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

IA376I/EA099I

Tópicos em Engenharia de Computação VII

Análise de Dados Visual (*Visual Analytics*)

Profa. Wu, Shin - Ting

Março de 2024

Conteúdo

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 [9]. Diante da impossibilidade de encontrar soluções exatas para esses 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 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 fornecer mecanismos apropriados para armazená-los, é fundamental desenvolver ferramentas e estratégias adequadas para acessar, 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 campo de pesquisa inovador 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 crucial 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 incentivando 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 de Dados Visual, 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 fornecem as ferramentas teóricas e práticas necessárias para entender a variabilidade nos dados, realizar inferências estatísticas, sumarizar e simplificar dados, extrair *insights* valiosos a partir de conjuntos de dados complexos, e contribuir para a solução aproximada de problemas com base nos 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 distintos: estatística descritiva e estatística de inferência. A **estatística descritiva** emprega métodos como média, mediana e desvio padrão para resumir dados, oferecendo uma visão abrangente de suas características. Já a **estatística de inferência**, baseada em abordagens probabilísticas, vai além ao realizar testes de hipóteses e estabelecer intervalos de confiança. Em conjunto, essas abordagens possibilitam a extração de conclusões sólidas a partir de amostras, gerenciando a incerteza e embasando decisões.

A interligação entre probabilidade e estatística de inferência 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 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 mais precisos ao longo do tempo, com base na "experiência" (memória) e treinamento contínuo com dados atualizados.

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 que você deseja testar, 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 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, 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 desempenhem um papel crucial ao 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 de seu potencial máximo.

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 do tipo específico 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 qualquer, 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, 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 para o nível de confiança. 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, o valor crítico correspondente é geralmente aproximadamente 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 [10].

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.

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 [14], 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.

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

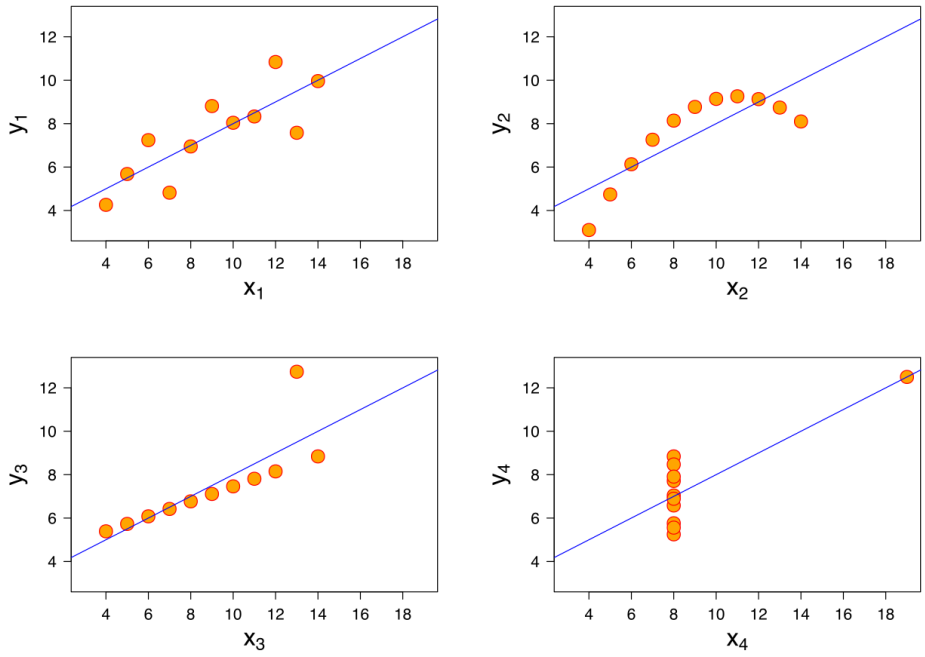


Figura 1.1: O quarteto de Ascombe. (Fonte: [14])

os cientistas de dados na escolha de variáveis e algoritmos adequados ao tipo de problema e aos dados disponíveis [31].

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 identificados. 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 vastamente explorada. Foi, no entanto, o estatístico americano John Wilder Tukey quem formalizou, em 1977, por meio de seu livro *Exploratory Data Analysis* [90], a interação visual com dados brutos através de gráficos, diagramas e cartas, permitindo análise sem condicionamento a um modelo específico. Pesquisadores, como Tufte [89] e Ware [93], 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 [93]. A ideia consiste essencialmente em utilizar técnicas de computação gráfica para representar graficamente os dados a fim de destacar padrões,

tendências e anomalias nos dados, facilitar a compreensão e interpretação e propiciar classificação, agrupamento, sequenciamento ou sumarização adequada desses dados.

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. 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 sensores, 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 : Os cientistas de dados desenvolvem modelos estatísticos ou de aprendizado de máquina para fazer previsões, classificações ou agrupamentos com base nos dados.

Avaliação de Modelos : Após a criação dos modelos, é importante avaliar o desempenho

destes modelos para garantir utilidade e precisão. Isso pode envolver métricas de avaliação e validação cruzada.

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 de Dados Visual

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 crucial de combater futuros ataques terroristas nos Estados Unidos e em todo o mundo. Uma agenda para programas de pesquisa e desenvolvimento de ferramentas que facilitam 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 ocultas ao longo da sua exploração e mineração.

O termo **Análise de Dados Visual** (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*" [107]. 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 [43] o estado-da-arte das pesquisas e desenvolvimento em análise de dados visual na Europa.

A Análise de Dados Visual 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 de Dados Visual 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 de Dados Visual 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 de Dados Visual 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. IPython[6] 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 programação, enquanto Jupyter Lab ou Jupyter Notebook[7], 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[8], 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[9], dplyr[10], caret[11] e ggplot2[12]. O ambiente R é favorecido por estatísticos e pesquisadores, sendo RStudio[13] uma IDE popular para desenvolvimento 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

[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://ipython.org/
[7] https://jupyter.org/install
[8] `https://www.r-project.org/`
[9] `https://cran.r-project.org/web/packages/RcppArmadillo/index.html`
[10] `https://dplyr.tidyverse.org/`
[11] `https://topepo.github.io/caret/`
[12] `https://ggplot2.tidyverse.org/`
[13] `https://posit.co/download/rstudio-desktop/`

especialmente valiosas na Ciência de Dados, onde **a transparência e a documentação adequada são essenciais**. Tanto o R Markdown[14] do RStudio quanto o Jupyter Notebook com nbconvert[15] em Python 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, como o pacote Seaborn de Python e o pacote ggplot2 de R[16] permite uma análise colaborativa e reproduzível de dados, contribuindo para sua relevância na Ciência de Dados. Uma análise comparativa entre essas duas linguagens é apresentada por Luna no espaço Datacamp[17].

Outra característica interessante é que os dois ambientes de desenvolvimento RStudio e Jupyter são projetados para permitir o acesso a funções de outras linguagens, o que oferece uma grande vantagem em termos de interoperabilidade[18]. Isso significa que os usuários podem tirar proveito de recursos e bibliotecas 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

[14] `https://rmarkdown.rstudio.com/`
[15] `https://nbconvert.readthedocs.io/en/latest/`
[16] `https://medium.com/mlearning-ai/an-alliance-python-and-r-seaborn-and-ggplot2-233864b77bc4`
[17] `https://www.datacamp.com/blog/python-vs-r-for-data-science-whats-the-difference`
[18] `https://www.datacamp.com/tutorial/using-both-python-r`

de interação e visualização da Computação Gráfica. Essa subárea específica da Ciência de Dados é conhecida como Análise de Dados Visual. 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 [17, 43]. É sobre essa parte da Ciência de Dados que este texto 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.

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, você trabalha 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 que você define. 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 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 R quanto Python possuem uma interface fluida com o Markdown, tornando fácil gerar documentações com diferentes níveis de detalhes nos ambientes de desenvolvimento integrados (*Integrated Development Environments*, em inglês) RStudio e Jupyter, como discutiremos na Seção 2.3.

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 para 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 programar em Python, é necessário instalar um **interpretador Python**. Explicações detalhadas da instalação de um interpretador em diferentes sistemas operacionais podem ser encontradas em [63]. 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 (sigla de *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
```

IPython é instalado como parte do ambiente Jupyter. Pode-se verificar se IPython foi instalado corretamente digitando o seguinte comando no *prompt* de comando:

```
ipython
```

Isso arirá 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 [8].

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. A declaração de variáveis nessa linguagem é implícita: 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), números de ponto flutuante (float), números complexos (complex) e *strings* (str).

Na manipulação, análise estatística e visualização de dados, a abstração em tipos de dados numéricos e categóricos desempenha um papel crucial. **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, do tipo sequência, 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[95]:

Tuplas : São estruturas de dados **imutáveis** que podem armazenar elementos de diferentes tipos ordenados. 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. 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 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() do pacote pandas (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.

O seguinte trecho de códigos ilustra a definição de variáveis dos 4 tipos 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 diferenciada em Python, permitindo que uma variável mude de tipo ao longo do programa conforme necessário.

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 5.2.1 em [35]), 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 5.2.2 em [35]), *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 de 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.

2.1.6 Funções Incorporadas

As funções incorporadas são componentes essenciais do conjunto padrão de funcionalidades de uma linguagem de programação, dispensando a necessidade de importar módulos ou pacotes adicionais para utilizá-las. Além das funções relacionadas às estruturas de sequência, como listas e tuplas, algumas das funções incorporadas mais comuns e amplamente utilizadas em Python incluem: print(object), type(object), round (number, digits), input(message), 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. Pode-se omitir os nomes dos argumentos na chamada da função se fornecer os argumentos na ordem correta. Isso é conhecido como **chamada de função posicional**. No entanto, é importante notar que, ao fazer isso, é necessário fornecer os argumentos na ordem exata em que foram definidos na função. 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 [95], as bibliotecas Python essenciais para análise de dados são NumPy, pandas, matplotlib e SciPy. Para utilizar as funções dessas bibliotecas, é necessário importar a biblioteca correspondente antes de seu uso no *script* ou programa, utilizando o comando import.

O uso da biblioteca NumPy é bastante comum, pois ela oferece suporte para arranjos multidimensionais e diversas operações sobre esses objetos. Por exemplo, para criar um vetor de valores de 0 a 10 com um passo de 2, podemos importar os métodos do pacote 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* o pacote enumpy para np.

O pacote 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 oferecem 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 você 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 "́", você pode 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. Uma análise comparativa desses quatro pacotes é realizada por Dennis e discutida em [109].

2.1.8 Funções Personalizadas

Em Python, a capacidade de criar funções personalizadas oferece uma flexibilidade fundamental para os programadores, 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 à função é indentado, e o uso da palavra-chave return é opcional para especificar o valor que a função deve retornar (Apêndice em [95]). 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 demonstra que o R incorpora uma série de conjuntos de dados como conjuntos básicos incorporados (*builtin* em inglês). Por outro lado, o Python não inclui conjuntos de dados na instalação padrão do interpretador, alinhado à filosofia da linguagem, que valoriza a simplicidade e a minimalidade na instalação.

O Python é um ecossistema altamente modular, o que significa que os desenvolvedores têm a liberdade de escolher os pacotes e bibliotecas específicos que desejam utilizar para suas tarefas. Mesmo que o Python não inclua conjuntos de dados diretamente na instalação padrão, existem bibliotecas de terceiros, como scikit-learn, seaborn, nltk e outras, que oferecem conjuntos de dados prontos para uso. Esses conjuntos de dados podem ser facilmente acessados e utilizados após a instalação das respectivas bibliotecas específicas. Por exemplo, a biblioteca scikit-learn oferece diversos conjuntos de dados para tarefas de aprendizado de máquina, enquanto a seaborn pode ser usada para visualização dos conjuntos de dados implementados no pacote.

2.2 R

A linguagem R é destinada para estatística e análise de dados, oferecendo uma ampla gama de ferramentas e recursos incorporados (*builtin*) 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 desenvol-

vimento 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 [41] os seus conhecimentos sobre R de forma didática.

Para programar em R, você precisa instalar o ambiente de desenvolvimento R e, opcionalmente, uma interface de desenvolvimento (IDE) dedicada como RSutdio. Uma referência rápida sobre RStudio se encontra em [72].

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.

Ao contrário do Python, em R, o escopo de uma função 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* em R é apresentada em [39, 41]. Especificamente relativos à análise de dados em R pode ser encontrada em [96].

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), números de ponto flutuante (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, R possui estruturas de dados mais complexas que facilitam a organização e manipulação de informações. Essas estruturas expandem consideravelmente as capacidades da linguagem em lidar com uma variedade de dados. Essa capacidade contribui significativamente para a adaptabilidade e versatilidade do R em diversas aplicações relacionadas à Ciência de Dados, abrangendo desde análise exploratória até modelagem estatística avançada. Para criar e manipular essas estruturas complexas, R oferece uma variedade de funções específicas para cada tipo de estrutura. Incluem-se entre as estruturas básicas [41]:

Vetores : São arranjos que armazenam elementos de diferentes tipos 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, 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 [53], integrado no pacote tidyverse. Essa nova versão oferece funcionalidades adicionais e uma interface mais amigável para manipulação de dados, tornando a análise de dados mais eficiente e intuitiva.

Factors : São estruturas de dados que representam variáveis categóricas. **Variáveis categóricas** são aquelas que possuem um número finito e predefinido de categorias ou níveis, como por exemplo, cores, grupos de tratamento, 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

[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).

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 summary() integrada.

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 (¿=) [41]. 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 *built-in* (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 e estatísticas. 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, ...)
```

como também as mesmas regras de omissão dos nomes dos argumentos em chamadas adotadas em Python.

Além das funções relacionadas aos tipos de dados de estrutura mais complexa, porém básica, como apresentado 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 [101] é amplamente utilizada para manipulação de dados organizados em tabelas, especialmente com o tipo de objeto conhecido como data frame (Seção 2.2.2). Para importá-la 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>))
```

Outra biblioteca amplamente utilizada para visualização de dados em R é o ggplot2 [99]. 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.

Em R, existe um pacote denominado tidyverse, que representa um conjunto integrado de pacotes projetados para processar de forma coesa na manipulação, visualização e análise de dados organizados de maneira *tidy* (Capítulo 6). Esse conjunto abrange não apenas o pacote dplyr e o pacote ggplot2,

mas também inclui o pacote tidyr para manipulação e organização de dados no formato *tidy*, e o pacote readr para facilitar a importação e exportação de dados.

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 pertencente à função, enquanto em R, utiliza-se chaves para delimitar o bloco de código, como ilustra o seguinte exemplo de código [96]:

```
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, 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 [83] fornece uma guia rápida de uso desses conjuntos de dados incorporados.

2.3 Documentação

Tanto R quanto Python, junto com o Markdown [58], 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 [108].

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 [79]. 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 (*chunk* em inglês): 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, PDF e os compatíveis com o Word. O yaml, que lida com dados no formato YAML (acrônimo de *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 [96] ou no material *online* em [26].

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 multilíngues. 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"[78]. 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 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 [40].

Ao instalar o Jupyter Notebook, geralmente são incluídos os pacotes jupyter e nbconvert, que

[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 [20].

[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.

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 [62, 78].

A utilização de pacotes, como numpy para operações numéricas [18], pandas para manipulação de dados em formato tabular [92], e matplotlib para a criação de gráficos [87], 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 a função 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
```

Por padrão, ao gerar uma documentação a partir de um *notebook*(.ipynb), todas as células Markdown são renderizadas. Isso inclui células Markdown que contêm texto formatado, equações matemáticas e elementos multimídia, como imagens ou vídeos incorporados. Quanto às células de código, a renderização depende do tipo de exportação. Em geral, as células de código são executadas e os

resultados são incluídos no documento. No entanto, essas configurações podem ser modificadas usando métodos como metadados especiais, extensões do Jupyter Notebook, ou configurações específicas ao exportar o *notebook* para outros formatos.

O método de metadados especiais envolve a adição direta de comentários especiais no JSON (do inglês *JavaScript Object Notation*) do *notebook*. Por exemplo, pode-se adicionar um comentário {"hide_input": true} ou {"hide_output": true} na primeira linha de uma célula para ocultar a entrada ou saída dessa célula, respectivamente. Outra opção é instalar extensões no Jupyter Notebook, como nbextensions ou Hide Input, que permitem adicionar metadados diretamente às células do *notebook* para controlar sua exibição na versão exportada. Essas extensões devem ser instaladas separadamente e, uma vez instaladas, é possível utilizar os botões ou opções fornecidos na interface do Jupyter Notobook para definir os metadados de ocultação em células individuais.

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 básica de R. Em teoria, pode ser usado com qualquer versão do R que suporte pacotes. Eçe é compatível com Python2 e Python3. Pode-se verificar a disponibilidade de funções Python para acessos em R através do seguinte comando

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

2.5 Considerações Finais

Neste capítulo, exploramos as nuances de 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 e Jupyter Notebook, 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

   - R: da Seção 4.4, 6.3 em [96]
   - Python: da Seção 4.4 em [35], complementado com os seguintes itens:

     (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://towardsdatascience.com/jupyter-notebooks-avoid-making-t` os erros comuns em Jupyter Notebook.

     (d) Sintetize com base em `https://www.linkedin.com/advice/3/what-best-ways-troubleshoot-e` algumas formas para diagnosticar os 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 de Dados Visual

A interface gráfica homem-máquina desempenha um papel crucial como uma conexão dinâmica entre o usuário e os dados, estabelecendo uma plataforma onde a análise de dados visual pode ocorrer de forma natural e eficiente. Como enfatizado por Yi e colaboradores [111], fatores como codificação visual inadequada, baixa usabilidade e falta de organização das informações na interface podem prejudicar a percepção de detalhes cruciais para a resolução de problemas.

Além de apresentar dados complexos de maneira eficaz, um sistema de análise visual de dados deve ser ágil ao responder às interações do usuário, promovendo uma sensação de fluidez e eficácia, aprimorando assim a experiência do usuário durante a análise em diferentes níveis de abstração. Conforme apontado por Jakob Nielsen [36], uma resposta em até 1 segundo a uma interação do usuário permite que ele mantenha seu fluxo de pensamento sem distrações significativas. Portanto, ao projetar a interface para um sistema de análise visual de dados, é essencial considerar não apenas a qualidade e eficiência dos algoritmos de renderização aplicados, mas também o tempo necessário para processar os dados e fornecer uma resposta alinhada com a intenção do usuário.

Além dos aplicativos gráficos interativos convencionais, um sistema de suporte à construção de modelos mentais e estratégias alternativas de resolução deve facilitar a exploração dos dados através de diferentes níveis de abstração de um mesmo conjunto de dados. A visualização coordenada desses diferentes níveis não apenas facilita a exploração, mas também proporciona uma compreensão mais profunda das informações potenciais ocultas nos dados [71]. No entanto, isso requer abordagens para resolver desafios relacionados ao espaço na tela, ao desempenho dos computadores, à capacidade de assimilação dos usuários e à coordenação da resposta a uma interação em uma representação específica com os outros níveis de abstração.

Tipicamente, uma interface gráfica é dividida em duas áreas principais como ilustra a Figura 3.1: uma área de elementos de interação e uma área de visualização. Essa abordagem visa organizar e estruturar a apresentação de informações. A **área de elementos de interação** abriga botões, menus, campos de entrada e outros elementos interativos que permitem aos usuários realizar ações ou interagir

com o sistema. É onde se encontram os controles que os usuários utilizam para navegar, realizar ações ou inserir informações. A **área de visualização** é destinada à apresentação de informações, dados ou conteúdo relevante para o usuário. Pode incluir gráficos, texto, imagens ou qualquer outro tipo de conteúdo visual. É onde os usuários veem os dados em questão e recebem realimentação sobre suas interações.

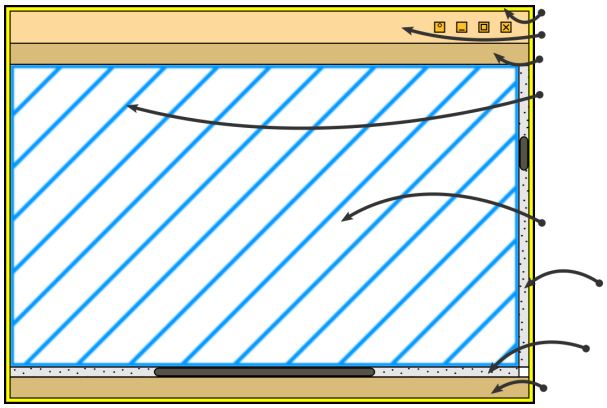


Figura 3.1: Uma janela numa interface gráfica com uma área de visualização (representada pela área hachurada) e uma área de interação (contendo elementos ao redor da área hachurada).

No intuito de facilitar a compreensão da interface e permitir que os usuários se concentrem nas tarefas sem distrações desnecessárias, inúmeros pesquisadores procuraram formular princípios ou diretrizes no *design* dos elementos de interação e visualização em uma interface gráfica. O cientista de computação Ben Shneiderman, o psicólogo cognitivo Colin Ware e os estatísticos Edward R. Tufte e Leland Wilkinson deixaram marcas distintas em suas áreas de especialização. Interessado em criar interfaces que permitissem aos usuários interagir diretamente com a tela do computador, sem depender do teclado e *mouse*, Shneiderman desenvolveu conceitos fundamentais e princípios que orientam a **interação direta** entre humanos e computadores, promovendo uma interação efetiva na exploração de dados. Mostramos na Seção 3.1 que seus princípios garantem que uma interface seja clara, previsível e capacitadora para o usuário.

Ware, dedicado à compreensão da influência da percepção e cognição visual na geração de imagens informativas e efetivas, e Tufte, centrado na apresentação eficiente de dados complexos em análises estatísticas, estavam mais interessados nos aspectos visuais e na apresentação efetiva de dados na área de visualização do que a disposição dos elementos de interação. Contudo, é crucial destacar que os dados a serem exibidos na área de visualização podem abranger uma diversidade de naturezas distintas conforme evidenciada na Seção 3.4. Apesar de Ware e Tufte terem focos distintos em suas áreas de especialização, ambos os pesquisadores desenvolveram princípios visando criar representações visuais eficazes e compreensíveis. A Seção 3.2 enfatiza que Ware ressalta a importância de representações visuais alinhadas à percepção humana (Capítulo 4), garantindo uma interface não apenas funcional, mas também cognitivamente eficiente. Por sua vez, a Seção 3.3 destaca que Tufte prioriza a clareza na apresentação de dados abstratos complexos, com princípios que favorecem a minimização de elementos

não essenciais e a maximização da integridade visual, promovendo uma compreensão mais rápida e precisa.

Em 1999, o estatístico Leland Wilkinson formalizou uma gramática dos gráficos 2D com o objetivo de padronizar o processo de *design* da área de visualização para análise estatística [104]. Essa iniciativa visava facilitar a comunicação entre *designers* e proporcionar uma base teórica sólida para a criação de gráficos coerentes e informativos. Na Seção 3.5, são apresentadas as regras e estruturas que orientam os projetistas na composição visual dos elementos gráficos, onde os dados são mapeados. Posteriormente, em 2005, Hadley Wickham e colegas desenvolveram e implementaram uma gramática dos gráficos específica para a linguagem de programação R. O pacote resultante, denominado **ggplot2**, tornou-se rapidamente uma ferramenta amplamente adotada na comunidade de ciência de dados e análise estatística. Embora os princípios fundamentais da gramática dos gráficos sejam estensíveis aos gráficos 3D, é importante reconhecer que essas representações tridimensionais apresentam desafios adicionais, especialmente relacionados às percepções espaciais.

3.1 Princípios de Schneiderman

A abordagem para explorar um conjunto de dados é profundamente influenciada pela maneira como seus potenciais usuários raciocinam. A decisão sobre estratégias alternativas para apresentar dados e os tipos de interação com esses dados precisa ser personalizada. Ben Schneiderman, precursor das pesquisas relacionadas à interface homem-máquina, publicou em [80] a Mantra de Busca de Informações Visuais (em inglês, *Visual Information-Seeking Mantra*), conhecida por **Mantra de Schneiderman** (*Overview first, zoom and filter, then details on demand*), envolvendo 7 tarefas sobre 7 tipos de dados (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 subseção 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 sobre 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 abordagem sistemática para a exploração eficaz de dados. Consolidando suas vastas experiências em *design* de interfaces de usuário e destacando princípios fundamentais, Schneiderman propôs as "Oito Regras de Ouro" em [73], também reconhecidas como "Oito Princípios para a Interface de Usuário". Esses princípios visam aprimorar a usabilidade e a experiência do usuário:

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 exigir 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 [106], 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.

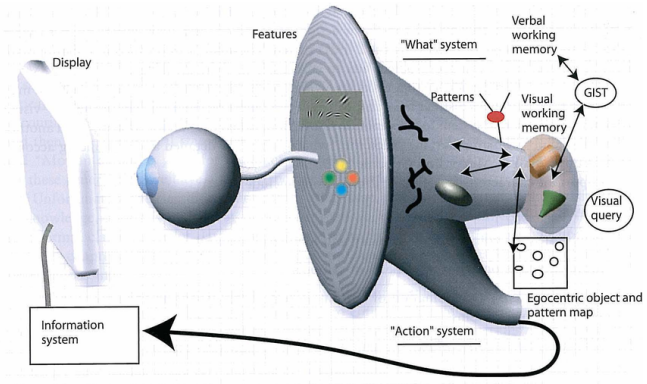


Figura 3.2: Um modelo de três estágios para processamento visual humano proposto pelo Ware: pré-atentivo (*features*), percepção de padrão (*patterns*) e processamento orientado à busca visual (*visual query*). (Fonte: `https://www.researchgate.net/figure/A-three-stage-model-of-human-visual-information-processing_fig6_224285723`)

Em [93], 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:

Resolução de Detalhes : Desconsiderar detalhes sutis que podem ser perdidos ou visualmente indiferenciados, não sendo notados pelos usuários. Isso reduz a sobrecarga visual com informações irrelevantes e otimiza o uso do recurso computacional sem

comprometer a qualidade da visualização, aproveitando as limitações naturais na resolução perceptual.

Suavização de Variações de Níveis de Cinza : evitar transições abruptas entre níveis de cinza adjacentes. Isso assegura uma percepção precisa da luminosidade na visualização, promovendo uma interpretação mais fiel das representações visuais. No entanto, em certos contextos, transições abruptas podem ser estrategicamente exploradas, como para realçar contornos aplicando o efeito de Cornsweet mostrado na Figura 3.3

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

Representação de Cores em Cromas : Alinhar mais de perto com a forma como nosso sistema visual processa as informações de cor. Isso otimiza a interpretação de das visualizações e aprimora a clareza e a fidelidade da visualização aos dados originais. Sendo a percepção visual humana mais sensível às variações de croma (pureza da cor) do que à luminosidade (intensidade da luz), o *design* de uma visualização pode se beneficiar de uma representação mais eficiente e intuitiva das informações de cor. No entanto, é importante observar que os canais cromáticos possuem aproximadamente 1/3 da capacidade de transmissão de detalhes finos quando comparados ao canal de luminância. Deve-se equilibrar o contraste em luminância para garantir uma visualização eficaz.

Processamento Pré-atentivo : Utilizar propriedades visuais, tais como forma, cor, textura, movimento e profundidade estereoscópica, que são prontamente percebidas pelos distintos canais perceptuais paralelos no primeiro estágio do sistema visual humano. Isso permite destacar de maneira eficiente informações relevantes sem exigir esforço cognitivo, aproveitando a capacidade da visão de processar automaticamente certos estímulos visuais. Ware também destaca como esses canais podem influenciar a percepção integrada ou separada de diferentes atributos físicos, sugerindo sua aplicação

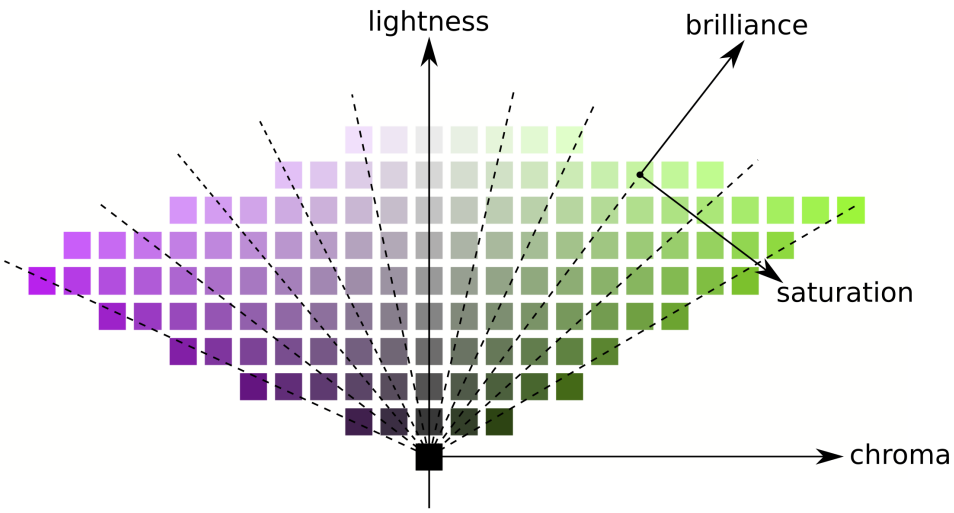


Figura 3.4: Croma e luminosidade num espaço de cores.

no *design* de visualização para criar representações mais eficazes e compreensíveis.

Organização Perceptual : Mapear atributos físicos de interesse em propriedades visuais, tais como posições espaciais, forma, cor, tamanho, de forma que os dados possam ser visualizados seguindo uma regra de estruturação. Isso propicia a percepção de padrões que ocorre naturalmente no segundo estágio do sistema visual humano e facilita uma interpretação eficaz. A aplicação de animações a esses elementos visuais organizados ao longo do tempo proporciona uma compreensão mais profunda dos dados, permitindo a observação das mudanças nos padrões ao longo de diferentes períodos. Isso não apenas melhora a interpretação dos dados, mas também adiciona uma dimensão temporal à visualização. Cabe ressaltar que Ware reconhece o valor do movimento e da animação na visualização de dados, mas também destaca a necessidade de usá-los com moderação e com um propósito claro para evitar distrações.

Reconhecimento de um Objeto : Utilizar diagramas que ilustrem os componentes de um objeto e suas interconexões. Isso favorece a organização dos elementos visuais de maneira a refletir a estrutura dos objetos no mundo real, facilitando assim o reconhecimento desses objetos no terceiro estágio do sistema visual. A capacidade inata do nosso sistema visual para interpretar visualizações complexas promove uma compreensão mais rápida e intuitiva. Além disso, incorporar conhecimento prévio sobre o objeto, fornecendo informações de contexto não-visuais, contribui significativamente para a eficácia do reconhecimento.

Reconhecimento ao Invés de Lembrança : Minimizar a carga cognitiva,, priorizando elementos facilmente reconhecíveis em vez de exigir que os usuários memorizem informações sobre esses elementos.

Visualização 3D orientada a Tarefa : Ao selecionar técnicas que fornecem pistas de

profundidade, é crucial considerar a finalidade específica da percepção 3D em relação às ações planejadas para os dados renderizados. Isso permite uma criação mais direcionada e eficaz de interfaces, garantindo que a visualização 3D atenda de forma otimizada às necessidades da interação do usuário com os dados tridimensionais.

Limitações da Memória de Trabalho Visual : Minimizar o tempo necessário para fornecer informações aos usuários, considerando a limitada capacidade da memória de trabalho visual e verbal humana. Isso visa preservar a contextualização cognitiva estabelecida para resolver tarefas reais, garantindo a continuidade do fluxo de pensamento durante sua resolução, sem interrupções.

Vale ressaltar que Ware enfatiza a viabilidade dessas diretrizes de *design* baseadas na percepção em situações relativamente simples. Em face de requisitos mais complexos, recai sobre o desenvolvedor a responsabilidade de fazer escolhas criteriosas e utilizar os recursos gráficos de maneira perspicaz.

3.3 Princípios de Tufte

O estatístico Edward Tufte é reconhecido por suas notáveis contribuições ao campo da visualização de dados estatísticos, evidenciando-se por sua abordagem focada em representações gráficas desses dados. Seu trabalho não apenas resgata métodos eficazes de representação gráfica dos dados de uma população, mas também os reinterpreta, destacando a continuidade desses princípios ao longo do tempo e empenhando-se em aprimorá-los. Tufte não apenas destaca a importância histórica da visualização, mas também busca constantemente aprimorar e modernizar as técnicas, garantindo que sua relevância perdure na era contemporânea.

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 conjuntos, 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 [89], Tufte estabelece uma definição fundamentada na clareza, precisão e eficiência na comunicação de ideias através de gráficos. Para Tufte, o princípio de **Excelência em um Gráfico** reside na capacidade de revelar dados de maneira transparente, priorizando a apresentação das informações em detrimento das computações estatísticas subjacentes. Nessa perspectiva, o propósito principal de um gráfico é servir como um veículo eficaz para a transmissão de mensagens e *insights* contidos nos

dados, destacando-se pela sua capacidade de ser compreendido de forma rápida e precisa. A sua excelência reside na sua capacidade de serem veículos eficazes de comunicação visual que transcendem a 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 e a integridade dos dados apresentados num gráfico. Para Tufte, um gráfico estatístico deve ser uma representação honesta e precisa dos dados, evitando distorções, omissões ou manipulações que possam conduzir a interpretações equivocadas. A integridade, nesse contexto, refere-se à responsabilidade do *designer* em assegurar que a representação visual seja fiel à realidade subjacente aos dados, permitindo que os leitores extraiam conclusões válidas e informadas. Em contraste com práticas que podem distorcer intencionalmente a percepção, o princípio de integridade reforça a ética na construção de gráficos estatísticos, enfatizando a transparência e a precisão como elementos fundamentais 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.

Com base numa Teoria de Gráficos de Dados em [89], Tufte postulou que os gráficos estatísticos devem direcionar a atenção do leitor para o significado e a substância dos dados, e não para outros elementos irrelevantes. A ênfase deve estar na comunicação efetiva das informações contidas nos dados, evitando distrações ou elementos visuais que não contribuem para a compreensão do conteúdo. Para garantir que a representação visual seja clara, direta e que facilite a interpretação correta dos dados apresentados, Tufte propus o terceiro princípio, **Maximização da Taxa de Tinta de Dados** em gráficos desenhados sobre folhas de papel

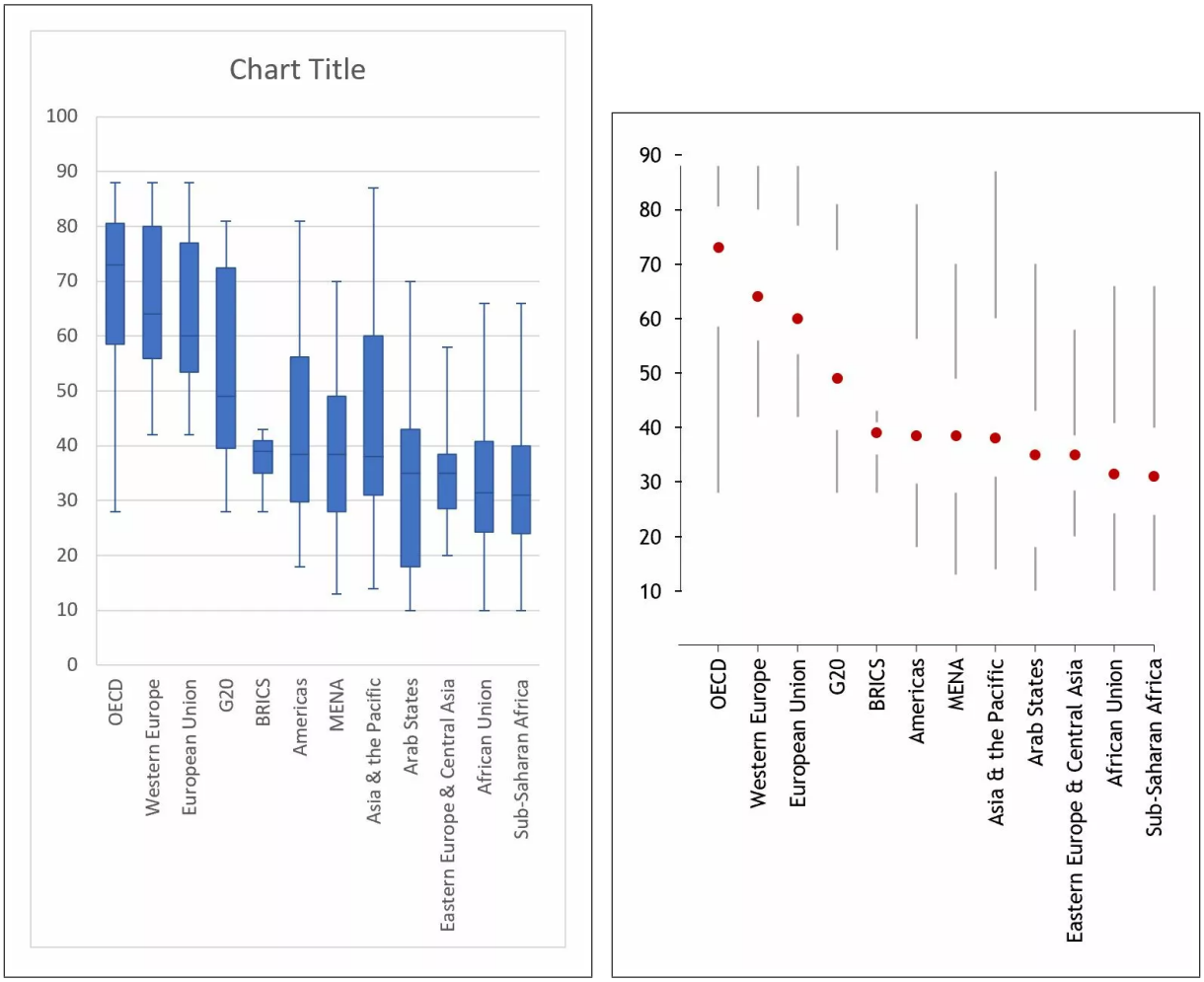
$$\text{Taxa de tinta de dados} = \frac{\text{Tinta de dados}}{\text{Quantidade total de tinta usada na impressão do gráfico}},$$

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

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

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 [47] e [4] exploram as funções disponíveis em R e Python, respectivamente, para criar gráficos estatísticos seguindo o paradigma de Tufte.



(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`)

Com o avanços da tecnologia de computação gráfica, Tufte expressa preocupação com a proliferação 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.

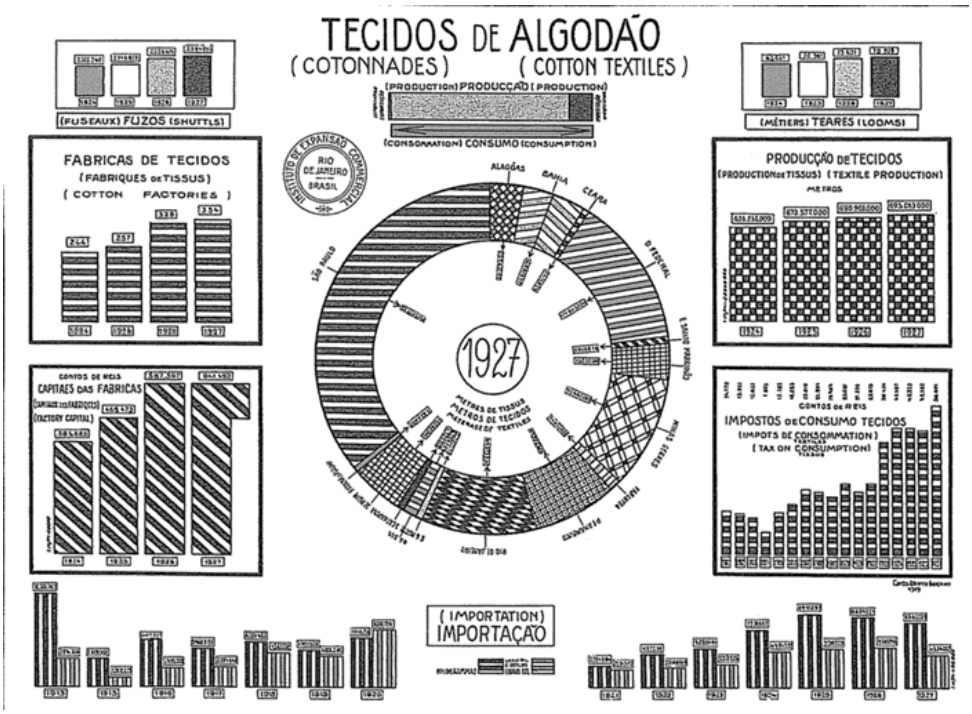


Figura 3.6: *Chartjunk*, como o efeito de moiré para sugerir vibrações ou movimentos, não contribui para a eficácia comunicativa dos gráficos de dados estatísticos; ao contrário, pode ser distrativo e até mesmo visualmente irritante. (Fonte: `https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/datapresentation/DataPresentation4.html`)

Para uma eficiente utilização da tinta (ou elementos gráficos) em um gráfico para representar informações relevantes, Tufte propôs mais um princípio, **Elemento Gráfico Multifuncional** para maximizar a eficácia de um elemento gráfico na transmissão de informações. Ele defende que a mesma tinta, ou elemento gráfico, pode servir a mais de um propósito, carregando informações de dados e, ao mesmo tempo, desempenhando funções de *design* que geralmente seriam atribuídas a elementos gráficos não relacionados a dados. Os gráficos estatísticos Isso permite uma representação mais compacta e eficiente de dados complexos e multivariados, como demonstra o gráfico na Figura 3.7 em que cada "barra" construída por números que são as divisões americanas presentes na França no perído de Junho de 1917 a Outubro de 1918 durante a Primeira Guerra Mundial [46]. No entanto, Tufte destaca que a implementação de elementos gráficos com múltiplas funções requer cuidado e sutileza. Quando bem projetados, esses elementos podem mostrar várias facetas de dados numa representação visual simplificada. No entanto, ele alerta para o perigo de elementos multifuncionais, pois podem criar quebra-cabeças gráficos, onde as codificações só podem ser compreendidas pelo criador do gráfico, tornando a interpretação mais desafiadora para o público em geral.

Na busca pela maximização da eficácia na comunicação visual de dados, Tufte introduziu e desen-

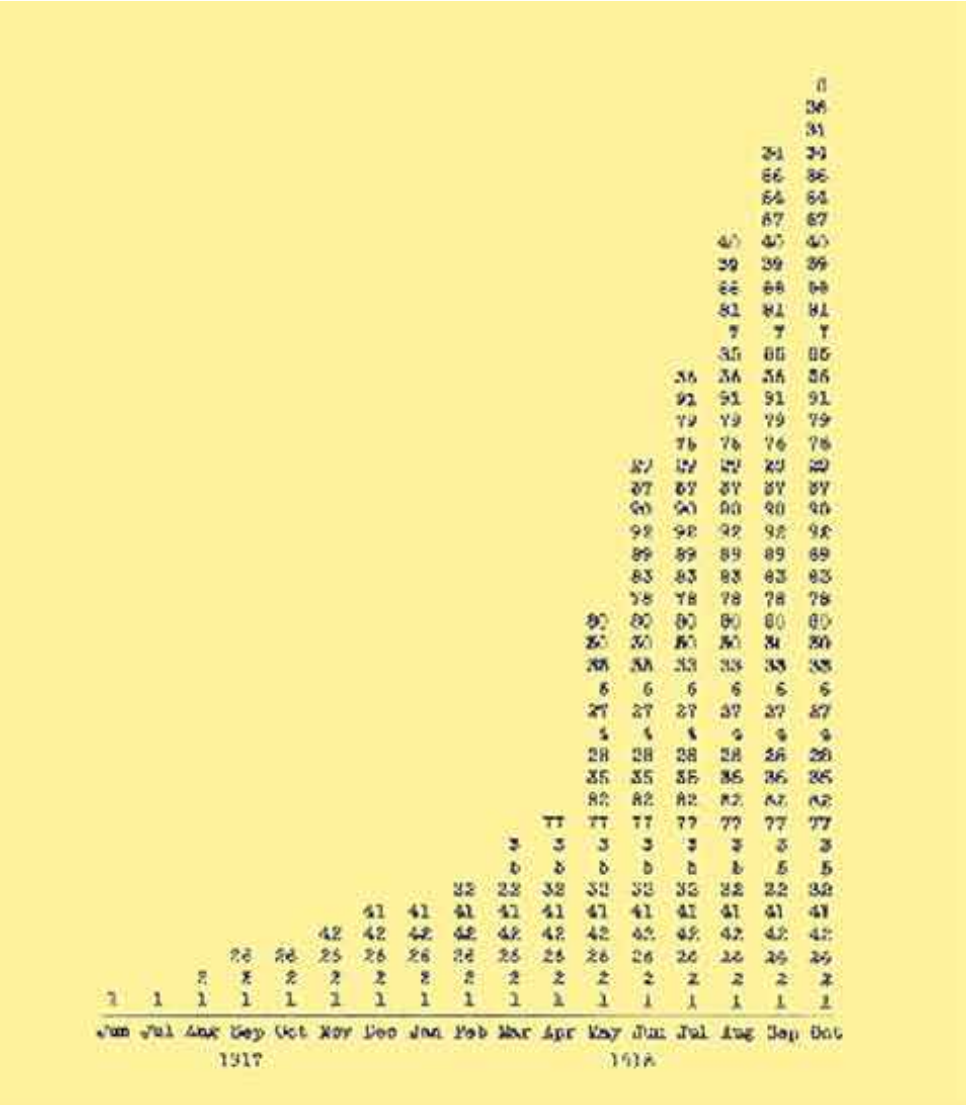


Figura 3.7: "Barra" multifuncional constituída pelos números das divisões americanas presentes no território francês durante o período de Junho de 1917 a Outubro de 1918 na Primeira Guerra Mundial. (Fonte: [46])

volveu o conceito de densidade de dados (*data density* em inglês) 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 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 mapeamento visual, que se alinham à linha de raciocínio desenvolvida nesta apostila.

3.4.1 Valores e Relações

A classificação de dados em **valores** e **relações** foi introduzida por Jacques Bertin [7]. Essa classificação é fundamental para o entendimento do modelo de dados relacional desenvolvido pelo Edgard F. Codd [13], no qual os **dados-valores** são representados por **atributos**. No contexto do modelo de dados relacional, os atributos representam características específicas associadas a uma **entidade** (como pessoas, lugares ou coisas), enquanto as **relações** referem-se às tabelas que armazenam esses atributos. As relações representam as associações entre diferentes entidades por meio de **chaves estrangeiras**, proporcionando uma maneira eficaz de organizar e representar dados complexos e fa-

cilitando a compreensão das interações e conexões em um sistema. 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).

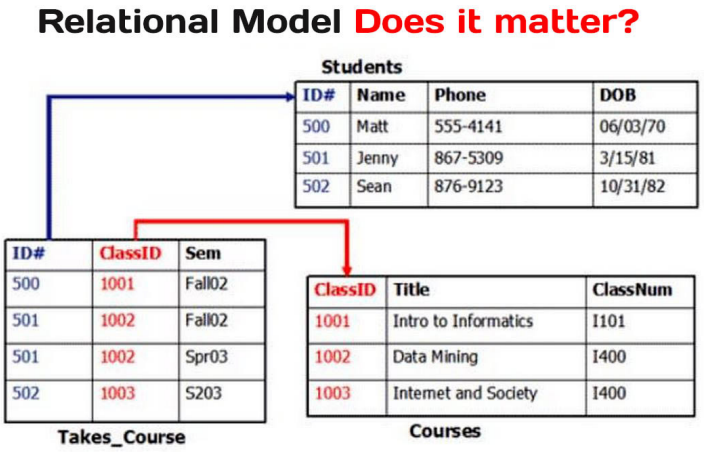


Figura 3.8: Modelo relacional de um conjunto de dados de alunos e disciplinas cursadas.

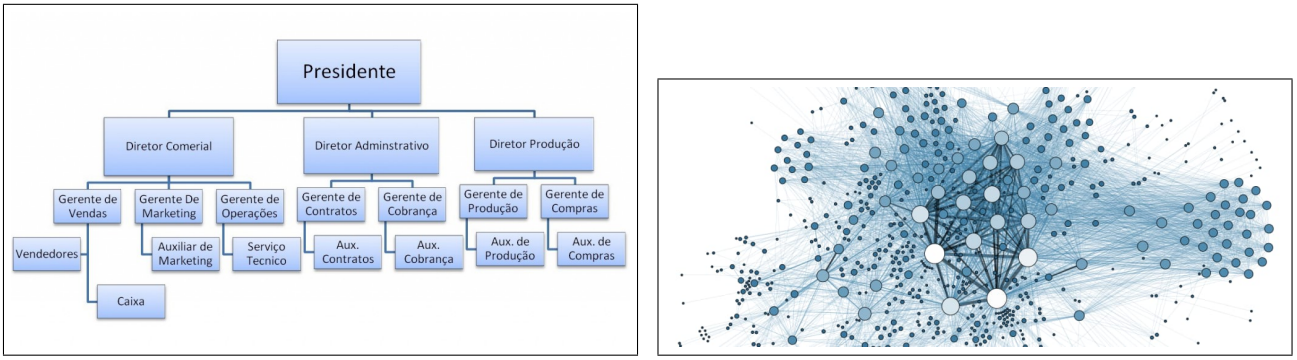
Em busca de um mapeamento sistemático dos valores dos dados em elementos gráficos durante a análise visual de dados, é relevante distinguir entre valores quantitativos e qualitativos. Os **valores quantitativos** podem ser subdivididos em discretos e contínuos. **Valores discretos** representam contagens ou números inteiros específicos, enquanto **valores contínuos** representam grandezas que podem assumir qualquer valor dentro de um intervalo da reta real. Por outro lado, os **valores qualitativos**, ou **categóricos**, incluem categorias que, por sua vez, podem ser nominais ou ordinais. Os **valores nominais** representam categorias sem uma ordem intrínseca, como cores ou tipos de produtos. Já os **valores ordinais** representam categorias com uma ordem, mas a distância entre elas não é significativa, como níveis de educação ou grau de satisfação de um serviço.

Fazendo uma correlação com os tipos de dados apresentados na Seção 3.1, os tipos de dados 1D, 2D, 3D e multidimensional propostos por Schneiderman representam 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. Além disso, a relação TAKES_COURSE é classificada como um tipo de dado 1D. Quando os valores de um atributo de uma entidade estão relacionados ao aspecto temporal dos dados em estudo, a entidade é considerada por Schneiderman do tipo de dado **temporal**.

Os tipos de dados em árvore e em rede (*networks*), destacados por Schneiderman, representam as relações entre os itens. No contexto de **árvores**, as relações podem ser estruturadas hierarquicamente, seguindo um padrão de ramificação que parte de um item central para outros itens, formando uma estrutura semelhante a uma árvore, como ilustra a Figura 3.9a. Essa estrutura é adequada para repre-

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

sentar relações de subordinação e dependência. Já **em redes**, as relações são mais complexas e podem ser estruturadas ou não. Nesse cenário, os itens podem ser representados por nós interconectados por arestas, cada uma indicando uma relação. **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 grafos, 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.

É relevante observar que padrões, tendências informativas e causas-efeitos emergem não apenas das relações entre itens, mas também das relações dentro dos próprios itens, como evidenciado na análise da interconexão entre diversas medidas climáticas. Essas relações intra-itens, frequentemente não explícitas, desempenham um papel crucial na compreensão dos dados, fornecendo *insights* sobre como as variáveis individuais se relacionam e influenciam mutuamente. Ao explorar tanto as relações inter-ítens quanto as relações intra-itens, os analistas de dados podem obter uma visão mais completa e significativa do contexto e da dinâmica subjacente aos dados em estudo. Em diversos cenários, a descoberta dessas relações, tanto inter quanto intra-itens, a partir de um amontoado de dados-valores brutos, representa um dos maiores desafios para os cientistas de dados, sendo objeto de pesquisa constante na área da análise de dados visual.

3.4.2 Mapeamento Visual

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 manifestações físicas, conhecidos como **dados abstratos**. Dados físicos compreendem medidas tangíveis, como volume, comprimento e posição espacial, enquanto dados abstratos encapsu-

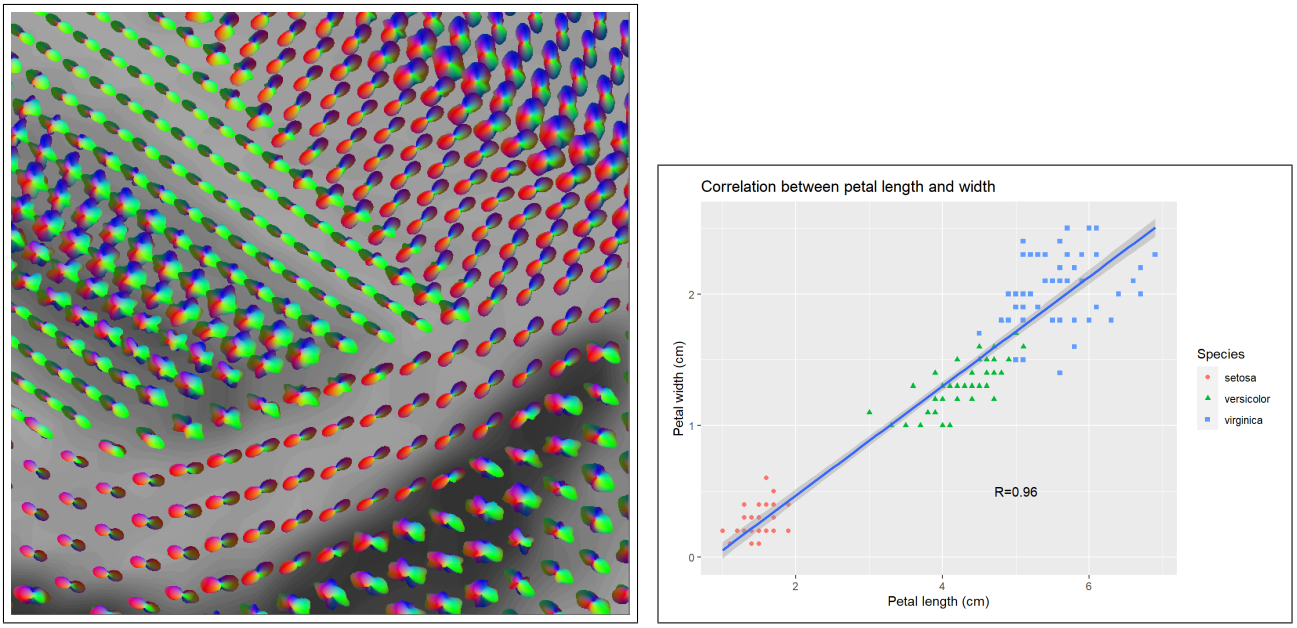
lam conceitos intangíveis, como sentimentos ou tendências. Seguindo a classificação de Schneiderman, os dados com manifestação física podem ainda ser categorizados em 1D, 2D e 3D, dependendo da quantidade de coordenadas espaciais associadas a eles. Os dados 1D são representados por curvas (1 coordenada), os 2D por superfícies (2 coordenadas) e os 3D por sólidos (3 coordenadas). Na representação visual desses dados, a aplicação do princípio de **Consistência e Padronização** direciona o uso das coordenadas espaciais para posicionar os elementos de maneira coerente na tela do computador. Além disso, em diversas situações, observa-se que o princípio de **Organização Perceptual** é atendido, facilitando a identificação e compreensão de padrões relevantes que contribuem para a resolução efetiva de problemas.

Para os dados abstratos, que carecem de uma realização física, é necessário estabelecer uma correspondência arbitrária entre os dados e uma realização física em *pixels*. Esse processo representa um desafio adicional para os cientistas na área da análise de dados visual. A abstração se revela também na interpretação de dados que possuem características tangíveis e mensuráveis, quando os padrões complexos, tendências e relações subjacentes não são diretamente visíveis na forma bruta dos dados-valores. Por exemplo, embora os dados meteorológicos, como temperatura, pressão atmosférica e umidade, estejam relacionados a fenômenos físicos concretos, a disposição desses dados numa tela, de acordo com a expressão física, nem sempre revela padrões significativos. A complexidade desses dados exige uma análise que vá além da simples associação a amostras espaciais. Mesmo que esses dados tenham uma base tangível, uma análise abstrata é preferível para extrair informações significativas sobre padrões climáticos, mudanças sazonais ou eventos extremos. Porisso, esses dados são considerados dados abstratos.

A análise de dados visual representa uma abordagem integrada que combina elementos da **visualização científica** dos dados físicos (Seção 3.2) e da **visualização de informação** dos dados abstratos (Seção 3.3), juntamente com a facilitação da **interação homem-computador** (Seção 3.1). Ao unir esses componentes, a análise de dados visual proporciona uma abordagem abrangente para explorar e comunicar informações a partir de uma diversidade de dados.

A **visualização de dados físicos** frequentemente utiliza extensivamente técnicas tridimensionais de renderização, representando curvas, superfícies, volumes e estruturas complexas, aproximando-se o máximo possível da nossa forma de percepção. 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 **visualização de dados abstratos** se concentra na comunicação de informações mais amplas, muitas vezes utilizando **gráficos estatísticos** (bidimensionais) para facilitar a compreensão. Nesse caso, o princípio de Organização Perceptual orienta que os dados sejam posicionados na tela de modo a propiciar a percepção de padrões relevantes. 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

[15]: 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 atributos, comprimento e largura, são mapeados em coordenadas x e y, respectivamente.



(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.

3.5 Gramática dos Gráficos

A Gramática dos Gráficos (*Grammar of Graphics (GoG)* em inglês), conceito desenvolvido por Leland Wilkinson, é uma abordagem sistemática e compreensível para a construção de **gráficos estatístivos**, independentemente da complexidade dos dados. Em vez de depender de um conjunto fixo de gráficos predefinidos, a GoG permite a combinação de elementos gráficos básicos para criar uma ampla variedade de visualizações. Em [104], Wilkinson propôs uma linguagem unificada para a criação de gráficos estatísticos. A biblioteca ggplot2 da linguagem R [96] é uma implementação prática dos princípios da Gramática dos Gráficos desenvolvida por Hadley Wickham.

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 visuais ajudam a garantir a precisão e a interpretação adequada das informações contidas nas representações visuais de dados. Em [105], ele destaca que os gráficos estatísticos são diferentes de outras visualizações, como mapas, diagramas e representações tridimensionais de volumétricas. A gramática dos gráficos estatísticos é fundamentada em conceitos matemáticos que permitem representar visualmente funções estatísticas aplicadas sobre os dados em análise.

A Gramática dos Gráficos define dois tipos de dados fundamentais, variável e conjunto de variáveis (*variable set – varset* em inglês). Uma **variável** X é uma função f que associa um conjunto de objetos O a um conjunto de valores V provenientes dos dados-valores brutos. Esses dados podem ser organizados em uma ou mais tabelas, como ilustra a Figura 3.8. Por outro lado, um **conjunto de variáveis** X é a função f que associa um conjunto de valores a um conjunto $\tilde{O}$ composto por todas as possíveis combinações ordenadas de objetos geradas a partir do conjunto O. A GoG estabelece 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 para guiar a criação e interpretação de gráficos estatísticos, facilitando a comunicação visual de padrões e tendências nos dados:

Variáveis (*Variables*): Transformar cada tabelas de dados-valores num 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 uma distância ou intervalo entre marcas nos eixos, que ajudam na interpretação e leitura dos dados renderizados, para os conjuntos de variáveis de acordo com as dimensões da área onde os gráficos estatísticos serão renderizados.

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

Geometria (*Geometry*): Especificar os tipos de marcas visuais, como pontos, arestas, linhas, polígonos, barras e contornos, utilizadas para representar os dados.

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

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

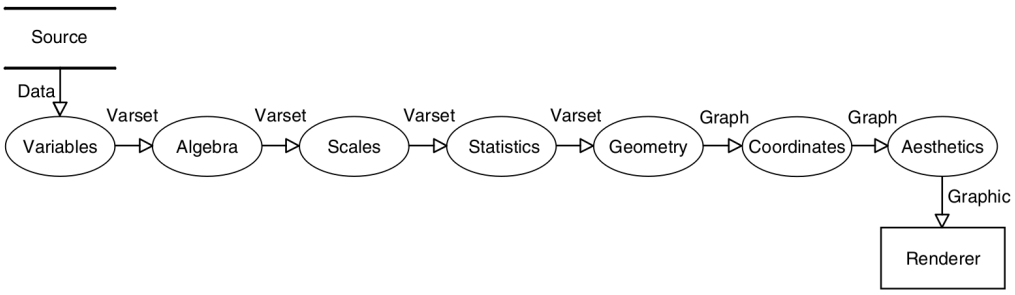


Figura 3.11: Fluxo de dados de acordo com as normativas estabelecidas pela Gramática dos Gráficos proposta por Leland Wilkinson. (Fonte: [105]

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 [97]. 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 [99]. A **folha de dicas do ggplot2** (*ggplot2 cheat sheet*, em inglês) [23] é um recurso de referência rápida que resume os principais conceitos e funcionalidades do ggplot2. Essas folhas de dicas são úteis para usuários iniciantes e experientes, fornecendo uma visão geral dos recursos do ggplot2 e exemplos de código para criar gráficos comuns. Uma descrição

completa de todas as funcionalidades de ggplot2 se encontra em [100].

O pacote plotnine [1, 51] é uma implementação em Python de uma gramática de gráficos, modelada após o ggplot2. Assim como seu homólogo em R, essa gramática permite a composição de gráficos ao mapear explicitamente variáveis em um dataframe para os elementos visuais que compõem o gráfico. Janssen compilou uma série de heurísticas [37] para auxiliar na transição de código do ggplot2 em R para o plotnine em Python, cujas funções são resumidas em [32]. 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 [70], ainda não foram diretamente implementadas para Python como pacotes independentes sob os mesmos nomes. Existem, no entanto, alternativas similares que podem ser reproduzidas com bibliotecas Python padrão ou através da combinação de várias bibliotecas de visualização de dados. É importante destacar que essas alternativas não podem ser usadas diretamente como novas camadas em plotnine.

Ambos os pacotes ggplot2/plotnine ilustram o poder de uma gramática de gráficos tanto em R quanto em Python. Com eles, gráficos personalizados (e de outra forma complexos) são fáceis de pensar e construir incrementalmente, enquanto os gráficos simples continuam simples de criar.

3.5.3 Exemplo

Esta seção demonstra, utilizando o conjunto de dados (*dataset* em inglês) DIAMONDS disponível no pacote tidyverse [96], algumas das funcionalidades das classes da Gramática dos Gráficos implementadas no pacote ggplot2. As linhas de comando usadas são inclusas para possibilitar a reprodução do processo.

Primeiramente, o conjunto de dados DIAMONDS do pacote tidyverse é carregado com as seguintes duas linhas de comando:

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

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

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)
```

```
> 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.

```
> 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*).

gera um novo conjunto de variáveis

Para ilustrar o efeito da etapa **Escalas** visualmente, é conveniente mostrar o efeito da etapa **Geometria** que mapeia cada variável a um elemento visual. Ao mapearmos as variáveis PRICE, CARAT e CUT em coordenada x, coordenada y e cor, obtivemos o gráfico apresentado na Figura 3.15.

```
ggplot (data = algebra) +
```

```
> 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 as variáveis CARAT, CUT e PRICE do conjunto de dados DIAMONDS.

```
  geom_point (mapping = aes(x=price,y=carat,color=cut))
```

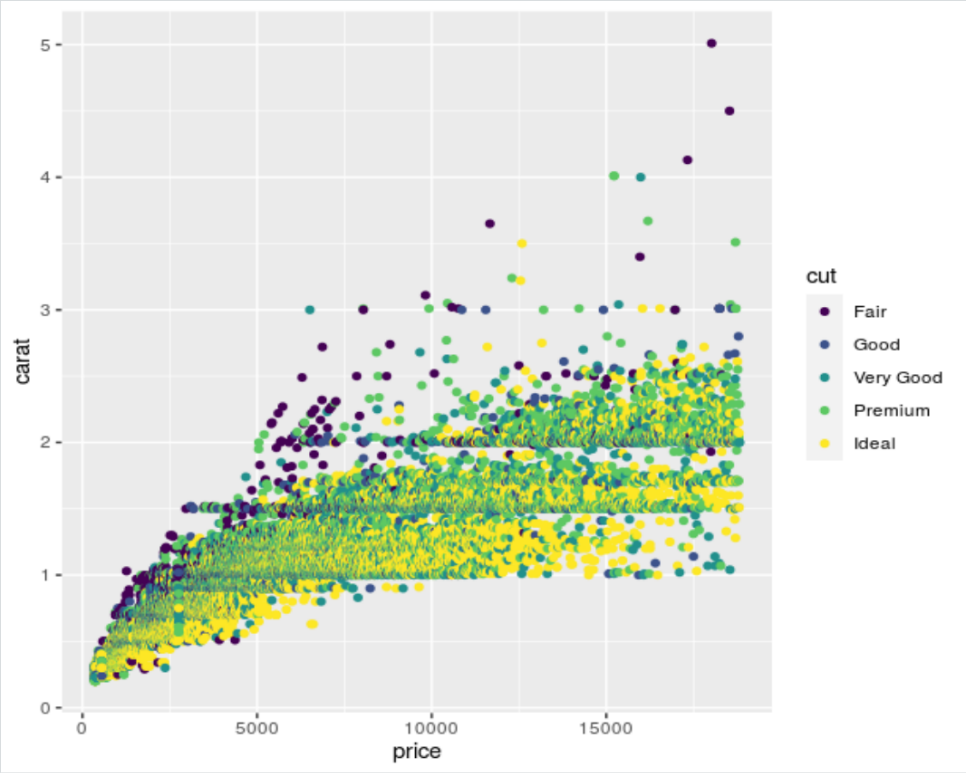


Figura 3.15: Operação de produto entre as variáveis CARAT, CUT e PRICE do conjunto de dados DIAMONDS.

Note que nada foi especificado sobre os intervalos entre as marcas ao longo dos dois eixos. O sistema usou os **valores padrão** estabelecidos na **Hierarquia de *defaults***. No entanto, se o usuário quiser personalizar os intervalos entre as marcas, pode reconfigurá-los na etapa **Escalas** como ilustra as seguintes linhas de comando.

```
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))
```

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

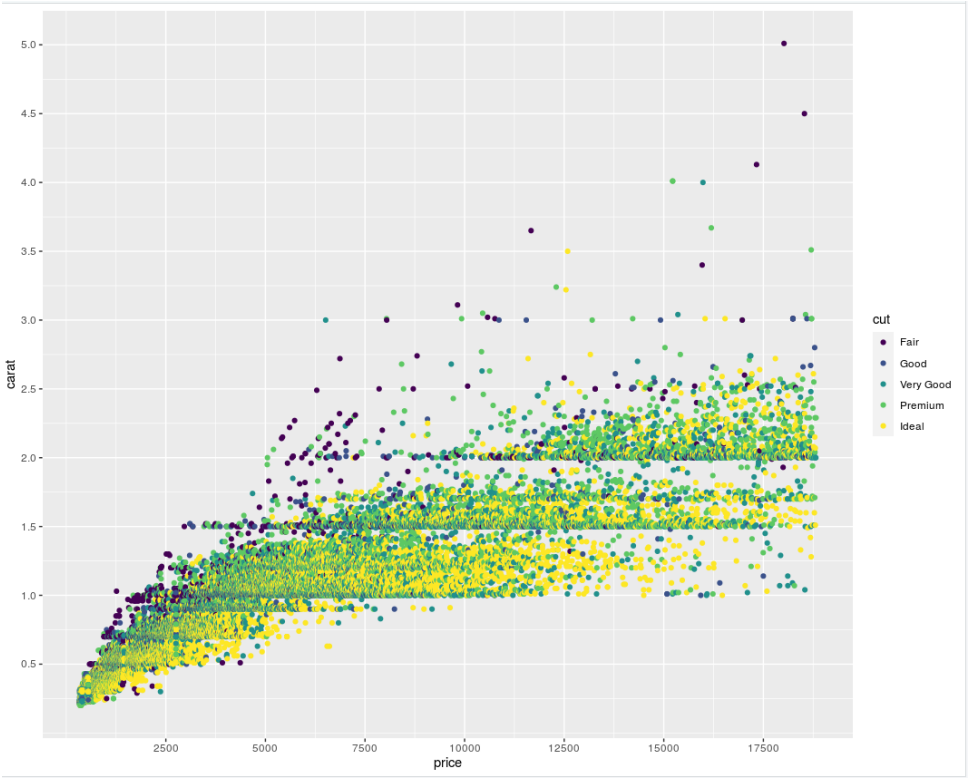


Figura 3.16: Personalização das escalas dos gráficos estatísticos.

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)
```

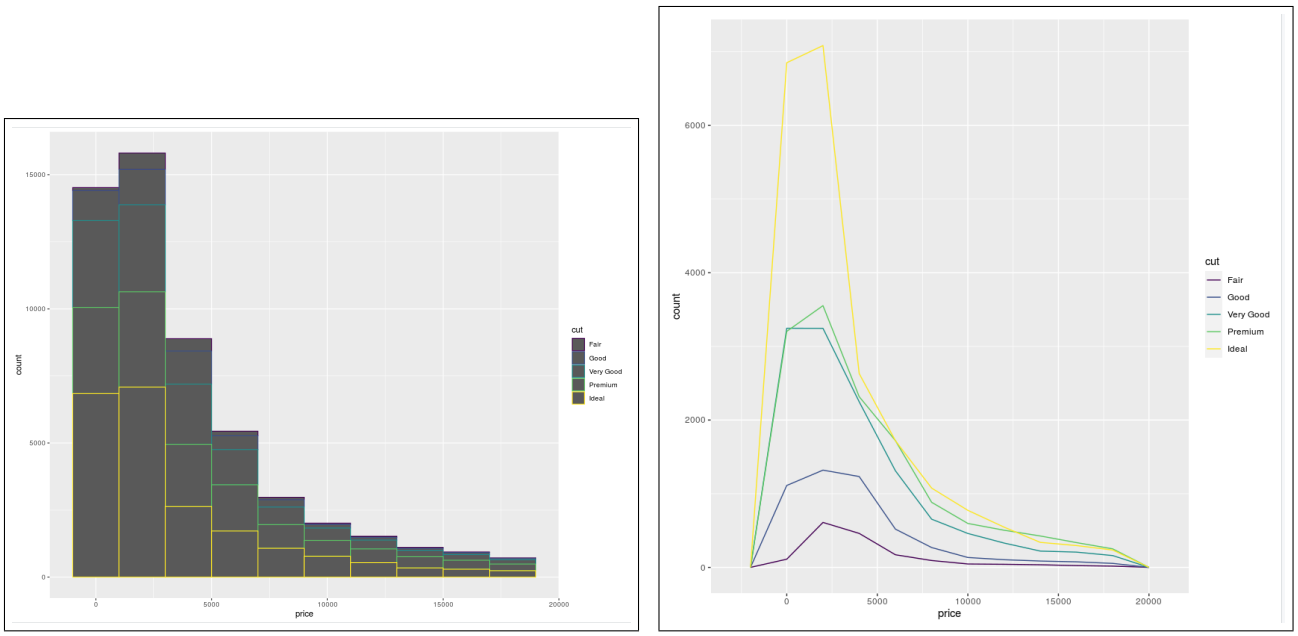
Os resultados foram $\mu_{carat} = 0.7979397$, $\sigma_{carat} = 0.4740112$, $Md_{carat} = 0.7$, μ_{price} = US$ 3932.8, σ_{price} = US$ 3989.44, Md_{price} = US$ 2401.

Na etapa **Geometria** é especificada a forma geométrica dos gráficos visualizados. Os seguintes dois comando definem, respectivamente, a geometria de histograma e de gráfico de frequências de ocorrência de cada valor de preço nos 53.940 itens do conjunto DIAMONDS. Conforme Ismay e Kim argumentam, com um conjunto básico de cinco gráficos - gráficos de dispersão, gráficos de linha,

gráficos de caixa, histogramas e gráficos de barras - é possível visualizar uma ampla variedade de tipos de variáveis diferentes. Esses gráficos foram denominados os **5 Gráficos Nomeados** (5NG, do inglês *5 Named Graphs*) [34].

```
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)
```

Figura 3.17 apresenta duas versões de gráficos estatísticos para visualizar uma mesma informação.



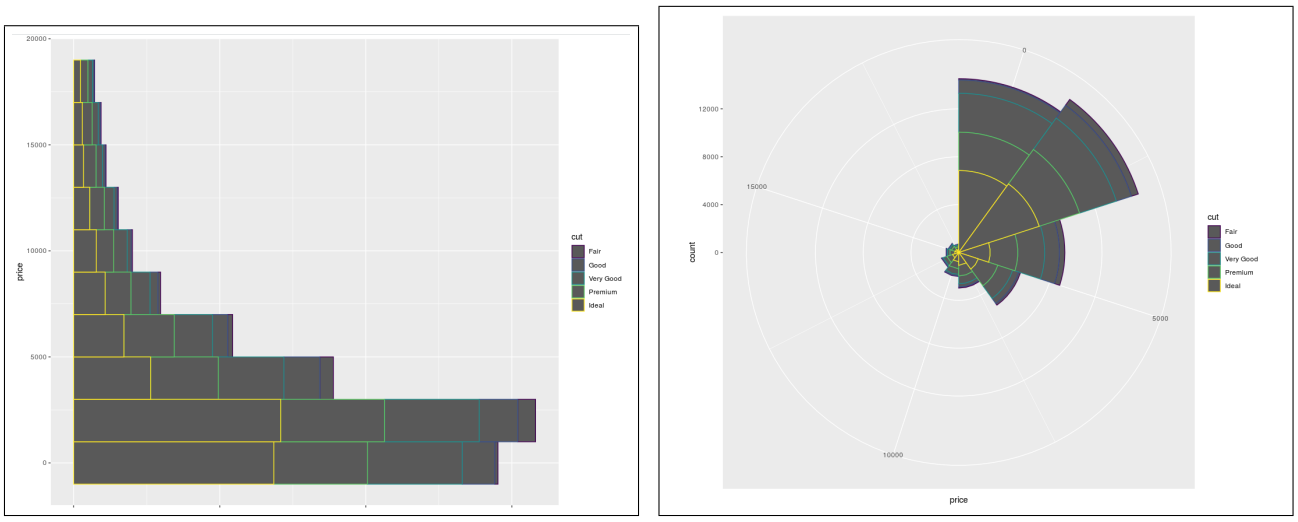
(a) Histograma (b) Gráfico de Frequências

Figura 3.17: Diferentes geometrias para expressar frequências de ocorrência.

Na Figura 3.18 são apresentadas as funções integradas na etapa **Coordenadas**, as quais possibilitam alternar facilmente as representações de um mesmo conjunto de dados em diferentes sistemas de coordenadas. Isso é alcançado por meio dos dois comandos subsequentes:

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

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



(a) Inversão de Eixos (b) Coordenadas Polares

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

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

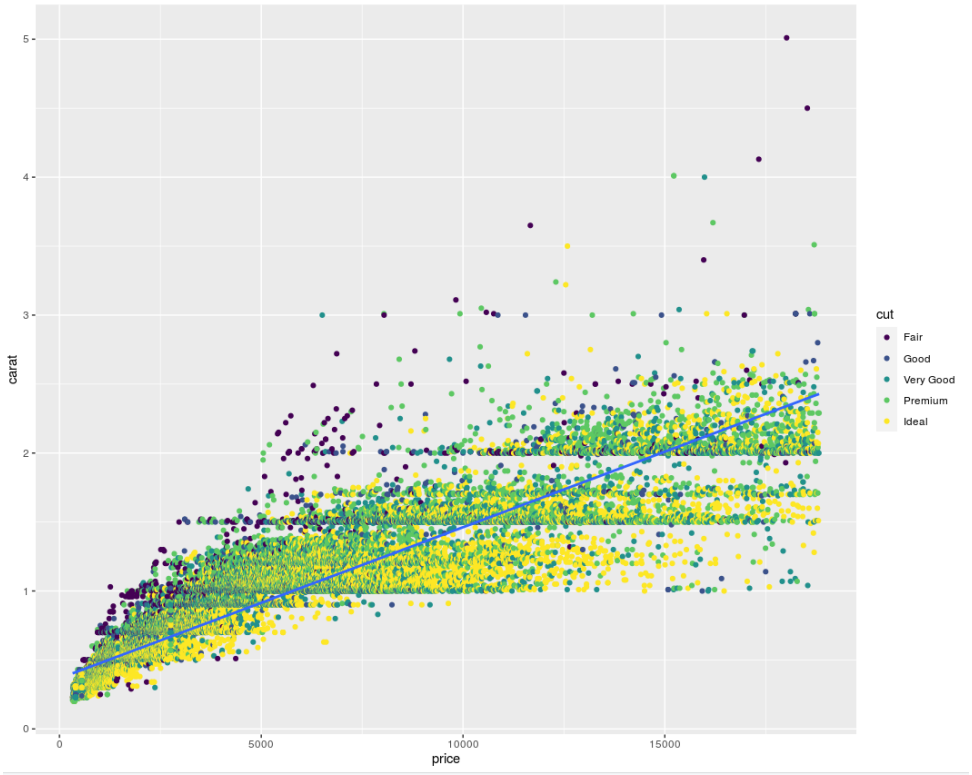


Figura 3.19: Gráfico composto de duas camadas: uma camada de gráfico de dispersão e uma camada de correlação entre o preço e o quilate dos diamantes.

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

```
  geom_smooth(method = "lm", se = FALSE)
```

Quando se deseja analisar a distribuição ou relação entre variáveis em diferentes categorias, proporcionando uma interpretação mais clara, a visualização multifacetada é empregada. Através do seguinte comando, o gráfico de dispersão da Figura 3.15 é desdobrado em pequenos gráficos de dispersão individuais para cada uma das 5 categorias de cortes de diamantes.

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

A Figura 3.20 revela 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.

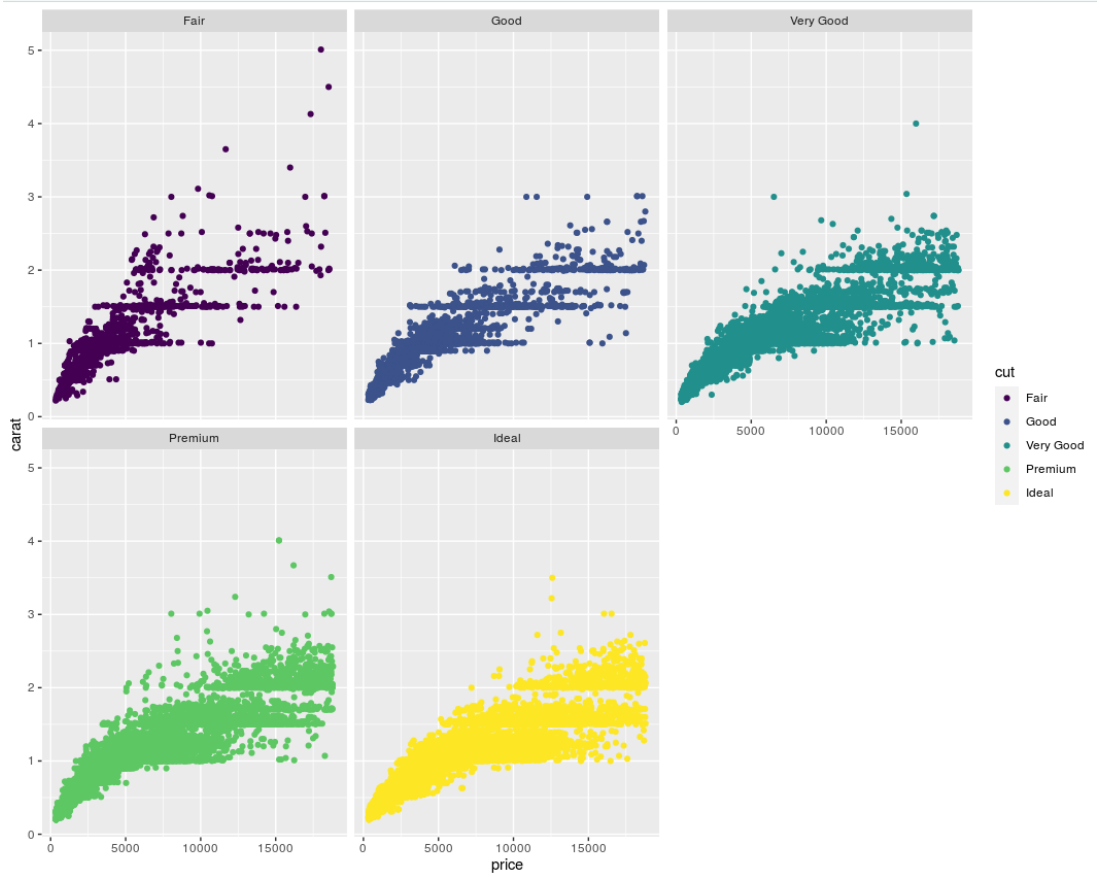


Figura 3.20: Uso de pequenos múltiplos de gráficos de dispersão para identificar padrões específicos em diferentes categorias de cortes de diamantes.

3.6 Considerações Finais

Ao explorar o *design* de interfaces com base nos princípios de Schneiderman, Ware e Tufte, percebemos a necessidade de uma abordagem cuidadosa, destacando a importância da clara separação entre as

áreas de controle e visualização. A ênfase na usabilidade, *feedback* imediato e interação intuitiva proposta por Schneiderman na área de controle é fundamental para garantir uma experiência do usuário eficaz e satisfatória (Seção 3.1). Ware, por sua vez, ressalta a distinção entre dados com e sem expressão espacial na área de visualização, uma diferenciação crucial para orientar as estratégias de representação e renderização.

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 [85] e abstratos [54] complexos. No entanto, o verdadeiro desafio reside na habilidade de mapear esses dados em elementos gráficos que não apenas sejam esteticamente agradáveis, mas que também facilitem a extração de informações cruciais 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.

A capacidade de transformar dados em narrativas visuais claras e envolventes é um aspecto essencial do *design* de interfaces efetivas. Neste contexto, a Gramática dos Gráficos Estatísticos emerge como uma ferramenta poderosa no arsenal de um *designer* de interfaces. Ao seguir os princípios da gramática, é possível criar visualizações coesas e significativas, garantindo que a representação gráfica seja precisa e compreensível para análise de dados visual. A flexibilidade proporcionada pela gramática permite adaptar as visualizações de acordo com a natureza dos dados abstratos, permitindo que a interface transcenda as limitações dos dados físicos. Integrar os princípios de Schneiderman, Ware e Tufte, juntamente com a aplicação cuidadosa da Gramática dos Gráficos Estatísticos, representa uma abordagem holística para criar interfaces que capacitam os usuários a explorar e compreender **dados abstratos** de maneira eficiente e a construir modelos subjacentes a esses dados.

3.7 Exercícios

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

   Interface do usuário para aplicativos móveis ou *web* `https://synodus.com/blog/web-development/bad-ui-design/`

   Coleção de erros comuns de *design* de interface `http://hallofshame.gp.co.at/shame.htm`

   Exemplos de *design* de interface inadequdo de forma geral `https://www.interaction-desi`
   `org/literature/article/bad-ui-design-examples`

   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.

2. Reescreva em Python, usando a gramática dos gráficos implementada no pacote plotnine [51], os códigos em R no

   - exemplo na Seção 8.1 na referência [65].
   - exemplo explorado na Seção 3.5.3.
   - caso 1 da Seção 10.1 na referência [65].
   - caso 2 descrito na Seção 10.5 na referência [65].

   Explique na documentação Markdown os passos aplicados na geração de cada gráfico. Os três conjuntos de dados usados podem ser, respectivamente, carregados seguindo o seguinte procedimento:

   murders : Baixe de `https://www.dca.fee.unicamp.br/cursos/IA376I/datasets/murders.csv` e carregálo com o comando

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

   Obs.: O conjunto de dados MURDERS é do pacote dslabs de R. Uma outra alternativa para carregá-lo em Python é usar comandos R no ambiente Python via rpy2 conforme mostrado na Seção 2.4.

   diamonds : É integrado no pacote plotnine. Basta carregá-lo com a linha de comando from plotnine.data import diamonds

   gapminder : Deve-se instalá-lo e carregá-lo usando os comandos

   ```
   pip install gapminder
   from gapminder import gapminder
   ```

   Para verificar as características carregadas, use o comando

   ```
   gapminder.head()
   ```

3. Utilize a função ggtitle para adicionar uma camada de título principal para os 4 gráficos construídos.

Capítulo 4

Percepção 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, desempenhando um papel crucial na análise de dados. Em um contexto onde a capacidade de identificar rapidamente informações relevantes é crucial, uma abordagem pré-atentiva na interface gráfica pode desempenhar um papel significativo para aprimorar a eficácia da análise.

Neste capítulo, 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 de dados visual, 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 divrsas 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, convergindo na retina, localizada na porção posterior do olho. A retina contém cerca de centenas de milhões de fotoreceptores, incluindo bastonetes e cones. 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 (*point spread function* – PSF, em inglês). Os **movimentos oculares**, como movimentos sacádicos, são, de fato, os responsáveis pela captura de uma sequência de outros ângulos dos objetos no campo visual e, portanto, pela exploração do ambiente e pela distinção inconsciente de detalhes.

A retina é coberta por células fotoreceptoras conhecidas como bastonetes e 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, mas são cruciais em ambientes de baixa luminosidade. 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), distinguindo 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.

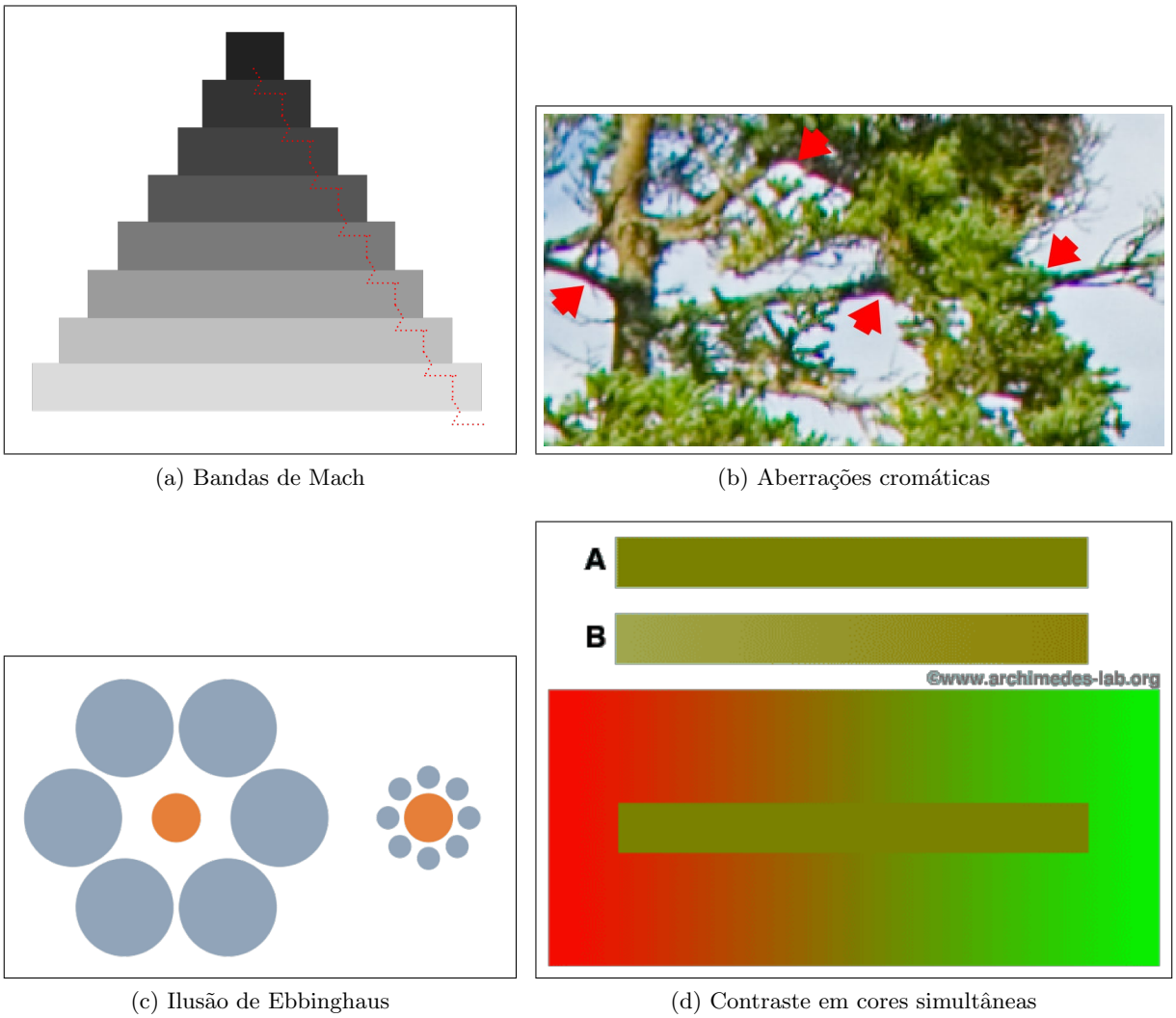
Canal de Cor Vermelha-Verde : é formado pela diferença na atividade dos cones de comprimento de onda longo (L) e médio (M). A variação na atividade desses cones cria uma resposta relativa à luz nas porções vermelha e verde do espectro de cores. O canal de cor vermelha-verde é essencial para discriminar 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 é crucial para discriminar 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. Posteriormente, essas características simples são agrupadas em padrões (visão de baixo nível). Conforme a atenção do observador, objetos armazenados em sua memória são "evocados" 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 exploradas com mais detalhes na Seção 4.3.

Essa complexa cadeia de eventos destaca a importância da **acuidade visual**, onde a mínima diferença entre dois elementos, seja em cor, forma ou tamanho, é fundamental para a percepção distinta. Além disso, a **ação inibidora** fora do centro do campo receptivo de um fotoreceptor contribui para a percepção visual, fazendo 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, 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, revelando as limitações e as peculiaridades do sistema visual humano. (Fontes: `https://pt.wikipedia.org/wiki/Bandas_de_Mach`, `https://www.phototraces.com/b/chromatic-aberration-photography/`, `https://www.cleartelligence.com/post/optical-illusions-and-data-visualization` e `https://www.archimedes-lab.org/color_optical_illusions.html`)

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 [33], no entanto, mesmo nesses casos, a percepção de profundidade não é tão comprometida. Outros elementos, 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 destaca pela estabilidade na percepção de certos atributos fundamentais (como tamanho, forma e cor) de forma consistente, independentemente das variações nas condições de observação, como distância, ângulo e iluminação. Este fenômeno é reconhecido como a **constância perceptiva**, onde a apreensão consistente desses atributos essenciais destaca a notável capacidade do sistema visual em filtrar variações externas, mantendo uma perceção estável e confiável. 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 como o cérebro leva em consideração o contexto ao processar as 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 conjuntos de dados. 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 e propósito. Elas podem oferecer uma abordagem única e interessante para a apresentação visual dos dados, como abordar a incerteza inerente aos conjuntos de dados.

Por último, devido às nossas limitações nos recursos cognitivos, particularmente na memória, ao nos envolvermos em ambientes visuais saturados de informações, inúmeros estímulos competem pela nossa atenção. Apenas alguns, com características visuais distintas, conseguem destacar-se e são processados de maneira **pré-atentiva**, ocorrendo em um nível inconsciente. Por outro lado, diversos outros estímulos podem ser descartados antes mesmo de passarem pelo processo de reconhecimento. Vale notar que, em ambientes complexos, alterações sutis, mesmo que relevantes, muitas vezes passam despercebidas. Esse fenômeno é conhecido como **cegueira às mudanças** (*change blindness* em inglês), evidenciando como pequenas modificações em um cenário podem escapar da nossa percepção, especialmente quando nosso foco está em outros elementos como demonstra o vídeo em [12].

4.2 Atributos Pré-atentivos

Para otimizar a busca visual por dados de interesse em um extenso conjunto de informações, 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 [93], 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 [93] elementos pré-atentivos e atentivos por meio de uma série de exemplos. Ele considera 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.

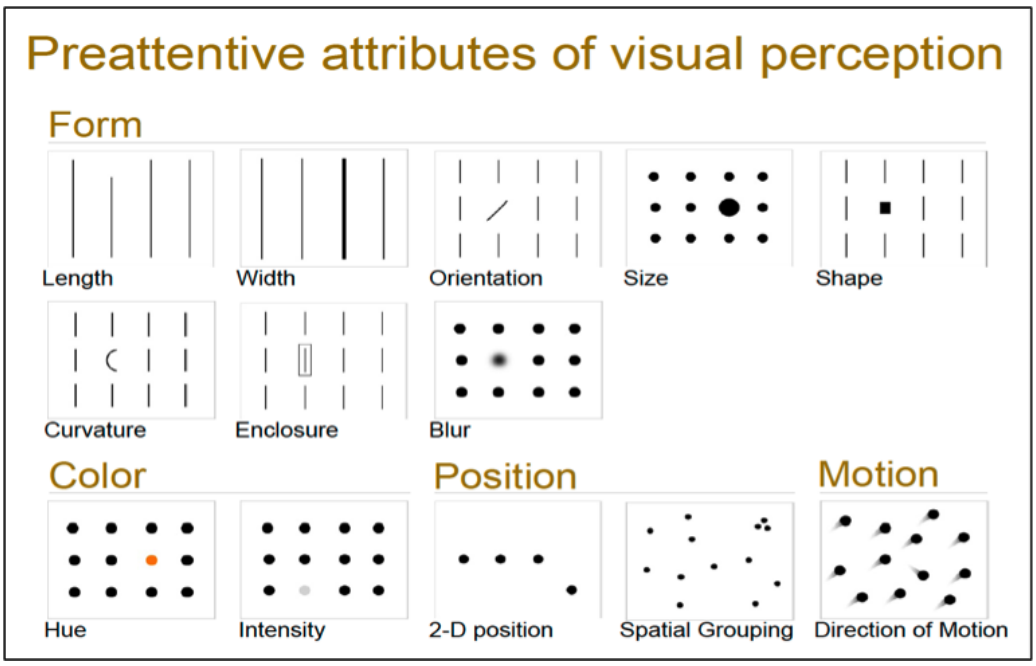


Figura 4.2: Atributos pré-atentivos para percepção visual humana. (Fonte: `https://www.relationshipone.com/blog/consumable-analytics-steps-to-making-your-dashboards-delicious/`

Cabe salientar que 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 se destacar de outras marcas dentro do mesmo canal. Por exemplo, a incorporação de marcas, denominadas por Ware como **assimetrias** (*asymmetries*, em inglês), 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 os dados físicos, cujo domínio "nativo"geralmente é contínuo, é que, mesmo quando amostrados em dados discretos, a visualização contínua dos valores de um atributo escalar, como a temperatura em um mapa de aquecimento global, tende a ser mais intuitiva. Por esse motivo, é comum realizar interpolação nos valores dos elementos gráficos nos quais os atributos físicos são mapeados. Dessa forma, existe uma preferência por elementos gráficos interpoláveis, como cores e iso-níveis, como exemplificado em [86], 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:

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

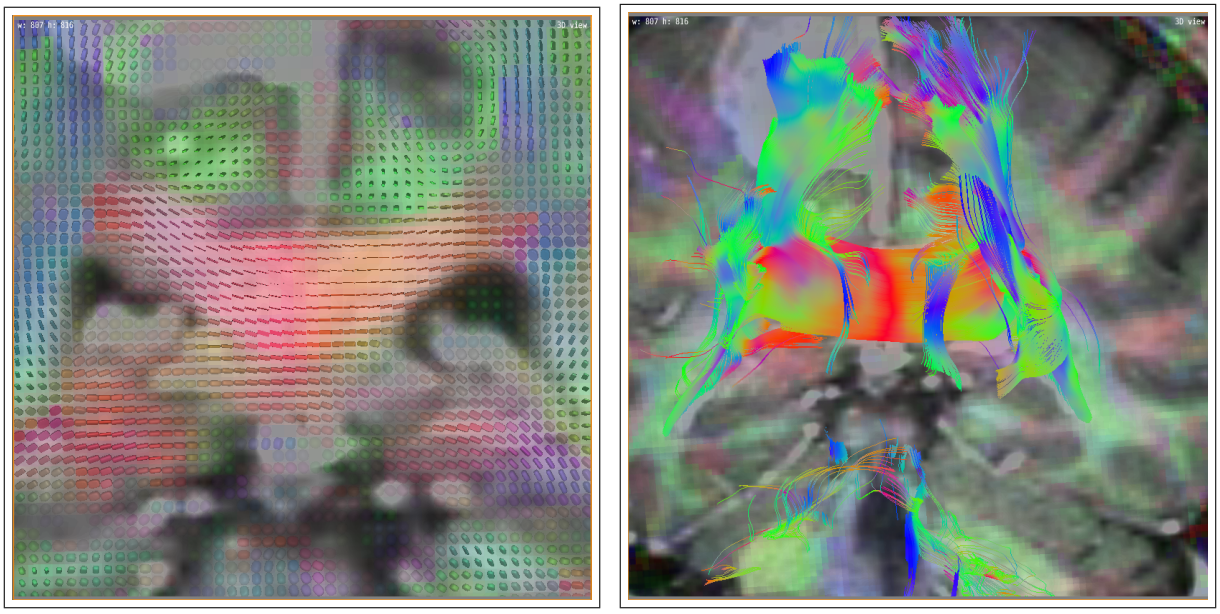
Outra questão explorada pelo Ware é a forma de codificar os valores multidimensionais dos dados em elementos gráficos correspondentes. A prática mais comum adota o uso de elementos discretos conhecidos como **glifos**. Quando as dimensões codificadas nos glifos podem ser visualmente integradas em um 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 [93] uma escala de combinações de características visuais para atributos bidimensionais, indo desde 100% integradas até 100% separadas. 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 ou glifos gráficos:

Proximidade: símbolos e/ou 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.

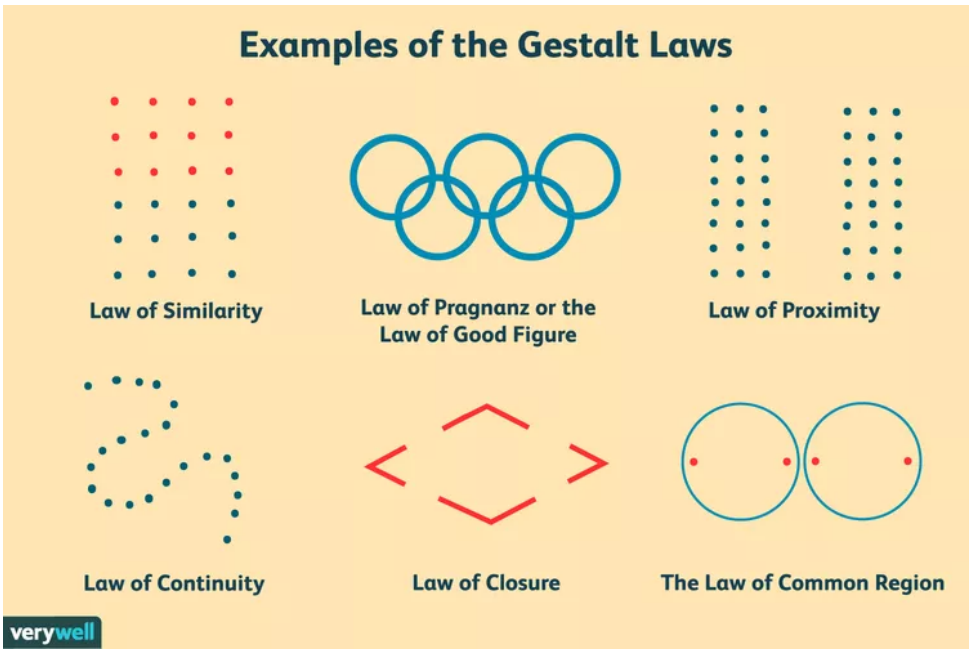


Figura 4.4: Leis de Gestalt: princípios psicológicos que descrevem como os seres humanos percebem e organizam elementos visuais em padrões significativos e estruturados. (Fonte: `https://www.verywellmind.com/gestalt-laws-of-perceptual-organization-2795835`)

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

Conectividade: conexões entre os símbolos ou glifos gráficos 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 símbolos ou glifos gráficos.

Simetria: disposição simétrica dos símbolos ou glifos gráficos facilita comparações.

Fechamento: contornos fechados ou áreas de uma região destacadas propiciam a percepção dos símbolos ou glifos gráficos contidos nas áreas como grupos/padrões.

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

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

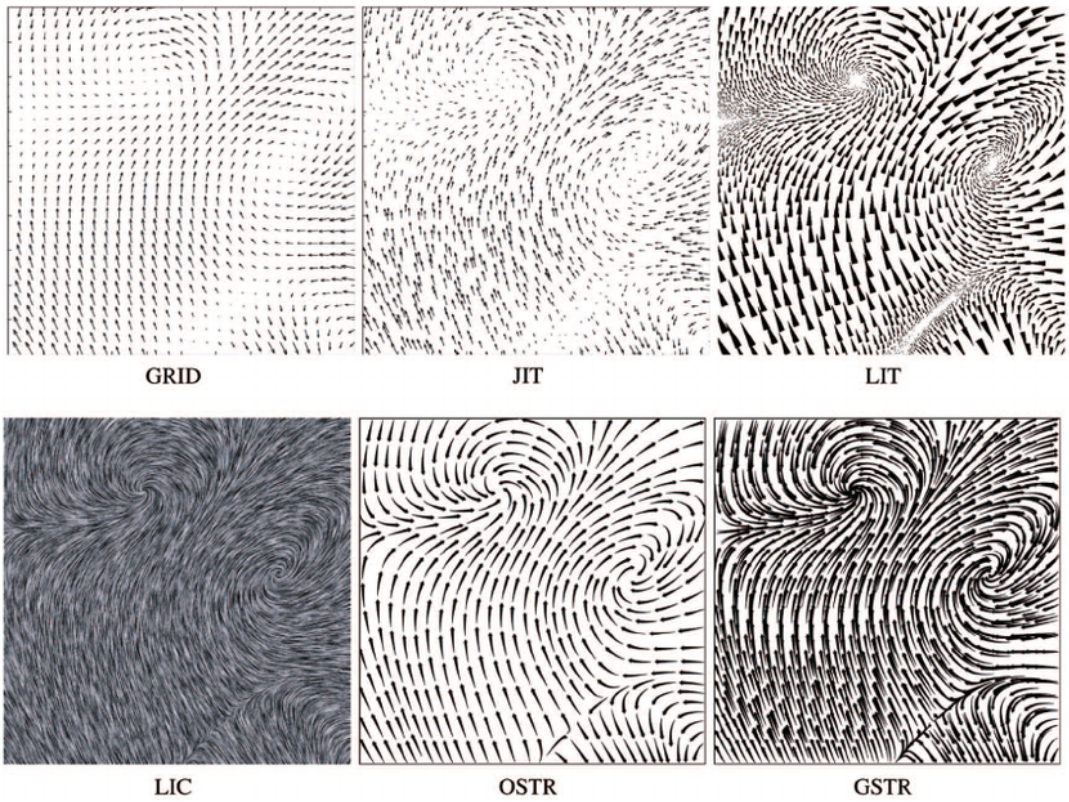


Figura 4.5: Campo vetorial representado por diferentes glifos. (Fonte: `https://www.researchgate.net/figure/One-of-the-approximately-500-vector-fields-visualized-with-each-of-the-six-visualifig1_8100150`)

Aplicando esses princípios, diferentes ícones, ilustrados na Figura 4.5), foram usados para representar as amostras de um campo (físico) vetorial em [45]:

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: É aplicada a técnica de convolução de integral de linha (*Line Integral Convolution* em inglês).

OSTR: São linhas de fluxo cujo posicionamento é guiado pela imagem.

GSTR: São 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 [93].

4.4 Psicologia Cognitiva

De acordo com a Wikipedia[16], 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 mais profunda 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 bem simplista, em três estágios (Figura 3.2) [93]:

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** (*recognition* em inglês) e **lembrança** (*recall* em inglês) [93]. 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". Para isso, evidências ambientais podem ser úteis na recuperação de conhecimentos da memória de longo prazo, desde que sejam cognitivamente simples. Este entendimento ampliado destaca 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 esse processo para chegar a uma solução. Ter uma compreensão dos mecanismos envolvidos na resolução de um problema nos permite estabelecer diretrizes claras para os recursos que devem estar presentes no sistema. Em [55], 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 de dados visual** 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:

1. na interpretação visual dos dados, identificando com precisão a origem do problema, os padrões não óbvios, monitorando resultados e avaliando a adequação de possíveis soluções;
2. na recuperação de casos semelhantes para auxiliar na modelagem do problema e no planejamento da solução;
3. na reformulação de problemas e estratégias de abordagem caso os resultados obtidos não atendam às expectativas; e
4. na aquisição de *insights* claros e uma visão mais profunda para auxiliar na tomada de decisões.

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 crucial de extrair *insights* capazes de orientar decisões assertivas. Os sistemas de análise de dados visual 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 nos permitem 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 alinhar esses princípios fundamentais da percepção visual com os princípios de *design* de interfaces gráficas para visualização de dados complexos, apresentado no Capítulo 3, os desenvolvedores se deparam com a oportunidade ímpar de conceber representações visuais intuitivas e implementar mecanismos de interação amigáveis. Essas representações e interações simplificam a exploração, viabilizando a identificação de padrões, tendências e anomalias nos dados.

4.7 Exercícios

1. Da coleção de ilusões ópticas do Museu Americano de História Natural [3], 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 11.1 a 11.12 em [65], o estatístico Rafael A. Irizarry destaca 11 conjuntos de problemas que podem aparecer nos gráficos gerados pela biblioteca ggplot2 (R)/plotnine (Python) e discute, quando pertinentes, soluções em R para contorná-los. Porte essas soluções em R para Python.

3. Na Seção 11.14 em [65], 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 entre 1928 e 2011, utilizando o conjunto de dados us_contagious_diseases [91]. 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

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 (*clustering*, em inglês), 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 dos dados não apenas revela padrões, tendências e relações não óbvias, mas também destaca valores discrepantes (*outliers*, em inglês) que podem influenciar a análise. Identificar e lidar adequadamente com esses *outliers* é crucial para assegurar a robustez e confiabilidade dos resultados da análise. Na análise de dados, a preparação dos dados é frequentemente considerada uma etapa crítica e desafiadora, conforme discutido em detalhes no Capítulo 6. A qualidade dos dados utilizados exerce um impacto significativo nos resultados da análise, e a preparação adequada dos dados é essencial para garantir a validade e confiabilidade das conclusões. Essa preparação de dados se beneficia substancialmente das técnicas estatísticas, tanto numéricas quanto gráficas.

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 5.1, 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 5.2, 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 5.3 explora medidas que revelam o grau de similaridade ou proximidade entre dois conjuntos de dados. Por fim, a Seção 5.4 detalha a ferramente de clusterização e suas variantes.

5.1 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.

5.1.1 Organização de Dados

A coleta de dados de uma população de interesse gera um conjunto inicial de **dados brutos** que requer organização para tornar-se 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 [48]. 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 [64]. Esse histograma revela que houve arredondamentos das notas ligeiramente abaixo do limite de aprovação para o limite de aprovação.

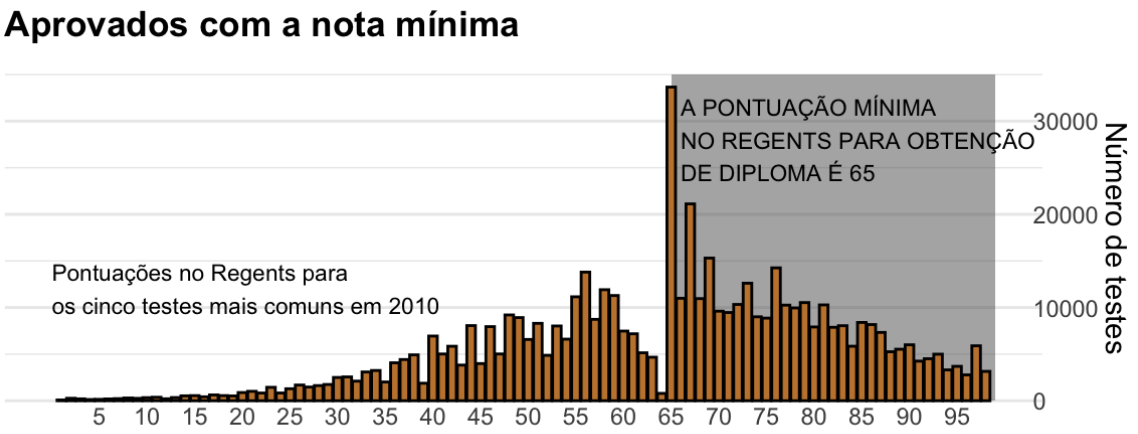


Figura 5.1: Histograma dos resultados de exames *Regents* revela um comportamento pouco esperado em torno da pontuação mínima. (Fonte: [64])

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 5.2):

Distribuição Uniforme: Esta distribuição descreve situações em que todos os valores

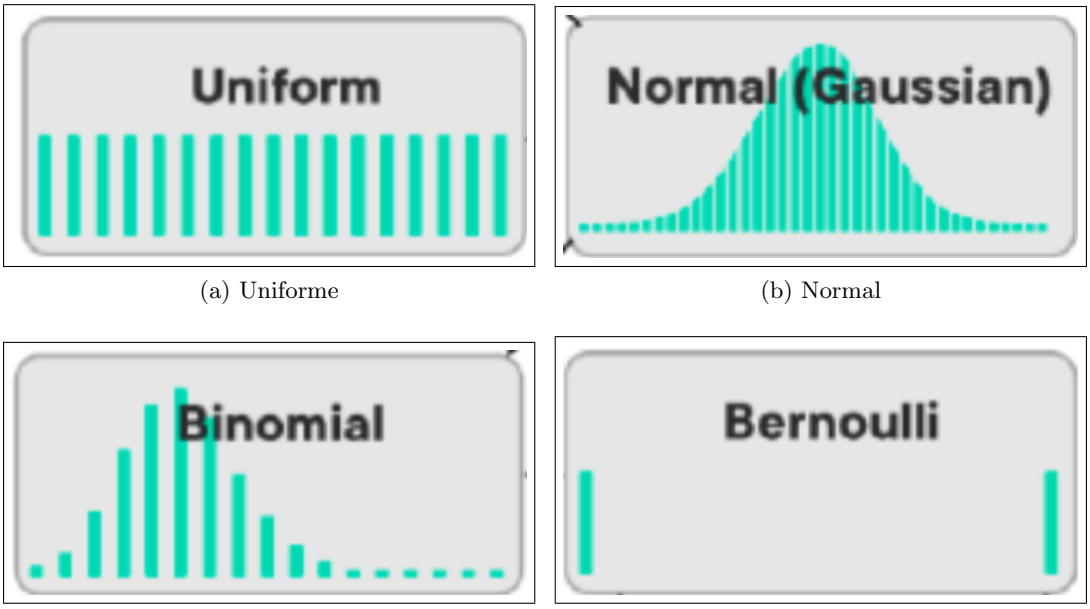
[1]Embora sejam similares visualmente, os histogramas são diferentes do gráficos de barras (*bar charts* ou *bar plots*, em inglês) na natureza das variáveis que eles representam. Um gráfico de barras representa dados categóricos ou quantitativos discretos e um histograma representa dados quantitativos.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

dentro de um intervalo têm a mesma frequência de ocorrência (Figura 5.2a).

Distribuição Normal (Gaussiana): Em forma de sino (Figura 5.2b), 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 5.2a 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 5.2d).



(a) Uniforme (b) Normal

(c) Binormal (d) Bernoulli

Figura 5.2: Exemplos de distribuições de frequência de ocorrência dos valores de uma variável. (Fonte: `https://bernard-mlab.com/post/probability-distribution/`)

Quando os dados não são discretos, é comum que a maioria dos valores será redundantemente única. Relatar a frequência de cada valor individual passa a não ser uma forma eficaz de 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 (*empirical cumulative distribution function*, em inglês) e é comumente denotada por F[65]:

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

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 5.3 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.

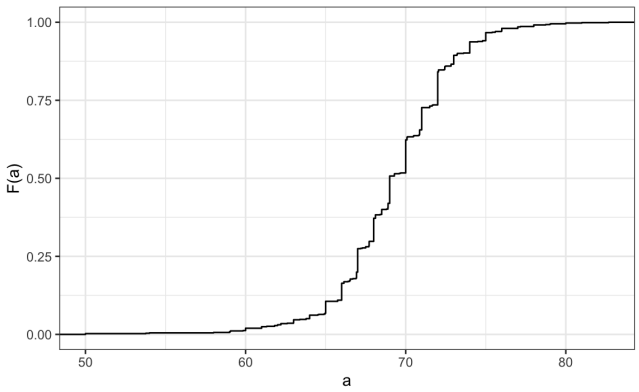


Figura 5.3: Gráfico de distribuição acumulada das alturas em polegadas dos estudantes masculinos do conjunto heights do pacote dslabs. (Fonte: [65])

5.1.2 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

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i. \tag{5.1}$$

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

$$meio\_da\_faixa = \frac{max(X) + min(X)}{2}. \tag{5.2}$$

A Figura 5.4 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 distruibuição 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.

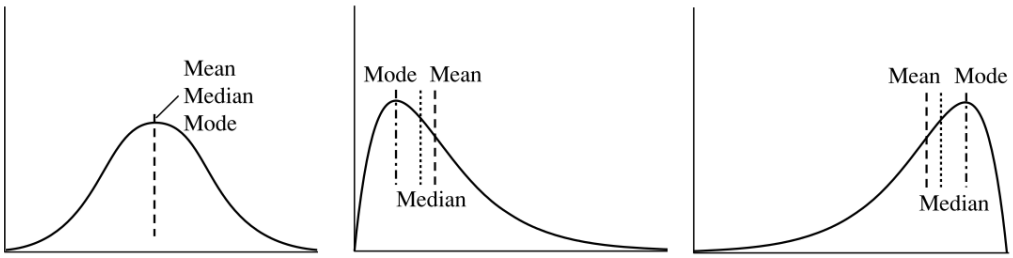


Figura 5.4: Tendências centrais em diferentes distribuições de frequência de dados: (a) distribuição simétrica, (b) distribuição assimétrica com a inclinação para valores mais baixos, (c) distribuição assimétrica com a inclinação para valores mais altos. (Fonte: [30])

5.1.3 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 5.5. 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 5.4.

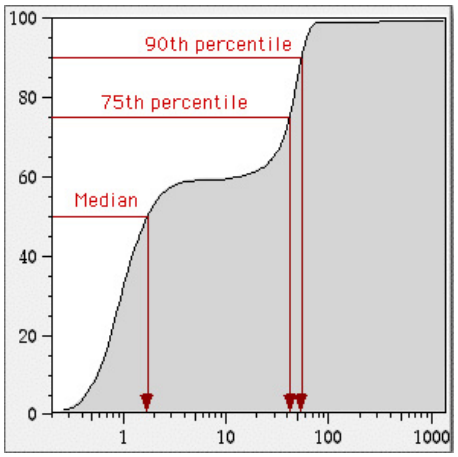


Figura 5.5: Dispersão de dados numa distribuição acumulada de frequência de dados revelada pela identificação direta de estatísticas como a mediana e outros valores que dividem os dados em termos de proporções de ocorrências. (Fonte: `https://docs.flowjo.com/flowjo/graphs-and-gating/data-visualization-and-display/gw-cdf/`)

Essa observação sugere que diferentes medidas de dispersão podem exigir diferentes tipos de gráficos para uma representação mais eficaz. Isso nos leva a considerar a existência de outros gráficos mais adequados para essa finalidade. De fato, existem tais gráficos disponíveis para representar as diferentes medidas de dispersão que 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*.

$$amplitude = max(X) - min(X). \tag{5.3}$$

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 5.5 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 (*InterQuartile Range*, em inglês): Corresponde à diferença entre o terceiro quartil (Q_3) e o primeiro quartil (Q_1) em uma distribuição ordenada. Ele

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

é menos sensível a *outliers* do que a amplitude.

$$IQR = Q_3 - Q_1. \tag{5.4}$$

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** (*boxplots*, em inglês) (Figura 5.6) 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 5.6): (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 5.6 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 de tamanho N e a média desta população.

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2. \tag{5.5}$$

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 de distribuição de dados e gráficos de caixa podem ser úteis para visualizar a dispersão dos dados e ajudar na compreensão da variabilidade dos valores de uma variável.

Desvio-padrão (σ): É a raiz quadrada da variância e fornece uma medida mais intuitiva

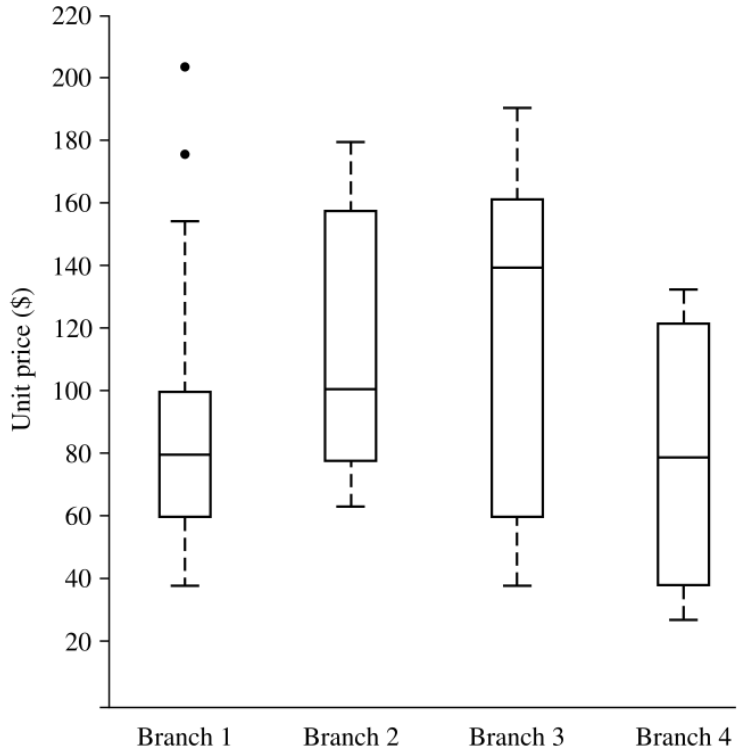


Figura 5.6: Gráficos de caixa representando os preços unitários para itens vendidos em quatro filiais de uma loja *on-line* durante um período de tempo. (Fonte: [30])

da dispersão dos dados, pois está na mesma escala que os próprios dados:

$$\sigma = \sqrt{\sigma^2} \tag{5.6}$$

Assim como a variância, o desvio padrão é uma medida de dispersão que não pode ser lida diretamente de um gráfico, mas tem uma interpretação gráfica simples, como mostra a Figura 5.7: 68,3% dos valores da variável estão situados a uma distância de um desvio padrão em torno da média de uma distribuição normal.

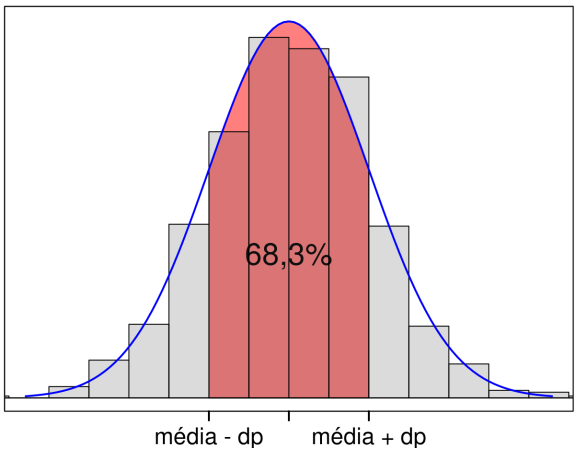


Figura 5.7: Desvio-padrão: uma medida de dispersão dos dados em relação à média numa distribuição de frequência normal, indicando o quanto os valores tendem a se afastar da média.

Coeficiente de dispersão (cv): É o desvio-padrão em relação à média da população. Também conhecido por **dispersão relativa**. 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.

$$cv = \frac{\sigma}{\mu}. \tag{5.7}$$

Em análises estatísticas, é comum empregar uma medida de posição relativa para avaliar a distância de um valor específico em relação à média de uma população. Essa medida é conhecida como **escore-z**, ou **escore padrão**. O escore-z quantifica em termos de desvios padrão a distância entre um valor específico x e a média μ da população, fornecendo uma referência sobre a posição relativa desse valor na distribuição estatística:

$$z = \frac{x - \mu}{\sigma}. \tag{5.8}$$

É importante destacar a relevância da distribuição normal na interpretação dos dados. Quando a distribuição de frequência dos valores de uma variável se assemelha à distribuição normal ou Gaussiana (Figura 5.2b), podemos descrever completamente essa distribuição usando apenas dois parâmetros: a média e o desvio padrão. Isso simplifica a análise estatística, pois esses dois parâmetros fornecem uma descrição concisa e abrangente da forma e variabilidade dos dados.

Muitos métodos estatísticos e modelos de inferência de dados presumem que os dados são distribuídos de acordo com uma distribuição normal. Por isso, é fundamental verificar se os dados realmente se ajustam a essa distribuição antes de aplicar esses métodos. Uma ferramenta comumente usada para essa verificação é o **gráfico quantil-quantil**, ou *Q-Q plot*. Neste gráfico, os quantis dos dados observados são plotados no eixo vertical em relação aos quantis esperados de uma distribuição normal no eixo horizontal. Se os dados seguem a distribuição normal, os pontos no gráfico geralmente formam uma linha diagonal, como ilustra a Figura 5.8. Quando os pontos se aproximam da diagonal, isso sugere que os dados se distribuem de forma semelhante a uma distribuição normal. Caso contrário, pode ser necessário explorar outras distribuições ou considerar transformações nos dados antes da análise.

5.1.4 Programação

O pacote dplyr em R e os pacotes pandas e numpy em Python são usados para manipulação e análise de dados. Ambos fornecem uma variedade de funções sumarizadoras úteis para resumir conjuntos de valores associados a uma variável e a grupos predefinidos nesse conjunto. No dplyr, a função summarize() é usada para resumir os dados, enquanto no numpy, a função aggregate() pode ser usada para realizar operações equivalentes. Essas funções incluem média, mediana, soma, desvio padrão,

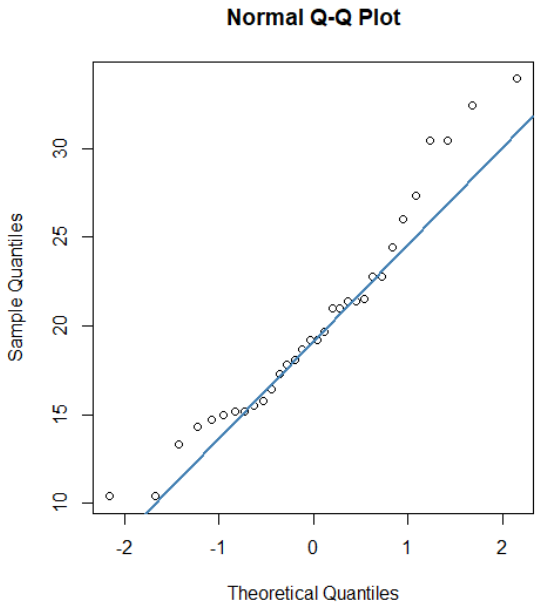


Figura 5.8: Gráfico quantil-quantil em que uma distribuição de frequência no eixo vertical é comparada com a distribuição de frequência normal no eixo horizontal. (Fonte: `https://www.learningtree.ca/blog/interpret-q-q-plot/`)

entre outras, como mostra a Tabela 5.1.

Tabela 5.1: 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)

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 [29]. Isso inclui opções como geom_bar() para gráficos de barras e geom_histogram() para histogramas, como ilustrado na Figura 3.17a, ao lidar com distribuição de frequências. A geometria geom_boxplot() é, por sua vez, uma escolha comum para representar os resumos dos cinco números. Para criar um gráfico de distribuição acumulada ou um gráfico quantil-quantil, é necessário calcular previamente a distribuição acumulada dos dados e os

quantis dos dados, respectivamente. Esses cálculos são essenciais antes de mapear os dados em uma camada de geometrias no ggplot().

Esse processo é simplificado ao incorporarmos, como camadas de **Estatísticas** em ggplot() ou plotnine(), a função estatística stat_ecdf() em R [28] ou em Python [60] na criação de um gráfico de distribuição acumulada de frequências e a função stat_qq() em R [27] ou em Python [61] na comparação de duas distribuições de frequências. Essas funções adicionam os dados transformados aos respectivos gráficos usando camadas de **Geometria** geom_line()/geom_step() em gráficos de ECDF e geom_qq() em gráficos quantil-quantil.

5.2 Covariância e Correlação

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 5.9 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 5.9: Tabela de frequência conjunta. (Fonte:[48])

A correlação e a covariância são medidas que proporcionam *insights* sobre a **relação entre as duas variáveis**. A **covariância**, 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 entre A e B é expressa por

$$Cov(A, B) = \frac{1}{N}\sum_{i=1}^{N}(a_i - \mu_A)(b_i - \mu_B). \tag{5.9}$$

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 não é diretamente interpretável, pois depende das escalas das variáveis. Para contornar isso, a **correlação** é frequentemente utilizada, 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" de covariância pelos desvios-padrão:

$$r_{A,B} = \frac{Cov(A, B)}{\sigma_A \sigma_B} \tag{5.10}$$

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

A melhor maneira de identificar a covariação ou correlação é visualizar a relação entre duas variáveis através de **gráficos de dispersão** (em inglês, *scatter plots*), em que 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 de 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 5.10. Se os pontos estiverem concentrados em torno de um padrão de curva, pode ajudar a entender o tipo de relação entre as duas variáveis.

Para calcular a correlação de duas variáveis 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 [96, 38]. 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 [52]. O exemplo que se segue usa o coeficiente de Pearson implementado no pacote pandas:

```
import pandas as pd
```

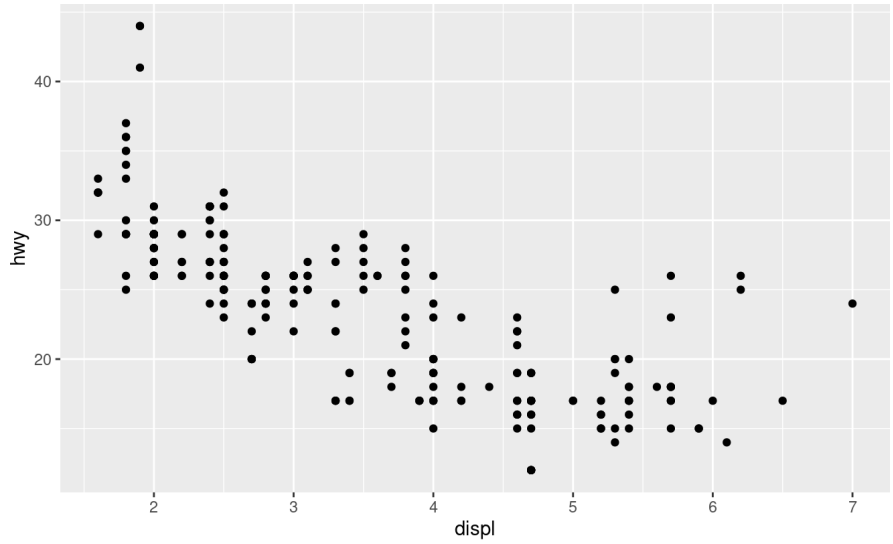


Figura 5.10: Gráfico de dispersão representado a relação entre o tamanho do motor (displ) e a eficiência de combustível (hwy). (Fonte: [96]

```
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.
```

Tanto em R quanto em Python, uma maneira de visualizar as relações entre os valores das variáveis em gráficos de dispersão é empregar a função ggplot() disponível nos pacotes ggplot2 (R) ou plotnine. O seguinte código exemplifica essa prática ao visualizar a 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 [91]:

```
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 várias variáveis simultaneamente, pode-se recorrer às matrizes de gráficos de dispersão (*scatterplot matrix*, em inglês). A função ggpairs() do pacote ggally, uma extensão do ggplot2, em R [66] e a função pairplot() do pacote seaborn em Python [77] 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 5.11.

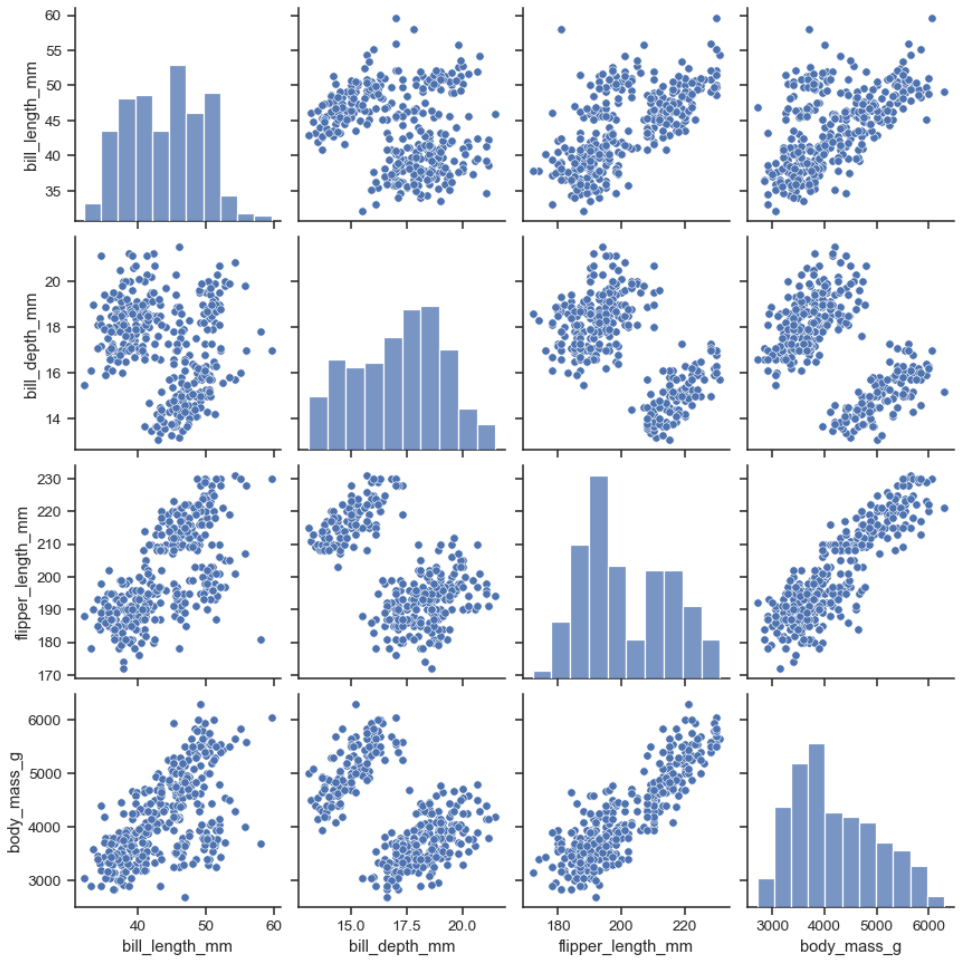


Figura 5.11: Uso de uma matriz de gráficos de dispersões para mostrar as relações entre as quatro características do conjunto de dados pinguins do pacote seaborn: comprimento e profundidade do bico e comprimento e massa da nadadeira. (Fonte: [77])

5.3 Similaridade

A Seção 5.1.2 introduziu as medidas resumo calculadas para os valores de uma variável quantitativa em um experimento, enquanto a Seção 5.2 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 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:

$$sim(i, j) = 1 - d(i, j) \tag{5.11}$$

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 [49]:

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.

$$d(P,Q) = \parallel P - Q \parallel = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{5.12}$$

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 espeços onde os movimentos são restritos a uma grade.

$$d(P,Q) = \parallel P - Q \parallel = \sum_{i=1}^{n} |p_i - q_i| \tag{5.13}$$

Distância Minkowski: É uma generalização que inclui tanto a distância euclidiana quanto a de Manhattan. É definida como:

$$d(P,Q) = \parallel P_i - Q_i \parallel_m = (\sum_{i=1}^{n} |p_i - q_i|^m)^{\frac{1}{m}}. \tag{5.14}$$

x 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 (*clustering*, em inglês), 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
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].

Uma das abordagens mais comuns para visualizar a similaridade dos dados é o uso de **gráficos de dispersão**, onde pontos no espaço representam observações e sua proximidade sugere similaridade.

Além disso, técnicas de redução de dimensionalidade, como a Análise de Componentes Principais (*Principal Component Analysis*, em inglês) [2, 112], são muito utilizadas para projetar dados de alta dimensão em um espaço bidimensional ou tridimensional, preservando ao máximo possível a estrutura e a proximidade relativa dos pontos originais.

5.4 Clusterização

Um **grupo** (*cluster*, em inglês) é uma coleção de objetos de dados que compartilham semelhanças entre si dentro do próprio grupo, mas são diferentes de objetos em outros grupos [31]. 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**. 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 5.3. Para esses dados providos de relação espacial, que chamamos de **pontos de dados**, destacam-se os seguintes métodos de clusterização [31]:

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 *cluster*. 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.

Segue uma explicação detalhada dos quatro algoritmos mencionados, juntamente com as funções correspondentes em R e Python. Além disso, exploramos uma abordagem visual que facilita o processo de agrupamento mental usando **mapas de calor**.

5.4.1 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*,

$$J = \sum_{j=1}^{k} \sum_{i=1}^{m_j} \omega_{i,j} \cdot d(x_{ji}, c_j)^2, \tag{5.15}$$

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 [69]):

```
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 [44], 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")
)
```

5.4.2 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 [31]):

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

$$W(S_j, S_k) = \frac{n_j n_k}{n_j + n_k} |m_j - m_k|^2. \tag{5.16}$$

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 5.12 ilustra um dendrograma em R que revela a clusterização hierárquica das observações do conjunto USArrests [91].

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* [68], e o uso do pacote ggdendro para a visualização de dendrogramas dentro do *framework* do ggplot2 em R [59]:

```
# 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)
```

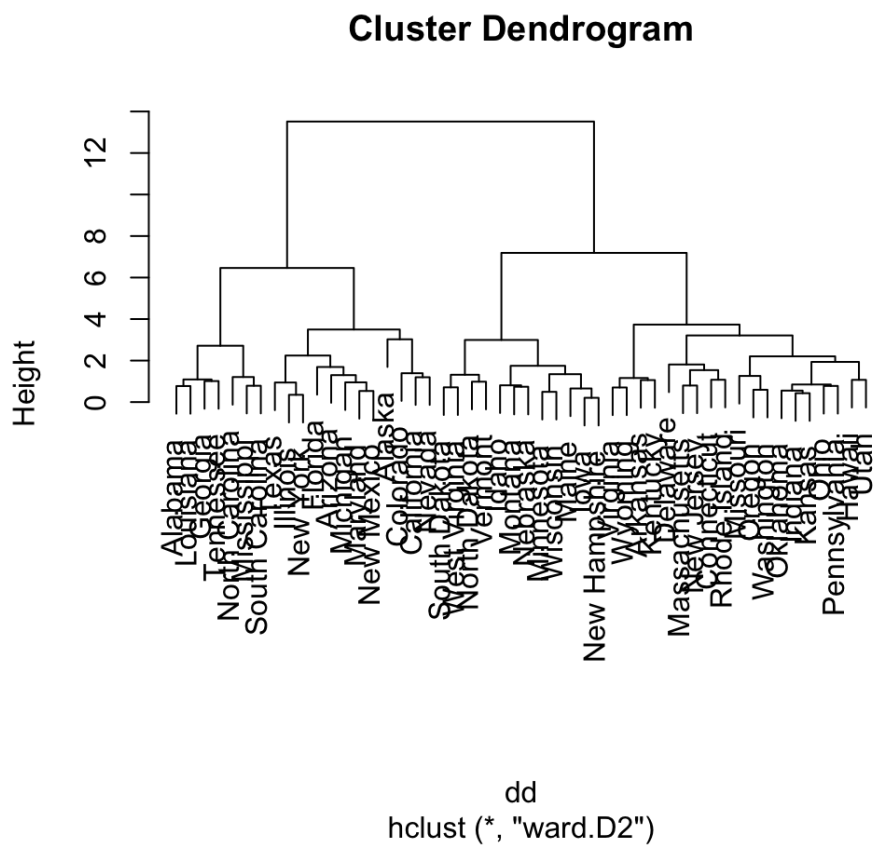


Figura 5.12: Dendrograma do conjunto de dados USArrests que contém estatísticas, em número de prisões por 100.000 residentes, para agressão, assassinato e estupro em cada um dos 50 estados dos EUA em 1973. (Fonte: `http://www.sthda.com/english/wiki/beautiful-dendrogram-visualizations-in-r-5-must-known-methods-unsupervised-machine-learning`)

```
# 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 [74] 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')
)
```

5.4.3 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 [50]:

```
# 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 [75]:

```
%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"))
```

5.4.4 CLARA

CLARA (do inglês, *Clustering Large Applications*) foi desenvolvida por Kaufman e Rousseeuw, em 1990 [25]. 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 [67].

```
# 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()
```

5.4.5 Mapas de Calor

Mapas de calor (*heat map*, em inglês), também chamados **imagens coloridas falsas**, podem ser utilizados para uma visualização simultânea de observações e características em uma grade bidimensional. Cada célula da grade representa a combinação de um par (observação,característica), onde o valor da característica é representado pela intensidade ou cor da célula. Geralmente, os valores das características de cada observação são organizados ao longo do eixo horizontal da grade, e a intensidade ou cor de cada célula é determinada pelo valor da métrica ou medida associada. Cores mais escuras geralmente representam valores mais altos, enquanto cores mais claras representam valores mais baixos. Figura 5.13 apresenta um mapa de calor do conjunto de dados mtcars [91], onde as cores amarela e vermelha correspondem aos valores mais baixos e mais altos, respectivamente.

Os mapas de calor podem ser úteis para explorar a clusterização subjacente dos dados. Isso ocorre porque os mapas de calor representam visualmente a densidade ou intensidade dos dados em uma grade bidimensional, onde cada célula da grade pode ser colorida de acordo com o valor associado

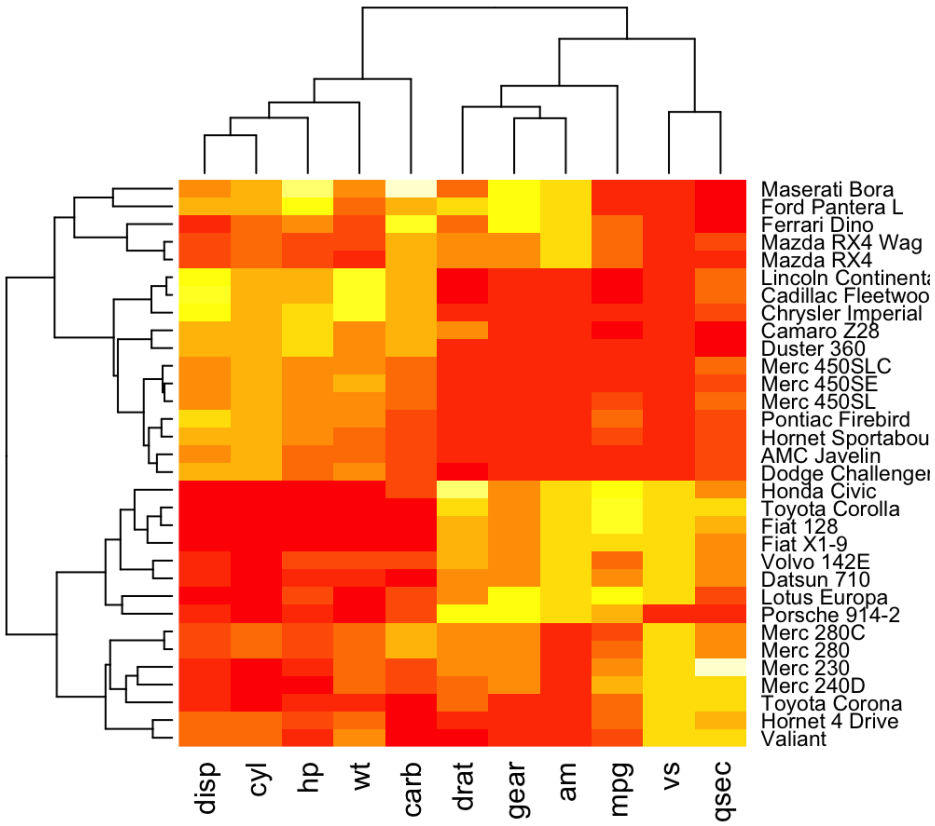


Figura 5.13

a ela. Esses mapas de calor podem destacar regiões da grade onde os pontos de dados estão mais densamente agrupados, indicando a presença de *clusters*. Isso pode ajudar os analistas a identificar padrões de agrupamento, tendências e correlações nos dados e avaliar a eficácia dos algoritmos de clusterização em separar os dados em grupos distintos. Além disso, os mapas de calor também podem ser combinados com dendogramas para fornecer uma representação visual completa da estrutura de agrupamento dos dados, como mostra a Figura 5.13.

5.5 Considerações Finais

Neste capítulo, exploramos as nuances da estatística descritiva, fornecendo uma visão abrangente das medidas resumo e técnicas de visualização que nos permitem entender melhor a distribuição e o comportamento de variáveis numéricas associadas a uma população. Introduzimos gráficos, como histogramas e diagramas de caixa, complementados por códigos em R e Python, para facilitar a aplicação dessas técnicas na prática.

Discutimos os conceitos de correlação e covariância, examinando como essas medidas podem nos dar *insights* sobre a relação entre duas ou mais variáveis. Destacamos gráficos de dispersão como uma forma de visualização das relações entre duas variáveis e matriz de gráficos de dispersão para visualizar as relações entre multivariáveis. Foram apresentadas ainda as medidas de similaridade, que

nos permitem comparar observações de um fenômeno. As técnicas de clusterização, por sua vez, junto com as representações visuais de dendrogramas e mapas de calor, emergiram como uma ferramenta essencial para identificar grupos inerentes nos dados, delineando estruturas e padrões que podem não ser evidentes à primeira vista.

Entretanto, é imperativo ressaltar algumas considerações. Tanto as técnicas de análise de similaridade quanto as de agrupamento, em última instância, fundamentam-se em comparações numéricas. Embora essa abordagem seja valiosa, é importante destacar que ela pode não capturar integralmente a riqueza semântica subjacente a esses conceitos complexos. O mapeamento inadequado dos valores numéricos para as coordenadas dos pontos pode resultar em análises equivocadas. Em algumas situações, torna-se essencial que usuários experientes e habilidosos explorem os dados e apliquem mapeamentos apropriados dos valores numéricos, proporcionando uma revelação mais precisa de padrões e tendências relevantes para a análise.

Nossa abordagem concentra-se principalmente em variáveis numéricas organizadas em tabelas, deixando de lado a consideração de variáveis categóricas. Embora tenhamos oferecido uma visão abrangente das técnicas estatísticas aplicáveis a dados quantitativos, reconhecemos que a estatística descritiva é uma disciplina ampla, com muito mais a explorar, especialmente no que diz respeito a variáveis categóricas. Também é importante destacar as limitações das representações tabulares dos dados, embora a maioria das técnicas estatísticas consiga manipulá-las eficientemente. Muitos fenômenos do mundo real vão além das tabelas bidimensionais. A organização inicial dos dados brutos em tabelas, como parte do pré-processamento para análise, é uma solução que requer consideração a médio e longo prazo.

Apesar dessas ressalvas, este capítulo proporciona uma base sólida para compreender e interpretar dados, mas é apenas o começo. À medida que avançamos, é crucial expandir nossos horizontes para incluir uma análise mais abrangente, considerando variáveis categóricas, semântica dos dados e estruturas além das tabulares, além de explorar outras técnicas avançadas. A estatística, como uma ferramenta poderosa na tomada de decisões, continua a oferecer um vasto território de descobertas e insights à medida que aprofundamos nosso entendimento e aplicação desses métodos.

5.6 Exercícios

1. Faça os itens 1 – 7 da Seção 12.10 da referência [65], usando o conjunto de dados Galton em [91].

2. Usando o mesmo conjunto de dados Galton, plote:

   (a) gráficos de distribuição de frequência das alturas dos pais e das crianças para avaliar o grau de simitria das distrituições.

   (b) gráficos de distribuição acumulada de frequências das alturas das crianças e dos pais para

determinar as medianas das duas distribuições.

(c) gráficos de caixa para comparar as estatísticas dos pais com as das crianças.

(d) gráficos de quantil-quantil para verificar se a distribuição das alturas dos pais segue uma distribuição Gaussiana.

(e) gráfico de dispersão para verificar se há correlação entre as alturas das crianças e as alturas dos pais.

(f) mapa de calor dos dados para identificar a quantidade de *clusters* subjacentes e compare com a quantidade de grupos de pontos de dados mostrados no gráfico de dispersão no item 2e.

Capítulo 6

Preparação de Dados

Antes de iniciar a exploração e análise de dados em ambientes de desenvolvimento integrado (em inglês, *Integrated Development Environment* – IDE), é crucial realizar um procedimento cuidadoso para garantir a qualidade destes dados. De acordo com Han e colegas (Seção 2.4.1 em [30]), 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 (*Accuracy* em inglês): Refere-se à exatidão dos dados em relação à realidade.

Completude (*Completeness* em inglês): Diz respeito à presença de todos os dados necessários, sem omissões significativas.

Consistência (*Consistency*, em inglês): Envolve a uniformidade e coesão dos dados, garantindo que não haja contradições ou discrepâncias.

Oportunidade (*Timeliness* em inglês): Relaciona-se à atualidade dos dados, ou seja, quão recentes são em relação ao momento em que são necessários.

Credibilidade (*Believability* em inglês): Refere-se à confiabilidade e fonte dos dados, avaliando se podem ser considerados como informações confiáveis.

Interpretabilidade (*Interpretability* em inglês): Diz respeito à facilidade com que os dados podem ser compreendidos e utilizados de maneira eficaz para atender aos objetivos específicos.

Observa-se que cerca de 80% dos analistas de dados passam a maior parte do tempo em preparação de dados e não na análise propriamente dita [98]. Na Ciência de Dados (tabulares), é conhecido por *data wrangling*, ou *data munging*, esse procedimento de preparação de dados que transforma conjuntos de dados brutos em conjuntos limpos, organizados e otimizados, de maneira ***tidy***, para análises estatísticas, cognitivas e a aplicação de algoritmos de aprendizado de máquina. Tais *tidy*

data, popularizados pelo cientista de dados Hadley Wickham [98], seguem os seguinte princípios de organização (Figura 6.1)

- cada tipo de unidade observacional forma uma tabela,
- cada observação forma uma linha, e
- cada variável forma uma coluna.

	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 *Untidy*

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 *Tidy*

Figura 6.1: Dados organizados em formato tabular: (a) Dados que não estão arrumados (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 que estão organizados (em inglês, *tidy*), porque cada linha corresponde a uma observação e cada coluna a uma variável. (Fonte: [98])

Segundo Hadley Wickham, conjuntos de dados no formato *tidy* proporcionam uma abordagem padronizada para vincular a estrutura física de um conjunto de dados à sua semântica, ou seja, ao seu significado. Embora a organização inicial desses dados demande algum esforço, os benefícios a longo prazo são significativos. Ao ter os dados organizados de acordo com o formato *tidy*, conseguimos utilizar de forma mais eficiente as ferramentas disponíveis nos pacotes R e Python que são projetadas especialmente para esse formato, como o pacote ggplot2 no R [99] e o pacote plotnine em Python [109]. Isso resulta em menos tempo gasto na manipulação dos dados entre diferentes representações, possibilitando dedicar mais tempo às perguntas analíticas específicas.

O termo **dados relacionais** se refere a uma coleção de tabelas (unidades observacionais) organizadas de maneira a possibilitar a extração de informações específicas para determinados objetivos. Essa organização é fundamentada nos conceitos de chaves primárias e estrangeiras, característicos dos modelos relacionais. A **chave primária** é composta por uma variável (coluna) ou conjunto de variáveis em uma tabela que proporciona uma identificação única para cada linha, garantindo unicidade. Por sua vez, a **chave estrangeira** é a variável que estabelece uma relação com a chave primária de outra tabela, conectando e relacionando informações entre tabelas.

O processo de preparação de dados *tidy* geralmente abrange três estágios: importação de dados, organização e estruturação de dados, e transformação de dados. A **importação de dados** em ambientes como RStudio ou IPython/Jupyter Notebook, a ser explorada na Seção 6.1, implica transferir

conjuntos de dados de diferentes fontes para RStudio/Jupyter Notebook. Para facilitar a análise exploratória, visualização e modelagem estatística nesses ambientes, é necessário que esses dados passem por uma **organização e reestruturação**, a ser discutida na Seção 6.2, e por uma **transformação** dos valores brutos em valores mais propícios, mantendo a integridade dos dados originais, a ser explorada na Seção 6.3. Nesse processo, diversas ferramentas estatísticas podem ser utilizadas para detectar e corrigir discrepâncias, com o objetivo de reduzir redundâncias e corrigir inconsistências, preparando os dados para serem utilizados em aplicativos disponíveis para análise e interpretação.

A reestruturação e transformação dos dados se revelam altamente adaptáveis, moldando-se conforme as soluções almejadas e a natureza da análise planejada. Apesar da disponibilidade de diversas ferramentas que oferecem suporte a esses processos, sua combinação é personalizada para atender às nuances de cada cenário. Em contextos mais simples, pode ser suficiente reorganizar e filtrar valores de dados, ou até mesmo transpor as colunas por linhas nos conjuntos tabulares originais. No entanto, em situações mais complexas, a integração de dados provenientes de fontes diversas se faz necessária, envolvendo uma reestruturação mais profunda e a inclusão de informações ausentes. Há também cenários cruciais em que a transformação dos valores dos dados, por meio de sumarizações ou expansões, é fundamental. Essa flexibilidade implica que a ordem de apresentação dos estágios de preparação de dados neste capítulo não reflete rigidamente a sequência exata de manipulações que um analista de dados adotaria em cada situação. A tomada de decisão quanto à sequência específica permanece como uma habilidade artística que não pode ser plenamente automatizada até o momento.

6.1 Importação de Dados

A **importação de dados** para IDEs, como RStudio ou Jupyter Notebook, consiste na transferência de conjuntos de dados provenientes de diversas fontes, frequentemente em formatos de "arquivos planos" (em inglês, *flat files*), para estruturas de dados em formato tabular, tipicamente data.frames/tibbles no R e DataFrames em Python. Essas estruturas são especialmente projetadas para serem processadas por funções de manipulação e análise de dados disponíveis nesses ambientes.

No início do processo de importação, é essencial identificar a origem dos dados, que pode ser um arquivo local, um banco de dados externo ou até mesmo uma Interface de Programação de Aplicações (em inglês, *Application Programming Interface* – API). Dependendo da fonte, diferentes métodos e funções são empregados para realizar a transferência eficiente e precisa dos dados para o ambiente de análise.

Após a importação dos dados, é prática comum realizar uma verificação inicial, examinando algumas linhas e colunas para garantir o sucesso do processo. Uma vez concluída essa etapa, os dados estão prontos para serem submetidos a processos adicionais. Especificamente, ocorre a organização dos dados em formatos que se alinham às estruturas ideais para as funções implementadas, seguida

pela transformação para aprimorar a qualidade. Esses dados aprimorados podem então ser explorados e analisados de maneira efetiva.

Esta seção oferece uma introdução básica às funções de importação e exportação em ambientes R e Python.

6.1.1 Arquivos

A função read_csv() é comumente utilizada para importar, em R, dados de arquivos CSV (sigla de *Comma-Separated Values*) (Capítulo 8 em [96]), enquanto a função read_tsv() é empregada para arquivos TSV (sigla de *Tab-Separated Values*) ou arquivos de extensão .txt. Essas funções são implementadas no pacote readr, integrante da tidyverse que pode ser instalado e carregado com as seguintes linhas de comando:

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

Entre os argumentos das funções de importação figuram o caminho do arquivo cujos dados devem ser importados e, opcionalmente, a forma de manipulação de cabeçalhos, o formato das linhas de comentários e formatação de dados, como ilustra o seguinte exemplo de importação de dados do arquivo caminho/do/seu/arquivo/ex1.csv no data.frame df1 do R:

```
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
```

O pacote readr também oferece funções, como write_csv() e write_tsv(), que exportam os dados do tipo data.frame/tibble para arquivos em formato CSV e TSV, respectivamente. Alternativas para importar e exportar fontes de dados em outros formatos são exploradas em [82].

Da mesma forma, no Jupyter Notebook (utilizando Python), funções equivalentes das bibliotecas pandas, altair e numpy são frequentemente empregadas para importar e manipular dados tabulares provenientes de diversos formatos, conforme detalhado no Capítulo 11 em [35]. Mais especificamente, para importar os dados dos arquivos de texto no formato CSV e TSV, são utilizadas as funções read_csv() e read_table(), respectivamente. A instalação e importação do pacote pandas podem ser realizadas com os seguintes comandos:

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

Note que o ponto de exclamação (!) antes da linha de comando "pip3 install pandas" é uma sintaxe específica do Jupyter Notebook que indica que o comando deve ser executado como um comando de sistema, fora do ambiente Python. Diferente do R, os métodos de exportação são associados aos objetos dos dados em Python. Por exemplo, o método to_csv() da classe de objetos DataFrames é usado para exportar os dados do tipo DataFframes para um arquivo de formato CSV. As seguintes linhas de comando ilustram a importação de dados em Python a partir do arquivo caminho/do/seu/arquivo/ex1.csv e a exportação para um outro arquivo caminho/do/seu/arquivo/resultado.csv sem os índices de 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 visão abrangente sobre processamento de entrada/saída de dados provenientes de diversas fontes, recomenda-se a leitura de uma explanação detalhada em [81].

6.1.2 Banco de Dados

R e Python oferecem suporte para a importação de dados de bancos de dados relacionais, disponibilizando pacotes específicos que simplificam a conexão e permitem a manipulação dos dados por meio de Linguagem de Consulta Estruturada (em inglês, *Structured Query Language* – SQL).

No contexto do 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, utilize 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*:

```
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 [96].

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, usa-se as funções de manipulação de dados da biblioteca pandas importada como pd:

```
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 oferece uma abstração de alto nível para Python interagir com bancos de dados relacionais, incluindo SQLite, MySQL, PostgreSQL e outros. Ela fornece uma API consistente para trabalhar com diferentes sistemas de gerenciamento de banco de dados (SGBDs) e oferece recursos adicionais, como mapeamento objeto-relacional (ORM) para representar tabelas de banco de dados como objetos Python. Há pacotes específicos para interagir com bancos de dados instalados localmente, como sqlite3 para o banco de dados SQLite, mysql.connector para MySQL e psycopg2 para PostgreSQL. Recomenda-se a leitura da referência [19].

6.2 Organização e Estruturação de Dados

A organização e reestruturação de dados importados de fontes heterogêneas é fundamental na preparação para análise. Esse processo envolve a **limpeza dos dados** e eventuais ajustes em situações

como valores ausentes, erros de digitação e formatos inconsistentes. Além disso, a **integração de dados** provenientes de diferentes fontes, que podem apresentar redundâncias e inconsistências, requer uma abordagem cuidadosa para garantir a uniformidade e coerência do conjunto de dados consolidado. A busca por um formato unificado não apenas facilita a manipulação e análises subsequentes, mas também fortalece a confiabilidade e a qualidade dos dados, fundamentais para decisões informadas e análises precisas.

6.2.1 Limpeza de Dados

É comum que os dados brutos apresentem erros ou falhas, tornando-os inadequados para os algoritmos disponíveis. Três tipos de erros comuns na aquisição de dados brutos são [30]:

Dados faltantes (*data missing* em inglês): 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 (*inconsistent data* em inglês): 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 6.2.3), ou mesmo devido a alterações nas regras que não foram refletidas de maneira consistente nos dados.

Dados ruidosos (*noisy data* em inglês): 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 [30]:

- 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).

Depois de identificar discrepâncias nos dados, é necessário selecionar a técnica mais adequada para abordá-las. Alguns desvios nos dados podem ser corrigidos manualmente, incluindo a exclusão de observações, a inserção manual de valores ausentes ou a atribuição automática de um valor padrão. No entanto, a maioria dos erros requer transformações nos dados. Após a identificação das discrepâncias, é comum definir e aplicar uma série de transformações para corrigi-las (Seção 6.3). Entre as técnicas para lidar com dados faltantes, a mais popular é atribuir valores mais prováveis, computados por meio de árvores de decisão, regressão e inferência Bayesiana. Por outro lado, técnicas como *binning* (ou segmentação) e regressão são amplamente empregadas para suavizar dados ruidosos. A abordagem interativa de limpeza e transformação de dados é uma estratégia promissora [30].

Tanto R quanto Python oferecem ferramentas para tratamento de dados faltantes. Em R, os dados faltantes são representados explicitamente por NA (sigla de *Not Available*) ou implicitamente por omissão (Capítulo 9 em [96]). Essas entradas podem ser detectadas usando a função filter do pacote dplyr com o argumento is.na() para filtrar as entrada como dados faltantes (Capítulo 3 em [96]). Aplicando em r_df1:

```
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. 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*) em estruturas 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) ou notnull() (entradas presentes). Destacam entre os métodos de tratamento de dados faltantes nas 2 classes de objetos dropna() para descartar entradas com NA, por unidade em Series ou por linhas/colunas em DataFrames, fillna() para substituir as entradas NA por algum valor especificado, incluindo estatísticas de resumo de dados como a média, e interpolate() para preencher os valores faltantes com os valores interpolados dos valores presentes [57, 56]. O Capítulo 5 em [95] explora o potencial desses métodos no tratamento de dados faltantes.

Para identificar dados inconsistentes ou ruidosos, é possível aplicar **estatísticas de resumo** (Figura 6.2) 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.

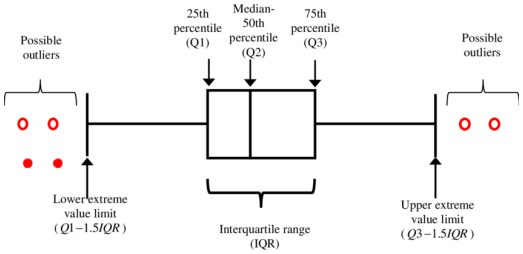


Figura 6.2: Detecção dos *outliers* através do gráfico de caixa (*boxplot*) (Fonte: `https://www.researchgate.net/figure/Box-plot-with-outliers-detection-range_fig3_322129094`).

6.2.2 Formatação em Dados *tidy*

Com a crescente popularidade dos dados *tidy*, a maioria das funções *off-the-shelf* espera que seus dados de entrada estejam no formato *tidy*. Esse formato é caracterizado pela organização dos dados em uma estrutura tabular, onde cada observação corresponde a uma linha e cada variável da observação ocupa uma coluna (Figura 6.3). Diante desse contexto, muitos usuários ainda optam por realizar o pré-processamento dos dados, convertendo-os para o formato "tidy" usando linguagens de programação

de propósito geral ou ferramentas de processamento de texto UNIX, como o sed ou o awk.

Figura 6.3: Dados no formato *tidy*. (Fonte: [96]

Contudo, tanto R quanto Python dispõem funções e bibliotecas dedicadas que facilitam a reorganização eficiente dos dados para o formato tabular, onde os valores de uma variável das observações são consolidados em uma única coluna. Isso facilita o processamento desses dados pela maioria das bibliotecas em R, como dplyr e ggplot2, e em Python, como pandas e altair. Para organizar os dados no formato *tidy*, é crucial entender a semântica dos dados e distinguir as observações e as variáveis associadas a cada observação.

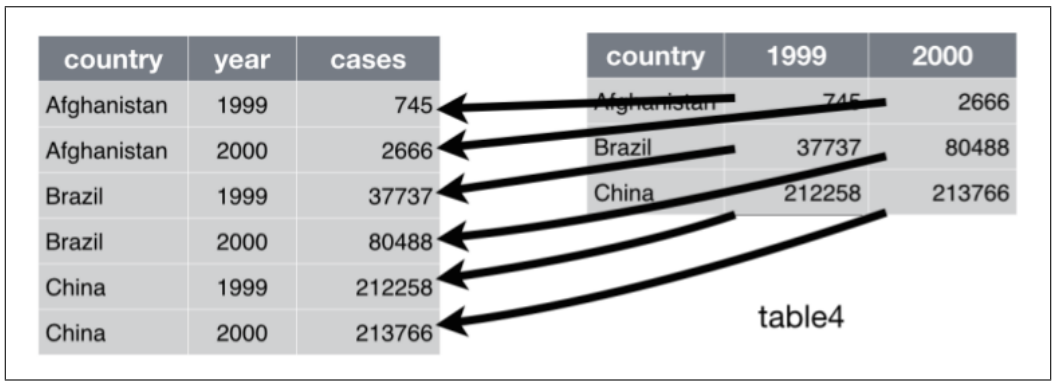
Em R, as bibliotecas tidyr e dplyr são especialmente reconhecidas por disponibilizarem ferramentas eficazes para a manipulação de dados em R. A biblioteca tidyr se concentra na organização e reestruturação de dados, facilitando a transformação de conjuntos de dados para o formato tabular. A dplyr, por sua vez, oferece uma variedade de funções para realizar operações de manipulação de dados de maneira eficiente.

Entre as funções do pacote tidyr, destacam-se gather(), spread(), separate() e unite() (conforme abordado no Capítulo 9 de [96]). A função gather() combina várias colunas em uma única, criando duas novas colunas: uma para armazenar os nomes originais das colunas (agora representadas como linhas) e outra para armazenar os valores correspondentes (consulte a Figura 6.4a). Por outro lado, a função spread() distribui os valores de uma coluna em várias colunas, usando os valores de outra coluna como cabeçalhos (consulte a Figura 6.4b). Normalmente, gather() transforma tabelas amplas (ou "wide"), onde as variáveis são representadas por colunas, em tabelas mais estreitas e longas (ou "long"), onde as variáveis e seus valores correspondentes são organizados em duas colunas distintas. Enquanto isso, a função spread() realiza o oposto, convertendo tabelas longas em tabelas mais curtas e largas.

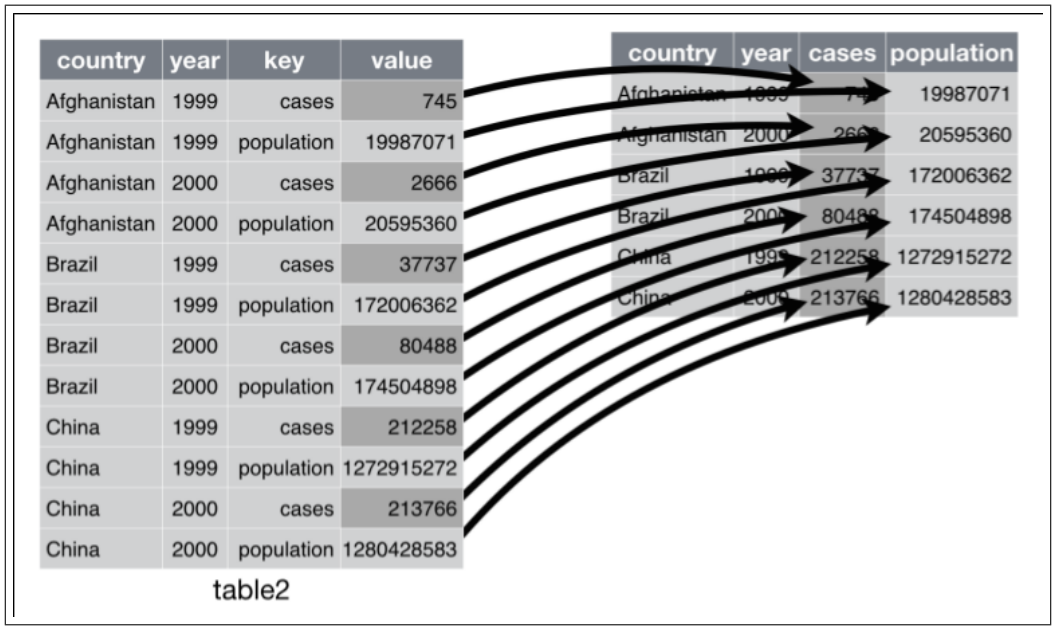
Segue-se um trecho de código que transforma dados organizados em tabelas *untidy*, dados_mat e dados_port, em dados *tidy*:

```
#Carregar bibliotecas
library(dplyr)
library(tidyr)


# Criando dados frame, contendo vetores de notas por ano
```



(a) gather()



(b) spread()

Figura 6.4: 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, (b) dispersão dos valores de uma coluna para diversas colunas, onde cada coluna representa um valor de chave da coluna "key" de chaves. (Fonte: [96])

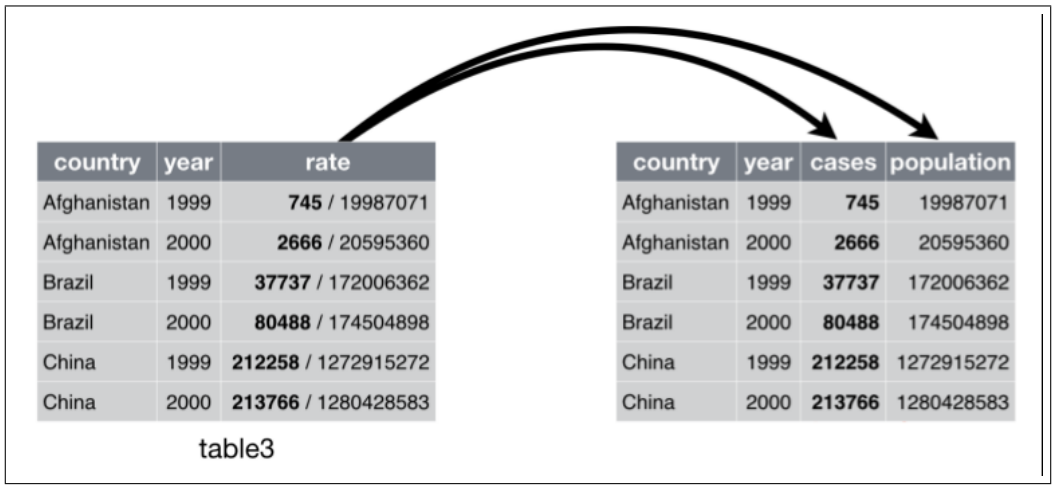
```
dados_mat <- data.frame(
  ID = c(1, 2, 3),
  Sexo = c("M", "F", "M"),
  '2020' = c(90, 80, 70),
  '2022' = c(75, 80, 60),
  '2023' = c(65, 95, 85)
)


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
```

As funções separate() e unite(), por sua vez, são úteis para manipular variáveis categóricas ou texto, permitindo compactar diferentes valores numa única coluna ou separar um valor em várias colunas. Mais especificamente, separate() divide os valores de uma única coluna em múltiplas colunas (Figura 6.5a e unite() combina os valores de várias colunas em uma única coluna (Figura 6.5b.

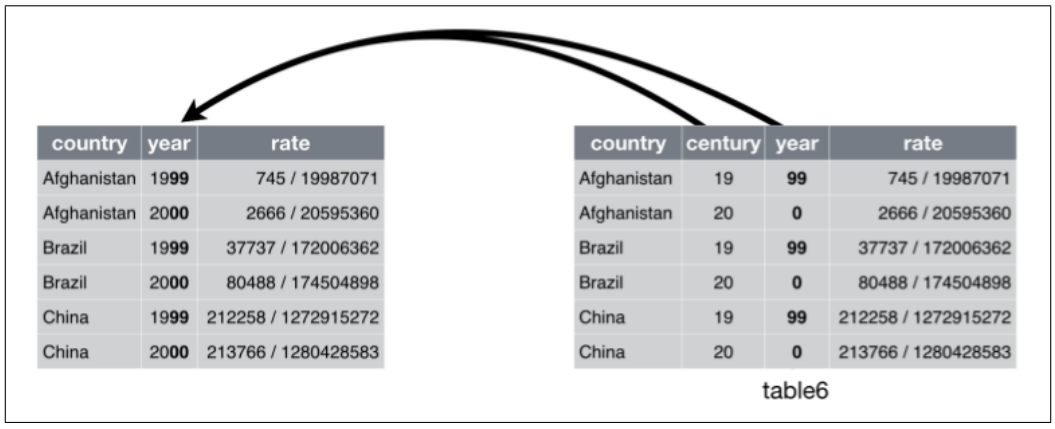


country	year	rate
Afghanistan	1999	**745** / 19987071
Afghanistan	2000	**2666** / 20595360
Brazil	1999	**37737** / 172006362
Brazil	2000	**80488** / 174504898
China	1999	**212258** / 1272915272
China	2000	**213766** / 1280428583

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 6.5: 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) junção de partes de um valor alocadas em diferentes colunas num único valor alocado numa única coluna. (Fonte: [96])

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, altair e numpy são amplamente utilizadas para reformatar os dados no formato *tidy* (Capítulo 12 em [35]). A função melt do pandas é equivalente à função gather() do tidyr em R, enquanto a função pivot() tem função análoga ao spread(). Por exemplo, a seguinte sequência de 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
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 ao 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)
```

6.2.3 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 [30]). A heterogeneidade semântica e a estrutura dos dados apresentam grandes desafios na integração de dados. Problemas, como identificação de entidades, redundância em valores associados a um atributo, duplicação de registros de dados, e dados conflitantes, 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** (*entity identification problem*, em inglês) se refere ao desafio de associar entidades 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 entidades em uma fonte de dados correspondem às mesmas entidades em outra fonte, 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.

A **redundância**, caracterizada pela repetição desnecessária de informações em um conjunto de dados, é outra questão importante na integração de dados. Um atributo num esquema pode ser, por exemplo, "derivado" de outro atributo ou conjunto de atributos de um outro esquema. Inconsistências na nomenclatura de atributos ou dimensões também podem causar redundâncias no resultado. Cabe lembrar aqui aque a remoção de dados redundantes é uma prática comum no armazenamento dos dados, incluindo no projeto de banco de dados, para otimizar a eficiência e manter a consistência dos dados. O procedimento de eliminação de dados redundantes em bancos de dados relacionais é denominado normalização. A **normalização** (dos dados relacionais) consiste em identificar e remover dependências funcionais entre as colunas de uma tabela, resultando na criação de tabelas menores ou longas interconectadas por meio de chaves estrangeiras para otimizar o armazenamento.

Um outro problema envolvido na integração de dados surge quando os valores dos atributos para a mesma entidade do mundo real diferem entre diferentes fontes. Essa disparidade pode ser atribuída a diferenças na representação, escala ou codificação adotadas por cada fonte. Tal divergência pode resultar em inconsistências nos dados, introduzindo divergências ou conflitos nos atributos associados à mesma entidade.

No contexto da análise de **dados relacionais**, onde os dados já estão no formato tabular normalizado após a importação de arquivos ou bancos de dados, é crucial notar que muitas funções analíticas e ferramentas de manipulação de dados operam de maneira mais eficaz em tabelas largas. Essas tabelas, que contêm todas as características/variáveis informativas organizadas em colunas, são preferidas em análises estatísticas devido à sua conveniência computacional, mesmo que essa abordagem possa introduzir alguma redundância. A **preferência por tabelas largas** simplifica o problema de integração, reduzindo-o à tarefa de juntar tabelas menores em uma única tabela que contenha todas as variáveis essenciais para uma análise eficiente. Essa integração facilita a detecção de dados correlacionados por meio de testes de correlação, a identificação de duplicatas de observações por meio da contagem das observações em uma tabela consolidada e a padronização na representação dos valores, definindo claramente os tipos de valores utilizados.

Essencialmente, existem duas classes de junções de tabelas (Capítulo 10 em [96]): as **junções baseadas na combinação** das variáveis das observações (*mutating joins*, em inglês), onde as tabelas são mescladas com base em uma variável específica compartilhada, e as **junções baseadas na filtragem** das observações de uma tabela m relação às observações contidas na segunda tabela (*filter join*,

em inglês). 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). 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.

Traduzindo para 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).

Dentre as **junções baseadas na combinação** das variáveis das observações de duas tabelas A e B, destacam-se

"$A \cap B$" (*inner join*, em inglês): a nova tabela contém as observações das duas tabelas, cujos valores das variáveis especificadas estão contidos nas observações de ambas as tabelas (Figura 6.6a).

"$A \cup B$" (*full join*, em inglês): 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 6.6b).

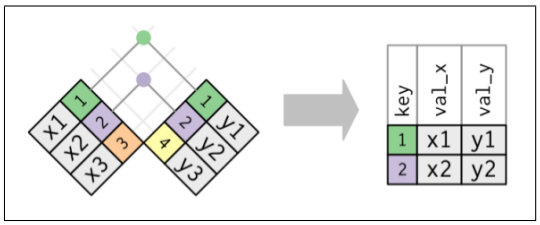
"$(A - B) \cup (A \cap B)$" (*left join*, em inglês): a nova tabela contém as observações das duas tabelas, cujos valores das variáveis especificadas estão contidos na tabela A (Figura 6.6b).

"$(B - A) \cup (A \cap B)$" (*right join*, em inglês): a nova tabela contém as observações das duas tabelas cujos valores das variáveis especificadas estão contidos nas observações da tabela B (Figura 6.6b).

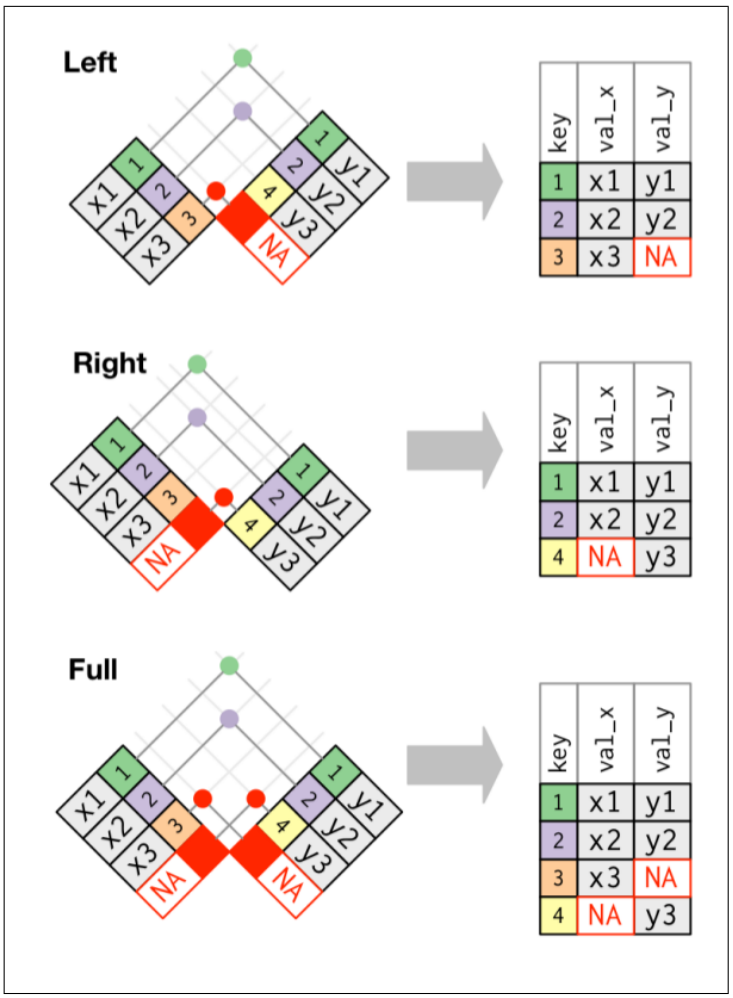
E das **junções baseadas em filtragem** das observações em A em relação aos valores das variáveis especificadas na tabela B, distinguem-se

filtragem por inclusão (*semi-join*, em inglês): 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 6.7a).

filtragem por exclusão (*anti-join*, em inglês): 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 6.7b).

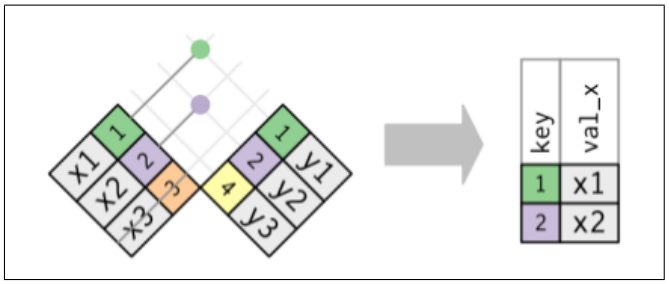


(a) *Inner Join*

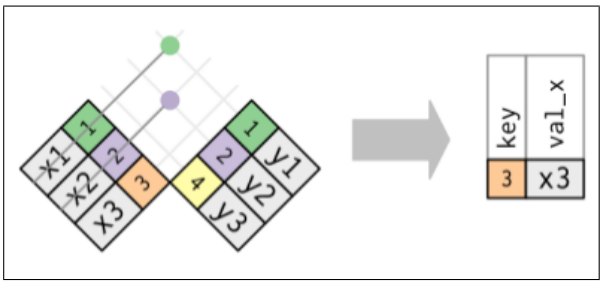


(b) *Outer Join*

Figura 6.6: 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: [96])



(a) *Semi Join*



(b) *Anti Join*

Figura 6.7: 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: [96])

A escolha entre esses métodos depende da estrutura dos dados e dos objetivos analíticos, 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 6.2.2, 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 [95]):

```
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 [95].

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 procecimento abaixo 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
```

Caso seja necessário **renomear** as colunas, pode-se usar o comndo rename do pacote dplyr em R (Capítulo 3 em [96]), como a seguinte instrução que 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 (Seção 5.4 em [35]):

```
py_dados_mat_tidy.rename(columns={’Mat’:’Matemática’})
```

Para **adicionar** colunas, usa-se tipicamente a função mutate em R (Capítulo 3 em [96]) e a função assign em Python (Seção 5.5 em [35]). 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
```

6.3 Transformação de Dados

Na transformação de dados, os dados tabulares são modificados ou consolidados em formas apropriadas para a sua análise [30]. Através de uma transformação de dados adequada, o processo de análise de dados visual pode ser mais eficiente, e os padrões encontrados podem ser mais fáceis de compreender.

Diversas estratégias para transformação de dados foram desenvolvidas. Nesta seção abordaremos as seguintes técnicas de transformações mais populares, a filtragem de dados, a reorganização de dados, a normalização de valores de dados, a redução de dados e a redução de dimensão dos dados. A **filtragem de dados** (Seção 6.3.1) extrai um subconjunto de observações que atendem a determinadas condições, enquanto a **reorganização** (Seção 6.3.2 reordena as observações, para necessidades específicas da análise. A **normalização** de valores de dados (Seção 6.3.3), diferente da normalização de dados relacionais (Seção 6.2.3), consiste em escalar os valores dos atributos para se situarem dentro de uma faixa menor, como $[-1,0;1,0]$ ou $[0,0;1,0]$. A **redução de dados** (Seção 6.3.4), por sua vez, transformam os dados de entrada em uma representação reduzida, muito menor em volume, mas que mantém de perto a integridade dos dados originais. A **redução de dimensionalidade** (Seção 6.3.5) consistem em reduzir a quantidade de variáveis aleatórias ou atributos ou características de interesse, removendo aqueles que são irrelevantes ou redundantes.

6.3.1 Filtragem

A Seção 6.2.1 abordou a limpeza de dados, destacando a importância das ações imediatas realizadas logo após a importação dos dados. Essas ações têm como objetivo lidar com desafios como valores ausentes, erros de digitação, inconsistências e outras imperfeições presentes nos dados brutos. Por outro lado, a **filtragem de dados** é uma etapa subsequente que ocorre após a integração de dados ou em estágios específicos do processo de análise. Essa fase envolve a seleção de observações com base em critérios específicos, como valores em determinadas colunas, intervalos de datas ou qualquer condição relevante para simplificar o processo de interpretação, tornando as análises mais claras e eficazes. Dentre os diversos cenários possíveis para filtragem, dois se destacam por sua frequência: a filtragem de duplicatas e a filtragem condicional de observações. Além disso, é comum também a aplicação de filtros para selecionar variáveis específicas das observações.

A **filtragem de duplicatas** é uma prática fundamental para assegurar a integridade dos dados, removendo **observações repetidas** que podem comprometer a precisão das análises. Observações duplicads podem surgir de várias fontes, como erros de coleta, integração de conjuntos de dados ou problemas nos processos de entrada. A identificação e remoção dessas duplicatas garantem que cada observação seja única, evitando distorções nos resultados e proporcionando uma base sólida para análises confiáveis.

A **filtragem condicional de observações** permite selecionar especificamente as observações (linhas) e variáveis (colunas) que atendem a critérios predefinidos. Esses critérios podem envolver nomes das variáveis, valores específicos em uma ou mais colunas, intervalos de tempo ou qualquer condição relevante para o contexto da análise. Essa abordagem permite focar nas observações ou nas variáveis mais pertinentes para o objetivo da investigação, eliminando ruídos e otimizando a eficiência da análise.

Em R, além da função distinct() do pacote dplyr para filtragem de linhas duplicatas, existem três funções básicas relacionadas com a filtragem de dados (Capítulo 3 em [96]):

duplicated() que retorna um vetor lógico indicando se cada elemento em um vetor ou uma coluna de um data frame é duplicado. Se o valor é TRUE, isso significa que o elemento é duplicado; se é FALSE, o elemento é único até aquele ponto no vetor/na coluna.

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 unique()

```
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 selected():

```
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 (Capítulo 7 em [95]), 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 (consulte a Seção 5.2 em [35]), e da função filter() para a seleção de colunas (características) com base em seus 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, respectivamente, linhas e colunas com base em critérios específicos, a função drop() em pandas é utilizada para remover linhas ou colunas específicas de um *data frame* com base em rótulos ou índices (Seção 5.4 em [35]). 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'])
```

6.3.2 Reorganização 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, a ordenação proporciona *insights* adicionais.

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 (Capítulo 3 em [96]) 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 (Seção 5.3 em [35]) são:

```
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.

6.3.3 Normalização de Valores

Para evitar dependências na escolha das unidades de medida, é recomendável 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 [30]):

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 $v_i\prime \in [new\_min_A, new\_max_A]$

$$v_i\prime = \frac{v_i - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A \tag{6.1}$$

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

$$v_i\prime = \frac{v_i - \overline{A}}{\sigma_A} \tag{6.2}$$

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

$$v_i\prime = \frac{v_i}{10^j} \tag{6.3}$$

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

$$\begin{aligned} -1 \leq log_{10}(\frac{max(|v_i\prime|}{10^j}) \leq 0 \\ -1 \leq log_{10}(max(|v_i\prime|) - log_{10}(10^j) \leq 0 \\ -1 \leq log_{10}(max(|v_i\prime|) - j \leq 0 \\ log_{10}(max(|v_i\prime|) - 1 \leq j \leq log_{10}(max(|v_i\prime|). \end{aligned} \tag{6.4}$$

É importante destacar que nos três métodos de normalização apresentados, é útil calcular medidas resumo, tais como médias, máximos e mínimos, conforme detalhado na Seção 5.1. 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. 6.4

```
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. 6.1, 6.2 e 6.3.

```
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. 6.3, basta usar a seguinte instrução:

```
mutate(dados_nots_full_n, 'Mat' = ifelse('Mat' == Mat,  Mat/10^j, 'Mat'))
```

Em Python, a Eq. 6.4 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. 6.3 pode ser feita pela seguinte linha de atribuição:

```
py_dados_notas_full_n['Mat'] = py_dados_notas_full_n['Mat']/(10**j)
```

6.3.4 Redução de Observações

A redução do volume de observações é uma estratégia essencial para lidar com grandes conjuntos 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.

Uma técnica comum para realizar a redução de observações é a **discretização**, que transforma variáveis contínuas em intervalos discretos (Seção 2.5.2 em [31]). As técnicas de discretização podem ser classificadas de acordo com como realizam a divisão, se utilizam informações de classe ou em

qual direção procedem (isto é, de cima para baixo versus de baixo para cima). Se o processo de discretização utiliza informações de classe, então dizemos que é uma **discretização supervisionada**. Caso contrário, é uma **discretização não-supervisionada**. Se o processo começa encontrando primeiro um ou alguns pontos, denominados **pontos de divisão** ou **pontos de corte**, para dividir toda a faixa do atributo e depois repete isso recursivamente nos intervalos resultantes, é chamado de **discretização de cima para baixo** ou **divisão**. Isso contrasta com a **discretização de baixo para cima** ou **fusão**, que começa considerando todos os valores contínuos como pontos de divisão potenciais, remove alguns mesclando valores vizinhos para formar intervalos e, em seguida, aplica recursivamente esse processo aos intervalos resultantes.

Dentre as técnicas básicas de discretização, destacam-se a discretização por intervaloes (*binning*), análise de histograma, análise de *cluster*, análise de árvore de decisão e análise de correlação. A discretização por análise de *cluster*, análise de árvore de decisão e análise de correlação podem ser consideradas mais orientadas à **supervisão**, uma vez que essas abordagens levam em conta a estrutura de classe dos dados. A **análise de *cluster*** agrupa os dados com base em semelhanças (Seção 5.4), enquanto a **análise de árvore de decisão** pode usar a classe como critério para dividir os dados em subgrupos. Embora a **análise de correlação** (Seção 5.2) não seja diretamente uma técnica de discretização, ela pode ser utilizada para identificar relações entre variáveis, o que pode ser relevante para a discretização, dependendo do contexto. As três técnicas demandam um conhecimento profundo dos dados e envolvem maiores interações com os analistas de dados. Os especialistas desempenham um papel fundamental na tomada de decisões sobre qual técnica aplicar, com base em seus conhecimentos, experiências e intuições. Assim, o suporte computacional se limita a fornecer uma variedade de ferramentas para clusterização, construção de árvores de decisão e correlação, deixando a escolha da abordagem nas mãos dos especialistas.

A discretização por intervalos (*binning*) e a discretização por análise de histograma (Seção 5.1.1) são geralmente consideradas **técnicas não-supervisionadas**, pois não dependem diretamente das informações de classe dos dados. O ***binning*** divide o intervalo contínuo dos dados em **intervalos discretos** ou ***bins*** com base nos valores observados, sem considerar explicitamente as classes a que pertencem. A **análise de histograma**, por sua vez, pode ser considerada uma forma de *binning*, pois envolve a contagem de ocorrências dentro de intervalos. Ela difere do *binning* em apresentações. O resultado do *binning* é uma representação tabular dos intervalos e da frequência com que os dados caem em cada intervalo, enquanto o histograma proporciona uma visualização da distribuição dos dados resultante do *binning*, destacando esta distribuição de uma maneira mais intuitiva.

Em técnicas não-supervisionadas, a divisão do intervalo contínuo dos dados em intervalos discretos pode ser pela **técnica de largura igual** (em inglês, *equal-width*), em que a largura de cada intervalo é uniforme, ou pela **a técnica de frequência igual** (em inglês, *equal-frequency*, em que os intervalos são criados de forma que o número de amostras em cada intervalo seja aproximadamente constante.

E, em seguida, substitui-se cada valor do intervalo discreto pela média, conhecida por **suavização por médias de intervalos**, ou mediana do intervalo, conhecida por **suavização por medianas de intervalo**.

Outra abordagem para reduzir o volume de dados é a **amostragem** (Seção 2.5.4 em [31]), que envolve a seleção de um subconjunto representativo dos dados. É uma estratégia eficaz para reduzir o volume de N observações, especialmente quando lidamos com grandes conjuntos de dados e o processamento de todo o conjunto é impraticável. O custo de obter uma amostra é proporcional ao tamanho da amostra n, em oposição a N. Portanto, a complexidade da amostragem é potencialmente sublinear em relação ao tamanho dos dados. Outras técnicas de redução de dados podem exigir pelo menos uma passagem completa por N observações. Para um tamanho de amostra n fixo, a complexidade da amostragem aumenta apenas linearmente à medida que a quantidade k de atributos/característica das observações aumenta, enquanto técnicas que usam histogramas, por exemplo, poderiam aumentar exponencialmente em k. Isso ocorre porque, ao aumentar o k, o espaço de características se expande exponencialmente.

Quando aplicada à redução de dados, a amostragem é predominantemente usada para estimar a resposta a uma **consulta sumarizada**. É possível, usando o Teorema Central do Limite, determinar um tamanho de amostra suficiente para estimar uma determinada função dentro de um grau especificado de erro. O **Teorema Central do Limite** é um princípio estatístico fundamental. Conforme detalhado na Seção 7.3, ele estabelece que, para um grande número N de amostras, cada uma de tamanho n, a distribuição das médias dessas amostras (ou de outras estatísticas) tende a se aproximar de uma distribuição normal, independentemente da forma da distribuição subjacente dos dados. Isso permite que os estatísticos utilizem as características bem conhecidas da distribuição normal para realizar inferências estatísticas, tais como a estimativa de médias populacionais ou a condução de testes de hipóteses, conforme explorado no Capítulo 8 sobre estatística de inferência.

Embora a estatística de inferência é uma abordagem comum para reduzir volumes de dados, pois permite extrair conclusões sobre uma população com base em uma amostra representativa dela, as estatísticas descritivas também podem ser úteis na redução da quantidade de dados. As estatísticas descritivas resumem e descrevem as características essenciais de um conjunto de dados, como média, mediana, moda, variância e desvio padrão. Ao aplicar estatísticas descritivas, é possível categorizar, agrupar ou resumir os dados de maneira que forneçam *insights* significativos sem a necessidade de examinar todos os dados individualmente.

Tanto R quanto Pyhton oferecem funções de categorização dos valores numéricos, e agrupamento e sumarização dos valores 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 de resumo sobre os valores agrupados com base em critérios específicos.

Em R, as funções cut()/qcut() 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, o R oferece a função group_by(), que, em conjunto com a função summarize() (Seção 6.3.3), permite reduzir as observações de uma tabela, agrupando-as com base em um conjunto predefinido de variáveis, e calcular estatísticas de resumo 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() 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 por um histograma, usando a função ggplot() do pacote ggplot2 em R/plotnine em 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
```

6.3.5 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 [31]). Essa abordagem é particularmente valiosa para analistas de dados visuais, pois proporciona uma maneira mais acessível de explorar e interpretar padrões em conjuntos de dados com uma grande quantidade de características. O resultado é a obtenção de *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 (PCA, do inglês *Principal Component Analysis*). 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. 6.2), 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 dados. 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 robustas funcionalidades para realizar redução de dimensionalidade. Em R, a função prcomp() do pacote básico realiza a Análise de Componentes Principais (PCA) [22], enquanto em Python, a função PCA() do módulo sklearn.decomposition executa a mesma técnica [76].

Embora o PCA seja amplamente conhecido por reduzir a quantidade de informações em conjuntos de dados complexos, as técnicas mais inovadoras para reduzir dimensões são aquelas baseadas em aprendizado de máquina. As técnicas de aprendizado de máquina procuram identificar os padrões relevantes ou estruturas nos dados que são mais siginificativas para a tarefa em questão e extrair características informativas a partir desses padrões, mantendo o máximo de informações importantes para análises ou modelagens futuras.

6.4 Considerações Finais

Neste capítulo dedicado à preparação de dados na Ciência de Dados, exploramos de maneira abrangente as três fases cruciais: importação, reorganização e reestruturação, e transformação de dados. Cada fase desempenha um papel vital no processo de garantir que os dados estejam prontos para análise, proporcionando insights valiosos e confiáveis. Ao longo do capítulo, proporcionamos uma visão prática, fornecendo exemplos e ferramentas tanto em R quanto em Python predominantemente com base em [96, 35][1], para capacitar os profissionais de dados a enfrentar desafios e extrair o máximo valor de seus conjuntos de dados. Cabe ressaltar que o capítulo foca em **dados relacionais** em que as classes de dados são organizadas em tabelas. 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 fase de importação (Seção 6.1), discutimos estratégias eficazes para trazer dados para o ambiente de análise, com foco nos ecossistemas RStudio e Jupyter Notebook, usando as ferramentas disponíveis em R e em Python. Reconhecendo a importância de carregar dados de diversas fontes, apresentamos exemplos práticos de como utilizar funções simples para essa tarefa em ambas as linguagens. Além da diversidade de formatos suportados por R e Python, suas funções de importação são altamente configuráveis, oferecendo uma extensa lista de argumentos para atender às convenções locais de formatação. É, portanto, crucial compreender as nuances de cada função, indo além dos exemplos básicos, para realizar escolhas mais alinhadas com os requisitos de um projeto específico. Por leis de transparência, muitos dados púbilicos nacionais estão abertos como arquivos que podem ser analisados por qualquer cidadão. Destacam-se os dados do sistema SUS no formato dbc [21], dados abertos da Agência Nacional de Energia Elétrica (ANEEL) no formato csv [5] e o portal de dados abertos do Tribunal Superior Eleitoral no formato csv [88]. Essas bases de dados proporcionam uma oportunidade para praticar a importação de dados de fontes de dados reais.

A reorganização e reestruturação dos dados (Seção 6.2) compreendem a segunda fase, onde nos

[1] 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.

concentramos na limpeza, formatação em *tidy* e integração de dados. Demonstramos, por meio de exemplos, como realizar a organização eficiente dos dados, garantindo sua consistência e tornando-os aptos para análises mais avançadas. Apresentamos algumas das inúmeras ferramentas específicas em R, como pacotes como dplyr e tidyr, e em Python, bibliotecas como pandas e numpy, que facilitam essa etapa. É importante ressaltar que as três etapas, limpeza, formatação e integração, não necessariamente ocorrem em uma sequência rígida. Elas podem se entrelaçar, dependendo da situação. Por exemplo, após a integração de dados, valores NA podem surgir e precisar ser removidos. As ferramentas apenas oferecem uma variedade de funções e alternativas discretas, deixando a cargo do analista de dados combiná-las de forma eficaz e confiável para realizar tarefas específicas.

Realizar uma boa análise exploratória dos dados (EDA do inglês *Exploratory Data Analysis*) é uma das etapas mais importantes em um projeto de ciência de dados. Uma boa EDA permite adquirir melhor conhecimento dos dados, gerar *insights* que podem auxiliar os cientistas de dados determinar a modelagem dos dados que impactarão de forma direta no desemepnho dos algoritmos de análise efetiva de dados, utilizando por exemplo aprendizado de máquina. Tanto R [42] quanto Python [110] forncem ferramentas adequadas para levantar perfis dos dados (*data profiling*, em inglês). Desenvolvido com o principal objetivo de proporcionar uma EDA consistente e rápida, tanto a biblioteca funModeling (R) quanto ydata-profiling (Python) geram relatórios completos com apenas uma linha de código. Cabe lembrar 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 de dados e suporte a transações atômicas.

A transformação de dados (Seção 6.3), nossa terceira fase, visa a padronização e simplificação dos dados. Isso ocorre por meio de processos como filtragem, redução de observações ou diminuição de variáveis, sempre com o cuidado de preservar informações cruciais para a análise desejada. A complexidade desta etapa se revela na diversidade de técnicas de aprendizado de máquina aplicadas para resolver problemas abertos, onde soluções analíticas não são viáveis. Navegamos por algumas funções simples disponíveis em ambas as linguagens R e Python, ilustrando partes dessa tarefa complexa e mostrando como preparar os dados para análises mais avançadas. Vale ressaltar que as técnicas baseadas em aprendizado de máquina estão fora do escopo deste texto.

6.5 Exercícios

1. Leia o texto em `https://www.kaggle.com/code/evertonsilva/data-wrangling-cleaning` e sintetize os principais passos de preparação de dados para aprendizado de máquina. Compare-os como os passos apresentados neste capítulo.

2. Reproduza os exemplos fornecidos nos Capítulos 6, 7 e 8 em [95] (Python) ou nos Capítulos 9 a 16

em [96] (R). Em ambas as referências, são abordadas diversas funções adicionais de manipulação dos dados, além das apresentadas neste capítulo, proporcionando uma visão mais abrangente das capacidades das respectivas linguagens de programação.

3. Leia o texto no apêndice C de [34] e sintetize as 7 dicas relacionadas com *data wrangling*.

4. A *Lahman Baseball Database* é uma das fontes de dados mais abrang entes e respeitadas para estatísticas e informações sobre *baseball* nos Estados Unidos e no Canadá. Foi criada por Sean Lahman e contém uma ampla variedade de dados históricos e estatísticas relacionadas ao *baseball*, incluindo informações sobre jogadores, equipes, estatísticas de temporada regular e pós-temporada, registros de jogos, salários, prêmios e muito mais. O *link* da base de dados é `http://seanlahman.com/`. Acesse os dados da ultima versão e faça os exercícios propostos na Seção 22.4 em [65].

5. Faça o exercício 13 da Seção 23.5 em [65]. Técnicas de raspagem da dados da rede em R e Python são, respectivamente, apresentadas no Capítulo 23 em [65] e em [24].

Capítulo 7

Probabilidade

A estatística descritiva, como discutido no Capítulo 5, 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 completa dos valores das características ao calcular tais medidas.

No vasto campo da 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 a medida da chance de ocorrência de um resultado possível 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 maior certeza na ocorrência do evento. Antes de explorar a relação entre probabilidade e estatística descritiva, é essencial estabelecer conceitos fundamentais de probabilidade. Na Seção 7.1, abordamos esses conceitos. Em seguida, na Seção 7.2, discutimos as operações básicas para calcular e manipular probabilidades, essenciais para compreender e aplicar princípios probabilísticos em análises estatísticas de inferência. Apresentamos, na Seção 7.4, uma abordagem probabilística para calcular as estatísticas descritivas discutidas no Capítulo 5. Definimos na Seção 7.3, variáveis aleatórias e exploramos seu papel na interpretação e análise dos dados de uma população, sem coletar todos os dados. e, na Seção 7.5, mostramos modelos de funções de probabilidade comumente usados na atribuição de probabilidade aos valores de uma variável de uma

população. Por fim, na Seção 7.6, introduzimos uma forma de simular um experimento sob diversas condições para capturar a variabilidade inerente a um sistema e compreender seu comportamento.

7.1 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 subjacente [30]. Uma **amostra** é um conjunto de observações, ou instâncias de medição, obtidas num experimento aleatório, estudo estatístico ou 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 [48]. 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 probabilidades idênticas a todas as faces. Na Seção 7.5 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 evento.

7.2 Operações Fundamentais de Probabilidade

Para construir um espaço de probabilidades (Ω, $\mathcal{F}$, 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. $\mathcal{F}$ 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 pertence 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 probabilidades de eventos compostos e analisar relacionamentos entre diferentes eventos num experimento aleatório ao aplicarmos as seguintes propriedades de probablidade [48]:

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

$$0 \leq P(A) \leq 1. \tag{7.1}$$

Probablidade do Complemento: A probabilidade do complemento A^c de um evento A é igual a 1 menos a probabilidade de A, ou seja,

$$P(A^c) = 1 - P(A). \tag{7.2}$$

Probablidade do Conjunto Vazio: A probabilida de um conjunto vazio é 0, ou seja,

$$P(\emptyset) = 0. \tag{7.3}$$

Probabilidade de União de Eventos: Sejam A e B eventos de Ω, então a probabilidade da união de A e B é

$$P(A \cup B) = P(A) + P(B) - P(A \cap B). \tag{7.4}$$

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,

$$P(A \cap B) = P(A)P(B). \tag{7.5}$$

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,

$$P(A \cap B) = 0 \quad \text{e} \tag{7.6}$$

$$P(A \cup B) = P(A) + P(B). \tag{7.7}$$

Quando lidamos com eventos que não são independentes ou 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

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \quad P(B) > 0. \tag{7.8}$$

Combinando as Eqs. 7.5 e 7.8, 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. 7.6 e 7.8, obtemos 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 ocorre porque a ocorrência de um evento mutuamente exclusivo torna a ocorrência do outro evento impossível, como expresso 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 pode ser visto 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}
```

7.3 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 para o qual a variável aleatória mapeia representa um conceito de interesse no 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 5 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, o valor numérico para o qual a variável aleatória mapeia (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, como por exemplo, se o evento ocorreu $(Y(\omega) = 1)$ ou não ocorreu $(Y(\omega) = 0)$.

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. A menos das ressalvas, as variáveis que exploramos são univariadas.

A **função de probabilidade**[1] $f(x)$ é uma função que descreve as probabilidades associadas aos possíveis valores que uma variável aleatória pode assumir. Ela atribui a cada valor $x \in X$ uma medida de probabilidade $P(X = x)$, respeitando as seguintes propriedades:

1. $f(x)$ é não-negativa, e
2. a soma de todas as probabilidades $P(X = x)$ para todos os valores possíveis em X é igual a 1.

Com base no tipo de variáveis aleatórias, a função de probabilidade é classificada em função de massa de probabilidade e função de densidade de probabilidade[2]. 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

[1] Muitas vezes, o termo **distribuição de probabilidades** (*probability distribution* em inglês) é utilizado de forma genérica para se referir tanto à distribuição de probabilidade discreta quanto à distribuição de probabilidade contínua. Optamos pelo termo função de probabilidade para distinguí-lo da **distribuição de frequências** de ocorrência dos dados observados empiricamente, introduzida na Seção 5.1.1.

[2] 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 para referí-las.

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** (PDF, do inglês *probability density function*) 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 7.1:

$$P(c \leq X \leq d) = \int_c^d f(x)dx \tag{7.9}$$

A PDF satisfaz, portanto, duas propriedades:

1. $f(x) \geq 0, \quad \forall x \in (-\infty, \infty)$, e
2. $\int_{-\infty}^{\infty} f(x)dx = 1$

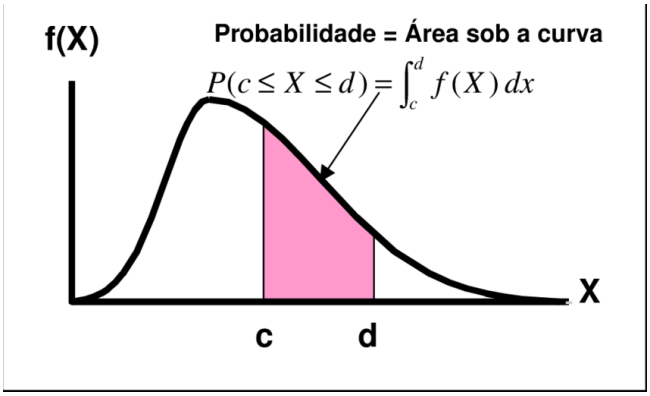
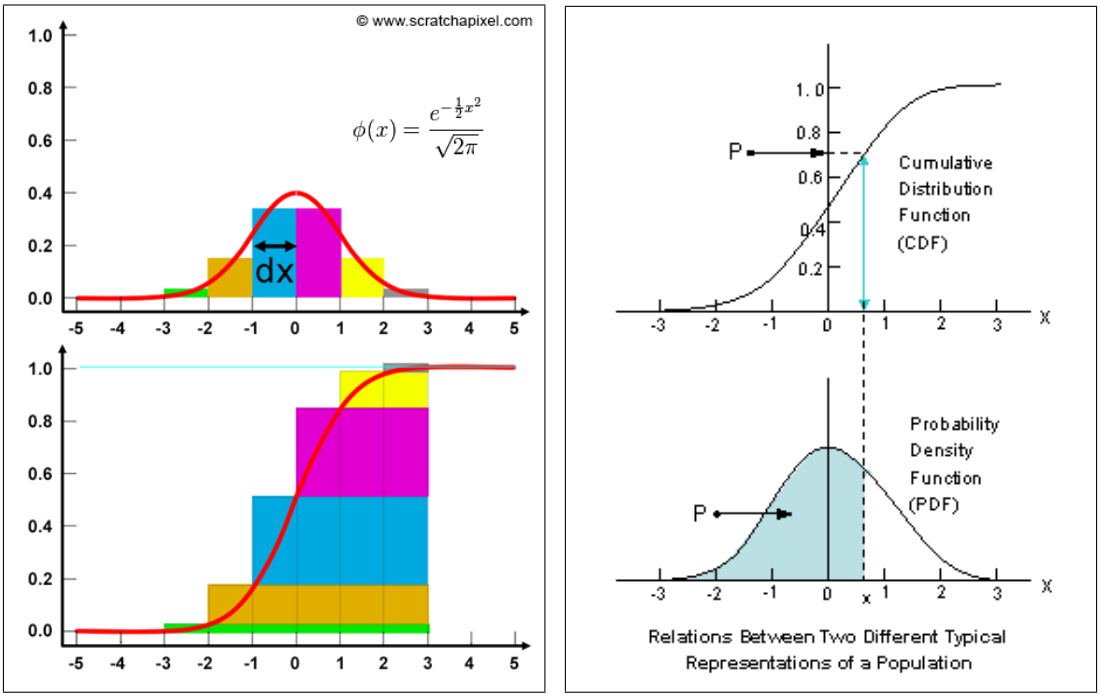


Figura 7.1: Função de densidade de probabilidade $f(x)$ e probabilidade $P(c \leq X \leq d)$. (Fonte: `https://www.researchgate.net/figure/Figura-37-Probabilidade-de-Variaveis-Aleatorias-Continuas_fig4_330703739`)

A partir da função de probabilidade $f(x)$, podemos definir a **função de distribuição acumulada** (CDF, do inglês *cumulative distribution function*) $F(x)$ que representa a probabilidade acumulada de que a variável aleatória X seja menor ou igual a um determinado valor x, ou seja,

$$F(x) = P(X \leq x) = \sum_{x_i \leq x} f(x_i) = \sum_{x_i \leq x} P(X = x_i), \tag{7.10}$$

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



(a) Discreta (b) Contínua

Figura 7.2: 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: `https://www.scratchapixel.com/lessons/mathematics-physics-for-computer-graphics/monte-carlo-methods-mathematical-foundations/pdf-and-cdf.html`), (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: `https://tex.stackexchange.com/questions/515670/probability-density-function-and-cumulative-distribution-function-for-normal-dis`).

de escada, e fornece a probabilidade acumulada para todos os valores de $x \in X$ no intervalo real como apresentada na Figura 7.2. A Seção 7.5 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 5.2, 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 $\widetilde{X}$ e $\widetilde{Y}$, cada uma com sua própria distribuição de probabilidade, que são chamadas de **funções de probabilidade marginais**. Ao atribuir uma probabilidade a cada combinação possível $(\tilde{x}, \tilde{y})$ de valores das variáveis aleatórias, indicando a chance de cada combinação ocorrer no

conjunto de dados, estaremos definindo uma **função de probabilidade conjunta**, denotada por

$$P[(\widetilde{X} = \tilde{x}) \cap (\widetilde{Y} = \tilde{y})] = P(\widetilde{X} = \tilde{x}, \widetilde{Y} = \tilde{y}),$$

onde $\tilde{x}$ e $\tilde{y}$ são os valores assumidos pelas variáveis aleatórias $\widetilde{X}$ e $\widetilde{Y}$, respectivamente. Isso nos permite quantificar a incerteza subjacente às combinações dos dados e entender a distribuição de possíveis resultados de um evento ou experimento.

Voltando ao problema da impossibilidade de coletar todos os dados de uma população exposto na introdução deste capítulo, as variáveis aleatórias surgem como candidatas ideais para representar amostras representativas devido à sua habilidade em capturar a complexidade e variabilidade dos dados. Fundamentadas na teoria da probabilidade e estatística, essas variáveis aleatórias oferecem um conjunto robusto de ferramentas para análise, interpretação e inferência de dados, incluindo o cálculo de médias, desvios padrão, intervalos de confiança e a realização de testes de hipóteses. Além disso, a validade desse enfoque é sustentada pelo Teorema Central do Limite.

O **Teorema Central do Limite** estabelece que, **quando as observações são independentes e o tamanho da amostra é grande**, a distribuição de probabilidade **das médias amostrais** se aproxima da distribuição normal. Isso ocorre independentemente da forma da distribuição original da população da qual as amostras são extraídas, como demonstrado na Figura 7.3. Nessa 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 assemelha-se ao formato de um sino, característico da distribuição normal.

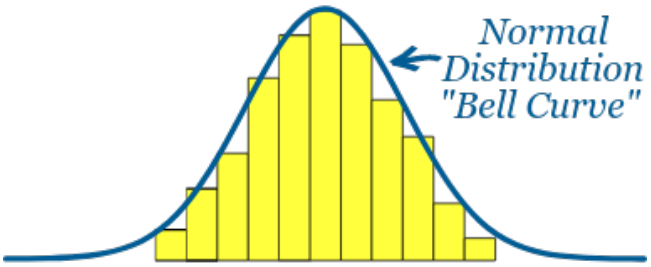


Figura 7.3: Teorema Central do Limite: distribuição normal das médias amostrais para qualquer população, se as observações forem independentes e a quantidade de repetições de amostragem for suficientemente grande. (Fonte: `https://www.mathsisfun.com/data/standard-normal-distribution.html`)

Isso tem importantes implicações para a inferência estatística e a tomada de decisões com base em amostras de dados, porque a distribuição normal é bem compreendida e tem propriedades matemáticas que permitem fazer inferências precisas sobre a população a partir das amostras mesmo quando a distribuição subjacente não é normal.

7.4 Medidas Resumo em Abordagem Probabilística

Usar uma abordagem probabilística para calcular medidas de estatística descritiva envolve o uso de variáveis aleatórias e distribuições de probabilidade para descrever os dados, oferecendo um rigor matemático sólido. A estatística descritiva frequentemente analisa conjuntos de dados como amostras de uma população maior. Ao aplicar conceitos de probabilidade e variáveis aleatórias, podemos quantificar a incerteza associada às medidas estatísticas, avaliando a precisão das inferências. Isso assegura consistência e precisão nos métodos estatísticos.

Além disso, uma abordagem probabilística é altamente generalizável e aplicável a diversos domínios, desde ciências naturais até finanças, tornando-se uma ferramenta versátil e robusta para análise de dados. Ao compreender as funções de probabilidade relacionadas aos dados, os especialistas podem fazer escolhas mais informadas, considerando tanto os resultados mais prováveis quanto os cenários de risco. A linguagem probabilística oferece uma maneira clara e precisa de comunicar resultados e incertezas, facilitando a interpretação e a comunicação dos resultados da análise estatística para diferentes públicos.

Segue-se uma abordagem fundamentada em métodos probabilísticos, contrastando com a abordagem anteriormente apresentada na Seção 5.1.2, 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 pelo seu correspondente probabilidade p_i e somando todos os produtos

$$E(X) = \mu_X = \mu = \sum_{i=1}^{N} x_i p_i. \tag{7.11}$$

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:

$$E(X) = \mu_X = \mu = \int_{-\infty}^{\infty} x f(x) dx. \tag{7.12}$$

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

$$\sigma^2 = Var(X) = \sum_{i=1}^{N}(x_i - \mu)^2 p_i, \tag{7.13}$$

enquanto para uma variável contínua X, as substituições análogas às da média de X são feitas:

$$\sigma^2 = Var(X) = \int_{-\infty}^{\infty}(x - \mu)^2 f(x)dx. \tag{7.14}$$

A **propriedade de linearidade da variância** nos diz que a 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:

$$\sigma = \sqrt{\sigma^2}. \tag{7.15}$$

Mediana: A mediana de uma função de probabilidade, representada por M_d, é o valor em que a probabilidade acumulada até esse ponto é igual a 0,5. Em outras palavras, metade das observações de uma variável X é menor ou igual a M_d, e metade é maior or igual a M_d. Isso é expresso pelas seguintes condições:

$$P(X \geq M_d) = 0.5 \;\text{ e }\; P(X \leq M_d) = 0.5. \tag{7.16}$$

Essas condições garantem que a mediana divida a probabilidade acumulada da função de probabilidade 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 probabilidade, 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 probabilidade 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 probabilidade, 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 $percentil \times 0.01$:

$$P(X \leq P_{percentil}) = percentil \times 0.01. \tag{7.17}$$

Por exemplo, $P(X \leq P_3) = 0.03$.

Moda: A moda M_o pode ser identificada como os máximos da função de probabilidade. 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,

$$P(X = x_i) = p(x_i) = p_i, \tag{7.18}$$

a moda pode ser expressa como

$$P(X = M_o) = max(p_1, p_2, \cdots, p_N). \tag{7.19}$$

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 satisfaça a condição

$$f(M_o) = max f(x). \tag{7.20}$$

A **covariância** quantifica como duas variáveis variam conjuntamente (Seção 7.3, 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. 5.9 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

$$Cov(X,Y) = E[XY] - E[Y]E[X] = 0. \implies E[XY] = E[X]E[Y].$$

Para padronizar a medida de relação entre as variáveis X e Y, introduziu-se a **correlação**, uma medida que normaliza a covariância em relação aos desvios-padrão populacionais (Seção 5.2):

$$\rho(X,Y) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}. \tag{7.21}$$

Uma 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[3] em R e as funções dos pacotes numpy e pandas em Python calculam estatísticas descritivas diretamento sobre os dados observados usando médias aritméticas, e não esperanças matemáticas. Usar as esperanças matemáticas requer conhecimento prévio sobre a distribuição dos dados ou um modelamento preciso de dados, que não é tão usual na prática em análise de dados exploratória.

7.5 Modelos de Funções de Probabilidade

Uma função de probabilidade é uma ferramenta sistemática e precisa para atribuir probabilidades a eventos ou 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 utilizam uma variedade de ferramentas e modelos matemáticos para explorar as funções de probabilidade associadas aos eventos e a variáveis aleatórias. Identificar padrões nessas funções é crucial não apenas para condensar dados complexos em formas matemáticas compreensíveis, mas também para aprimorar a interpretação dos dados e realizar inferências estatísticas mais robustas.

Muitos fenômenos aleatórios compartilham comportamentos semelhantes, frequentemente descritos por um conjunto comum de modelos de função 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

[3] 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. No entanto, 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.

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.

Cabe ressaltar aqui que embora possam haver distribuições de frequência, ilustradas na Figura 5.2, que se assemelham a certos modelos de funções de probabilidade, eles não são os mesmos. Enquanto as distribuições de frequência representam a contagem de valores específicos em um conjunto de dados observados, os modelos de função de probabilidade descrevem distribuições teóricas de variáveis aleatórias.

7.5.1 Variáveis Discretas

Para variáveis aleatórias discretas, alguns exemplos comuns de modelos de funções de probabilidade, análogos às distribuições de frequência de ocorrência de valores de uma variável apresentadas na Seção 5.1.1, incluem [48]:

Função de probabilidade uniforme: Atribui-se a mesma probabilidade a todos os k possíveis valores de uma variável aleatória X:

$$P(X = x_j) = \frac{1}{k}, \quad \forall j = 1, 2, 3, \cdots, k \tag{7.22}$$

Função de probabilidade 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

$$P(X = k) = p^k (1 - p)^{(1-k)}, \quad x = 0, 1, \tag{7.23}$$

onde p representa a probabilidade de sucesso.

Função de probabilidade Binomial: É expressa através da equação

$$P(X = k) = \left(\begin{array}{c} n \\ k \end{array} \right) p^k (1 - p)^{(n-k)}, \quad k = 0, 1, \cdots, n \tag{7.24}$$

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 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 Poisson é:

$$P(X = k) = \frac{e^{-\lambda}\lambda^k}{k!}, \tag{7.25}$$

onde λ é a taxa média de ocorrência de eventos no intervalo considerado.

7.5.2 Variáveis Contínuas

Entre os exemplos dos modelos de funções de probabilidade para variáveis contínuas, temos [48]:

Função de probabilidade uniforme: Atribui-se a mesma densidade de probabilidade a todos os valores de uma variável aleatória no intervalo $[a, b]$:

$$f(x) = \begin{cases} \frac{1}{b-a}, & a \leq x \leq b \\ 0, & \text{caso contrário.} \end{cases} \tag{7.26}$$

Assim, a média e a variância de uma distribuição uniforme são, respectivamente,

$$\begin{aligned} \mu &= \int_a^b x \cdot f(x)dx = \frac{b+a}{2} \\ \sigma^2 &= \int_a^b (x-\mu)^2 \cdot f(x)dx = \frac{(b-a)^2}{12} \end{aligned} \tag{7.27}$$

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.

$$f(x) = \begin{cases} \alpha e^{-\alpha x}, & x \geq 0 \\ 0, & \text{caso contrário.} \end{cases} \tag{7.28}$$

É possível demonstrar através de cálculos de integração que, para uma distribuição exponencial,

$$\begin{aligned} \mu &= \int_0^\infty x \cdot f(x)dx = \frac{1}{\alpha} \\ \sigma^2 &= \int_0^\infty (x-\mu)^2 \cdot f(x)dx = \frac{1}{\alpha^2} \end{aligned} \tag{7.29}$$

Função de probabilidade normal: É representada através da expressão

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, -\infty < x < \infty \tag{7.30}$$

A distribuição normal é uma das mais importantes na Estatística. São **propriedades fundamentais de uma distribuição normal** é que a sua média e variância são iguais aos parâmetros da distribuição

$$\begin{aligned} \mu &= \int_{-\infty}^{\infty} x \cdot f(x)dx = \mu \\ \sigma^2 &= \int_{-\infty}^{\infty} (x-\mu)^2 \cdot f(x)dx = \sigma \end{aligned} \tag{7.31}$$

As variáveis aleatórias de muitos fenômenos se comportam próximas a esta distribuição. 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. Existe uma **tabela de distribuição normal padrão** $N(Z; 0, 1)$ [6], também conhecida como **distribuição normal padronizada**, que permite consultar a probabilidade de qualquer valor z de uma variável aleatória normalizada Z especificada em escore-z (Eq. 5.8)[4]. A nova variável Z nos permite consultar todas as probabilidades $P(0 \leq Z \leq z)$ na tabela de distribuição normal padrão.

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 p de sucesso em uma única tentativa

$$f(k) = P(X = k) = \begin{pmatrix} n \\ k \end{pmatrix} \cdot p^k \cdot (1-p)^{n-k} \tag{7.32}$$

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 p. Para a variância, usamos o fato de que a

[4] 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, preservando as relações de ordem e as proporções entre os valores.

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, $p(1-p)$. Ou seja,

$$\begin{aligned}\mu &= n \cdot p \\ \sigma^2 &= Var(X_1 + X_2 + \cdots + X_n) = nVar(X) = n \cdot p \cdot (p-1)\end{aligned} \tag{7.33}$$

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

$$\begin{aligned}Var(X) &= \sum_{i=1}^{2}(x_i - \mu)^2 p_i = (0-p)^2(1-p) + (1-p)^2 p = p^2(1-p) + (1-2p+p^2)p \\ &= p^2 - p^3 + p - 2p^2 + p^3 = p - p^2 = p(1-p).\end{aligned} \tag{7.34}$$

Funçao 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 pela expressão:

$$f(t \mid \nu) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\Gamma(\frac{\nu}{2})}(1 + \frac{t^2}{\nu})^{-\frac{\nu+1}{2}}, \tag{7.35}$$

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

$$\Gamma(t) = \int_0^\infty t^{z-1}e^{-t}dt, \tag{7.36}$$

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.

$$f(x) = \frac{1}{2^{\frac{d}{2}}\Gamma(\frac{d}{2})} x^{\frac{d}{2}-1} e^{-\frac{x}{2}}, \quad x \geq 0. \tag{7.37}$$

A sua média e o desvio-padrão são dados pelas seguintes equações:

$$\begin{aligned} \mu &= d \\ \sigma^2 &= 2d. \end{aligned} \tag{7.38}$$

7.5.3 Gráficos de Distribuições em R e Python

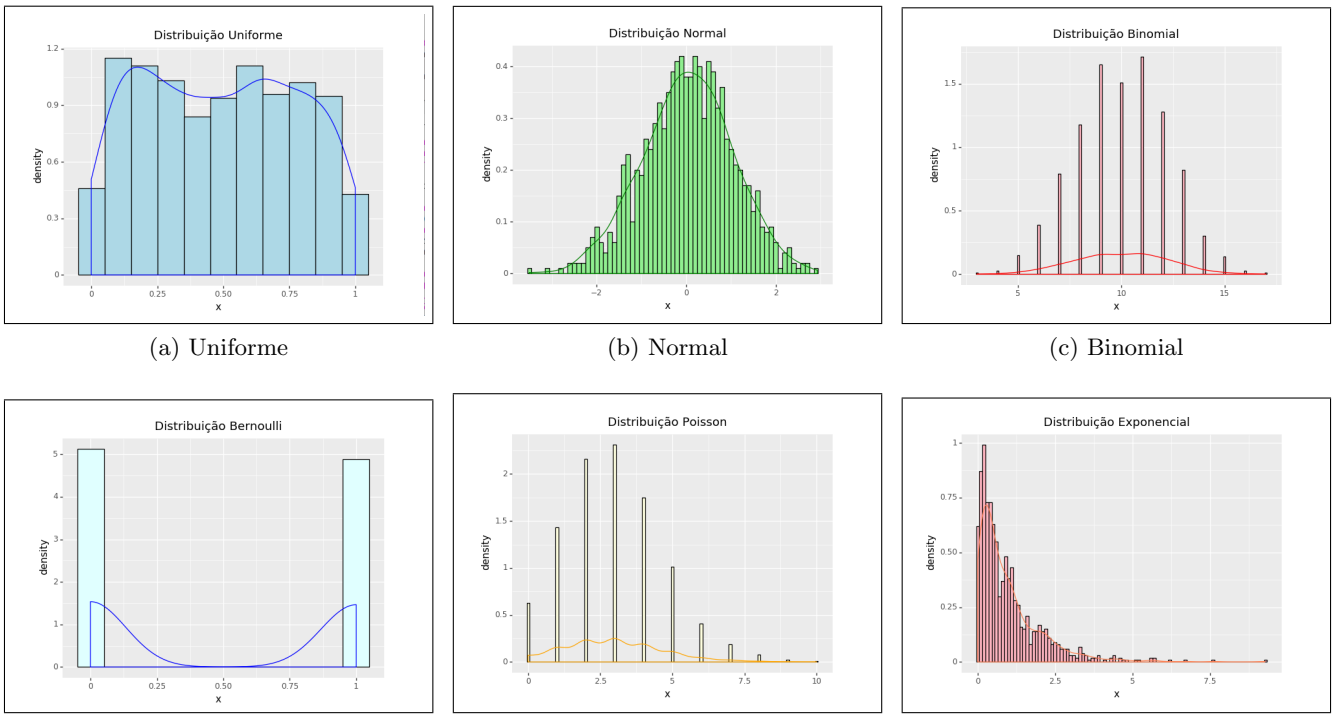
Tanto em R quanto em Python, é possível criar gráficos que representam as distribuições de frequência dos valores de uma variável e traçar curvas de densidade de probabilidade que melhor se ajustam a essas distribuições. Para realizar essa tarefa, podemos utilizar bibliotecas como ggplot2 em R e plotnine 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 7.4.

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")
```



(a) Uniforme (b) Normal (c) Binomial

(d) Bernoulli (e) Poisson (f) Exponencial

Figura 7.4: 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.

```
# 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 os gráficos de distribuições usando o pacote plotnine, é essencial carregar as distribuições implementadas no pacote scipy.stats. Ao carregar scipy.stats, que faz parte da biblioteca SciPy, é importante também carregar o NumPy para garantir o funcionamento adequado de todas as funcionalidades da SciPy. A SciPy depende do NumPy para representações de matrizes, vetores e outras estruturas de dados numéricos, bem como para operações de manipulação de dados e cálculos numéricos subjacentes. Além disso, é necessário carregar o pandas para transformar os dados gerados aleatoriamente em uma estrutura DataFrame, que pode ser processada eficientemente 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")
)
```

7.6 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 através da geração de vários conjuntos diferentes de valores aleatórios, como mostrado na Seção 1.2.1, que seguem certas funções de probabilidade das variáveis envolvidas no modelo. Cada conjunto é considerado uma realização possível do experimento. Ao repetir a simulação muitas vezes, as estatísticas e as distribuições das saídas das simulações são usadas para estimar as propriedades do sistema ou modelo em questão e resultados em sistemas complexos e determnísticos.

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 funções de probabilidade apropriadoss 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.

7.6.1 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 k amostras, cada uma consistindo de n lançamentos. 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 k vezes para obter uma distribuição das N proporções amostrais de frequências e estimar 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 nas N repetições segue uma distribuição binomial.

Geração de Amostras Aleatórias : Usamos funções disponíveis em R/Python para gerar N amostras aleatórias seguindo 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)
```

7.7 Considerações Finais

Ao longo deste capítulo, buscamos fornecer uma base sólida sobre a probabilidade de sua relação com as estatísticas descritivas. Iniciamos explorando os conceitos essenciais de probabilidade na Seção 7.1, delineando as noções básicas que são fundamentais para qualquer análise subsequente. Na sequência, na Seção 7.2, abordamos as operações básicas para calcular e manipular probabilidades. Esses princípios são essenciais para computar incertezas inerentes às análises estatísticas de inferência usando as variáveis aleatórias introduzidas na Seção 7.3. Ao definir e explorar o papel dessas variáveis, pudemos compreender a sua relevância na representação e no cômputo de incertezas nos fenômenos aleatórios e na formulação de modelos estatísticos.

Na Seção 7.4, introduzimos uma abordagem probabilística para calcular as estatísticas descritivas mencionadas no Capítulo 5. 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 estabelece uma base sólida para 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 das probabilidades através dos gráficos das funções de probabilidade, que mostram como os dados são distribuídos, ajudam 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 7.5.3 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 funções de probabilidade 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 de Monte Carlo, na Seção 7.6. Essa técnica oferece uma maneira de **simular** 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.

7.8 Exercícios

1. Faça os *learning checks* LC7.1 – LC7.7 propostos em [34].

2. Após revisar a Seção 14.9 em [65], descreva, em uma sentença, um equívoco comum associado à Lei dos Números Grandes.

3. 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.

4. 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 contraintuitiva, que desafia a intuição inicial. Ambos os problemas envolvem situações em que a probabilidade aparente de um evento pode ser enganosa e contraintuitiva, levando muitas pessoas a tirarem conclusões incorretas. Leia a Seção 13.7 em [65] para explorar os dois problemas, e descubra como esses desafios podem ser abordados empiricamente usando a técnica de Monte Carlo. Para verificar se os resultados das simulações (`stick`, `switch` e `results`) seguem uma distribuição normal, faça um um histogram e um gráfico quantil-quantil (QQ-plot) com a distribuição normal para cada resultado. Use a função `ggplot`.

Capítulo 8

Estatística de Inferência

Embora a estatística descritiva, explorada no Capítulo 5, seja uma ferramenta valiosa para resumir e visualizar conjuntos de dados, sua aplicação direta em populações muito grandes torna-se impraticável ou impossível devido ao tamanho, custo ou complexidade dos dados envolvidos. Imaginem tentar calcular medidas descritivas, como média, desvio padrão e quartis, para uma população com milhões ou bilhões de observações. Além de ser extremamente demorado e exigir recursos computacionais consideráveis, esse processo seria logisticamente inviável.

Em vez de tentar examinar toda a população, os estatísticos fazem uso de técnicas probabilísticas, introduzidas no Capítulo 7, para extrair conclusões válidas de amostras representativas. Uma **amostra** é considerada **representativa** quando reflete fielmente as características importantes da população de onde foi retirada. A Seção 7.1 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 poderosas para resumir e explorar dados observados de uma população, e a probabilidade nos oferece uma medida da proporção de ocorrência de eventos específicos em um conjunto de resultados possíveis, a **estatística de inferência** foca em fazer inferências abrangentes sobre as características da população, como média, variância, proporção e outras, utilizando apenas uma coleção N de amostras de 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 de inferência está o conceito de **estimativa**, que envolve o cálculo de **estimativas** ou valores aproximados para os **parâmetros**

populacionais ou estatísticas (resumidas) da população (Seção 5.1), com base nas amostras coletadas.

A estatística de inferência permite que profissionais de diversas áreas obtenham *insights* valiosos, realizem análises significativas, tomem decisões informadas, façam previsões confiáveis e desenvolvam estratégias eficazes com base em informações limitadas, mas representativas, da população em estudo. Na Seção 8.1, veremos como a estimativa pontual busca prever um único valor representativo de um parâmetro específico da população, como a média populacional derivada da média da amostra. Já na Seção 8.3, os intervalos de confiança fornecem uma faixa de valores plausíveis para o parâmetro populacional, refletindo a incerteza associada à estimativa pontual. Os testes de hipóteses, explorados na Seção 8.4, são empregados para fazer afirmações sobre os parâmetros populacionais com base em evidências amostrais, envolvendo a formulação de uma hipótese nula e uma hipótese alternativa.

Na estatística, todas as formas de inferência dependem do entendimento das distribuições de estatísticas amostrais, que são abordadas na Seção 8.2. Para realizar estimativas pontuais de parâmetros populacionais, como a média, é comum usar uma distribuição das observações da população. Os intervalos de confiança são construídos com base nas propriedades das distribuições amostrais subjacentes aos procedimentos de inferência, como a distribuição amostral. Nos testes de hipóteses, comparamos estatísticas derivadas dos dados amostrais com distribuições subjacentes, como a distribuição t-Student ou a distribuição normal padrão.

Nesse contexto, a visualização dos dados desempenha um papel crucial 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, sendo fundamentais para a correta aplicação das técnicas de inferência estatística. Ao longo do capítulo, são explorados os recursos gráficos que proporcionam uma compreensão mais profunda do comportamento dos dados subjacentes aos resultados de uma estatística de inferência.

8.1 Estimativas Pontuais

Um dos tipos mais comuns de inferência estatística é a estimativa de uma estatística de interesse de uma população, como a média, a proporção ou a variância. Costuma-se representar essa estatística com a letra grega, tais como θ, μ e σ. Essas estatísticas são denominadas **parâmetros populacionais** quando são valores fixos e desconhecidos. Para calcular uma suposição sobre um parâmetro θ, usamos uma função chamada de **estimador amostral**. Este estimador calcula a partir de amostras de n observações da população uma **estimativa pontual**, também referida como estatística da amostra ou simplesmente estimativa, de θ. Normalmente, denotamos os estimadores usando o símbolo do

parâmetro de interesse correspondente com um acento circunflexo, como $\hat{\theta}$, $\hat{\mu}$ e $\hat{\sigma}$.

As características de uma população podem variar de uma amostra para outra devido à aleatoriedade inerente à amostragem de dados observados. É, portanto, conveniente tratar cada valor individual x_i da variável como uma variável aleatória X_i. Assim, definimos $\hat{\theta}$ como uma função de n variáveis aleatórias correspondentes às n observações de uma amostra:

$$\hat{\theta} = f(X_1, X_2, \cdots, X_n). \tag{8.1}$$

Aos valores específicos que obtemos ao aplicar esta função às n observações chamamos de **estimativas pontuais**, ou simplesmente **estimativas**.

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 estimador pode ser, também, uma função que assume o valor do segundo valor sorteado na amostra

$$\hat{\mu}_2(X_1, X_2, \cdots, X_n) = X_2.$$

O exemplo apresentado na Seção 7.6 ilustra, de fato, uma estimativa da frequência de ocorrência da cara (C) no lançamento de uma moeda, a partir de 1000 amostras de 10 lançamentos usando como estimador a frequência de ocorrência de C em cada amostra de 10 observações para obter 1000 estimativas.

Na estatística de inferência, os estimadores são ferramentas fundamentais para fazer inferências sobre parâmetros desconhecidos da população com base em amostras. Um estimador $\hat{\theta}$ pode ser avaliado quanto a várias características importantes que determinam sua qualidade e utilidade em estimar o parâmetro θ de interesse. Seguem-se as definições das principais características de $\hat{\theta}$ [48]:

Imparcialidade: $\hat{\theta}$ é considerado **não-viciado**, ou **não-viesado**, se, em média, seu valor estimado $\hat{\theta}$ é igual ao valor real do θ que está sendo estimado. Em outras palavras, o valor esperado do estimador é igual ao valor verdadeiro do parâmetro

$$E(\hat{\theta}) = \theta.$$

Isso significa que, ao repetir o processo de amostragem e aplicação do estimador várias vezes, a média das estimativas converge para o valor real do parâmetro.

Eficiência: $\hat{\theta}$ é considerado **eficiente**, se tiver uma variância pequena em relação a outros

estimadores não tendenciosos para o mesmo parâmetro. Em outras palavras, um estimador $\hat{\theta}_1$ é mais eficiente do que $\hat{\theta}_2$, se

$$Var(\hat{\theta}_1) < Var(\hat{\theta}_2).$$

Dizemos que um estimador eficiente fornece estimativas precisas com um mínimo de variação em torno do valor verdadeiro do parâmetro.

Consistência: $\hat{\theta}$ é **consistente** se, à medida que o tamanho da amostra aumenta indefinidamente, a probabilidade do estimador se aproximar do valor verdadeiro do parâmetro θ também aumenta. Em outras palavras, o estimador converge para o valor real do parâmetro 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

$$\hat{\theta}_1(T(X)) < T(\hat{\theta}_2(X)).$$

Isso significa que se aplicarmos uma transformação T à variável de interesse, o estimador deve permanecer o mesmo ou se transformar de uma maneira conhecida.

Robustez: $\hat{\theta}$ é considerado **robusto** se suas propriedades permanecerem estáveis mesmo quando os pressupostos subjacentes são violados ou quando há presença de *outliers* nos dados. Um estimador robusto é capaz de fornecer boas estimativas mesmo em condições adversas.

Essas são algumas das características principais pelas quais os estimadores são avaliados. Escolher um bom estimador é um desafio para os estatísticos, pois envolve considerar várias características e propriedades do estimador, como sem viés, eficiência, consistência, robustez, invariância, entre outros. Equilibrar essas características para obter estimativas precisas e confiáveis do parâmetro de interesse é fundamental para garantir a qualidade das análises estatísticas e das conclusões derivadas delas.

Tipicamente, os estimadores "naturais" são derivados das estatísticas amostrais, que consistem nos valores específicos observados de uma amostra de tamanho n de uma população, modelados como variáveis aleatórias $\{X_1, X_2, \cdots, X_n\}$:

Média amostral: É calculada como a média aritmética dos valores observados em uma amostra específica:

$$\hat{\mu} = \overline{X} = \frac{\sum_{i=1}^{n} X_i}{n}. \tag{8.2}$$

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:

$$s^2 = \frac{\sum_{i=1}^{n}(X_i - \overline{X})^2}{n-1}. \tag{8.3}$$

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:

$$s = \sqrt{s^2}. \tag{8.4}$$

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:

$$\hat{p} = \frac{\text{número de ocorrências de } X_i \text{ na amostra}}{n}. \tag{8.5}$$

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 [65].

Embora os estimadores baseados em estatísticas 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 de interesse. Métodos como a abordagem bayesiana [102] e o método da máxima verossimilhança [103] são também empregados devido à sua eficiência, robustez, adequação ao modelo estatístico subjacente ou requisitos específicos do problema em questão.

8.2 Distribuições de Estatísticas Amostrais

As **distribuições de estatísticas amostrais** fornecem uma descrição da variabilidade das estatísticas derivadas de diferentes amostras retiradas da mesma população. Elas desempenham um papel crucial na estatística inferencial, permitindo-nos fazer inferências sobre parâmetros populacionais com base em amostras limitadas e fornecendo *insights* sobre a precisão das estimativas e a incerteza associada aos

resultados. Dois tipos comuns de distribuições amostrais são a distribuição amostral, que representa a variabilidade de uma estatística de interesse, como a média ou o desvio padrão, calculada a partir de todas as possíveis amostras da população, e a distribuição *bootstrap*, que é construída a partir de amostras repetidas, com reposição, de uma amostra original.

8.2.1 Distribuições Amostrais

Quando realizamos amostragens repetidas de uma população, obtemos diversas estimativas $\overline{X}_i$ para cada estatística amostral i. Essas estimativas, ao serem reunidas, formam uma distribuição de frequência que descreve a frequência com que cada valor ocorre entre os estimadores em N amostras de tamanho n retiradas da população (Seção 5.1.1). Essa frequência pode ser vista como uma medida empírica da probabilidade associada a cada valor, aproximando-se da distribuição amostral, como ilustra a Figura 8.1. Em outras palavrs, as **distribuições amostrais** representam de maneira probabilística as estimativas obtidas de diferentes amostras da mesma população, apresentando características próprias, como forma geral, tendência central e variabilidade. Embora nem sempre reproduzam exatamente a distribuição de probabilidade populacional, as distribuições amostrais são amplamente aplicadas na estatística de inferência dos parâmetros populacionais, mostrando como essas amostras podem variar em torno desses parâmetros.

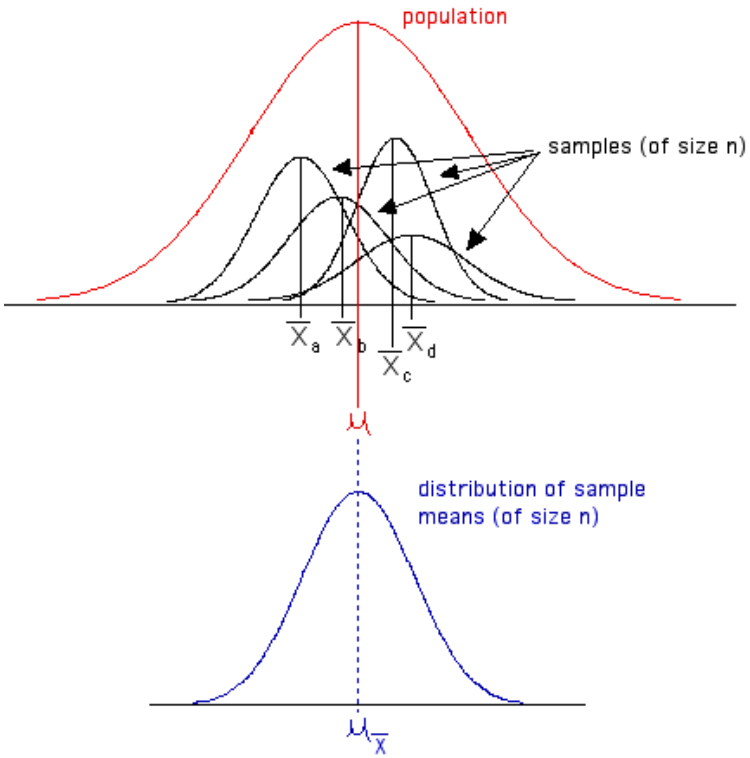


Figura 8.1: Variabilidade das estatísticas das diferentes amostras de mesmo tamanho de uma população refletidas em distribuições amostrais (Fonte: `https://psychology.illinoisstate.edu/jccutti/psych340/fall02/oldlecturefiles/images/sampdist.GIF`)

Por exemplo, quando a distribuição populacional é normal, com média μ e desvio padrão σ, e o

tamanho das amostras n é grande o suficiente, suas distribuições amostrais também serão normais (simétricas). Elas apresentarão propriedades específicas para a tendência central, com a média tendendo a se aproximar de μ, e variabilidade dependente do tamanho da amostra $\frac{\sigma}{\sqrt{n}}$. Em certos cenários, quando a distribuição populacional é assimétrica e o tamanho da amostra n é suficientemente grande, a **distribuição amostral da média**, por exemplo, tende a se aproximar de uma distribuição normal. Isso ocorre independentemente da forma da distribuição populacional original, conforme estabelecido pelo Teorema Central do Limite, apresentado na Seção 7.3. A Figura 8.2 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 a simetria das distribuições amostrais aumenta à medida que o tamanho da amostra cresce.

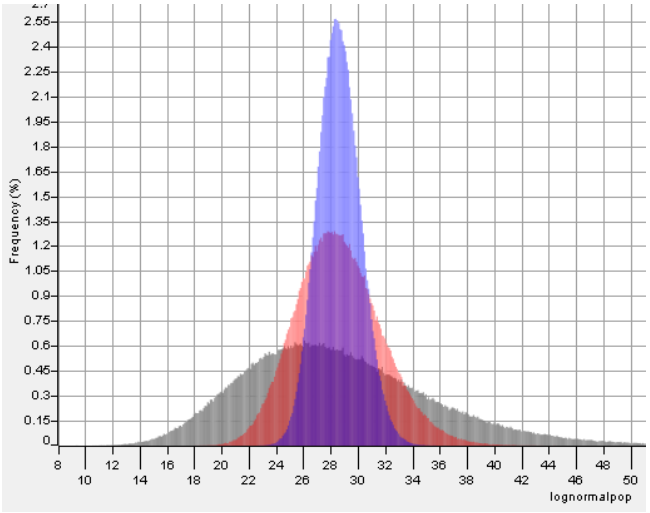


Figura 8.2: Distribuição populacional dos pesos de um conjunto de maçãs, com μ=100 e σ=15 (gráfico em cinza), juntamente com duas distribuições amostrais geradas a partir deste conjunto. As amostras têm tamanhos de n=5 (gráfico em vermelho) e n=20 (gráfico em azul), respectivamente, com 500.000 amostras em cada caso. (Fonte: `https://statisticsbyjim.com/hypothesis-testing/sampling-distribution/`)

As distribuições amostrais fornecem informações importantes sobre as propriedades de um **estimador**, que é um procedimento utilizado para calcular estimativas (Seção 8.1). Por exemplo, considerando uma característica populacional representada por X, com média μ e variância $\sigma^2 = Var(X)$, a distribuição amostral de N amostras de tamanho n pode fornecer informações sobre:

Tendência central: A tendência central das estimativas obtidas a partir do estimador, geralmente representada pela média da distribuição amostral, é expressa por

$$\mu_{\overline{X}} = \frac{\sum_{i=1}^{N} \overline{X}_i}{N}, \tag{8.6}$$

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 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. Para a média de uma distribuição amostral $\overline{X}$ e assumindo que as variâncias de cada observação X_{ij} em relação à média amostral correspondente $\overline{X}_i$ são iguais à variância populacional $Var(X)$, a variância da média amostral do estimador pode ser calculada da seguinte maneira

$$Var(\overline{X}) = Var(\frac{\overline{X}_1 + \overline{X}_2 + \cdots + \overline{X}_N}{N}).$$

Podemos expandir esta expressão para

$$\begin{aligned} \frac{\sum_{i=1}^{N} Var(\overline{X}_i)}{N} &= \frac{\sum_{i=1}^{N} \frac{\sum_{j=1}^{n} Var(X_{ij})}{n^2}}{N} = \frac{\sum_{i=1}^{N} \frac{nVar(X)}{n^2}}{N} \\ &= \frac{N\frac{Var(X)}{n}}{N} = \frac{Var(X)}{n} = \frac{\sigma^2}{n}, \end{aligned} \tag{8.7}$$

onde n é o tamanho da amostra e σ^2, a variância populacional. Eq. 8.7 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 dizer que quanto maior o tamanho da amostra n, menor é a variabilidade e, portanto, maior a precisão do estimador. Isso significa que, com amostras maiores, o estimador tende a se concentrar mais em torno do verdadeiro valor do parâmetro populacional. Figura 8.2 mostra graficamente 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), também conhecido por **viés**, do estimador é definido como o desvio padrão da distribuição das estimativas amostrais. Uma maneira de interpretar o erro padrão é considerá-lo como a diferença entre a média amostral $\mu_{\overline{X}}$ e a média populacional μ. Quanto menor o erro padrão, mais próximas as médias amostrais tendem a estar da verdadeira média populacional, o que indica uma estimativa mais precisa. O cálculo do erro padrão $SE(\overline{X})$ envolve o desvio padrão $\sigma_{\overline{X}}$ da distribuição amostral, que por sua vez, segundo o Teorema Central do Limite, pode ser aproximado pelo desvio padrão σ da população e pela quantidade n de amostras:

$$SE(\overline{X}) = \sigma_{\overline{X}} = \sqrt{Var(\overline{X})} = \frac{\sigma}{\sqrt{n}}. \tag{8.8}$$

Isso reflete o fato de que, à medida que a quantidade da amostras aumenta, a dispersão das médias amostrais ao redor da média populacional tende a diminuir. A Figura 8.3 ilustra graficamente a diferença entre o desvio padrão de uma distribuição amostral e o erro padrão do estimador.

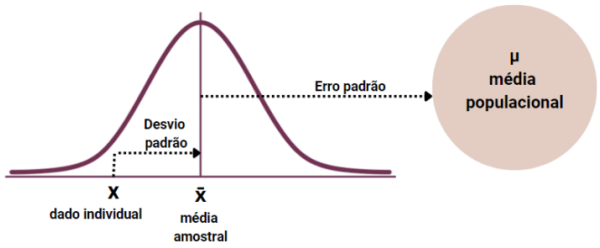


Figura 8.3: Desvio padrão e erro padrão. (Fonte: `https://blog.proffernandamaciel.com.br/desvio_erro_padrao/`)

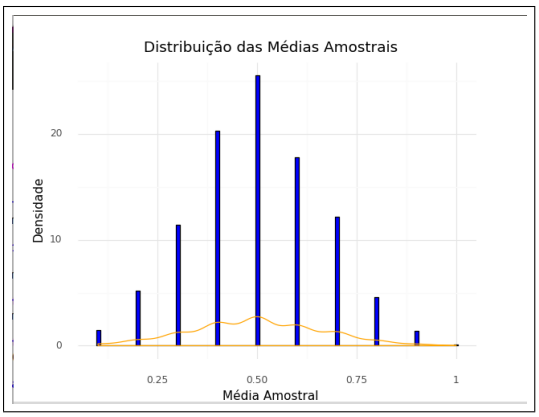
Estimar o parâmetro de **proporção populacional** é uma prática comum e útil na estatística inferencial, especialmente em pesquisas de opinião, estudos de mercado, saúde pública e muitos outros campos. O exemplo apresentado na Seção 7.6.1 é uma aplicação do Teorema Central do Limite na **estimativa da 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, a média de cada variável $E(Y_i) = \mu_{Y_i} = p$, e a variância de cada variável é dada pela Eq. 7.34. Um estimador não viciado $\hat{p}$ para p usando n variáveis aleatórias é dado por

$$E(\hat{p}) = E(\sum_{i=1}^{n} \frac{Y_i}{n}) = p \quad Var(\hat{p}) = Var(\sum_{i=1}^{n} \frac{Y_i}{n}) = \frac{p(1-p)}{n}$$

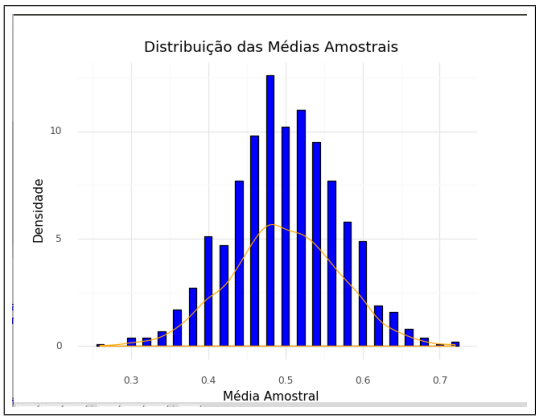
Quando n é grande o suficiente para que a aproximação normal seja válida para a distribuição de proporções amostrais, é vantajoso recorrer a ela para calcular probabilidades relacionadas à proporção amostral. Isso simplifica significativamente a análise estatística e a interpretação dos resultados, devido à familiaridade e facilidade de cálculo associadas à distribuição normal.

Além disso, as distribuições amostrais também podem fornecer informações sobre a forma da distribuição, possíveis valores extremos e a presença de assimetria. As distribuições amostrais são, de fato, as **distribuições de probabilidade de estimadores**. 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. Algumas das distribuições mais conhecidas de estimadores incluem a distribuição normal, a distribuição t de Student e a distribuição qui-quadrado (Seção 7.5.2), que são frequentemente utilizadas em diferentes contextos inferenciais, dependendo das características dos dados e do tipo de estimador em questão.

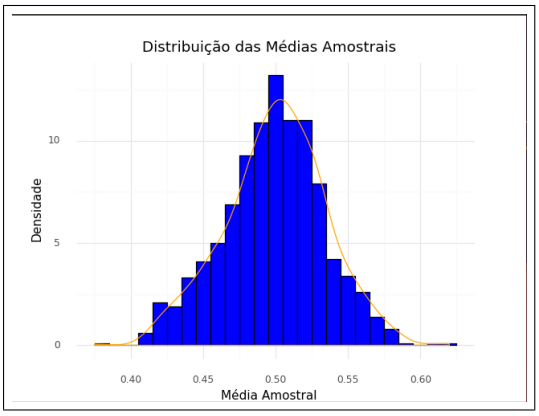
Pode-se plotar a distribuição de frequência de ocorrência das proporções amostrais usando geom_histogram disponível na biblioteca ggplot2 (R) ou plotnine (Python). Por ilustrar, vamos plotar as médias amostrais mean_samples geradas no exemplo da Seção 7.6.1 como mostrado na Figura 8.4a. Adicionalmente, 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 8.4 como os erros padrão diminuem e a curva de distribuição aproxima da curva de distribuição normal à medida que n cresce.



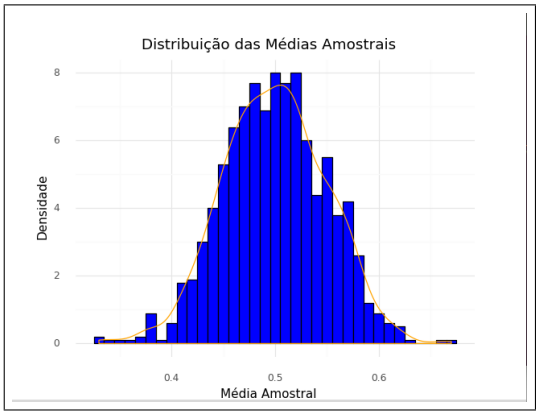
(a) n=10, $\mu_{\overline{X}}$=0.4956, $SE(\overline{X})$=0.004897



(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 8.4: Distribuição amostral das 1000 amostras, cada uma composta por n lançamentos. As amostras de $n = 10$ lançamentos são as do exemplo dado na Seção 7.6.1. Note como a distruibução representada pela curva em laranja tende a uma distribuição normal quando n cresce.

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 a consistência e facilitar a manipulação dos dados posteriormente, caso 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.

```
# 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 7.6.1, 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 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 a partir de uma população de 2400 bolinhas vermelhas e azuis previamente definidas. A Figura 8.5 ilustra uma distribuição amostral gerada.

R: São usadas as bibliotecas rsample e ggplot2. A biblioteca rsample oferece uma variedade de métodos para realizar amostragem e reamostragem.

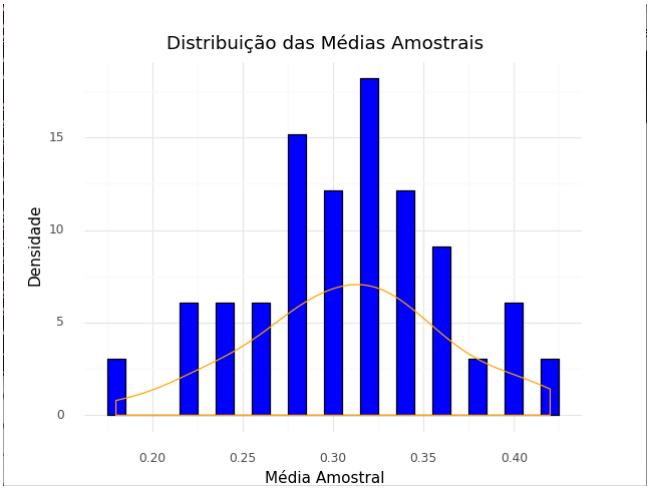


Figura 8.5: Distribuição amostral de $N = 33$ amostras, cada uma com tamanho $n = 50$, obtidas de uma população de 2400 bolinhas azuis e vermelhas, com uma proporção de 70% para azuis e 30% para vermelhas.

```
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 Amostr
  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())
```

8.2.2 Distribuições *Bootstrap*

A inferência baseada em distribuições amostrais pode ser desafiadora por várias razões. Primeiro, o desconhecimento da distribuição populacional subjacente dos dados dificulta a inferência sobre parâmetros populacionais e estatísticas de interesse. Além disso, muitas técnicas estatísticas tradicionais dependem de suposições sobre a distribuição dos dados, como normalidade e homogeneidade de variância, que nem sempre são realistas ou verificáveis na prática. Por fim, em algumas situações, o tamanho da amostra pode ser pequeno, o que pode resultar em estimativas imprecisas ou não confiáveis da distribuição amostral.

As **distribuições *bootstrap*** oferecem soluções para esses desafios de várias maneiras. Primeiramente, não exigem suposições teóricas sobre a distribuição dos dados. Além disso, são altamente flexíveis e aplicáveis a uma ampla variedade de situações estatísticas, independentemente da forma da distribuição dos dados ou do tamanho da amostra. Por fim, fornecem uma aproximação empírica da distribuição amostral da estatística de interesse, permitindo inferências estatísticas sem a necessidade de suposições teóricas sobre a distribuição dos dados. Essas distribuições são obtidas através de uma técnica conhecida como reamostragem *bootstrap* (*bootstrap resampling* em inglês).

Introduzida por Bradley Efron em 1979, a técnica *bootstrap*[1] 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

[1] 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".

tamanho da amostra é grande o suficiente, a distribuição *bootstrap* pode se aproximar da distribuição normal, como estabelece o Teorema Central do Limite.

Cabe ressaltar que, para estimar a distribuição de uma estatística de interesse a partir de uma única amostra de dados, sem a necessidade de fazer suposições sobre a distribuição teórica dos dados, é necessário que a amostra original seja uma representação razoável da população subjacente.

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 8.2.1, utilizando tanto R quanto Python. A Figura 8.6 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 8.5, é bem significativa a semelhança entre as duas distribuições.

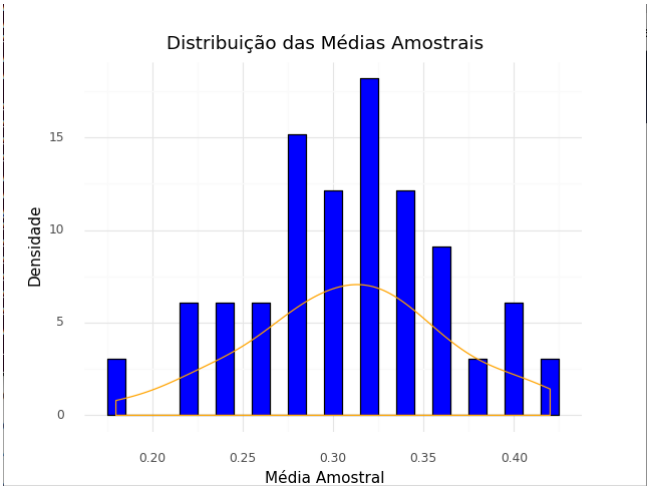


Figura 8.6: Distribuição *bootstrap* de $N = 33$ amostras, cada uma com tamanho $n = 50$, obtidas de uma amostra original de tamanho n=50 extraída a partir de uma população com uma proporção de 70% para azuis e 30% para vermelhas.

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())
```

8.2.3 Distribuições Padronizadas

A normalização das distribuições amostrais 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. Além disso, o uso de *bootstrap* pode se beneficiar da normalização, facilitando a estimativa de intervalos de confiança e a avaliação da incerteza estatística.

A **normalização** consiste em transformar os valores de uma variável de forma que ela siga uma distribuição normal ou se aproxime dela. Isso pode ser feito por meio de diferentes técnicas, sendo a mais comum a padronização. A **padronização** é um tipo específico de normalização em que os valores são transformados de modo que a média $\mu_{\overline{X}}$ ((Eq. 8.6) seja igual a 0 e o desvio padrão σ seja igual a 1 (Eq. 8.8)

$$\text{estimativa normalizada com variância conhecida} = \frac{\overline{X} - \mu_{\overline{X}}}{\frac{\sigma}{\sqrt{n}}}. \tag{8.9}$$

No entanto, se o desvio padrão populacional σ é desconhecido, é comum aproximar esse valor pelo desvio padrão amostral s (Eq. 8.4)

$$\text{estimativa normalizada com variância desconhecida} = \frac{\overline{X} - \mu_{\overline{X}}}{\frac{s}{\sqrt{n}}}. \tag{8.10}$$

A padronização resulta em uma distribuição em que a maioria dos valores está centrada em torno de zero e a variabilidade é comparável entre diferentes variáveis.

Dentre as distribuições de estatísticas amostrais, as distribuições normais padronizadas da **estatística z** e as distribuições t-Student padronizadas da **estatística t**, ambas introduzidas na Seção 7.5.2, desempenham papéis fundamentais na realização de inferências estatísticas e na construção de intervalos de confiança.

A **distribuição normal padronizada**, também conhecida como **distribuição Z**, é uma distribuição de probabilidade que, conforme o Teorema Central do Limite, muitas estatísticas amostrais seguem quando o tamanho da amostra é suficientemente grande. A distribuição normal padrão (Z) possui média igual a 0 e desvio padrão igual a 1. Por outro lado, a **distribuição t padronizada** é uma distribuição estatística que surge quando estamos lidando com médias de amostras pequenas e populacionais com variância desconhecida. Ela é semelhante à distribuição normal padronizada, mas varia com o número de graus de liberdade ($df = \nu = (n-1)$) e suas caudas são mais pesadas. A distribuição t-Student padronizada tem também média 0 e desvio padrão igual a 1 para todos os graus de liberdade.

A relação entre as duas distribuições reside no fato de que, à medida que o tamanho da amostra aumenta, a distribuição t padronizada se aproxima da distribuição normal padrão. Quando o tamanho da amostra é grande (geralmente considerado como $n \geq 30$), as duas distribuições são praticamente idênticas, como ilustra a Figura 8.7. Devido a essa convergência [48], as tabelas de probabilidade para a distribuição t-Student padronizada [84] são limitadas a valores de graus de liberdade menores ou iguais a 120. Para graus de liberdade superiores a 120, as probabilidades são obtidas a partir da tabela da distribuição normal padrão [6].

O **gráfico quantil-quantil** (Figura 5.8) é uma ferramenta visual eficaz para avaliar a normalidade dos dados. Nesse gráfico, os quantis (valores ordenados) dos seus dados são plotados no eixo y, enquanto os quantis esperados de uma distribuição teórica (como a distribuição normal) são plotados no eixo x. Se os seus dados seguem aproximadamente uma distribuição normal, os pontos no QQ-plot devem formar uma linha aproximadamente diagonal. Se os pontos desviam significativamente dessa linha, isso sugere que os seus dados podem não seguir uma distribuição normal.

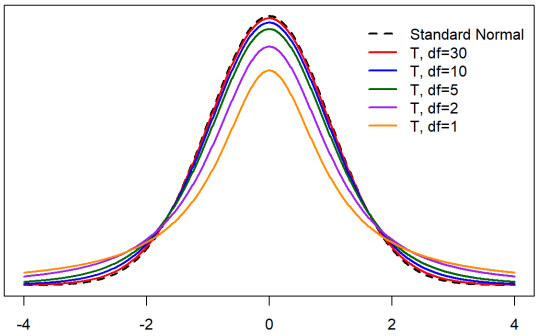


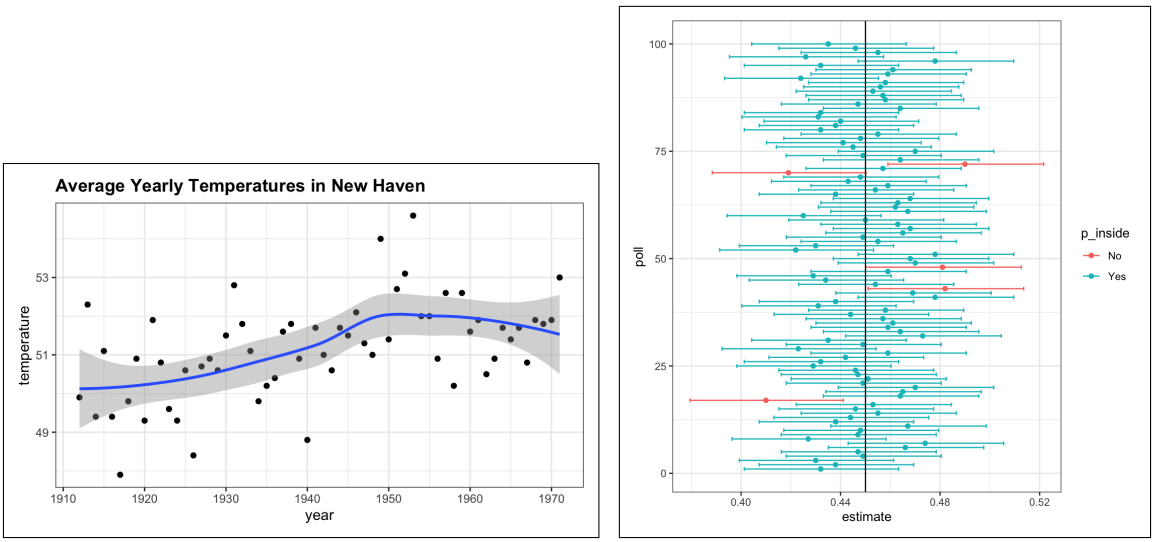
Figura 8.7: Visão comparativa entre a distribuição normal padrão (estatística z) e as distribuições t-Student de distintos graus de liberdade $df = \nu = (n-1)$, onde n é o tamanho da amostra. (Fonte: `https://bookdown.org/lgpperry/introstats/IntroStats_files/figure-html/unnamed-chunk-84-1.png`)

8.3 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 8.1, 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 potencialmente infinito de estimativas que poderiam ser obtidas de diferentes amostras da mesma população. Na Seção 8.2, exploramos a noção de distribuição amostral, que é uma distribuição que engloba todas as possíveis estimativas pontuais que poderiam ser calculadas a partir de diferentes amostras da população. Essa distribuição amostral possui propriedades fundamentais, como a média, que coincide com o parâmetro populacional que estamos tentando estimar (Figura 8.1), e o desvio padrão, que reflete a variabilidade das estimativas pontuais em torno da verdadeira média populacional.

A partir das distribuições amostrais, podemos derivar um par de média e desvio padrão amostral para cada amostra. Esses pares fornecem informações sobre a variabilidade do estimador e nos permitem quantificar a incerteza em torno de uma estimativa pontual. Isso é ilustrado na Figura 8.8, onde podemos ver a dispersão das médias amostrais em torno da média populacional verdadeira, bem como a dispersão dos desvios padrão amostrais. Essa visualização ajuda a entender a precisão da estimativa e a avaliar a confiabilidade da inferência estatística feita a partir dos dados amostrais.

Uma **estimativa intervalar** é uma estimativa baseada em uma amostra que fornece uma faixa de valores plausíveis onde o parâmetro populacional provavelmente estará contido com uma certa incerteza como mostrado na Figura 8.8. 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, geral-



(a) Gráfico de faixa

(b) Gráfico de intervalos de confiança

Figura 8.8: Gráficos de distribuição das variações de um estimador ao calcular uma estatística de interesse. (Fonte: [65])

mente 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 de qual parâmetro populacional desejamos estimar. Isso pode ser a média, a proporção, a variância, a mediana ou qualquer outra medida de interesse.

Escolha do estimador: Escolha do método de estimativa apropriado para o parâmetro em questão. Isso pode envolver o uso de estimadores pontuais específicos, como a média amostral, a proporção amostral, a mediana amostral, etc.

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. Quanto maior o nível de confiança escolhido, maior será a largura do intervalo.

8.3.1 Distribuição Subjacente da Amostra

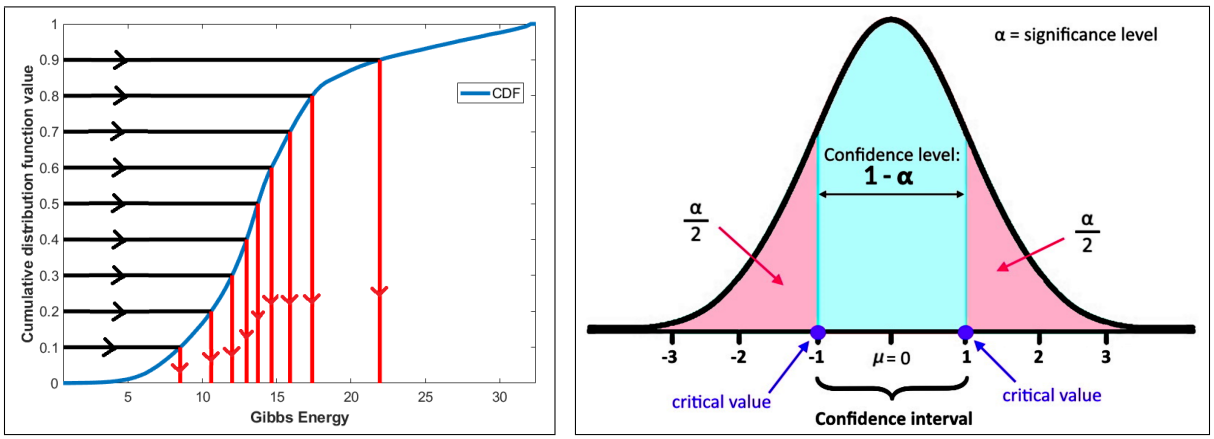
É importante ter uma noção da distribuição subjacente da amostra ao estimar um intervalo de confiança. A distribuição subjacente da amostra influencia diretamente a forma como calculamos o intervalo de confiança e como interpretamos seus resultados. Por exemplo, se a amostra segue uma distribuição normal, podemos usar os métodos baseados na distribuição normal, como a distribuição normal e t-Student padronizada, para calcular o intervalo de confiança. Caso contrário, pode ser necessário recorrer a métodos não paramétricos ou técnicas de reamostragem, como a distribuição *bootstrap*, para estimar o intervalo de confiança de forma mais precisa.

A técnica de *bootstrap* nos permite criar múltiplas amostras *bootstrap* a partir de uma amostra original, com reposição (Seção 8.2.2. 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.

8.3.2 Cálculo de Intervalos de Confiança

Para determinar os extremos do intervalo de confiança associado a um nível de confiança, ou a probabilidade, $(1-\alpha) \in [0,1]$, dois métodos comuns são frequentemente empregados [34]:

Método do percentil: Os extremos do intervalo de confiança são derivados dos percentis $\frac{\alpha}{2}$ e $1-\frac{\alpha}{2}$ da distribuição amostral da estatística de interesse. Esses percentis representam os pontos na distribuição em que a probabilidade acumulada é igual a



(a) Método do Percentil (b) Método do Erro Padrão

Figura 8.9: Cálculo dos extremos de um intervalo de confiança: (a) Método do percentil (Fonte: `https://dmn92m25mtw4z.cloudfront.net/img_set/stat-6-6-x-4-article/v1/stat-6-6-x-4-article-1256w.png`) e (b) Método do erro padrão. (Fonte: `https://www.researchgate.net/publication/359215074/figure/fig3/AS:1139021417840640@1648575276453/An-illustrative-example-of-the-estimation-of-CDF-percentile-feature-from-CDF.ppm`)

$\frac{\alpha}{2}$ e $1-\frac{\alpha}{2}$, respectivamente. Eles são determinados a partir da função de distribuição cumulativa, conforme detalhado na Seção 5.1.1. Embora simples e intuitivo, este método pode exigir esforço computacional considerável em grandes conjuntos de dados.

Método do erro padrão: Os extremos do intervalo de confiança são estabelecidos utilizando o erro padrão da estatística de interesse e o valor associado a $\frac{1-\alpha}{2}$, obtido a partir de tabelas de distribuição normal ou t-Student padronizadas. O erro padrão, calculado como descrito na Equação 8.8, representa a variabilidade esperada da estatística de interesse na amostra. Este método, fundamentado em propriedades teóricas das distribuições estatísticas, é frequentemente empregado em situações onde os conjuntos de dados são extensos ou a distribuição da estatística de interesse é conhecida.

Por exemplo, os **intervalos de confiança** para uma média populacional μ podem ser definidos como intervalos simétricos em torno da média amostral $\overline{X}$ da distribuição normal aproximada, com largura determinada pelo erro padrão da média $SE(\overline{X})$ e multiplicado pelo valor $z_{\frac{1-\alpha}{2}}$ da distribuição normal padrão $N(Z;0,1)$ associado ao nível de confiança desejado $1-\alpha$, onde $P(\mid Z \mid < z_{\frac{1-\alpha}{2}}) = 1-\alpha$

$$CI(\mu, 1-\alpha) = \overline{X} \pm (z_{\frac{1-\alpha}{2}} \times SE(\overline{X})) = \overline{X} \pm (z_{\frac{1-\alpha}{2}} \times \frac{\sigma}{\sqrt{n}}). \tag{8.11}$$

Na Figura 8.10, 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 $[-2\times SE(\overline{X}), +2\times SE(\overline{X})]$. Es-

ses 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 consultarmos a tabela de distribuição normal padrão [6]. Esses valores são exatamente os multiplicadores dos desvios-padrão da distribuição amostral apresentados na Eq. 8.11.

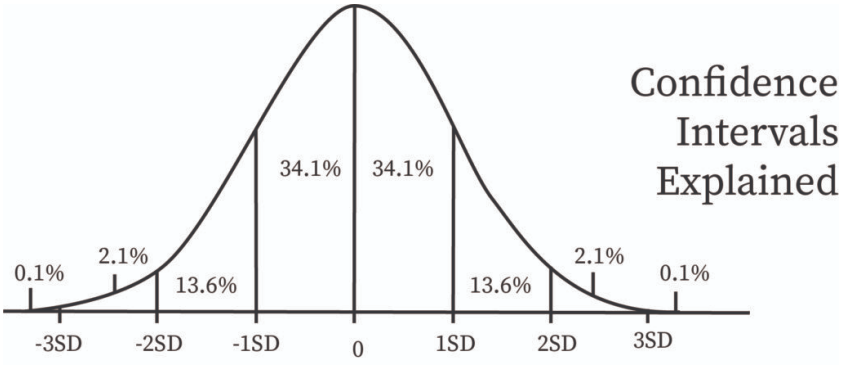


Figura 8.10: Intervalos de confiança para níveis de confiança $p = 68,26\%$ e $p = 95.4\%$: $[-SE(\overline{X}), +SE(\overline{X})]$ e $[-2\times SE(\overline{X}), +2\times SE(\overline{X})]$, respectivamente. (Fonte: `https://www.bartleby.com/subject/math/statistics/concepts/confidence-intervals`)

Quando a variância é desconhecida, podemos construir intervalos de confiança para μ usando o modelo t-Student. Através da tabela de distribuição t-Student com $(n-1)$ graus de liberdade [84], 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

$$CI(\mu, 1-\alpha) = \overline{X} \pm (t_{\frac{1-\alpha}{2}} \times SE(\overline{X})) = \overline{X} \pm (t_{\frac{1-\alpha}{2}} \times \frac{s}{\sqrt{n}}). \tag{8.12}$$

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 da análise. 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 7.5.2).

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** [48], como:

$$\hat{p} = \frac{\sum_{i=1}^{n} X_i}{n}, \text{ onde } X_i \in \{0,1\}$$
$$Var(\hat{p}) = \frac{\hat{p}(1-\hat{p})}{n}. \tag{8.13}$$

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 $1 - \alpha$.

Uma segunda **abordagem conservativa** é considerar que $p(1-p) \leq \frac{1}{4}$, ou seja,

$$\hat{p} = \frac{\sum_{i=1}^{n} X_i}{n}, \text{ onde } X_i \in \{0, 1\}$$
$$\text{Var}(\hat{p}) = \frac{\hat{p}(1-\hat{p})}{n} \leq \frac{\frac{1}{4}}{n}. \tag{8.14}$$

Ao aproximarmos a distribuição binormal pela distribuição normal, podemos obter o valor de $z_{\frac{1-\alpha}{2}}$ na tabela de distribuição normal padrão através do nível de confiança $1 - \alpha$. Substituindo $z_{\frac{1-\alpha}{2}}$ na Equação 8.11, 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.

Ao invés de depender da tabela de distribuição normal padrão [6] para encontrar os valores $z_{\frac{p}{2}}$, tanto R quanto Python oferecem funções que realizam esses cálculos automaticamente. 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 7.6.1. 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`.

8.3.3 Interpretação de Intervalos de Confiança

Para interpretar corretamente os intervalos de confiança, é fundamental 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 [34].

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. Amostras maiores tendem a resultar em intervalos mais estreitos, mantendo o mesmo nível de confiança. É 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.

8.4 Testes de Hipóteses

Os intervalos de confiança fornecem uma estimativa do parâmetro populacional, juntamente com uma medida de incerteza em torno dessa estimativa, utilizando amostras da população. Quando surge a necessidade de avaliar a plausibilidade de uma afirmação específica sobre o parâmetro populacional com base em evidências de uma amostra, recorremos aos testes de hipóteses. Esses testes envolvem o uso de uma estatística de teste e a formulação de uma hipótese nula (H_0) sobre essa estatística, que presume que o parâmetro avaliado não causa uma mudança real na população. Ao lado disso, há uma hipótese alternativa (H_1 ou H_a), que apresenta a afirmação contrária e pode levar a uma redistribuição amostral, conforme ilustrado na Figura 8.11.

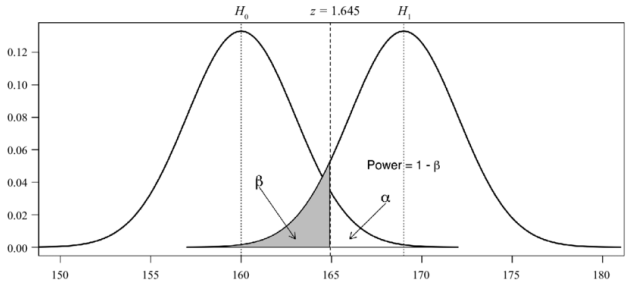


Figura 8.11: Distribuições amostrais para duas hipóteses de testes: nula (H_0) e alternativa (H_1). (Fonte: `https://www.researchgate.net/figure/Sampling-distributions-for-the-null-H-0-and-alternative-H-1-hypotheses_fig2_272117096`)

Um procedimento clássico de conduzir um teste de hipóteses Z (distribuição normal) ou T (distribuição t-Student) compreende tipicamente as seguintes etapas:

Formulação das hipóteses: Formular as hipóteses nula (H_0) e alternativa (H_a). Usualmente, a hipótese nula geralmente afirma que não há efeito ou diferença, enquanto a hipótese alternativa afirma o oposto.

Escolha do nível de significância: Escolher uma probabilidade que represente a tolerância às evidências contrárias à hipótese nula quando ela é verdadeira. Essa probabilidade é conhecida como **nível de significância**, geralmente denotada por α (Figura 8.11.

Seleção da estatística de teste apropriada: Com base na natureza dos dados e das hipóteses, escolhe-se uma estatística de teste apropriada.

Coleta de dados e cálculo da estatística de teste: Em seguida, coletam-se os dados relevantes e calcula-se a estatística de teste apropriada a partir desses dados.

Tomada de decisão: Com base no valor da estatística de teste calculada, compara-se com o **valor crítico** da distribuição de probabilidade correspondente ou calcula-se o **valor p**. Se o valor da estatística de teste estiver além do valor crítico ou se o valor

p for menor que o nível de significância escolhido, a hipótese nula é rejeitada. Caso contrário, a hipótese nula não é rejeitada.

Interpretação dos resultados: Por fim, interpreta-se os resultados do teste de hipótese à luz do contexto específico do problema em estudo, considerando as implicações práticas das descobertas.

8.4.1 Distribuição Subjacente da Amostra

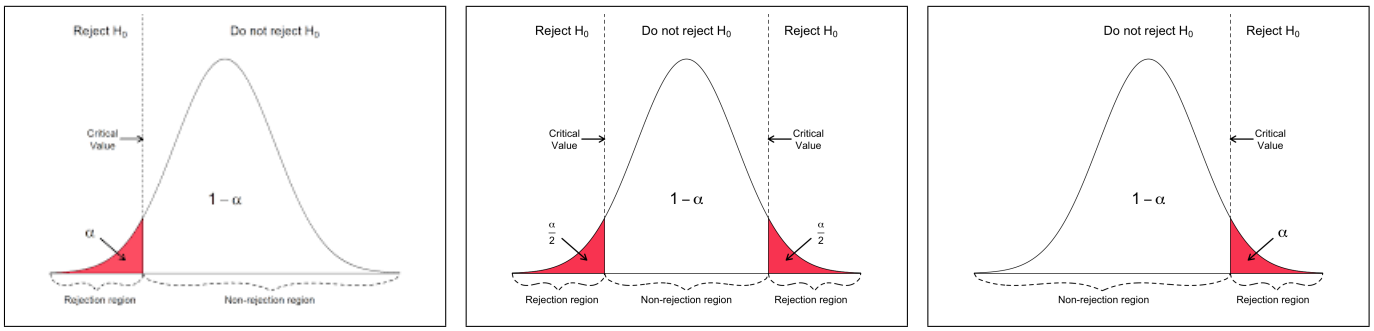
Muitos testes de hipóteses pressupõem que os dados sigam uma distribuição de probabilidade específica. No caso dos testes Z e T, é exigido que os dados sigam uma distribuição normal. Essas suposições são essenciais para garantir a confiabilidade e a interpretação correta dos resultados obtidos por meio desses testes. Uma estratégia útil para avaliar visualmente a normalidade dos dados, antes de aplicar os testes Z ou T, é através do uso do gráfico quantil-quantil (Figura 5.8). Esse tipo de gráfico compara os quantis dos dados da amostra com os quantis esperados de uma distribuição normal.

Se a distribuição das observações não for normal, é prudente considerar o uso de métodos não paramétricos ou técnicas de reamostragem, como a distribuição *bootstrap*. Os métodos não paramétricos são robustos em relação à distribuição dos dados e não requerem suposições sobre a forma da distribuição, como a técnica *bootstrap*. Essa técnica oferece uma alternativa para estimar intervalos de confiança, como já discutido na Seção 8.3.1, e realizar testes de hipóteses de forma robusta, mesmo quando as suposições sobre a distribuição dos dados não são atendidas.

A ideia básica por trás do *bootstrap* de permutação, ou *permutation test*, é que, se as hipóteses nula e alternativa são verdadeiras, então a atribuição das observações aos diferentes grupos ou tratamentos não deve afetar as diferenças observadas entre eles. Portanto, sob a hipótese nula, podemos embaralhar aleatoriamente ou permutar as observações entre os grupos e calcular a estatística de teste várias vezes para gerar uma distribuição de valores da estatística de teste que esperaríamos observar se a hipótese nula fosse verdadeira [34].

8.4.2 Cálculo de Valores Críticos

Um procedimento clássico de teste de hipótese envolve, essencialmente, o cálculo de uma estatística de teste, podendo ser uma estatítica resumo como média, variância, desvio padrão ou proporção, e a comparação dessa estatística com uma distribuição amostral conhecida sob a hipótese nula. Tipicamente, a decisão de **rejeitar a hipótese nula** H_0 é baseada no **nível de significância**, ou a probabilidade, α que estamos dispostos a tolerar. O ponto x_c em uma distribuição de probabilidade que separa as regiões de aceitação ($P(\mid \overline{X} \mid < x_c) = 1 - \alpha$) e rejeição da hipótese nula ($P(\overline{X} \geq x_c)$ ou $P(\overline{X} \geq x_c)$) é conhecido por **valor crítico**. O valor crítico está intimamente relacionado com as distribuições padronizadas (Seção 8.2.3).



(a) unilateral esquerda (b) bilateral (c) bilateral

Figura 8.12: 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: `https://www.geo.fu-berlin.de/en/v/soga-py/Basics-of-statistics/Hypothesis-Tests/Introduction-to-Hypothesis-Testing/Critical-Value-and-the-p-Value-Approach/index.html`)

Na Figura 8.12, podemos observar as duas direções possíveis da hipótese alternativa em relação à hipótese nula. Os **testes de hipóteses** são classificados como **bilaterais**, também conhecidos como bidirecionais, quando a hipótese alternativa considera a possibilidade de o parâmetro de interesse ser diferente do valor especificado na hipótese nula, tanto para valores maiores quanto para valores menores. Esses testes são empregados quando desejamos determinar se há uma diferença significativa em qualquer direção. Neste caso, a região de rejeição é dividida em ambas as extremidades da distribuição, com uma densidade de probabilidade igual a $\frac{\alpha}{2}$, como ilustra a Figura 8.12b. Por outro lado, quando a hipótese alternativa contempla apenas uma das duas direções, temos os **testes de hipóteses unilaterais**, que se dividem em duas categorias: direita e esquerda. Nos **testes unilaterais direita** (Figura 8.12c), a hipótese alternativa sugere que o parâmetro de interesse é maior do que o valor especificado na hipótese nula. Esses testes são utilizados quando buscamos verificar se há um aumento significativo no parâmetro. Em contrapartida, nos **testes unilaterais esquerda** (Figura 8.12a), a hipótese alternativa indica que o parâmetro de interesse é menor do que o valor especificado na hipótese nula. São empregados quando pretendemos determinar se há uma diminuição significativa no parâmetro.

As distribuições padronizadas são ferramentas fundamentais para determinar os valores críticos com base no nível de significância α desejado e nos parâmetros específicos do modelo estatístico, como os graus de liberdade ν (no caso da distribuição t-Student) ou a variância σ^2 conhecida (no caso da distribuição normal padrão), como ilustra a Figura 8.12. Por exemplo, para uma hipótese H_0 que segue um modelo normal com média μ e desvio padrão σ, o valor crítico x_c para um nível de significância α é calculado por

$$z_c = \frac{x_c - \mu}{\frac{\sigma}{\sqrt{n}}} \implies x_c = z_c \frac{\sigma}{\sqrt{n}} + \mu, \tag{8.15}$$

tal que $\alpha = P(Z < z_c) = 1 - P(\mid Z \mid \leq z_c)$. Através da tabela de distribuição normal padrão, podemos

obter $P(\mid Z \mid \leq z_c)$ para $1-\alpha$. Além disso, funções prontas para uso, como stats.ppf() do pacote scipy em Python ou qnorm() em R, que automaticamente calculam z_c com base no nível de significância. Para uma hipótese que segue o modelo t-Student, o valor crítico é dado por

$$t_c = \frac{x_c - \mu}{\frac{\sigma}{\sqrt{n}}} \Longrightarrow x_c = t_c \frac{\sigma}{\sqrt{n}} + \mu, \tag{8.16}$$

tal que $\alpha = P(T < t_c) = 1 - P(\mid T \mid \leq t_c)$. Similarmente, através das tabelas de distribuição t-Student padronizadas, podemos obter $P(\mid T \mid \leq t_c)$ para $1-\alpha$. Novamente, existem funções prontas para uso, como stats.t.ppf() do pacote scipy em Python ou qt() em R, que automaticamente calculam t_c com base no nível de significância.

A tomada de decisão em um teste de hipóteses é realizada comparando a estatística de teste com o valor crítico x_c. Para ilustrar este processo, suponhamos que tenhamos formulado a hipótese de que a proporção de caras em 50 lançamentos de uma moeda não enviesada é igual a 0.65 (estatística de teste), e que a distribuição das proporções de caras em repetidos lançamentos se aproxima de uma distribuição normal. Para avaliar a evidência estatística dessa hipótese em um nível de significância de 0.25, podemos utilizar os valores críticos calculados na Seção 8.3.2, $z_{0.025} = 0.38011$ e $z_{0.975} = 0.65989$, para definir as regiões de aceitação e rejeição.

Se a estatística de teste estiver dentro da região de aceitação, concluímos que não há evidência suficiente para rejeitar a hipótese nula H_0 em favor da hipótese alternativa. Por outro lado, se estiver dentro da região de rejeição, dizemos que há evidência estatística para rejeitamos a hipótese nula em favor da hipótese alternativa. Assim, se a estatística de teste estiver na região de aceitação, como neste exemplo, afirmamos que há evidência estatística para aceitar a hipótese nula H_0. Isso significa que, com base nos dados disponíveis e no nível de significância escolhido, não temos evidências estatísticas suficientes para rejeitar a hipótese nula e, portanto, não podemos concluir que a proporção de caras nos lançamentos é diferente de 0.65.

8.4.3 Erros em Testes de Hipóteses

Quando a estatística de teste se encontra na região da hipótese alternativa H_a, isso não implica automaticamente que podemos afirmar que H_0 seja falsa. Apenas indica que não temos evidências estatísticas suficientes para aceitar a hipótese H_0 como verdadeira. A ausência de evidências a favor de uma hipótese nula não significa que há evidências contra ela. É possível que a hipótese nula seja verdadeira mesmo quando a estatística de teste não fornece evidências para rejeitá-la. Nesse caso, cometemos um **erro do tipo I**, pois a hipótese nula é verdadeira e decidimos rejeitá-la, como ilustrado na região hachurada azul na Figura 8.13. Da mesma forma, é possível aceitar a hipótese nula mesmo que ela seja falsa, quando a estatística de teste se encontra na região de aceitação, conforme indicado pela região cinza na Figura 8.13. Nesse cenário, cometemos um **erro do tipo II**, pois

ocorre a aceitação de uma hipótese nula falsa. Note que os erros tipo I e tipo II são especificamente relacionados à decisão de aceitar ou rejeitar uma hipótese nula com base em evidências estatísticas.

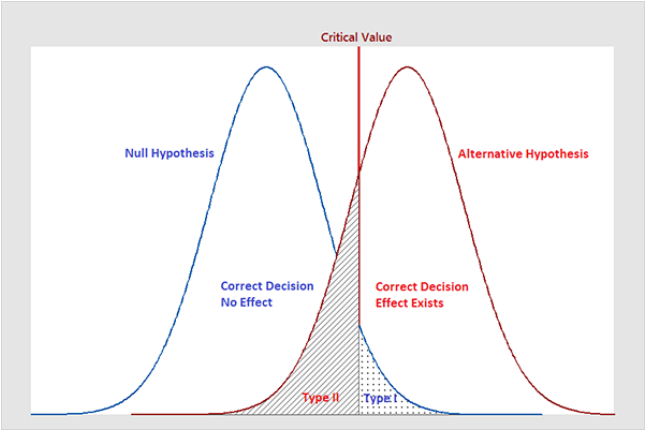


Figura 8.13: Tipos de erros numa tomada de decisão de aceitação (região esquerda do valor crítico) e de rejeição (região direita do valor crítico) de uma hipótese nula H_0: erro do tipo I (falso positivo) e erro do tipo II (falso negativo). (Fonte: `https://statisticsbyjim.com/hypothesis-testing/types-errors-hypothesis-testing/`)

Figura 8.11 mostra que o valor crítico x_c divide a densidade de probabilidade da distribuição amostral correspondente à hipótese H_0 em duas regiões, $1-\alpha$ e α, e a distribuição amostral correspondente a H_a em $1-\beta$ e β. Essas quatro probabilidades são comumente referidas da seguinte forma:

- $1-\alpha$**:** Este é o **nível de confiança** (*confidence level* em inglês) do teste. Representa a probabilidade de não cometer um erro do tipo I, ou seja, a probabilidade de corretamente não rejeitar a hipótese nula quando ela é verdadeira.
- α**:** É a probabilidade de cometer um erro do tipo I, também conhecido como **nível de significância** (*significance level* em inglês). Corresponde à probabilidade de rejeitar erroneamente a hipótese nula quando ela é verdadeira.
- $1-\beta$**:** É o **poder do teste** (*power of test* em inglês), ou seja, a probabilidade de corretamente rejeitar a hipótese nula quando ela é falsa. Representa a capacidade do teste de detectar uma diferença ou efeito verdadeiro, se existir.
- β**:** É a probabilidade de cometer um erro do tipo II, ou seja, a probabilidade de não rejeitar a hipótese nula quando ela é falsa. Conhecida por taxa de erro, ela indica a probabilidade de falhar em detectar uma diferença ou efeito verdadeiro, se existir.

No exemplo dado na Seção 8.4.2, 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 inferir que ocorreu um erro tipo I. Aumentando o tamanho da amostra, reduziríamos a região de aceitação como mostra o exemplo na Seção 8.3.2, proporcionando evidências estatísticas suficientes para rejeitar H_0.

8.4.4 Cálculo do P-Valor

No procedimento clássico de teste de hipóteses, geralmente começamos com um nível de significância α, que representa a probabilidade máxima que estamos dispostos a aceitar de cometer um erro tipo I (rejeitar a hipótese nula quando ela é verdadeira). Com base em α, construímos regiões de decisão para determinar se rejeitamos ou não a hipótese nula.

Uma abordagem alternativa é não fixar um valor específico para α, mas permitir que quem estiver utilizando os resultados do teste faça essa escolha. Nesse caso, assumindo que a hipótese nula de que a estatística de teste $\mu = \mu_0$ seja verdadeira, calculamos o p-valor, representado por α^*, que é a probabilidade, de obter uma estatística de teste $\overline{x}_{obs}$ tão extrema quanto aquela observada na amostra, ou ainda mais extrema na direção esquerda, quando a hipótese alternativa é $H_a < \mu_0$. Ou seja,

$$\alpha^* = P(\overline{X} < \overline{x}_{obs} \mid H_0 \text{ verdadeira}). \tag{8.17}$$

Quando a hipótese alternativa é $H_a > \mu_0$, o p-valor é a probabilidade, de obter uma estatística de teste $\overline{x}_{obs}$ tão extrema quanto aquela observada na amostra, ou ainda mais extrema na direção direita. Ou seja,

$$\alpha^* = P(\overline{X} > \overline{x}_{obs} \mid H_0 \text{ verdadeira}). \tag{8.18}$$

E quando s hipótese alternativa é $H_a \neq \mu_0$, o p-valor é a probabilidade, de obter uma estatística de teste $\overline{x}_{obs}$ tão extrema quanto aquela observada na amostra, ou ainda mais extrema em ambas direções. Ou seja,

$$\alpha^* = 2 \times P(\overline{X} > \overline{x}_{obs} \mid H_0 \text{ verdadeira}). \tag{8.19}$$

Para obter as probabilidades associadas às estatísticas de teste $\overline{x}_{obs}$, as estatísticas de teste são padronizadas subtraindo a média e dividindo pelo desvio padrão. Isso resulta em valores padronizados, denotados como z_{obs} para uma distribuição normal padrão (Z) (Eq. 8.9) e como t_{obs} para uma distribuição t-Student (Eq. 8.10). Em seguida, as probabilidades associadas aos valores padronizados podem ser obtidas de duas maneiras:

Leitura das tabelas estatísticas: Tabelas de distribuição normal padrão ou t-Student fornecem as probabilidades acumuladas para diferentes valores de z ou t. Consultando essas tabelas, é possível encontrar diretamente as probabilidades associadas aos valores padronizados.

Uso de funções prontas para uso em R e Python: Em R, pode-se utilizar a função `qnorm()` para distribuição normal padrão e `t.test()` para distribuição t-Student. Em Python, usando o pacote `scipy`, a função `stats.norm.cdf()` é usada para distribuição normal padrão, enquanto `stats.ttest_1samp()` é utilizada para distribuição t-Student.

O **p-valor** α^* obtido é então comparado com o nível de significância α escolhido para a tomada de decisão: determinar se rejeitamos ou não a hipótese nula. Se a probabilidade for menor que α, consideramos a evidência contra a hipótese nula como estatisticamente significativa e rejeitamos a hipótese nula. Se a probabilidade for maior que α, não temos evidências estatísticas suficientes para rejeitar a hipótese nula. O p-valor é, portanto, uma medida que nos permite avaliar a força das evidências contra a hipótese nula em um teste de hipóteses. Weiss [94] sugere as seguintes diretrizes para usar o p-valor na avaliação das evidências contra a hipótese nula H_0.

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 8.1: Evidências contra hipótese nula com base em valor-p.

8.4.5 Interpretação de Resultados dos Testes de Hipóteses

Interpretar os resultados dos testes de hipóteses vai além de apenas analisar os valores dos testes e os p-valores em relação às hipóteses nula e alternativa. É fundamental contextualizar os resultados, considerando o problema em questão e suas implicações práticas. Uma rejeição da hipótese nula não implica automaticamente na validade da hipótese alternativa; indica apenas que existem evidências suficientes para sugerir que a hipótese nula é improvável. É crucial estar ciente dos dois tipos de erros possíveis em um teste de hipóteses: erro tipo I (rejeitar a hipótese nula quando ela é verdadeira) e erro tipo II (falhar em rejeitar a hipótese nula quando ela é falsa). Ao interpretar os resultados, é importante ponderar sobre os possíveis impactos desses erros e avaliar a confiabilidade das conclusões. Ao relatar os resultados de um teste de hipóteses, é essencial fornecer uma interpretação estatística clara e detalhada, explicando as conclusões à luz do contexto do problema e destacando qualquer incerteza associada aos resultados.

8.5 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 uma única amostra, que é um subconjunto consideravelmente menor da população.

Na Seção 8.1, 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 subja-

centes da amostra, as abordagens clássicas, como testes Z e T, pressupõem uma distribuição normal subjacente da amostra.

Na Seção 8.2.1, abordamos noções sobre distribuições amostrais normais e sua natureza aleatória, incluindo suas versões padronizadas. É uma prática comum e válida avaliar a normalidade de uma amostra 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 alternativa eficaz, o gráfico quantil-quantil, para avaliar visualmente a normalidade dos dados.

Na Seção 8.3, 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.

Na Seção 8.4, analisamos os testes de hipóteses, utilizados para fazer afirmações sobre os parâmetros populacionais com base em evidências amostrais. Isso envolve a formulação de uma hipótese nula e uma hipótese alternativa, e apresentamos duas técnicas de rejeição/aceitação: por meio de valores críticos e por meio do p-valor. Embora os valores críticos sejam clássicos e mais intuitivos, o p-valor é frequentemente preferido devido à sua interpretação mais intuitiva, maior flexibilidade e facilidade de comparação entre estudos.

Além disso, introduzimos a técnica *bootstrap*, que se tornou uma alternativa robusta e não tendenciosa para fazer inferências estatísticas, como estimativas pontuais, intervalos de confiança (*bootstrap* de percentil) e testes de hipóteses (*bootstrap* de permutação). 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 [34].

8.6 Exercícios

1. Faça os exercícios de aprendizado LC7.1 – LC7.24 relacionados à **amostragem**, conforme proposto em [34]. 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 [34]. Verifique suas respostas no apêndice D do mesmo livro.

3. Faça os exercícios de aprendizado LC9.1 – LC9.15 relacionados aos **testes de hipótese**, conforme proposto em [34]. Verifique suas respostas no apêndice D do mesmo livro.

4. Explique com suas próprias palavras os textos nas Seções 9.6.2 e 9.6.3 em [34].

5. Identifique as pergunta, as estatísticas de teste e as amostras nos exemplos apresentados nos Capítulos 8 e 9 em [34].

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 em [11]. Na Seção 8.6 em [34], 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.

   (a) Qual é a distribuição subjacente das amostras?

   (b) Para visualizar a variabilidade do estimador $\hat{P}$ para 1000 amostras de tamanho 50, plote gráficos semelhantes aos da Figura 8.8.

7. Em 1974 foi publicado no *Journal of Applied Psychology* um estudo, usando os dados promotions disponíveis em [11], para responder a pergunta: "Será que seu gênero afetará suas chances de ser promovido?". Na Seção 9.1 em [34], os autores formularam a pergunta como duas hipóteses H_0 e H_1 e aplicaram a técnica de *bootstrap* para respondê-la. Plote o gráfico de distribuição das duas hipóteses, hachurando a área correspondente ao p-valor.

Bibliografia

[1] Hassan Kibirige. A Grammar of Graphics for Python. https://plotnine.org/.

[2] 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.

[3] 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.

[4] Andrew Nisbet. Tufte in Matplotlib. https://www.ajnisbet.com/blog/tufte-in-matplotlib, 2020.

[5] ANEEL. Dados Abertos - Agência Nacional de Energia Elétrica. https://dadosabertos.aneel.gov.br/.

[6] 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.

[7] Jacques Bertin. *Graphics and Graphics Information-Processing*. 1981. Original French edition published by Flammarion, Paris, 1977, translated by William J. Berg and Paul Scott.

[8] Matheus Budkewicz. Como instalar o Jupyter Notebook? (Windows e Linux). https://medium.com/horadecodar/como-instalar-o-jupyter-notebook-windows-e-linux-20701fc583c.

[9] 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.

[10] G. Casella and R.L. Berger. *Inferência Estatística*. CENGAGE DO BRASIL, 2010.

[11] Chester Ismay. Repositório de dados de Moderndive. https://github.com/moderndive/moderndive/blob/master/data-raw/process_data_sets.R. Acessado em Maio de 2024.

[12] Chris Chobris and Daniel Simons. The Invisible Gorilla. https://www.youtube.com/watch?v=D_m_9N_3u7o, 2012.

[13] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, June 1970.

[14] Wikipedia contributor. Ascombe's quartet. https://en.wikipedia.org/wiki/Anscombe%27s_quartet.

[15] Wikipedia contributor. Iris flower data set. https://en.wikipedia.org/wiki/Iris_flower_data_set.

[16] Wikipedia contributor. Psicologia cognitiva. https://pt.wikipedia.org/wiki/Psicologia_cognitiva.

[17] 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.

[18] The NumPy Steering Council. Numpy. https://numpy.org/.

[19] 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.

[20] Aaron Culich. Jupyter Logo - Design Brief. https://github.com/jupyter/design/wiki/Jupyter-Logo.

[21] Daniela Petruzalek. DATASUS: Conheça a Nova Ferramenta para Ler Arquivos DBC. https://pt.linkedin.com/pulse/datasus-conhe%C3%A7a-nova-ferramenta-para-ler-arquivos-dbc-petruzalek.

[22] DataCamp team. prcomp: Principal Components Analysis. https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/prcomp.

[23] Datacamp team. ggplot2 Cheat Sheet. https://www.datacamp.com/cheat-sheet/ggplot2-cheat-sheet, 2022. Acessado em Março de 2024.

[24] Dhanashree. Web Scraping with Python Tutorial. https://nanonets.com/blog/web-scraping-with-python-tutorial/. Acessado em Abril de 2024.

[25] J. E. Gentle, L. Kaufman, and P. J. Rousseuw. Finding groups in data: An introduction to cluster analysis. *Biometrics*, 47(2):788, June 1991.

[26] Garrett Grolemund. Introduction to R Markdown. https://rmarkdown.rstudio.com/articles_intro.html, 2014.

[27] 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.

[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. Compute empirical cumulative distribution. `https://ggplot2.tidyverse.org/reference/stat_ecdf.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. Plot basics. `https://ggplot2.tidyverse.org/reference/`. Acessado em Março de 2024.

[30] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, 1st edition, 2001.

[31] 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.

[32] Hassan Kibirige. Plotnine 0.13.4 API Reference. `https://plotnine.org/reference/`. Acessado em Abril de 2024.

[33] 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.

[34] 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.

[35] J. Hathaway and Katie Larson. Python for Data Science. `https://byuidatascience.github.io/python4ds/tidy-data.html`.

[36] Nielsen Jakob. Response times: The 3 important limits. `https://www.nngroup.com/articles/response-times-3-important-limits/`, Janeiro 1993.

[37] Jeroen Janssens. Heuristics for Translating Ggplot2 Code. `https://jeroenjanssens.com/heuristics/`, 2019. Acessado em Março de 2024.

[38] Jose Stotopoli and Leonardo Vils. Relação entre Variáveis - Correlações. `https://storopoli.io/Estatistica/5-Correlacoes.html`, 2021.

[39] João Luís Ferreira Batista. Curso Relâmpago de R. `http://cmq.esalq.usp.br/wiki/doku.php?id=publico:tutoriais:r-relampago:star`.

[40] Jupyter Development Team. nbconvert: Convert Notebooks to other formats. `https://nbconvert.readthedocs.io/en/latest/`.

[41] Robert I. Kabacoff. Quick-R by Datacamp. `https://www.statmethods.net/`.

[42] Kam Tin Seong. Data Profiling, Exploration and Analysis-funModeling Methods. `https://rpubs.com/tskam/R4DSA04-funModeling_methods`. Acessado em Abril de 2024.

[43] Daniel Keim, Jörn Kohlhammer, Geoffrey Ellis, and Florian Mansmann. *Mastering the Information Age Solving Problems with Visual Analytics*. Eurographics Association, 2010.

[44] Kevin Babitz. Introduction to k-Means Clustering with scikit-learn in Python. `https://www.datacamp.com/tutorial/k-means-clustering-python`, 2023.

[45] 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.

[46] LEONARD P. AYRES. THE WAR WITH GERMANY: A STATISTICAL SUMMARY. `http://www.gwpda.org/docs/statistics/statstc.htm`, 1919.

[47] Lukasz Piwek. Tufte in R. `http://motioninsocial.com/tufte/`, 2017.

[48] Marcos Nascimento Magalhães and Antonio Carlos Pedroso de Lima. *Noções de Probabilidade e Estatística*. edUSP, 7^a ed. edition, 2010.

[49] 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.

[50] Michael Hahsler. dbscan: Density-based Spatial Clustering of Applications with Noise (DBSCAN) . `https://www.rdocumentation.org/packages/dbscan/versions/1.1-12/topics/dbscan`.

[51] Miguel Garcia. Using ggplot in Python: Visualizing Data With plotnine. `https://realpython.com/ggplot-python/`.

[52] Mirko Stojiljković. NumPy, SciPy, and pandas: Correlation With Python. `https://realpython.com/numpy-scipy-pandas-correlation-python/`.

[53] Kirill Müller and Hadley Wickham. Tibbles. https://tibble.tidyverse.org/articles/tibble.html.

[54] University of Toronto Libraries team. Data Visualization – Tools & Tutorials. https://mdl.library.utoronto.ca/dataviz/tools-tutorials.

[55] Oz. Psicologia: Resolução de Problemas. https://mundodeoz.wordpress.com/2009/12/27/resolucao-de-problemas/.

[56] Pandas team. DataFrame. https://pandas.pydata.org/docs/reference/frame.html.

[57] Pandas team. Series. https://pandas.pydata.org/docs/reference/series.html.

[58] Pipz Academy team. Guia básico de Markdown. https://docs.pipz.com/central-de-ajuda/learning-center/guia-basico-de-markdown.

[59] Plotly Team. Dendrograms in ggplot2. https://plotly.com/ggplot2/dendrogram/. Acessado em Março de 2024.

[60] Plotnine Team. plotnine.stat_ecdf. https://plotnine.org/reference/stat_ecdf. Acessado em Março de 2024.

[61] Plotnine Team. plotnine.stat_qq. https://plotnine.org/reference/stat_qq. Acessado em Março de 2024.

[62] Benjamin Pryk. How to Use Jupyter Notebook: A Beginner's Tutorial. https://www.dataquest.io/blog/jupyter-notebook-tutorial/, 2020.

[63] Python team. Python For Beginners. https://www.python.org/about/gettingstarted/.

[64] Rafael A. Irizarry. Introdução à Ciência de Dados - Análise de Dados e Algoritmos de Previsão com R. http://rafalab.dfci.harvard.edu/dslivro/, 2020.

[65] Rafael A. Irizarry. Introduction to Data Science: Data Analysis and Prediction Algorithms with R. https://rafalab.dfci.harvard.edu/dsbook/, 2023.

[66] Ramzi W. Nahhas. An Introduction to R for Research. https://bookdown.org/rwnahhas/IntroToR/three-or-more-continuous-variables-scatterplot-matrix.html. Acessado em Março de 2024.

[67] RDocumentation team. clara: Clustering Large Applications. https://www.rdocumentation.org/packages/cluster/versions/1.3-5/topics/clara.

[68] RDocumentation team. hclust: Hierarchical Clustering. https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/hclust.

[69] RDocumentation team. kmeans: K-Means Clustering. https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/kmeans.

[70] 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.

[71] 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.

[72] Verônica Santana. Tutorial R/RStudio. https://edisciplinas.usp.br/pluginfile.php/4883125/mod_resource/content/1/Tutorial.pdf.

[73] Ben Schneiderman. The Eight Golden Rules of Interface Design. https://www.cs.umd.edu/users/ben/goldenrules.html.

[74] Scikit Learn team. sklearn.cluster.AgglomerativeClustering. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html.

[75] Scikit Learn team. sklearn.cluster.DBSCAN. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html.

[76] Scikit-learn team. sklearn.decomposition.PCA. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.

[77] Seaborn Team. seaborn.pairplot. https://seaborn.pydata.org/generated/seaborn.pairplot.html. Acessado em Março de 2024.

[78] Adam Shafi. How to Use Jupyter Notebooks: The Ultimate Guide. https://www.datacamp.com/tutorial/tutorial-jupyter-notebook, 2020.

[79] Shiny team. Easy web apps for data science without the compromises. https://shiny.posit.co/.

[80] 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.

[81] 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.

[82] STHDA team. Import and export data using R. http://www.sthda.com/english/wiki/import-and-export-data-using-r.

[83] STHDA team. R Built-in Data Sets. `http://www.sthda.com/english/wiki/r-built-in-data-sets`.

[84] STHDA team. t distribution table. `http://www.sthda.com/english/wiki/t-distribution-table`. Acessado em Abril de 2024.

[85] TechWeb team. Scientific Visualization Software Packages. `https://www.bu.edu/tech/support/research/training-consulting/online-tutorials/introduction-to-scientific-visualization-tutorial/software-packages/`.

[86] Alexandru C. Telea. *Data Visualization: Principles and Practice*. A K Peters Ltda, 2nd edition, 2008.

[87] The Matplotlib development team. Matplotlib. `https://matplotlib.org/`.

[88] TSE. Portal de Dados Abertos do TSE. `https://dadosabertos.tse.jus.br/dataset/`.

[89] E.R. Tufte. The visual display of quantitative information, 2001.

[90] John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[91] Vincent Arel-Bundock. Repositório de dados do aplicativo R. `https://vincentarelbundock.github.io/Rdatasets/datasets.html`. Acessado em Março de 2024.

[92] Volunteer Contributors. Pandas. `https://pandas.pydata.org/`.

[93] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Series in Interactive Technologies. Morgan Kaufmann, 3 edition, 2012.

[94] Neil A. Weiss. *Introductory Statistics*. Pearson, Addison-Wesley, 2015.

[95] McKinney Wes. Python for data analysis. `https://wesmckinney.com/book/`, 2012.

[96] H. Wickham and G. Grolemund. R for data science. `https://r4ds.had.co.nz/`, 2017.

[97] Hadley Wickham. A layered grammar of graphics. *Journal of Computational and Graphical Statistics*, 19(1):3–28, 2010.

[98] Hadley Wickham. Tidy data. *Journal of Statistical Software*, 59(10), 2014.

[99] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016.

[100] 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.

[101] 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.

[102] Wikipedia contributor. Bayes Estimator. `https://en.wikipedia.org/wiki/Bayes_estimator`. Acessado em Março de 2024.

[103] Wikipedia contributor. Maximum likelihood estimaation. `https://cran.r-project.org/doc/manuals/r-release/R-data.html`. Acessado em Março de 2024.

[104] L. Wilkinson, D. Wills, D. Rope, A. Norton, and R. Dubbs. The Grammar of Graphics. `https://books.google.com.br/books?id=_kRX4LoFfGQC`, 2005.

[105] Leland Wilkinson. The grammar of graphics. *WIREs Computational Statistics*, 2(6):673–677, October 2010.

[106] 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.

[107] Pak Chung Wong and J. Thomas. Visual analytics. *IEEE Computer Graphics and Applications*, 24(5):20–21, September 2004.

[108] Martin Woodward and Tali Herzka. Markdown - Math Support. `https://github.blog/2022-05-19-math-support-in-markdown/`.

[109] Yancy Dennis. Matplotlib, Seaborn, Plotly and Plotnine Comparison. `https://python.plainenglish.io/matplotlib-seaborn-plotly-and-plotnine-comparison-baf2db5a9c40`, 2023.

[110] Ydata profiling Team. Ydata Profiling 4.7. `https://docs.profiling.ydata.ai/latest/`. Acessado em Abril de 2024.

[111] 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.

[112] Zoumana Keita. Principal Component Analysis in R Tutorial. `https://www.datacamp.com/tutorial/pca-analysis-r`, 2023. Acessado em Março de 2024.