.

[91] TechWeb team. Scientific Visualization Software Packages. `https://www.bu.edu/tech/support/research/training-consulting/online-tutorials/introduction-to-scientific-visualization-tutorial/software-packages/`.

[92] Alexandru C. Telea. *Data Visualization: Principles and Practice*. A K Peters Ltda, 2nd edition, 2008.

[93] The Matplotlib development team. Matplotlib. `https://matplotlib.org/`.

[94] E.R. Tufte. The visual display of quantitative information, 2001.

[95] John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[96] Vincent Arel-Bundock. Repositório de dados do aplicativo R. `https://vincentarelbundock.github.io/Rdatasets/datasets.html`. Acessado em Março de 2024.

[97] Volunteer Contributors. Pandas. `https://pandas.pydata.org/`.

[98] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Series in Interactive Technologies. Morgan Kaufmann, 3 edition, 2012.

[99] Neil A. Weiss. *Introductory Statistics*. Pearson, Addison-Wesley, 2015.

[100] McKinney Wes. Python for data analysis. `https://wesmckinney.com/book/`, 2012.

[101] H. Wickham and G. Grolemund. R for data science. `https://r4ds.had.co.nz/`, 2017.

[102] H. Wickham, Mine Çetinkaya Rundel, and G. Grolemund. R for data science. `https://r4ds.hadley.nz/`, 2023.

[103] Hadley Wickham. A layered grammar of graphics. *Journal of Computational and Graphical Statistics*, 19(1):3–28, 2010.

[104] Hadley Wickham. Tidy data. *Journal of Statistical Software*, 59(10), 2014.

[105] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016.

[106] Hadley Wickham, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke, Kara Woo, Hiroaki Yutani, Dewey Dunnington, Teun van den Brand, and PBC Posit. Package 'ggplot2'. `https://cran.r-project.org/web/packages/ggplot2/ggplot2.pdf`. Acessado em Abril de 2024.

[107] Hadley Wickham, Romain François, Lionel Henry, Kirill Müller, and Davis Vaughan. dplyr: A grammar of data manipulation. `https://github.com/tidyverse/dplyr`, 2023. R package version 1.1.4.

[108] L. Wilkinson, D. Wills, D. Rope, A. Norton, and R. Dubbs. The Grammar of Graphics. `https://books.google.com.br/books?id=_kRX4LoFfGQC`, 2005.

[109] Leland Wilkinson. The grammar of graphics. *WIREs Computational Statistics*, 2(6):673–677, October 2010.

[110] Euphemia Wong. Shneiderman's Eight Golden Rules Will Help You Design Better Interfaces. `https://www.interaction-design.org/literature/article/shneiderman-s-eight-golden-rules-will-help-you-design-better-interfaces`, 2020.

[111] Pak Chung Wong and J. Thomas. Visual analytics. *IEEE Computer Graphics and Applications*, 24(5):20–21, September 2004.

[112] Martin Woodward and Tali Herzka. Markdown - Math Support. `https://github.blog/2022-05-19-math-support-in-markdown/`.

[113] Ydata profiling Team. Ydata Profiling 4.7. `https://docs.profiling.ydata.ai/latest/`. Acessado em Abril de 2024.

[114] Ji Soo Yi, Youn ah Kang, John T. Stasko, and Julie A. Jacko. Understanding and characterizing insights. In *Proceedings of the 2008 Workshop on BEyond time and errors: novel evaLuation methods for Information Visualization*. ACM, April 2008.

[115] Zoumana Keita. Principal Component Analysis in R Tutorial. `https://www.datacamp.com/tutorial/pca-analysis-r`, 2023. Acessado em Março de 2024.