CITHA | Inovação e Sustentabilidade na Amazônia

2025

CIÊNCIA DE DADOS

(Estudos Práticos de Análise e Exploração de Dados com Python)

DESIGNED BY FREEPIK

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Ciência de dados [livro eletrônico] : (estudos práticos de análise e exploração de dados com Python) / Raimundo Fagner Costa ... [et al.]. -- prefácio por Nivaldo Rodrigues e Silva ; revisão Alyson de Jesus dos Santos 2. ed. -- Manaus, AM : Ed. dos Autores, 2025.
PDF

Outros autores: Eduardo Palhares Júnior, Wenndisson da Silva Souza, Alexandre Lopes Martiniano, Nivaldo Rodrigues e Silva
Bibliografia
ISBN 978-65-01-80960-1

1. Ciência de dados 2. Dados - Análise 3. Python (Linguagem de programação para computadores) I. Costa, Raimundo Fagner. II. Palhares Júnior, Eduardo. III. Souza, Wenndisson da Silva. IV. Martiniano, Alexandre Lopes. V. Silva, Nivaldo Rodrigues e. VI. Santos, Alyson de Jesus dos

25-317825.0 CDD-005.73

Índices para catálogo sistemático:

1. Ciência de dados 005.73

Maria Alice Ferreira - Bibliotecária - CRB-8/7964

DOI: 10.5281/zenodo.17675244

Expediente do IFAM

Reitor
Jaime Cavalcante Alves

Pró-Reitor de Administração
Fábio Teixeira Lima

Pró-Reitor de Gestão de Pessoas
Leandro Amorim Damasceno

Pró-Reitora de Ensino
Rosângela Santos da Silva

Pró-Reitora de Extensão
Maria Francisca Morais de Lima

Pró-Reitor de Pesquisa, Pós-Graduação e Inovação
Paulo Henrique Rocha Aride

Diretor Geral do Campus Manaus Distrito Industrial
Nivaldo Rodrigues e Silva

Expediente do Projeto CITHA

Gestores
Nivaldo Rodrigues e Silva
Samirames da Silva Fleury
Alyson de Jesus dos Santos
Maria Cassiana Andrade Braga
Adanilton Rabelo de Andrade

Coordenadores
Tiago Francisco Andrade Diocesano
Jaidson Brandão da Costa
Elcivan dos Santos Silva
Martinho Correia Barros
Adelino Maia Galvão Filho

Expediente de Produção

Autores
Raimundo Fagner Costa
Wenndisson da Silva Souza
Eduardo Palhares Jr.
Alexandre Lopes Martiniano
Nivaldo Rodrigues e Silva

Avaliação Pedagógica
Samirames da Silva Fleury

Diagramadores
Wenndisson da Silva Souza
Eduardo Palhares Jr.
Fabio Serra Ribeiro Couto

Revisores de Texto
Alyson de Jesus dos Santos

Ciência de Dados
(Estudos Práticos de Análise e Exploração de Dados com Python)

Autores
Raimundo Fagner Costa
Wenndisson da Silva Souza
Eduardo Palhares Jr.
Alexandre Lopes Martiniano
Nivaldo Rodrigues e Silva

Prefácio por Nivaldo Rodrigues e Silva

Revisão
Alyson de Jesus dos Santos

2ª Edição

Manaus - AM
2025

Lista de Expressões para Enriquecimento de Conteúdo

Este material foi cuidadosamente estruturado para apoiar sua jornada de aprendizado. Ao longo dos capítulos, você encontrará diversas chamadas sinalizadas por ícones especiais, que ajudarão a destacar pontos-chave e enriquecer sua compreensão. Durante a diagramação, esses ícones serão inseridos conforme as indicações dos autores, guiando você para diferentes tipos de conteúdo e atividades que potencializam seu estudo.

Fique Alerta!

Destaque para conceitos, expressões e trechos fundamentais que merecem sua atenção especial para a compreensão do conteúdo.

Iniciando o diálogo...

Espaço para reflexão crítica. Aqui você será convidado(a) a problematizar os temas abordados, relacionando-os com sua experiência e buscando conexões relevantes para aprofundar seu aprendizado.

Conhecendo um pouco mais!

Indicação de fontes complementares, como livros, entrevistas, vídeos, aplicativos, links e outros recursos para ampliar seu conhecimento sobre o tema.

Caso Prático

Aplicação direta do conteúdo em exemplos concretos, para facilitar a fixação e demonstrar a utilidade do que foi aprendido.

Copie e Teste!

Trechos de código prontos para serem copiados e executados, para que você possa experimentar, validar e explorar na prática os conceitos estudados.

Tela do Terminal

```
Apresenta o resultado esperado após a execução de um
bloco de código. Serve como um gabarito para que você
possa verificar se sua implementação está funcionando
corretamente.
```

Prefácio

A era digital transformou a maneira como lidamos com a informação, tornando a Ciência de Dados uma ferramenta essencial para a análise e interpretação de grandes volumes de dados. Este e-book oferece um percurso estruturado para aqueles que desejam compreender os fundamentos dessa área e aplicá-los de forma prática, abordando desde conceitos básicos até técnicas avançadas de análise. Com um enfoque claro e objetivo, este material serve tanto para iniciantes quanto para profissionais que buscam aprimorar suas habilidades.

A estrutura do conteúdo foi cuidadosamente planejada para proporcionar um aprendizado progressivo. O primeiro capítulo apresenta os fundamentos da Ciência de Dados, incluindo conceitos estatísticos, programação em Python e manipulação de dados. No segundo capítulo, são exploradas ferramentas e técnicas de análise, capacitando o leitor a transformar dados brutos em informações valiosas. O terceiro capítulo foca em fontes de dados climáticos, demonstrando como a Ciência de Dados pode ser aplicada na sustentabilidade e na gestão de recursos naturais.

No capítulo final, a teoria ganha vida por meio de um experimento prático, permitindo aos leitores aplicar os conhecimentos adquiridos na análise de dados para a agricultura sustentável. A proposta é fomentar a integração entre teoria e prática, preparando profissionais para lidar com desafios reais e tomar decisões fundamentadas em dados. Essa abordagem interativa fortalece o aprendizado e promove uma visão ampla do impacto da Ciência de Dados em diferentes setores.

Este e-book é mais do que um guia técnico; é um convite à exploração do potencial transformador dos dados. Ao compreender e aplicar os conceitos aqui apresentados, o leitor estará apto a enfrentar os desafios da era digital com competência e inovação. Que esta jornada seja repleta de descobertas e insights valiosos para a construção de um futuro mais inteligente e sustentável.

Projeto de Capacitação e Interiorização em Tecnologias Habilitadoras na Amazônia - CITHA

O projeto CITHA surge com o objetivo de fortalecer a economia da Amazônia por meio do incentivo ao empreendedorismo local e do desenvolvimento sustentável. Sua proposta é capacitar profissionais e impulsionar a criação de startups voltadas para a bioeconomia, além de apoiar cooperativas locais na melhoria de seus processos produtivos. A implementação de tecnologias inovadoras é uma das estratégias centrais do projeto, visando oferecer soluções eficientes que atendam às necessidades regionais, como a otimização dos recursos naturais e a melhoria da infraestrutura local.

Ao longo de sua execução, o projeto se compromete a integrar os diversos stakeholders, como governos, empresas, ONGs e comunidades, por meio da capacitação da mão de obra local. O objetivo é formar um capital intelectual qualificado, capaz de apoiar uma governança eficiente, promover a inovação e assegurar a sustentabilidade. O CITHA dedica-se à criação de processos internos que incentivem o desenvolvimento de novos métodos e tecnologias, adaptáveis às particularidades do território amazônico.

Em síntese, o projeto CITHA visa criar um ciclo de desenvolvimento que não só incentive o empreendedorismo, mas também promova a modernização das estruturas locais, elevando a qualidade de vida das populações da Amazônia. Focado em áreas como bioeconomia, inovação e transferência de tecnologia, o projeto busca estabelecer um ecossistema mais forte e autossustentável, capaz de responder eficientemente às demandas do mercado e da sociedade.

Sumário

Capítulo 1

A Jornada dos Dados: Do Problema à Decisão

Iniciando o diálogo...

Você já imaginou como dados sobre o nível dos rios podem prever safras, ou como a análise de sons da floresta pode ajudar a proteger espécies ameaçadas? A Ciência de Dados não é uma ferramenta do futuro distante; ela é a linguagem que usamos hoje para fazer perguntas ao mundo e obter respostas que geram impacto. Neste capítulo, vamos descobrir como transformar simples números em decisões poderosas.

1.1 O que é Ciência de Dados?

A Ciência de Dados (*Data Science*) é uma área fundamentalmente interdisciplinar, e é crucial não confundi-la com o termo multidisciplinar. Um ambiente multidisciplinar sugere uma *coexistência*, como uma caixa de ferramentas onde um estatístico, um programador e um gestor trabalham lado a lado, mas cada um na sua "caixinha" separada. Em contraste, o espírito interdisciplinar é uma fusão: o verdadeiro valor nasce justamente da intersecção entre as áreas. O Cientista de Dados não é três especialistas em um, mas um profissional único que opera na fronteira entre esses campos. Essa fusão sinérgica é a combinação de três pilares essenciais.

Primeiro, o Conhecimento de Negócio (o Domínio) permite fazer as perguntas certas. Segundo, a Matemática e Estatística fornecem a base para formular as hipóteses corretas e validar se um padrão encontrado é real ou apenas uma coincidência. Por fim, a Ciência da Computação oferece o poder de testar essas hipóteses em escala massiva, processando milhões de registros de forma que seria impossível manualmente. Ela não é apenas programação, nem apenas estatística, nem apenas conhecimento de negócios. Ela é a fusão sinérgica das três, como visto na Figura **??**. Pense desta forma:

- Ter Conhecimento de Negócio permite fazer as perguntas certas.
- Com a Matemática e Estatística envolvidas podemos formular as hipóteses corretas e entender se um padrão é real ou apenas coincidência.
- A Ciência da Computação permite testar essas hipóteses em escala massiva (milhões ou bilhões de registros), algo impossível de se fazer manualmente.

O verdadeiro valor da Ciência de Dados não está em saber programar. Está em saber programar para aplicar um modelo estatístico que responda a uma pergunta de negócio crucial. É essa fusão que transforma dados brutos em decisões estratégicas.

Figura 1.1: A natureza interdisciplinar da Ciência de Dados.

Pense em um Cientista de Dados como um "detetive moderno". O "crime" é um problema de negócio (ex: "Nossas vendas caíram 20%") e os "dados" são as pistas (planilhas, logs de acesso, feedback de clientes). O cientista usa ferramentas computacionais para coletar e organizar as pistas, aplica a estatística para encontrar padrões que ninguém viu e, por fim, usa o conhecimento do negócio para contar a história do que realmente aconteceu e quem é o "culpado". Ou seja, a Ciência de Dados é o processo científico de extrair conhecimento útil e acionável a partir de dados, com o objetivo de auxiliar na tomada de decisões estratégicas.

1.1.1 Transformando Dados em Sabedoria

Para compreender o poder desse processo, precisamos de uma analogia. Muitas organizações modernas estão, como diz o ditado, "se afogando em dados, mas morrendo de sede por *insights*". Elas coletam terabytes de informações de planilhas, sensores e logs, mas falham em usá-los para tomar decisões estratégicas. Isso acontece porque há uma enorme diferença entre Dados brutos (o ruído) e Sabedoria (a decisão acionável). A Ciência de Dados é o motor que move um nível ao outro.

Para entender essa jornada da confusão à clareza e o poder desse processo, vamos diferenciar os níveis de compreensão. Usaremos um exemplo prático e regional: o de uma cooperativa extrativista na Amazônia que produz castanha-do-Pará.

- **Dado (Bruto):** Um número solto. Ex: `2200`. Sozinho, esse número não significa nada.
- **Informação (Contexto):** O dado com contexto. Ex: A planilha de 2016 registra `Chuva = 2200mm` e `Produção = 58 Ton`. Agora temos fatos, mas ainda não sabemos o "porquê".

- **Conhecimento (Padrão):** A análise de 10 anos de planilhas revela um padrão: "Anos em que a chuva ultrapassa 2100mm, como 2016, tendem a ter uma safra de castanha ≈15% maior no ano seguinte." Isso é um *insight*.
- **Sabedoria (Decisão):** Com base nesse conhecimento e na previsão do INMET de um "inverno chuvoso" (dado novo), a cooperativa toma uma decisão: "Vamos antecipar a negociação com transportadoras e contratar mais 10 coletores para a próxima safra, pois ela provavelmente será recorde."

Para reforçar, pense na visita a um médico. O processo é idêntico:

- **Dado (Bruto):** O termômetro marca `39.5C`.
- **Informação (Contexto):** O paciente (João, 40 anos) está com febre alta (`39.5C`).
- **Conhecimento (Padrão):** O médico analisa o padrão: "Febre alta + tosse seca + manchas vermelhas... esse padrão, que já vi antes, é um forte indicador de Sarampo."
- **Sabedoria (Decisão):** O médico toma uma ação: "Vou prescrever o tratamento X e recomendar isolamento imediato."

A Ciência de Dados é o motor que transforma Dados brutos em sabedoria estratégica, o processo que nos permite parar de apenas coletar dados (febre) e começar a tomar decisões (tratamento).

1.1.2 Onde a Ciência de Dados Atua na Amazônia?

A região amazônica é um território de superlativos, mas também de desafios logísticos, ambientais e sociais únicos no planeta. As "grandes perguntas" aqui são complexas e de alto impacto: Como podemos impulsionar a bioeconomia e gerar renda para as comunidades ribeirinhas sem acelerar a degradação? Como otimizar a complexa logística fluvial que conecta comunidades isoladas, transportando alimentos, remédios e produtos? Como monitorar milhões de hectares de floresta em tempo real para combater o desmatamento ou prever surtos de doenças antes que eles se espalhem?

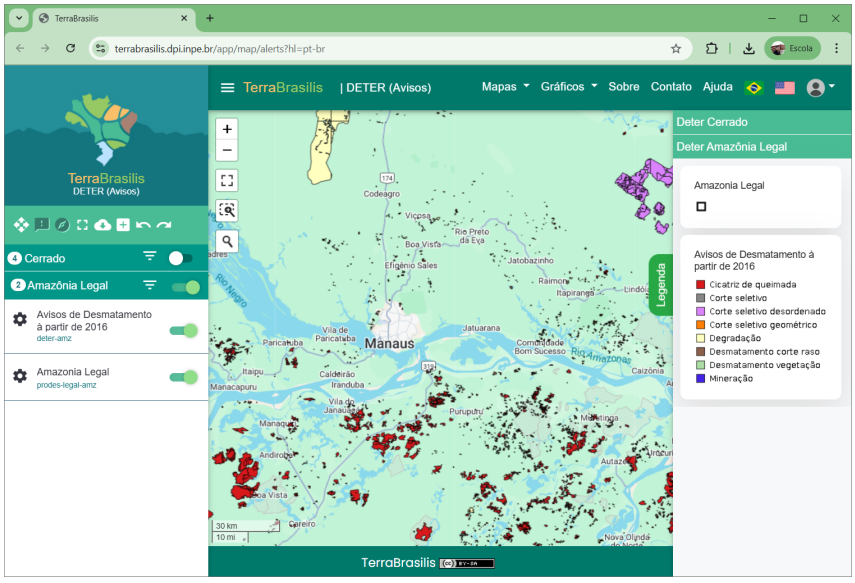

Figura 1.2: Exemplo de monitoramento por satélite da Amazônia Legal nas proximidades da cidade de Manaus.

Neste cenário, a Ciência de Dados deixa de ser uma ferramenta abstrata e torna-se um pilar estratégico para a sobrevivência e o desenvolvimento sustentável. Ela é a única ferramenta capaz de processar o imenso e variado volume de dados (de satélites, sensores climáticos, sensores acústicos, níveis dos rios e registros de saúde) e transformá-los em respostas acionáveis. A aplicação da Ciência de Dados é, portanto, central para a região amazônica, convergindo diretamente com os objetivos de inovação, capacitação e sustentabilidade do projeto CITHA. Outros exemplos de aplicações:

- **Bioeconomia e Cadeias Produtivas:** Previsão da produtividade de safras de açaí e cupuaçu com base em dados de solo, clima e imagens de satélite (sensoriamento remoto), otimizando a colheita e reduzindo perdas.
- **Monitoramento Ambiental:** Análise de imagens de satélite (comparando pixels ao longo do tempo) e dados de sensores acústicos (sons da floresta) para detectar desmatamento ou atividade de mineração ilegal quase em tempo real.
- **Logística Fluvial e Urbana:** O "Waze" dos rios. Otimização de rotas de barcos (transporte de pessoas e cargas) com base em dados de nível dos rios e histórico de navegação, reduzindo custos de combustível e tempo de viagem.
- **Saúde Pública:** Mapeamento preditivo de focos de doenças como malária e dengue, correlacionando dados de saúde (novos casos) com condições climáticas (umidade, chuva) e geográficas, permitindo que a vigilância sanitária atue preventivamente.

1.2 O Ciclo de Vida de um Projeto de Dados

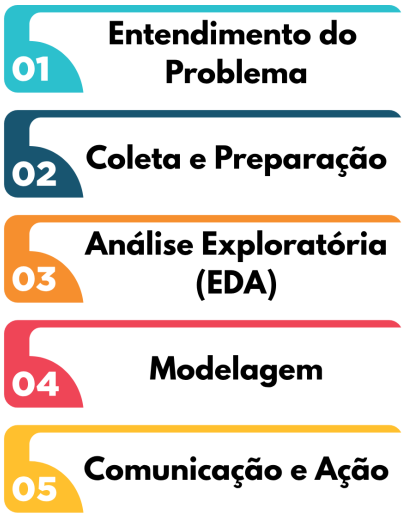

Figura 1.3: As Fases do Processo de Descoberta de Conhecimento em Dados (KDD). É um ciclo contínuo de aprendizado e refinamento.

Um projeto de Ciência de Dados não é um único bloco de código que se escreve de uma vez. É fundamental entender que ele é um processo iterativo (um ciclo), muito semelhante ao próprio método científico, que transforma uma pergunta de negócio vaga em uma resposta clara e baseada em evidências.

A palavra-chave aqui é ciclo: o que se descobre na fase de Análise Exploratória (passo 3), por exemplo, pode revelar que os dados coletados são insuficientes ou tendenciosos, forçando um retorno imediato à fase de Coleta e Preparação (passo 2) para buscar novas fontes. Da mesma forma, um modelo complexo (passo 4) pode se mostrar impraticável para o negócio, exigindo uma redefinição do problema original (passo 1).

Entender esse fluxo e saber navegar por ele é, indiscutivelmente, a habilidade mais importante de um cientista de dados, superando até o domínio de uma ferramenta específica. Saber a sintaxe de uma biblioteca de *Machine Learning* é útil, mas saber qual pergunta fazer primeiro, como validar os dados para ela e quando parar de modelar e focar na comunicação é o que realmente define um projeto bem-sucedido. Este ciclo, seja conhecido como KDD (Descoberta de Conhecimento em Dados) ou CRISP-DM, é o verdadeiro roteiro do cientista.

1.2.1 Entendimento do Problema (O "Porquê")

Esta é a fase **mais importante** e a que menos envolve código. Antes de abrir o Python, gastamos tempo com as pessoas, fazendo perguntas. Se errarmos aqui, construiremos uma solução incrível para o problema errado. O objetivo é traduzir um objetivo de negócio vago em uma pergunta de dados clara.

- **Problema de Negócio (Vago):** "Nossa cooperativa de açaí precisa ser mais eficiente."
- **Perguntas de Análise (Investigação):** "O que significa 'eficiente'? Gastar menos? Perder menos produto? Entregar mais rápido?"
- **Pergunta de Dados (Específica):** "Qual é a correlação entre o tempo de transporte fluvial e o percentual de perda de açaí por oxidação? Se reduzirmos o tempo de viagem em 24h, quanto economizamos?"

Um bom cientista de dados não é quem sabe as respostas certas, mas sim quem é capaz de fazer as perguntas certas.

1.2.2 Entendendo a Matéria-Prima: Tipos de Dados

Antes de coletar e limpar, precisamos entender a natureza fundamental dos nossos dados. Esta etapa não é um mero exercício acadêmico; ela é a fundação sobre a qual toda a nossa análise será construída. O tipo de dado define quais operações matemáticas, quais comparações estatísticas e quais gráficos podemos utilizar. Tentar calcular a "média" de uma coluna de "Municípios" (um dado nominal) não é apenas um erro de código, é um erro conceitual. Da mesma forma, tentar traçar um gráfico de linha para mostrar a "distribuição" de um dado categórico não faz sentido.

Quando carregamos dados no Python, ferramentas como o Pandas tentarão "adivinhar" esses tipos (ex: int64, float64, object). No entanto, é responsabilidade do cientista verificar e corrigir essa classificação. Uma coluna de "ID de Usuário" pode parecer um número (quantitativa), mas na verdade é uma categoria (qualitativa nominal), e tratá-la como um número em um modelo matemático levaria a resultados desastrosos. Portanto, esta classificação é o nosso primeiro filtro para definir a ferramenta de análise correta. Fundamentalmente, todos os dados se dividem em dois grandes grupos:

- **Dados Qualitativos (ou Categóricos):** Descrevem uma qualidade ou característica. São divididos em:

  - **Nominais:** Categorias que não possuem uma ordem intrínseca. Exemplos:
    * `Município`: ("Manaus", "Parintins", "Tefé")
    * `Produto`: ("Açaí", "Cupuaçu", "Mandioca")
    * `Gênero`: ("Masculino", "Feminino")
  - **Ordinais:** Categorias que possuem uma ordem lógica ou hierarquia. Exemplos:
    * `Escolaridade`: ("Fundamental", "Médio", "Superior")
    * `Classificação`: ("Ruim", "Bom", "Excelente")
    * `Tamanho`: ("Pequeno", "Médio", "Grande")

- **Dados Quantitativos (ou Numéricos):** Descrevem uma quantidade (números). São divididos em:
  - **Discretos:** Números que resultam de uma contagem (geralmente inteiros). Exemplos:
    * `Número de Coletores`: (10, 11, 12)
    * `Total de Sementes`
    * `Quantidade de barcos`: (1, 2, 3)
  - **Contínuos:** Números que resultam de uma medição (podem ter casas decimais). Exemplos:
    * `Temperatura`: (26.5 °C)
    * `Nível do Rio`: (14.82 m)
    * `Preço`: (R$ 10,50)
    * `Peso`: (120.4 kg)

Fique Alerta!

Entender essa diferença é vital. Não podemos calcular a "média" de um dado nominal (Qual a média entre "Manaus" e "Tefé"?). Da mesma forma, o gráfico de barras não é ideal para ver a "distribuição" de um dado contínuo. O tipo de dado define a ferramenta de análise.

1.2.3 A Realidade dos Dados: Coleta e Preparação

Agora que sabemos *o quê* procurar (os tipos de dados), podemos coletá-los de fontes diversas (sensores, APIs, bancos de dados, planilhas Excel, arquivos CSV). Aqui, nos deparamos com a primeira e mais importante lição da área: os dados do mundo real são sujos (*messy*). Esta não é uma exceção, é a regra. É nesta fase que a maioria dos projetos falha e onde os cientistas de dados, realisticamente, gastam até 80% do seu tempo.

Dados são "sujos" por inúmeras razões: erros humanos de digitação ("Manaus" vs "manaus"), falhas de sensores (valores ausentes ou *NaN*), sistemas diferentes que não conversam (um usa "10,50" e outro "10.50"), ou até mesmo vieses intencionais ou não nos registros históricos.

Nesta fase, o princípio mais importante é o GIGO (*Garbage In, Garbage Out*), ou "Lixo Entra, Lixo Sai". Podemos ter o modelo de *Machine Learning* mais avançado e complexo (que veremos na fase 4), mas se o alimentarmos com dados inconsistentes, incompletos ou incorretos, o modelo apenas aprenderá padrões errados. Ele nos dará previsões inúteis (ou até perigosas) com grande velocidade.

Para realizar essa limpeza, não usamos o Python puro. Nossa principal ferramenta será a biblioteca Pandas. Pense no Pandas como a bancada de trabalho ou a planilha inteligente do cientista de dados. Ele nos permite carregar arquivos (como CSVs ou Excel) para uma estrutura chamada DataFrame (essencialmente, uma tabela) onde podemos filtrar, modificar e limpar os dados com eficiência.

A maior parte das Ações de limpeza que veremos a seguir, como padronizar Manaus e manaus, são feitas com comandos do Pandas. A preparação, também chamada de *data cleaning* (limpeza de dados) ou *data wrangling*, é o processo de transformar esses dados brutos e caóticos em uma fonte única, limpa, estruturada e confiável, pronta para a análise.

Conhecendo um pouco mais!

Exemplos de "sujeira" e como limpamos:

- **Valores Ausentes (NaN):** O sensor de umidade do solo falhou. **O que fazer:** Removemos a linha ou preenchemos o buraco com a média dos valores vizinhos.
- **Dados Inconsistentes (Tipografia):** Na coluna `Município`, encontramos "Manaus", "manaus" e "MANAUS". **O que fazer:** Padronizamos tudo para letras maiúsculas ("MANAUS").
- **Formatos Errados:** O sistema brasileiro registra `Preço` como "10,50" (com vírgula), mas o Python espera "10.50" (com ponto). **O que fazer:** Substituímos todas as vírgulas por pontos e convertemos a coluna para número.
- ***Outliers* (Valores Extremos):** Um sensor de temperatura na floresta marcou 200°C. **O que fazer:** Isso é claramente um erro de medição. Removemos o registro ou o tratamos como "Valor Ausente".

1.2.4 Análise Exploratória de Dados

Com os dados limpos, iniciamos um "diálogo" com eles. Esta é, para muitos, a fase mais interessante e criativa do ciclo. A *Exploratory Data Analysis* (EDA) é o momento de ser cético e curioso, atuando como um detetive que interroga as evidências. Mesmo após a limpeza (passo anterior), os dados podem conter surpresas. A EDA é nossa rede de segurança para validar se nossas suposições estão corretas e se os dados realmente fazem sentido.

Por exemplo, a limpeza tratou os dados de Temperatura? Um histograma aqui validaria rapidamente se a faixa de valores (ex: 20°C a 40°C) é plausível para a Amazônia. Para isso, usamos duas ferramentas principais: estatística descritiva (como média, mediana, moda e desvio padrão) para resumir os dados, e a visualização (gráficos simples) para revelar suas formas e relações. O objetivo não é criar modelos complexos ainda, mas sim entender profundamente a matéria-prima, encontrar padrões óbvios, identificar anomalias (como *outliers* que a limpeza pode não ter pego) e testar nossas primeiras hipóteses. Perguntas típicas da EDA:

- "Qual é a distribuição dos meus dados contínuos?" (Ex: Histograma de Temperatura)
- "Qual é a contagem dos meus dados categóricos?" (Ex: Gráfico de Barras de Produção por Município)
- "Existe alguma relação (correlação) entre duas variáveis numéricas?" (Ex: Gráfico de Dispersão de Chuva vs. Produção)

- "Como uma variável numérica se compara entre diferentes categorias?" (Ex: Boxplot do Preço do Açaí por Município)

Como dialogamos com os dados? Usando nossos olhos. As ferramentas centrais da EDA são as bibliotecas de visualização, principalmente Matplotlib e Seaborn. O Matplotlib é o motor que desenha os gráficos, e o Seaborn é uma interface mais amigável que, com uma linha de código, nos permite criar os gráficos estatísticos complexos que você vê nesta lista (Histogramas, Boxplots, etc.). É através delas que respondemos a essas perguntas visualmente.

Nesta fase, um simples gráfico revela *insights* que nenhuma tabela complexa ou modelo avançado mostraria. Pular a EDA é o erro mais comum de um iniciante, que tenta ir direto para a modelagem (passo 4). Isso é como tentar construir um telhado sem antes inspecionar os alicerces: o resultado quase certamente será um fracasso, pois o modelo pode acabar aprendendo com dados ruidosos ou enviesados que uma simples análise exploratória teria revelado.

1.2.5 Modelagem e Análise Preditiva

Se a EDA (passo anterior) encontrou padrões fortes e relações promissoras, podemos ir além da simples descrição e tentar "prever" o futuro. É aqui que entramos no domínio do Aprendizado de Máquina (*Machine Learning*). E para construir essa receita matemática, nossa principal ferramenta é a biblioteca Scikit-Learn. Ela é a caixa de ferramentas de IA definitiva para a maioria das tarefas de Ciência de Dados, vindo com dezenas de modelos prontos para Regressão e Classificação. Ela nos permite treinar o modelo nos dados passados (com um comando chamado .fit()) e usá-lo para prever o futuro (com um comando chamado .predict()).

Construímos um modelo (uma "receita" matemática ou um "conjunto de regras" que o computador aprende). O objetivo de um modelo não é "acertar" perfeitamente os dados que ele já viu; é generalizar. Ele precisa aprender com os dados do passado (o conjunto de treino) para fazer previsões precisas sobre dados novos, que ele nunca viu (o conjunto de teste).

O maior perigo nesta fase é o *overfitting* (sobreajuste): quando o modelo é tão complexo que, em vez de aprender o *padrão* real dos dados, ele acaba "decorando" todo o ruído e as particularidades do conjunto de treino. Esse modelo "decorador" terá um desempenho perfeito nos dados de treino, mas falhará miseravelmente em dados novos. Portanto, a modelagem é um balanço delicado entre complexidade e generalização. Os dois tipos mais comuns de modelagem supervisionada são:

- **Regressão (Prever um número Quantitativo):** Usamos o passado para prever quanto. Ex: Com base nos dados de chuva e umidade, prever o `Nível do Rio` em milímetros para daqui a 30 dias.
- **Classificação (Prever uma categoria Qualitativa):** Usamos o passado para prever o quê. Ex: Com base em uma imagem de satélite (dados de pixels), classificar se uma área é "Floresta" ou "Desmatamento".

1.2.6 Data Storytelling

Esta é a fase final e, possivelmente, a mais subestimada do ciclo. É o momento em que todo o trabalho árduo de limpeza, análise e modelagem (os 99% técnicos do projeto) encontra o mundo real. De nada adianta um modelo matemático complexo com 99% de acurácia se o gestor da cooperativa ou o tomador de decisão não entender o que ele significa, ou pior, não confiar nele para tomar uma decisão. Um projeto de dados só tem valor real se ele muda uma ação ou uma estratégia no mundo real.

É aqui que entra o Data Storytelling. Esta não é apenas a arte de fazer "gráficos bonitos". É a habilidade de transformar *insights* técnicos complexos em uma narrativa clara, convincente e, acima de tudo, acionável. Um gestor não quer saber sobre a sua *curva ROC* ou o *P-valor* do seu teste estatístico. Ele quer respostas diretas para as perguntas do negócio: *"Onde estamos perdendo dinheiro?", "Qual é a causa?", e "O que devemos fazer a respeito?"*

Esta fase é a ponte entre análise e a ação, transformando *insights* técnicos em relatórios executivos e visualizações claras que contam uma história completa: o contexto (o problema), a descoberta (o padrão nos dados) e a recomendação (a decisão sugerida).

Fique Alerta!

Comunicação Fraca: *"O teste de correlação de Pearson entre Tempo de Transporte e Perda de Produto resultou em um P-Valor de 0.002 e um R^2 de 0.78."*

Comunicação Forte: *"Nossa análise prova que há uma forte relação entre o tempo de viagem e a perda de açaí. O modelo prevê que, para cada dia que reduzimos no transporte fluvial, economizamos R$ 15.000 em produto. Recomendamos investir em barcos mais rápidos para a rota de Coari, pois o retorno do investimento será de 6 meses."*

1.3 O Cientista de Dados: Um Perfil Interdisciplinar

O Cientista de Dados é o profissional responsável por coletar, organizar e analisar grandes volumes de dados para gerar informações que auxiliem na tomada de decisões estratégicas.

1.3.1 Habilidades Fundamentais

A principal ferramenta de um cientista de dados é a curiosidade. As habilidades técnicas podem ser aprendidas, mas a vontade de fazer perguntas e investigar é o que move a análise.

- **Programação e Lógica:** Domínio de linguagens como Python (para análise) e SQL (para buscar dados em bancos de dados), e a compreensão de como estruturar um problema.
- **Estatística e Matemática:** Fundamentos cruciais para realizar análises, gerar estatísticas e, o mais importante, entender a incerteza (saber se um resultado é um padrão real ou apenas uma coincidência).
- **Conhecimento de Negócio e Comunicação:** O analista deve entender o domínio em que atua (ex: processos agrícolas, logística fluvial) e possuir excelente capacidade de comunicação (verbal e escrita) para propor soluções adequadas.

1.3.2 A Disrupção e as Novas Carreiras

A computação moderna e a capacidade de processar dados em larga escala estão causando uma disrupção profunda na sociedade. Carreiras tradicionais focadas em tarefas repetitivas (como entrada de dados manual ou análise descritiva simples) estão sendo rapidamente automatizadas. No entanto, essa mesma força está criando uma explosão de novas carreiras que não existiam há uma década. Essas novas funções nascem exatamente da intersecção de áreas que antes não dialogavam:

- **Engenheiro de ML Ops (Machine Learning Operations):** A intersecção entre Engenharia de Software (DevOps) e Ciência de Dados. Foca em como colocar um modelo de IA em produção de forma confiável e escalável.
- **Analista de Bioinformática:** A intersecção entre Biologia/Genética e Ciência da Computação. Analisa sequências de DNA e dados genômicos para descobrir novos medicamentos ou entender doenças.
- **Especialista em NLP (Processamento de Linguagem Natural):** A intersecção entre Linguística e Computação. Cria *chatbots* inteligentes, sistemas de tradução ou ferramentas que analisam o sentimento em mídias sociais.
- **Cientista de Dados Jurídicos (Legal Tech):** A intersecção entre Direito e Estatística. Analisa milhões de documentos de processos para encontrar padrões, prever resultados de casos ou automatizar a revisão de contratos.

O poder do programador moderno não está mais apenas em "construir software", mas em "construir sistemas que aprendem". Estamos migrando de uma sociedade que registra dados para uma sociedade que toma decisões baseadas em dados, e o Cientista de Dados é o profissional que lidera essa transformação.

1.3.3 A Ética dos Dados

O trabalho vai além da técnica. Dados são reflexos de processos humanos e, portanto, podem conter vieses (preconceitos). Se um banco de dados histórico mostra que um grupo de pessoas recebeu menos crédito no passado, um modelo de IA treinado com esses dados "aprenderá" esse preconceito e continuará a discriminar esse grupo no futuro.

Imagine um modelo de IA treinado para aprovar empréstimos. Se os dados históricos (dos últimos 20 anos) mostram que o banco, por razões de preconceito social, aprovou menos empréstimos para um determinado grupo demográfico, o modelo de IA aprenderá esse padrão. O resultado? O modelo irá automatizar e perpetuar a discriminação, dando aprovações mais baixas para esse mesmo grupo, mas agora sob o pretexto de ser uma decisão objetiva da IA.

O papel ético do cientista de dados é mitigar ativamente esse risco. Na fase de Análise Exploratória (EDA), ele deve se perguntar: Meus dados estão balanceados? A taxa de aprovação histórica é a mesma entre todos os grupos?. Se um desequilíbrio for encontrado, ele deve aplicar técnicas de mitigação, como reamostragem dos dados (para forçar o balanceamento) ou usar métricas de justiça (fairness) para avaliar o modelo, garantindo que ele não penalize nenhum grupo. É fundamental garantir o uso ético e transparente dos dados, em conformidade com as legislações vigentes (como a LGPD - Lei Geral de Proteção de Dados).

Fique Alerta!

Sua função não se limita à técnica. É essencial que o profissional mantenha-se atualizado quanto às boas práticas, desenvolvendo uma visão multidisciplinar que una tecnologia, ética e senso de responsabilidade para questionar ativamente: *"O meu modelo está reforçando um problema social ou ajudando a resolvê-lo?"*

1.4 Aplicando seus conhecimentos

1. **Ciclo de Vida:** Se você fosse contratado para analisar o motivo da evasão escolar (abandono) em escolas rurais do Amazonas, quais seriam as duas primeiras perguntas que você faria na fase de Entendimento do Problema?

2. **Tipos de Dados:** Classifique cada uma das variáveis abaixo como *Qualitativa Nominal*, *Qualitativa Ordinal*, *Quantitativa Discreta* ou *Quantitativa Contínua*:

   (a) O `Preço` do litro do açaí.

   (b) O `Município` de origem da semente.

   (c) A `Classificação` de um fruto (Ex: "Tipo A", "Tipo B", "Tipo C").

   (d) A `Quantidade de frutos` em uma caixa.

3. **Limpeza de Dados:** Você recebe uma planilha com uma coluna "Idade" que contém os seguintes valores: `[25, 30, "vinte", 45, -10, 999, 28]`. Quais são os problemas (pelo menos 3) nesta coluna e que ações de limpeza você tomaria?

4. **Modelagem:** Explique a diferença entre a Modelagem de Regressão e a de Classificação (discutidas na Seção 1.2.5). Dê um exemplo prático para cada uma, diferente dos apresentados no texto, aplicado ao contexto amazônico.

1.5 Considerações deste Capítulo

Neste capítulo, você estabeleceu o mapa da nossa jornada: a Ciência de Dados é a união da tecnologia, estatística e conhecimento de negócio. O Ciclo de Vida do Projeto é o seu roteiro, e as bibliotecas Python são suas ferramentas. Você aprendeu a diferenciar os tipos de dados, o que é o primeiro passo para uma análise correta. Mais importante, você compreendeu seu papel como um profissional ético, multidisciplinar e, acima de tudo, curioso. No próximo capítulo, aprofundaremos as fases de Coleta e Preparação dos Dados, colocando a mão na massa com Pandas para transformar arquivos brutos e não confiáveis em DataFrames limpos e prontos para a análise.

Capítulo 2

Explorando Dados: Visualização e Estatística Descritiva

Iniciando o diálogo...

No Capítulo 1, desenhamos o "mapa" da jornada de um cientista de dados, desde a formulação de um problema até a apresentação de uma solução. Agora, neste capítulo, vamos construir nosso "laboratório" e aprender a usar as ferramentas fundamentais de análise. Quando um cientista de dados recebe um novo conjunto de dados, ele não começa aplicando algoritmos complexos. O trabalho se assemelha mais ao de um detetive: o primeiro passo é a exploração. Precisamos interrogar nossos dados, entender sua forma, sua estrutura e suas peculiaridades.

- *Quantos dados temos?*
- *Eles estão concentrados ou espalhados?*
- *Existe algum valor "estranho" ou atípico?*
- *Duas variáveis parecem estar relacionadas?*

Para responder a essas perguntas, não podemos confiar apenas na intuição. Precisamos de um alicerce sólido em Estatística.

2.1 O Experimento do Peixe

Para tornar esses conceitos claros e práticos, usaremos um cenário de laboratório ao longo de todo este capítulo. Imagine que somos analistas em uma cooperativa de piscicultura aqui na Amazônia. A cooperativa quer decidir em qual ração investir para o próximo ano e nos entrega um pequeno conjunto de dados de um experimento:

> *Temos dados de 30 peixes, onde 15 foram alimentados com a "Ração A" e 15 com a "Ração B". Para cada peixe, medimos seu crescimento em quilos e o quanto ele consumiu de ração. Qual ração é realmente melhor?*

Usaremos este *dataset* simples e controlado como nosso "fio condutor". Com ele, vamos aprender visualmente como descrever e resumir os dados usando a Estatística Descritiva.

Para criar este cenário, usamos o código Python abaixo para gerar um *dataset* fictício e reprodutível chamado `experimento_peixes.csv`.

Copie e Teste!

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

# 1. Definir o "seed" para que os resultados sejam sempre os
   mesmos (reprodutibilidade)
np.random.seed(42)

# 2. Definir o número de amostras (peixes) por grupo
N = 15

# Ração A: Crescimento CONCENTRADO, Correlação FORTE
# Média 5.0, Desvio Padrão 0.5
crescimento_A = np.random.normal(loc=5.0, scale=0.5, size=N)
# Consumo tem CORRELAÇÃO FORTE com crescimento
consumo_A = (crescimento_A * 1.5) + np.random.normal(loc=0, scale
   =0.3, size=N)

# Ração B: Crescimento DISPERSO, OUTLIER, Correlação FRACA
# Média ~4.2, Desvio Padrão 1.5 (N-1 para o outlier)
crescimento_B_base = np.random.normal(loc=4.2, scale=1.5, size=N
   -1)
# Adicionamos o OUTLIER
outlier_B = [25.0]
crescimento_B = np.concatenate((crescimento_B_base, outlier_B))
# Consumo tem CORRELAÇÃO FRACA
consumo_B = np.random.normal(loc=7.0, scale=2.0, size=N)

# --- Juntando os dados ---
df_A = pd.DataFrame({
    'tipo_racao': 'Ração A',
    'crescimento_kg': crescimento_A,
    'consumo_racao_kg': consumo_A
})
df_B = pd.DataFrame({
    'tipo_racao': 'Ração B',
    'crescimento_kg': crescimento_B,
    'consumo_racao_kg': consumo_B
})

# Combinando os grupos em um único DataFrame
df_peixes = pd.concat([df_A, df_B], ignore_index=True)
```

```
# --- Adicionando Features Bônus  ---
df_peixes['id_peixe'] = range(1, len(df_peixes) + 1)
df_peixes['meses_cultivo'] = np.random.choice([6, 7, 8, 9], size=
    len(df_peixes))
df_peixes['avaliacao_saude'] = np.random.choice(['Baixa', 'Média',
    'Alta'], size=len(df_peixes), p=[0.2, 0.5, 0.3])

# Reordenar colunas para melhor visualização
df_peixes = df_peixes[['id_peixe', 'tipo_racao', 'crescimento_kg',
    'consumo_racao_kg', 'meses_cultivo', 'avaliacao_saude']]

# Embaralhar o dataset para parecer mais realista
df_peixes = df_peixes.sample(frac=1, random_state=42).reset_index(
    drop=True)

# Salvar para uso futuro
df_peixes.to_csv('experimento_peixes.csv', index=False)

print("Dataset de Laboratório ('df_peixes') Gerado com Sucesso")
print(df_peixes.head(10))
```

Tabela 2.1: Dados de Cultivo e Saúde de Peixes.

ID Peixe	Tipo de Ração	Crescimento (kg)	Consumo (kg)	Meses Cultivo	Avaliação Saúde
0	Ração B	4.03	5.32	8	Média
1	Ração B	3.30	4.04	8	Alta
2	Ração B	2.21	5.65	8	Alta
3	Ração B	4.18	6.08	8	Média
4	Ração A	4.77	6.72	8	Média
5	Ração A	5.27	7.74	9	Média
6	Ração B	3.75	6.38	6	Média
7	Ração B	4.50	8.22	8	Alta
8	Ração A	5.12	7.79	8	Baixa
9	Ração A	5.25	7.70	7	Média

Nossa primeira ferramenta nesta investigação não será uma fórmula matemática complexa, mas sim nossos próprios olhos. Vamos começar aprendendo a visualizar os dados.

2.2 Visualizando Dados

Nossa primeira ferramenta de investigação não é uma fórmula, mas nossos próprios olhos. A visualização de dados é a maneira mais rápida e intuitiva de descobrir padrões, identificar problemas e entender a "forma" dos nossos dados. Cada tipo de gráfico é projetado para responder a um tipo diferente de pergunta. Vamos começar com o mais comum de todos: o gráfico de barras.

2.2.1 Gráfico de Barras

"Como um valor numérico se compara entre diferentes grupos?" O gráfico de barras é nossa ferramenta fundamental para comparar uma quantidade (uma variável quantitativa) entre diferentes categorias (uma variável qualitativa).

No nosso Experimento do Peixe, a pergunta mais óbvia que podemos fazer é: Qual ração teve o melhor crescimento médio? Para criar esse gráfico, nosso processo tem duas etapas:

1. **Agrupar os Dados:** Primeiro, precisamos calcular a média de `crescimento_kg` para cada `tipo_racao`.
2. **Visualizar os Dados:** Em seguida, usamos as médias calculadas para desenhar as barras.

Para essa tarefa utilizaremos duas de nossas principais ferramentas: Pandas para organizar e agrupar os dados (com o comando .groupby()) e Seaborn (apelidada de sns) para desenhar o gráfico (sns.barplot()).

Copie e Teste!

```
# --- 1. Preparação dos Dados para o Gráfico ---
# Para comparar as rações, primeiro calculamos a média de
   crescimento de cada grupo.
# Agrupa por 'tipo_racao' e calcula a média de 'crescimento_kg'
media_por_racao = df_peixes.groupby('tipo_racao')['crescimento_kg'
   ].mean().reset_index()

print("--- Dados Prontos para o Gráfico (Médias Calculadas) ---")
print(media_por_racao)
print("\n")

# --- 2. Criação do Gráfico de Barras ---
print("--- Gerando Gráfico de Barras ---")
plt.figure(figsize=(8, 6))

# Usamos o Seaborn (sns) para criar o gráfico de barras
sns.barplot(x='tipo_racao', y='crescimento_kg', data=
   media_por_racao)

# --- 3. Customização (Títulos e Rótulos) ---
plt.title('Crescimento Médio por Tipo de Ração')
plt.xlabel('Tipo de Ração')
plt.ylabel('Crescimento Médio (kg)')
plt.show()
```

Tabela 2.2: Médias de Crescimento por Tipo de Ração.

Tipo de Ração	Crescimento Médio (kg)
Ração A	5.00
Ração B	5.33

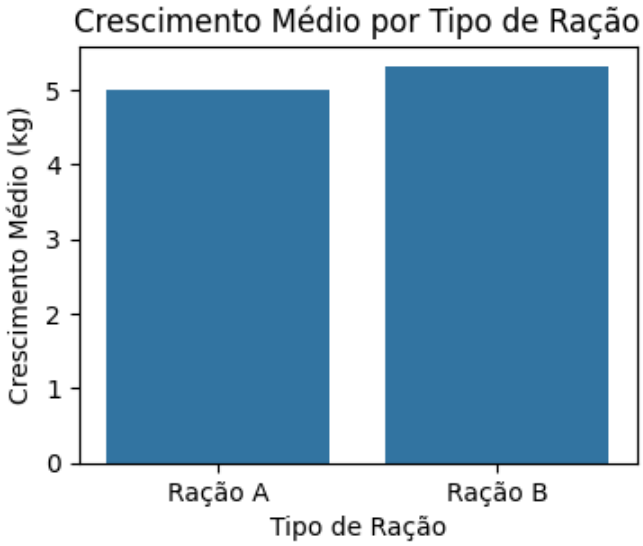

Figura 2.1: Gráfico de barras do crescimento médio por tipo de ração.

O Perigo da Primeira Vista

Ao olhar para o gráfico de barras gerado, sua primeira conclusão seria óbvia: a *"Ração B" parece ser significativamente melhor que a "Ração A"*, pois sua barra de crescimento médio é visivelmente mais alta (aproximadamente 5.33 kg contra 5.01 kg). No entanto, como cientistas de dados, aprendemos a ser céticos. Lembre-se, nós *projetamos* este *dataset* para ter um *outlier* (um peixe que cresceu 25.0 kg) no grupo da Ração B. Este gráfico está nos contando uma história incompleta. A média é uma métrica muito sensível a valores extremos. Esse único peixe "puxou" a média da Ração B inteira para cima, distorcendo nossa comparação.

O gráfico de barras é excelente para mostrar uma comparação de médias, mas ele é péssimo em mostrar a dispersão e os *outliers* dos dados. Ele nos deu uma resposta inicial, mas agora nos deu uma nova pergunta: "Essa média da Ração B é confiável?" Para investigar isso, precisamos de um gráfico que mostre a distribuição dos dados.

2.2.2 Gráfico de Linha

"Como uma variável mudou ao longo de um eixo ordenado (como o tempo)?" Enquanto o gráfico de barras é excelente para comparar categorias distintas, o gráfico de linha é a ferramenta ideal para mostrar tendências e sazonalidade. Ele conecta uma série de pontos de dados, facilitando a visualização de padrões de crescimento, queda ou ciclos que se repetem.

No nosso "Experimento do Peixe", temos a coluna `meses_cultivo`. Uma pergunta natural seria: *"O crescimento médio dos peixes muda conforme eles ficam mais velhos?"* Vamos investigar isso agrupando os dados por mês de cultivo e tipo de ração, e então plotar as médias.

Copie e Teste!

```
# --- 1. Preparação dos Dados para o Gráfico ---
# Queremos ver a média de crescimento POR MÊS e POR RAÇÃO.
# Usamos o .groupby() para agrupar por duas colunas.
media_por_mes = df_peixes.groupby(['meses_cultivo', 'tipo_racao'])
    ['crescimento_kg'].mean().reset_index()
```

```
print("--- Dados Prontos para o Gráfico (Médias por Mês) ---")
print(media_por_mes)
print("\n")

# --- 2. Criação do Gráfico de Linha ---
print("--- Gerando Gráfico de Linha ---")
plt.figure(figsize=(10, 6))

# Usamos o Seaborn (sns) para criar o gráfico de linha
sns.lineplot(data=media_por_mes, x='meses_cultivo', y='
    crescimento_kg', hue='tipo_racao', marker='o')

# --- 3. Customização (Títulos e Rótulos) ---
plt.title('Crescimento Médio dos Peixes ao Longo dos Meses')
plt.xlabel('Meses de Cultivo')
plt.ylabel('Crescimento Médio (kg)')
plt.legend(title='Tipo de Ração') # Adiciona uma legenda
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()
```

Tabela 2.3: Médias de Crescimento por Mês de Cultivo e Tipo de Ração.

Meses Cultivo	Tipo de Ração	Crescimento Médio (kg)
6	Ração A	4.82
6	Ração B	4.40
7	Ração A	4.97
7	Ração B	3.06
8	Ração A	4.91
8	Ração B	6.64
9	Ração A	5.29

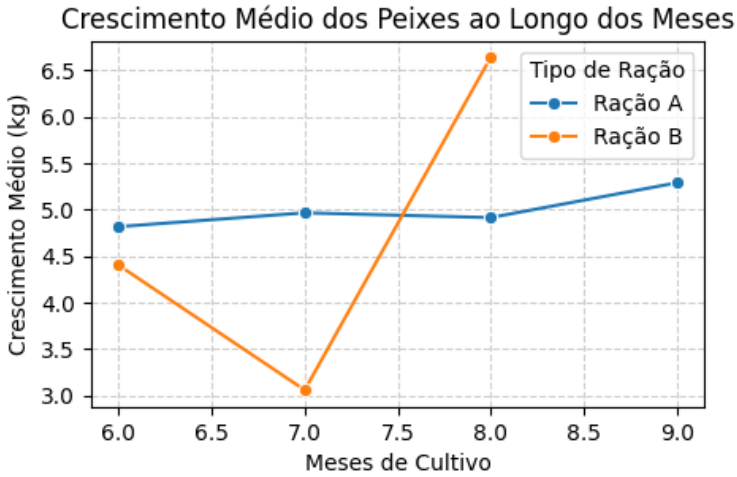

Figura 2.2: Gráfico de linhas do crescimento médio dos peixes ao longo dos meses.

O Poder das Tendências

Ao contrário do gráfico de barras, que nos deu uma "imagem" estática da média geral, o gráfico de linha nos dá um "filme" de como essa média evolui. No gráfico gerado, podemos comparar visualmente as duas rações ao longo do tempo.

- A Ração A (linha azul) pode mostrar um crescimento mais estável e previsível.
- A Ração B (linha laranja) provavelmente parecerá mais errática ou volátil (especialmente no mês 8, onde o *outlier* puxa a média para cima).

O gráfico de linha é a ferramenta principal para analisar Séries Temporais. Ele nos permite identificar sazonalidade (padrões que se repetem, como o ciclo de cheia e vazante dos rios) e tendências de longo prazo. Até agora, vimos como comparar categorias (barras) e como seguir tendências (linhas). Mas e se a nossa pergunta for sobre a forma e a composição dos nossos dados? Para isso, precisamos do histograma.

2.2.3 Histograma

"Como meus dados estão distribuídos? Onde os valores mais se concentram? Existem lacunas ou picos inesperados?" O histograma é um dos gráficos mais importantes na caixa de ferramentas de um cientista de dados. Ele pega uma variável quantitativa (ex. `crescimento_kg`), divide-a em faixas (ou *bins*), e conta quantas observações caem dentro de cada faixa.

O resultado é um gráfico de barras que revela a distribuição de frequência dos dados. No nosso experimento, o gráfico de barras anterior nos deu uma nova pergunta: *"Por que a média da Ração B é tão alta?"* Vamos usar o histograma para investigar a distribuição de cada grupo.

Copie e Teste!

```
# --- 1. Criação dos Histogramas ---
print("--- Gerando Histogramas de Distribuição ---")

# Criamos uma figura com dois gráficos lado a lado
fig, axes = plt.subplots(1, 2, figsize=(14, 6), sharey=True)
fig.suptitle('Distribuição do Crescimento (kg) por Tipo de Ração')

# Histograma da Ração A
sns.histplot(df_peixes[df_peixes['tipo_racao'] == 'Ração A']['
   crescimento_kg'], ax=axes[0], kde=True)
axes[0].set_title('Ração A')
axes[0].set_xlabel('Crescimento (kg)')
axes[0].set_ylabel('Frequência (N° de Peixes)')

# Histograma da Ração B
sns.histplot(df_peixes[df_peixes['tipo_racao'] == 'Ração B']['
   crescimento_kg'], ax=axes[1], kde=True)
axes[1].set_title('Ração B')
axes[1].set_xlabel('Crescimento (kg)')
axes[1].set_ylabel('') # Remove o rótulo Y do segundo gráfico

plt.show()
```

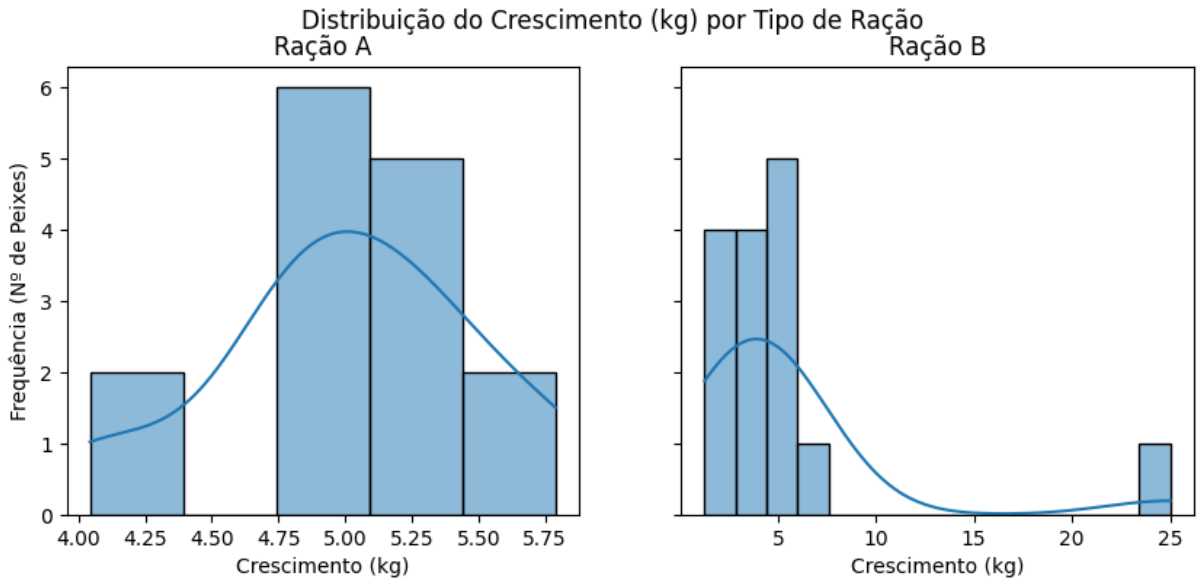

Figura 2.3: Histogramas da distribuição do crescimento (kg) por tipo de ração.

Análise: Expondo o *Outlier*

1. **Gráfico da Ração A:** Vemos uma bela "curva de sino" (uma distribuição normal). Os dados estão perfeitamente concentrados em torno de 5.0 kg, com pouca variação. Este é o visual de um processo consistente e previsível.

2. **Gráfico da Ração B:** Este gráfico é completamente diferente. Vemos um grupo de peixes concentrados em valores mais baixos (entre 1.0 e 7.0 kg) e, em seguida, um enorme espaço vazio, seguido por uma única barra solitária lá na direita, no valor de 25.0 kg.

O histograma expôs o *outlier*. Ele nos confirmou visualmente que a média alta da Ração B (que vimos no gráfico de barras) não se deve ao fato de todos os peixes crescerem muito, mas sim a um único peixe "superstar" que distorceu o cálculo. O histograma é ótimo para ver a forma da distribuição. Mas e se quiséssemos um resumo estatístico dessa forma? Para isso, usamos o Boxplot.

2.2.4 Boxplot

"Qual é o resumo estatístico dos meus dados? Onde estão a mediana, os 50% centrais dos dados e quais são os valores atípicos (outliers)?" O Boxplot (ou Diagrama de Caixa) nos permite ir além: ele resume a distribuição em 5 números estatísticos e é a melhor ferramenta visual para comparar a dispersão entre diferentes grupos. Um boxplot nos mostra:

- A Mediana (linha central): O valor do meio dos dados (percentil 50).
- A Caixa (IQR): Onde se concentram os 50% centrais dos dados (do percentil 25 ao 75).
- Os "Bigodes" (whiskers): Mostram o alcance principal dos dados (excluindo os *outliers*).
- Os *Outliers* (pontos): Valores que caem muito fora do alcance principal, identificados estatisticamente.

Vamos usá-lo para comparar diretamente a "Ração A" com a "Ração B".

Copie e Teste!

```
print("--- Gerando Boxplots Comparativos ---")
plt.figure(figsize=(10, 7))

# Usamos o Seaborn (sns) para criar o boxplot
# x = eixo categórico, y = eixo numérico
sns.boxplot(x='tipo_racao', y='crescimento_kg', data=df_peixes)

plt.title('Comparação da Distribuição do Crescimento (kg) por
   Ração')
plt.xlabel('Tipo de Ração')
plt.ylabel('Crescimento (kg)')
plt.show()
```

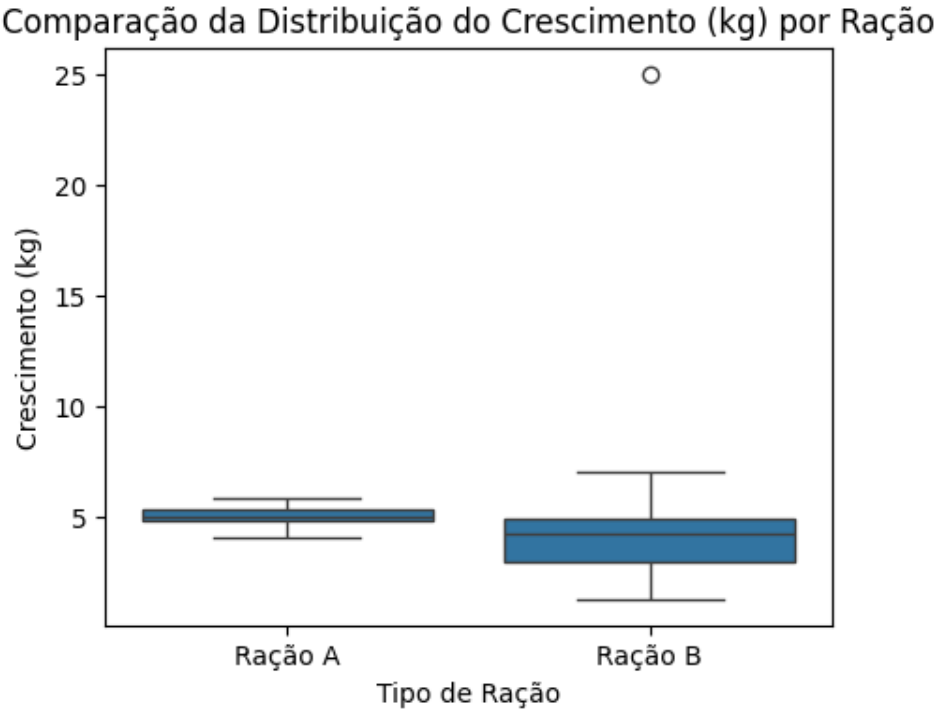

Figura 2.4: Boxplot da comparação da distribuição do crescimento (kg) por ração.

Análise: Avaliando a distorção causada pelo *Outlier*

Este gráfico é a prova definitiva do nosso storytelling e nos conta toda a história em uma única imagem:

- **Ração A:** É uma caixa "curta" e "simétrica". A mediana (linha central) está bem no meio. Os "bigodes" são curtos. Isso nos diz visualmente que os dados são consistentes, previsíveis e concentrados em torno de 5.0 kg.
- **Ração B:** É uma caixa "longa" e "assimétrica". A mediana está na parte de baixo da caixa. E, o mais importante, vemos um ponto solitário lá em cima (nosso outlier de 25.0 kg). Isso nos diz que os dados são dispersos, imprevisíveis e assimétricos.

O Boxplot é a ferramenta perfeita para complementar o histograma. Ele confirmou o outlier, mas a que custo?

Observe no gráfico 2.4 que a presença do valor extremo de 25.0 kg "achata" completamente o boxplot da Ração B. A escala do eixo Y precisa se ajustar para incluir o outlier, tornando a "caixa" (o IQR) e os "bigodes" da Ração B tão pequenos que são quase impossíveis de ler. Não conseguimos comparar de forma justa a mediana ou a variabilidade da Ração B com a Ração A.

Para resolver isso e analisar a distribuição principal dos dados, podemos criar um segundo gráfico, filtrando temporariamente esse outlier. Esta é uma técnica comum para compreender o comportamento do "grosso" dos dados.

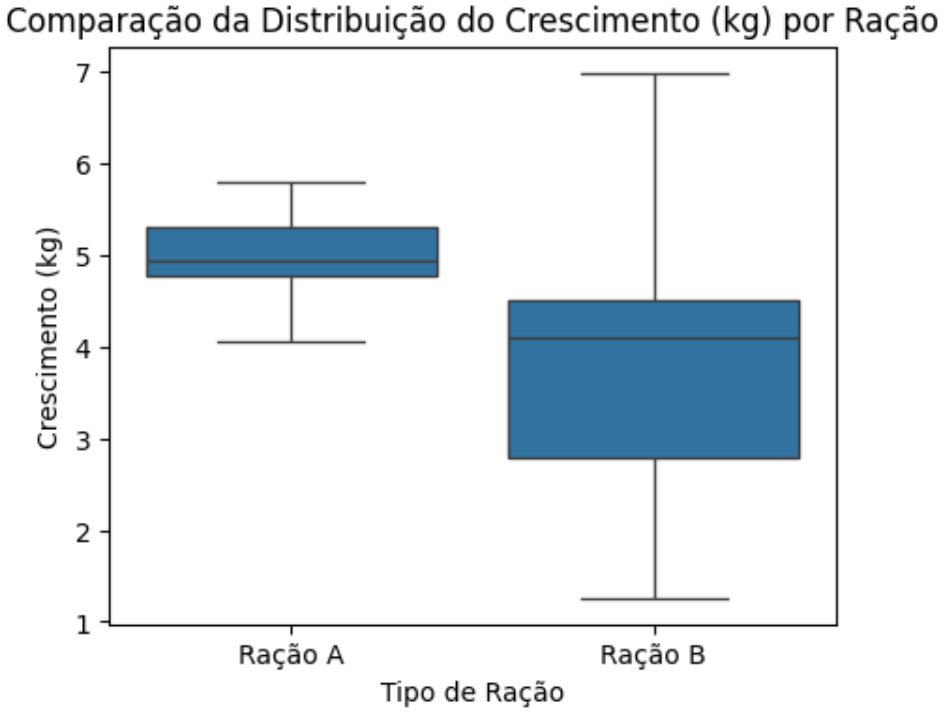

Figura 2.5: Boxplot alternativo da comparação da distribuição do crescimento (kg) por ração, corrigido pela remoção do outlier.

Análise: Avaliando a distribuição dos dados sem o *Outlier*

Agora com o gráfico 2.5 atualizado, podemos finalmente ver a história com mais clareza:

- **Ração A:** Continua a mesma, um grupo coeso e previsível.
- **Ração B:** A verdadeira forma da sua distribuição é revelada, ela é extremamente longa e assimétrica.
  - A mediana (linha central, 4.1kg) é visivelmente mais baixa que a da Ração A (4.9kg).
  - A variabilidade é imensa, com a caixa (IQR) muito mais alta que a Ração A, enquanto que o "bigode" inferior se estende até 1.2kg e o superior até 7.0kg.

O uso dos dois gráficos nos permitiu concluir o storytelling: A Ração A produz um crescimento consistente. A Ração B é altamente imprevisível; a maioria dos peixes tem um crescimento menor e mais variável que a Ração A, mas ela também tem o potencial de gerar um resultado extremo (o outlier de 25.0 kg), que pode ter sido um "super peixe" ou um erro de medição.

2.2.5 Gráfico de Dispersão

"Minhas variáveis se movem juntas? Quando uma sobe, a outra também sobe? Ou ela desce?" O Gráfico de Dispersão (ou *Scatter Plot*) é a nossa ferramenta para investigar a correlação entre duas variáveis quantitativas. Cada ponto no gráfico representa uma única observação (um peixe), posicionado de acordo com seus valores nos eixos X e Y. Ao olharmos o padrão formado por todos os pontos, podemos ver se existe uma relação.

No nosso "Experimento do Peixe", a pergunta mais importante é: *"O quanto um peixe come (`consumo_racao_kg`) influencia o quanto ele cresce (`crescimento_kg`)?"* E será que essa relação é a mesma para as duas rações?

Copie e Teste!

```
# --- 1. Criação dos Gráficos de Dispersão ---
print("--- Gerando Gráficos de Dispersão (Correlação) ---")

# Criamos uma figura com dois gráficos, um ao lado do outro
fig, axes = plt.subplots(1, 2, figsize=(14, 6))
fig.suptitle('Relação entre Consumo de Ração e Crescimento')

# Gráfico da Ração A
# Usamos sns.regplot() para já traçar a linha de tendência
sns.regplot(data=df_peixes[df_peixes['tipo_racao'] == 'Ração A'],
   x='consumo_racao_kg', y='crescimento_kg', ax=axes[0], ci=None,
   line_kws={'color':'red'})
axes[0].set_title('Ração A')
axes[0].set_xlabel('Consumo (kg)')
axes[0].set_ylabel('Crescimento (kg)')

# Gráfico da Ração B
sns.regplot(data=df_peixes[df_peixes['tipo_racao'] == 'Ração B'],
   x='consumo_racao_kg', y='crescimento_kg', ax=axes[1], ci=None,
   line_kws={'color':'red'})
axes[1].set_title('Ração B')
axes[1].set_xlabel('Consumo (kg)')
axes[1].set_ylabel('') # Remove o rótulo Y

plt.show()

# --- 2. Cálculo Numérico (O Coeficiente de Correlação) ---
# O gráfico é visual, o coeficiente é o número.
print("\n--- Coeficiente de Correlação (Pearson) ---")

# Filtra os dados de cada ração
df_A = df_peixes[df_peixes['tipo_racao'] == 'Ração A']
df_B = df_peixes[df_peixes['tipo_racao'] == 'Ração B']

# Calcula a correlação entre as duas colunas
corr_A = df_A['consumo_racao_kg'].corr(df_A['crescimento_kg'])
corr_B = df_B['consumo_racao_kg'].corr(df_B['crescimento_kg'])
```

```
print(f"Correlação Ração A: {corr_A:.4f}")
print(f"Correlação Ração B: {corr_B:.4f}")
```

Tela do Terminal

```
--- Coeficiente de Correlação (Pearson) ---
Correlação Ração A: 0.9571
Correlação Ração B: 0.2139
```

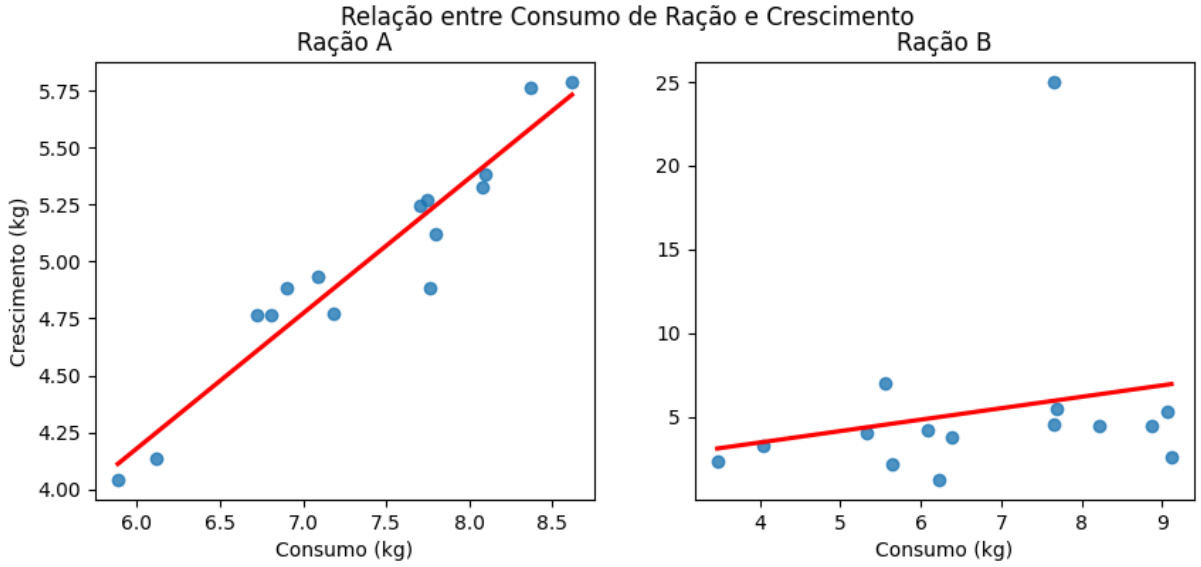

Figura 2.6: Diagrama de dispersão e relação entre consumo e crescimento para as Rações A e B.

Análise: Uma Relação Clara vs. o Caos

Os resultados aqui são incrivelmente reveladores e cumprem exatamente o que projetamos em nosso *dataset*:

- **Ração A:** Os pontos formam um padrão linear quase perfeito, subindo da esquerda para a direita. A linha de tendência vermelha captura essa relação com clareza. O coeficiente numérico de +0.9571 confirma uma correlação positiva muito forte.
  - **Tendência:** Com a Ração A, o crescimento é diretamente e previsivelmente ligado ao consumo. Mais comida = mais crescimento.
- **Ração B:** Os pontos estão espalhados por todo o gráfico como uma nuvem caótica (incluindo o *outlier* de 25kg, que está deslocado no topo). A linha de tendência fica quase na horizontal, mostrando que não há um padrão claro. O coeficiente de +0.2139 confirma que há uma correlação muito fraca.
  - **Tendência:** Com a Ração B, o quanto um peixe come *não* parece ter relação com o quanto ele cresce. Isso reforça nossa conclusão de que a Ração B é imprevisível.

O Gráfico de Dispersão é a nossa principal ferramenta para validar hipóteses de causa e efeito. Ele é a base visual para a seção de Correlação que veremos mais à frente.

2.2.6 Mapa de Calor

"Qual é a força da correlação entre todos os pares de variáveis numéricas do meu dataset?" No tópico anterior, investigamos a relação entre `consumo_racao_kg` e `crescimento_kg`. Mas o que fazer quando temos 5, 10, ou 50 variáveis numéricas? Criar um gráfico de dispersão para cada par seria impraticável. É aqui que o Mapa de Calor (Heatmap) se torna essencial. Ele exibe uma matriz onde cada célula mostra o coeficiente de correlação (um número entre -1 e +1) entre duas variáveis. As cores nos dão um diagnóstico visual imediato:

- **Cores Quentes (ex: vermelho):** Correlação Positiva Forte (próximo de +1).
- **Cores Frias (ex: azul):** Correlação Negativa Forte (próximo de -1).
- **Cores Neutras (ex: branco):** Sem Correlação (próximo de 0).

Copie e Teste!

```
# --- 1. Preparação dos Dados (Cálculo da Correlação) ---
# Selecionamos apenas as colunas numéricas
colunas_numericas = ['crescimento_kg', 'consumo_racao_kg', '
   meses_cultivo']
df_numerico = df_peixes[colunas_numericas]

# Calculamos a matriz de correlação
matriz_corr = df_numerico.corr()

print("--- Matriz de Correlação ---")
print(matriz_corr)
print("\n")

# --- 2. Criação do Mapa de Calor ---
print("--- Gerando Mapa de Calor (Heatmap) ---")
plt.figure(figsize=(8, 6))

sns.heatmap(matriz_corr, annot=True, cmap='coolwarm', fmt='.2f',
   linewidths=0.5)

plt.title('Mapa de Calor das Correlações')
plt.show()
```

Tabela 2.4: Matriz de Correlação.

	crescimento_kg	consumo_racao_kg	meses_cultivo
crescimento_kg	1.00	0.21	0.14
consumo_racao_kg	0.21	1.00	0.23
meses_cultivo	0.14	0.23	1.00

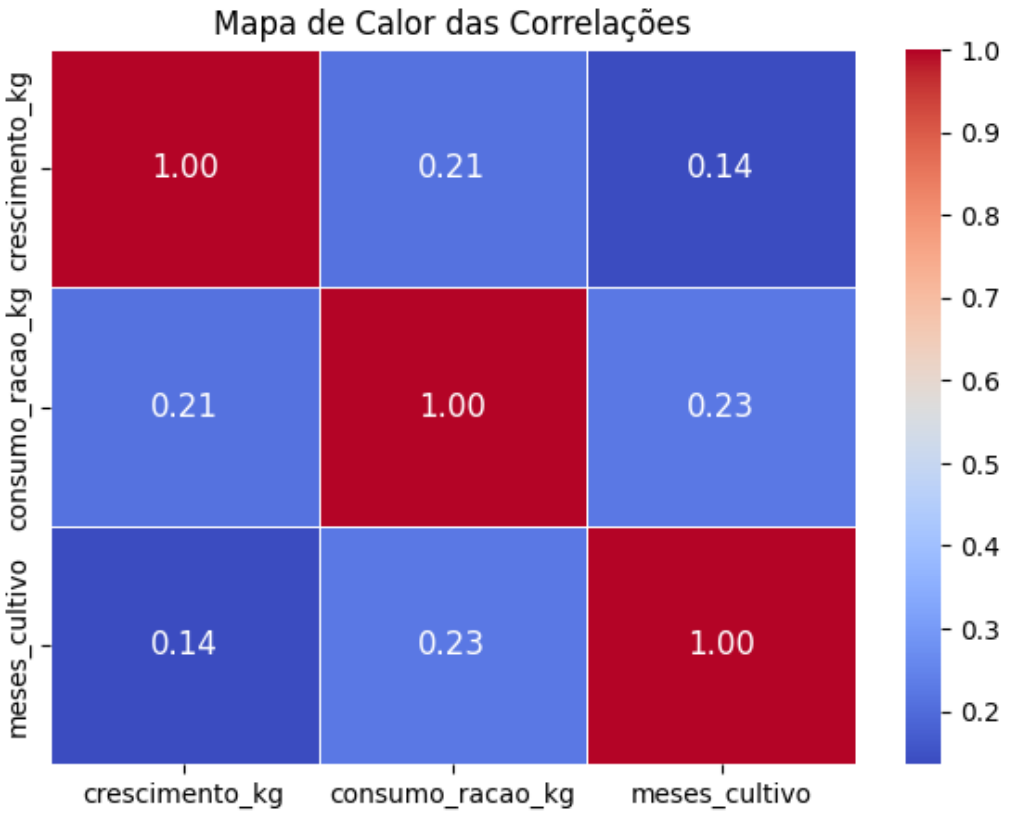

Figura 2.7: Mapa de calor da matriz de correlação.

Uma Visão Panorâmica

A matriz impressa e o mapa de calor nos dão uma visão rápida de todas as relações:

1. **Diagonal Principal:** A diagonal de (topo-esquerda para baixo-direita) é sempre 1.00 (vermelho escuro). Isso é óbvio, pois mostra a correlação de uma variável com ela mesma.
2. `crescimento_kg` vs. `consumo_racao_kg`**:** Vemos um valor fraco (0.21). Por quê? Porque o gráfico junta ambas as rações. A Ração A tem correlação forte (0.96) e a Ração B tem correlação fraca (0.21); O resultado geral é uma média "morna".
3. **Outras Relações:** Vemos que `meses_cultivo` tem uma correlação muito fraca com as outras variáveis.

O Mapa de Calor é a sua ferramenta de diagnóstico rápido para *datasets* complexos. Ele identifica quais variáveis parecem ter uma relação forte e que merecem uma investigação mais profunda com um Gráfico de Dispersão (como fizemos ao separar as Rações A e B).

2.2.7 Gráfico de Pizza

"Qual é a proporção ou porcentagem de cada categoria dentro de um todo?" Enquanto os gráficos de barras comparam os valores absolutos dos grupos, o Gráfico de Pizza (ou Gráfico de Setores) foca exclusivamente na composição percentual. Ele é usado para mostrar como um todo (100%) é dividido em partes. No nosso "Experimento do Peixe", temos a coluna `avaliacao_saude`. Uma pergunta interessante seria: *"Qual a proporção de peixes com saúde Alta, Média ou Baixa em nosso experimento completo?"*

Copie e Teste!

```
# Precisamos contar quantas vezes cada categoria aparece.
frequencia_saude = df_peixes['avaliacao_saude'].value_counts()

print("--- Tabela de Frequência (Dados para o Gráfico) ---")
print(frequencia_saude)
print("\n")

print("--- Gerando Gráfico de Pizza ---")
plt.figure(figsize=(8, 8))
plt.pie(frequencia_saude, labels=frequencia_saude.index, autopct='
   %1.1f%%', startangle=90)
plt.title('Proporção da Avaliação de Saúde dos Peixes')
plt.axis('equal')
plt.show()
```

Tabela 2.5: Tabela de Frequência da Avaliação de Saúde (Dados para o Gráfico).

Avaliação de Saúde	Frequência
Média	13
Alta	10
Baixa	7

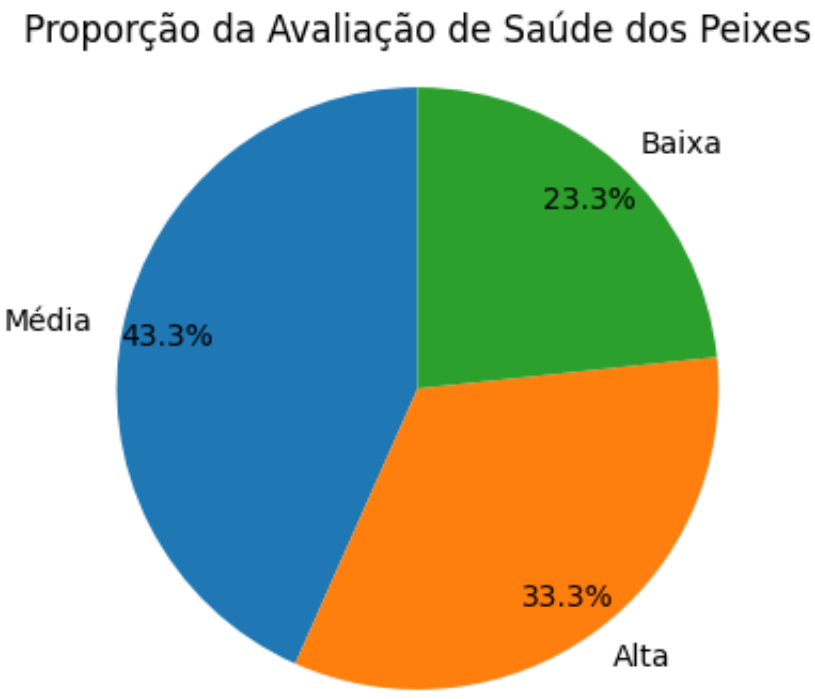

Figura 2.8: Gráfico de pizza da proporção da avaliação de saúde dos peixes.

Análise: Vendo as "Fatias"

O gráfico nos mostra instantaneamente a composição do nosso *dataset*. No nosso exemplo, a fatia Média (43.3%) é claramente a maior, seguida por Alta (33.3%) e Baixa (23.3%).

Fique Alerta!

O Perigo do Gráfico de Pizza
Embora popular, o Gráfico de Pizza deve ser usado com extremo cuidado. O cérebro humano tem muita dificuldade em comparar o tamanho de ângulos e áreas (fatias). **Evite o Gráfico de Pizza quando:**

1. Você tem muitas categorias (mais de 5 ou 6), pois o gráfico se torna um "arco-íris" ilegível.
2. As categorias têm valores muito próximos, assim fica quase impossível dizer visualmente se uma fatia de 23% é maior que uma de 25%.

Na dúvida, use um Gráfico de Barras. Um gráfico de barras (como o que vimos no tópico 2.2.1) mostraria as mesmas contagens de saúde de forma muito mais clara e fácil de comparar.

2.3 Estatística Descritiva

Nas seções anteriores, aprendemos a visualizar dados (com gráficos). Agora, vamos adicionar a segunda ferramenta da exploração: os números que resumem os dados. A estatística descritiva é o conjunto de técnicas que usamos para descrever e resumir as principais características de um conjunto de dados. Antes de construir um modelo complexo, precisamos primeiro entender o básico.

2.3.1 Tipos de Variáveis

O primeiro passo de *qualquer* análise é identificar os tipos de variáveis que temos. A escolha de qual gráfico usar (Seção 2.2) ou qual cálculo fazer (média, contagem, etc.) depende 100% dessa identificação. Em estatística, não estamos tão preocupados com os tipos de dados do computador (`int`, `float`, `str`). Estamos interessados nos tipos conceituais. Todas as variáveis se dividem em duas grandes famílias:

1. Variáveis Qualitativas (Categóricas)

Descrevem uma qualidade, um tipo ou uma categoria. Elas não podem ser somadas ou subtraídas. Elas se dividem em:

- **Nominal:** Categorias que não possuem uma ordem natural.
  - Exemplos: `tipo_racao` ("Ração A", "Ração B"), `especie` ("Tambaqui", "Pirarucu").
  - Análise Típica: Contagem de frequência (Tabela de Frequências) e Gráfico de Barras/Pizza.

- **Ordinal:** Categorias que possuem uma ordem ou hierarquia lógica.
  - Exemplos: `avaliacao_saude` ("Baixa", "Média", "Alta"), `tamanho` ("Pequeno", "Médio", "Grande").
  - Análise Típica: Tabela de Frequências, Mediana (para encontrar a categoria central) e Gráfico de Barras.

2. Variáveis Quantitativas (Numéricas)

Descrevem uma quantidade ou um número. São dados que podemos usar em operações matemáticas. Elas se dividem em:

- **Discreta:** Números que podem ser contados. Geralmente são inteiros e não existem valores "entre" eles.
  - Exemplos: `meses_cultivo` (6, 7, 8), `quantidade_de_peixes` (10, 11, 12).
  - Análise Típica: Média, Mediana, Gráfico de Barras ou Gráfico de Linha.
- **Contínua:** Números que podem ser medidos. Eles podem assumir qualquer valor dentro de um intervalo.
  - Exemplos: `crescimento_kg` (5.04 kg, 5.05 kg, 5.051 kg), `consumo_racao_kg`, `temperatura`.
  - Análise Típica: Média, Mediana, Desvio Padrão, Histograma, Boxplot, Dispersão.

Vamos rodar o comando `.info()` do Pandas para ver como o computador "enxerga" nossos dados e, ao lado, faremos nossa classificação estatística.

Copie e Teste!

```
print("--- Saída do df_peixes.info() ---")
df_peixes.info()
```

Tela do Terminal

```
--- Saída do df_peixes.info() ---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 6 columns):
# Column Non-Null Count Dtype
--- ------ -------------- -----
0 id_peixe 30 non-null int64
1 tipo_racao 30 non-null object
2 crescimento_kg 30 non-null float64
3 consumo_racao_kg 30 non-null float64
4 meses_cultivo 30 non-null int64
5 avaliacao_saude 30 non-null object
dtypes: float64(2), int64(2), object(2)
memory usage: 1.5+ KB
```

Análise: Classificação de Variáveis

Saber classificar suas variáveis é a habilidade número um da análise de dados. É ela que define todo o seu plano de ataque:

- Se a variável é **Qualitativa**, sua primeira pergunta será: *"Qual a frequência de cada categoria?"*
- Se a variável é **Quantitativa**, sua primeira pergunta será: *"Qual a tendência central e a dispersão?"*

Tabela 2.6: Classificação Estatística das Variáveis do Experimento.

Coluna	Tipo Pandas	Tipo Estatístico	Por Quê?
`id_peixe`	`int64`	Qualitativa Nominal	É um código. Não faz sentido calcular a "média" dos IDs.
`tipo_racao`	`object`	Qualitativa Nominal	É um rótulo de categoria sem ordem.
`crescimento_kg`	`float64`	Quantitativa Contínua	É um valor medido (peso).
`consumo_racao_kg`	`float64`	Quantitativa Contínua	É um valor medido (peso).
`meses_cultivo`	`int64`	Quantitativa Discreta	É um valor contado (número inteiro de meses).
`avaliacao_saude`	`object`	Qualitativa Ordinal	É uma categoria ("Baixa", "Média", "Alta") com ordem.

2.3.2 Descrevendo Variáveis Qualitativas: Tabela de Frequências

"Quantas vezes cada categoria aparece no meu dataset?" Para variáveis qualitativas (como `tipo_racao` ou `avaliacao_saude`), a primeira análise que fazemos é uma tabela de frequências. Ela simplesmente conta quantas observações pertencem a cada categoria. No Pandas, fazemos isso com o comando `.value_counts()`.

Copie e Teste!

```
# --- 1. Contagem de Frequência para 'tipo_racao' (Absoluta)
print("--- Frequência para 'tipo_racao' ---")
freq_racao = df_peixes['tipo_racao'].value_counts()
print(freq_racao)

# --- 2. Contagem de Frequência para 'avaliacao_saude' (Absoluta)
print("\n--- Frequência para 'avaliacao_saude' ---")
freq_saude = df_peixes['avaliacao_saude'].value_counts()
print(freq_saude)

# --- 3. Frequência em Porcentagem (Relativa)
# Passamos o parâmetro normalize=True
```

```
print("\n--- Frequência % para 'avaliacao_saude' ---")
freq_saude_pct = df_peixes['avaliacao_saude'].value_counts(
   normalize=True)
print(freq_saude_pct)
```

Tabela 2.7: Tabela de Frequência Completa para Avaliação de Saúde.

Classe	Freq. Absoluta	Freq. Rel. %	Freq. Abs. Acum.	Freq. Rel. Acum. %
Baixa	7	23.33	7	23.33
Média	13	43.33	20	66.67
Alta	10	33.33	30	100.00
Total	30	100.00	—	—

A Tabela de Frequências é o ponto de partida para descrever dados categóricos, inclusive contém os dados-fonte que usamos para construir Gráficos de Barras e Gráficos de Pizza. Ela nos mostra instantaneamente a composição do nosso *dataset*:

1. Os grupos de `tipo_racao` estão perfeitamente balanceados (15 em cada).
2. A maioria dos peixes (43.3%) teve uma avaliação "Média".

2.3.3 Descrevendo Variáveis Quantitativas: Tendência Central

"Qual é o valor 'típico' ou 'central' dos meus dados?" Para variáveis quantitativas (como `crescimento_kg`), não basta olhar o gráfico; buscamos um único número que resuma o "centro" da distribuição. O problema é que existem várias maneiras de definir o "centro". Vamos explorar as três principais: Média, Mediana e Moda.

Média Aritmética (Mean)

É a soma de todos os valores dividida pelo número total de observações. É a medida que todos aprendemos na escola como "a média".

Formalismo Matemático: A média amostral ($\bar{x}$) é:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i = \frac{x_1 + x_2 + ... + x_N}{N}$$

- **Intuição:** É o "ponto de equilíbrio" dos dados. Se você colocasse pesos iguais em uma régua em cada ponto de dado, a média seria o ponto onde a régua se equilibraria.
- **O Grande Perigo (Sensibilidade a *Outliers*):** A média é muito democrática: *todos* os pontos têm o mesmo peso. Isso significa que um único valor extremo (um *outlier*) pode "puxar" a média inteira em sua direção e distorcer o resultado.
- **Exemplo Prático (Salários):** Imagine uma pequena empresa com 5 funcionários.
  - **Salários:** [R$ 2.000, R$ 2.200, R$ 2.400, R$ 2.100, R$ 2.300].
  - **Média:** R$ 2.200. Esse valor parece ser um bom resumo do salário "típico".

  - **Cenário com *Outlier*:** Agora, o dono (que é funcionário) se paga R$ 50.000.
  - **Novos Salários:** [R$ 2.000, R$ 2.200, R$ 2.400, R$ 2.100, R$ 50.000].
  - **Nova Média:** R$ 11.740.

  Se você dissesse a um novo candidato que o "salário médio" é R$ 11.740, você estaria contando uma mentira estatística. O *outlier* distorceu completamente a realidade. É exatamente isso que suspeitamos que está acontecendo com a Ração B.

Mediana (Median)

A mediana é o valor que está exatamente no meio dos dados, depois que eles são ordenados. Metade dos dados (50%) está abaixo dela e metade (50%) está acima.

- **Intuição:** É o valor da "pessoa do meio". Se você enfileirar todos os seus pontos de dados do menor para o maior, a mediana é o valor daquele que está no centro da fila.
- **A Grande Vantagem (Robustez a *Outliers*):** A mediana não é sensível a *outliers*. Ela não se importa com o *quão* extremo é o maior ou o menor valor; ela só se importa com a *posição* central.
- **Exemplo Prático (Salários, de novo):** Vamos usar os mesmos salários com o *outlier*:
  - **Salários:** [R$ 2.000, R$ 2.200, R$ 2.400, R$ 2.100, R$ 50.000].
  - **1. Ordene os dados:** [R$ 2.000, R$ 2.100, R$ 2.200, R$ 2.400, R$ 50.000].
  - **2. Encontre o valor do meio:** A Mediana é R$ 2.200.

  Note como a Mediana (R$ 2.200) nos dá um resumo muito mais honesto e representativo do funcionário "típico" do que a Média (R$ 11.740).

Conhecendo um pouco mais!

E se o número de dados for par?
No exemplo dos salários, tínhamos 5 funcionários (ímpar), então foi fácil achar "o do meio". Se tivéssemos 6 funcionários com salários ordenados [R$ 2.000, R$ 2.100, R$ 2.200, R$ 2.400, R$ 2.500, R$ 50.000], não haveria um único valor central. Nesse caso, a regra é pegar os dois valores centrais (R$ 2.200 e R$ 2.400) e calcular a média deles. A mediana seria:

$$\left(\frac{2200 + 2400}{2}\right) = \text{R\$ } 2.300$$

Moda (Mode)

A moda é simplesmente o valor que mais aparece (o mais frequente) no conjunto de dados, ou seja, é o valor "mais popular". É a principal medida de tendência central para dados qualitativos (como vimos na Seção 2.3.2). Para dados quantitativos contínuos (como `crescimento_kg`), ela é raramente usada, pois a chance de dois peixes terem exatamente o mesmo peso (ex: 5.01234 kg) é muito pequena.

- **Exemplo Prático:** O dono de uma sapataria quer saber qual o "tamanho central" de sapato feminino que ele deve ter em estoque.

- A **Média** dos tamanhos vendidos é inútil, pois não existe sapato tamanho 37.83.
- A **Mediana** é melhor, pois 38 diz o tamanho que divide o mercado ao meio.
- A **Moda** é a informação mais acionável, pois 37 diz qual foi o "tamanho mais vendido".

Agora, vamos aplicar as duas medidas mais relevantes (Média e Mediana) ao nosso "Experimento do Peixe".

Copie e Teste!

```
# --- 1. Vamos calcular as métricas para a Ração A (Simétrica) ---
print("--- Análise da Ração A (Simétrica) ---")
crescimento_A = df_peixes[df_peixes['tipo_racao'] == 'Ração A']['
    crescimento_kg']
media_A = crescimento_A.mean()
mediana_A = crescimento_A.median()
print(f"Média:   {media_A:.2f} kg")
print(f"Mediana: {mediana_A:.2f} kg")

# --- 2. Vamos calcular as métricas para a Ração B (Assimétrica,
    com Outlier) ---
print("\n--- Análise da Ração B (Com Outlier) ---")
crescimento_B = df_peixes[df_peixes['tipo_racao'] == 'Ração B']['
    crescimento_kg']
media_B = crescimento_B.mean()
mediana_B = crescimento_B.median()
print(f"Média:   {media_B:.2f} kg  <-(Influenciada pelo outlier)")
print(f"Mediana: {mediana_B:.2f} kg  <-(Robusta ao outlier)")
```

Tela do Terminal

```
--- Análise da Ração A (Simétrica) ---
Média: 5.01 kg
Mediana: 4.93 kg

--- Análise da Ração B (Com Outlier) ---
Média: 5.33 kg <-(Influenciada pelo outlier)
Mediana: 4.18 kg <-(Robusta ao outlier)
```

Análise: Média vs. Mediana

Aqui está o momento "aha!" da nossa análise, que confirma o que vimos nos gráficos:

- **Ração A (Simétrica):** Note como a Média (5.01 kg) e a Mediana (4.93 kg) são praticamente idênticas. Isso é um comportamento típico de dados bem distribuídos e simétricos, como o "sino" que vimos no histograma da Ração A.
- **Ração B (Assimétrica):** Aqui a história é oposta. O *outlier* de 25.0 kg "puxou" a Média para 5.33 kg, fazendo-a parecer falsamente melhor que a Ração A. No entanto, a **Mediana**,

que é robusta e "ignora" o valor extremo, nos conta a verdade: o peixe "do meio" da Ração B cresceu apenas 4.18 kg.

Isso prova o que os gráficos nos mostraram: o peixe "típico" da Ração B (Mediana 4.18 kg) na verdade cresceu *menos* que o peixe típico da Ração A (Mediana 4.93 kg).

Fique Alerta!

Quando usar qual?

- Use a **Média** quando os dados forem simétricos (Ração A).
- Use a **Mediana** quando os dados forem assimétricos ou tiverem *outliers* (Ração B).

2.3.4 Descrevendo Variáveis Quantitativas: Dispersão

"Ok, eu sei o centro (média/mediana), mas quão 'espalhados' ou 'consistentes' são os meus dados?" Saber o centro não é suficiente. Precisamos de medidas que nos digam o quão dispersos os dados estão.

Amplitude

É a medida mais simples: a diferença entre o maior e o menor valor.

$$\text{Amplitude} = \text{Valor Máximo} - \text{Valor Mínimo}$$

Variância (s^2) e Desvio Padrão (s)

Estas são as medidas de dispersão mais importantes. Elas nos dizem, em média, o quão longe cada ponto de dado está da média do grupo.

- **Variância (s^2):** É a média dos quadrados das diferenças de cada valor para a média. É difícil de interpretar (ex: kg^2).
- **Desvio Padrão (s):** É a raiz quadrada da variância. Este é o número que usamos na prática, pois ele retorna à unidade original (ex: kg).

Um desvio padrão baixo (próximo de 0) significa que os dados estão muito concentrados em torno da média (consistente). Um desvio padrão alto significa que os dados estão muito espalhados (inconsistente).

Formalismo Matemático: O Desvio Padrão Amostral (s) é calculado como:

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

Copie e Teste!

```
# --- 1. Análise de Dispersão (Ração A - Simétrica) ---
print("--- Análise de Dispersão (Ração A) ---")
crescimento_A = df_peixes[df_peixes['tipo_racao'] == 'Ração A']['
    crescimento_kg']
```

```
amplitude_A = crescimento_A.max() - crescimento_A.min()
std_A = crescimento_A.std() # std = Desvio Padrão
print(f"Amplitude: {amplitude_A:.2f} kg")
print(f"Desvio Padrão: {std_A:.2f} kg")

# --- 2. Análise de Dispersão (Ração B - Com Outlier) ---
print("\n--- Análise de Dispersão (Ração B) ---")
crescimento_B = df_peixes[df_peixes['tipo_racao'] == 'Ração B']['
    crescimento_kg']
amplitude_B = crescimento_B.max() - crescimento_B.min()
std_B = crescimento_B.std()
print(f"Amplitude: {amplitude_B:.2f} kg  <-- (Inflacionada pelo
    outlier)")
print(f"Desvio Padrão: {std_B:.2f} kg  <-- (Inflacionado pelo
    outlier)")
```

Tela do Terminal

```
--- Análise de Dispersão (Ração A) ---
Amplitude: 1.75 kg
Desvio Padrão: 0.50 kg

--- Análise de Dispersão (Ração B) ---
Amplitude: 23.74 kg <-- (Inflacionada pelo outlier)
Desvio Padrão: 5.63 kg <-- (Inflacionado pelo outlier)
```

Análise: Avaliando a Dispersão

- **Ração A:** Tem um desvio padrão de apenas 0.50 kg. Isso significa que a maioria dos peixes teve um crescimento muito próximo da média (5.01 kg). É uma ração consistente e previsível.
- **Ração B:** Tem um desvio padrão de 5.63 kg (mais de 10x maior!). Isso significa que os dados estão *extremamente* espalhados. A Amplitude de 23.74 kg confirma que os resultados são caóticos. É uma ração inconsistente e imprevisível.

Juntando tudo (Tendência Central + Dispersão), podemos afirmar: **"Embora a média da Ração B parecesse maior (devido a um *outlier*), a mediana da Ração A é melhor, e seu desvio padrão é muito menor. Portanto, a Ração A é a escolha mais segura e consistente."**

2.4 Correlação

Até agora, nossa estatística descritiva focou em uma variável de cada vez (análise univariada). Mas as perguntas mais ricas da Ciência de Dados estão nas relações entre variáveis (análise bivariada). *"O consumo de ração influencia o crescimento? O tempo de cultivo afeta a avaliação de saúde?"* A correlação é uma medida estatística que descreve a força e a direção de uma relação linear entre duas variáveis quantitativas.

2.4.1 Gráfico de Dispersão e Coeficiente

Como vimos na seção de visualização 2.2.5, o Gráfico de Dispersão (Scatter Plot) é a nossa ferramenta visual para investigar a correlação. Ele nos permite "ver" o padrão. Para quantificar esse padrão, usamos o Coeficiente de Correlação de Pearson (r). Este coeficiente é um número que varia de -1.0 a +1.0.

- **+1.0:** Correlação positiva perfeita. (Quando X sobe, Y sobe na mesma proporção).
- **0.0:** Nenhuma correlação linear. (As variáveis não têm relação linear).
- **-1.0:** Correlação negativa perfeita. (Quando X sobe, Y desce na mesma proporção).

Formalismo Matemático: O Coeficiente de Correlação de Pearson (r) de uma amostra é calculado dividindo-se a covariância das duas variáveis (x e y) pelo produto de seus desvios padrão (s_x e s_y):

$$r = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}}$$

Felizmente, o Pandas faz todo esse cálculo para nós com o método `.corr()`. Vamos revisitar o código que usamos para gerar a matriz de correlação e o mapa de calor.

Copie e Teste!

```
# --- 1. Calcular a Matriz de Correlação Geral ---
# Selecionamos apenas as colunas numéricas
colunas_numericas = ['crescimento_kg', 'consumo_racao_kg', '
   meses_cultivo']
df_numerico = df_peixes[colunas_numericas]
matriz_corr = df_numerico.corr()
print("--- Matriz de Correlação (Geral) ---")
print(matriz_corr)

# --- 2. Visualizando a Matriz com Heatmap ---
plt.figure(figsize=(8, 6))
sns.heatmap(matriz_corr, annot=True, cmap='coolwarm', fmt='.2f',
   vmin=-1, vmax=1)
plt.title('Mapa de Calor das Correlações (Dataset Completo)')
plt.show()

# --- 3. Calcular Correlação Separada por Ração ---
corr_A = df_peixes[df_peixes['tipo_racao'] == 'Ração A']['
   consumo_racao_kg'].corr(df_peixes[df_peixes['tipo_racao'] == '
   Ração A']['crescimento_kg'])

corr_B = df_peixes[df_peixes['tipo_racao'] == 'Ração B']['
   consumo_racao_kg'].corr(df_peixes[df_peixes['tipo_racao'] == '
   Ração B']['crescimento_kg'])

print(f"\nCorrelação (r) Ração A: {corr_A:.4f} (Forte)")
print(f"Correlação (r) Ração B: {corr_B:.4f} (Fraca/Nenhuma)")
```

Tabela 2.8: Matriz de Correlação (Geral).

	crescimento_kg	consumo_racao_kg	meses_cultivo
crescimento_kg	1.00	0.21	0.14
consumo_racao_kg	0.21	1.00	0.23
meses_cultivo	0.14	0.23	1.00

Tabela 2.9: Correlação (Crescimento vs. Consumo) por Tipo de Ração.

Ração	Correlação (r)	Classificação
Ração A	0.96	Forte
Ração B	0.21	Fraca/Nenhuma

Análise: Avaliando a Dispersão

Os números confirmam o que vimos nos gráficos de dispersão na seção 2.2.5.

- **Ração A:** Apresenta um $r = +0.96$. Esta é uma correlação positiva muito forte. A conclusão é clara: para a Ração A, o consumo está diretamente ligado ao crescimento.
- **Ração B:** Apresenta um $r = +0.21$. Este valor é fraco, indicando baixa correlação linear.

O que nos leva ao alerta estatístico mais importante de todos!

2.4.2 Correlação NÃO implica Causalidade

Este é, talvez, o conceito mais importante deste capítulo. Você *precisa* entender a diferença entre correlação e causalidade para não tirar conclusões erradas.

- **Correlação** significa apenas que duas variáveis se movem juntas.
- **Causalidade** significa que a mudança em uma variável causa a mudança na outra.

Encontrar uma correlação forte **NÃO** prova que uma variável causa a outra. Quase sempre, existe uma "variável oculta" (ou *variável de confusão*) que explica a relação.

Fique Alerta!

Exemplo Clássico: Sorvete e Ataques de Tubarão
Se analisarmos dados de uma cidade de praia, encontraremos uma correlação positiva muito forte entre o número de sorvetes vendidos e o número de ataques de tubarão.

- **Conclusão Ruim (Causalidade):** "Vender sorvete *causa* ataques de tubarão! Vamos proibir a venda de sorvetes para salvar os banhistas." ("Isso é absurdo").
- **Conclusão Correta (Correlação):** Existe uma variável oculta: a temperatura (calor).

1. Quando faz mais calor, *mais pessoas* vão à praia e compram sorvete.
2. Quando faz mais calor, *mais pessoas* entram na água, aumentando a probabilidade de um encontro com um tubarão.

A venda de sorvetes e os ataques de tubarão não têm relação causal. Ambos são causados por uma terceira variável (o calor).

Em nosso experimento, encontramos uma correlação forte entre consumo e crescimento na Ração A. Isso sugere que o consumo causa o crescimento (o que é biologicamente plausível). Mas, como cientistas de dados, nosso trabalho é dizer: "Encontramos uma forte relação linear". A afirmação de causa só pode vir de um experimento controlado e do conhecimento de especialistas (biólogos, neste caso).

2.5 Aplicando seus conhecimentos

1. **Visualização:** Você viu na Seção 2.2.1 que um Gráfico de Barras, por mostrar apenas a média, pode ser enganoso. Se você quisesse investigar a consistência (dispersão) e os *outliers* do crescimento da Ração A vs. Ração B em um único gráfico, qual gráfico da Seção 2.2 você usaria e por quê?

2. **Média vs. Mediana:** No "Experimento do Peixe", a Média da Ração B (5.33 kg) foi maior que a da Ração A (5.01 kg), mas a Mediana da Ração B (4.18 kg) foi muito menor. Explique, em suas palavras, por que isso aconteceu e qual métrica (Média ou Mediana) você confiaria mais para descrever o peixe "típico" da Ração B?

3. **Desvio Padrão:** A Ração A teve um Desvio Padrão de 0.50 kg, enquanto a Ração B teve um de 5.63 kg (Seção 2.3.4). O que esse número (Desvio Padrão) nos diz sobre a *previsibilidade* do crescimento de cada ração?

4. **Correlação vs. Causalidade:** Você analisa dados de uma cidade e encontra uma correlação positiva muito forte ($r = +0.90$) entre o número de sorvetes vendidos e o número de afogamentos na praia. O seu colega diz: "Isso prova que tomar sorvete *causa* afogamentos!". Usando o conceito da Seção 2.4.2, qual é a "variável oculta" que provavelmente explica essa relação?

2.6 Considerações deste Capítulo

Neste capítulo, mergulhamos na primeira metade da análise: a Exploração e Descrição. Vimos, com o "Experimento do Peixe", a importância de *visualizar* os dados (com Gráficos de Barra, Histogramas e Boxplots) antes de confiar cegamente em um único número. Aprendemos a usar a Estatística Descritiva (Média, Mediana e, crucialmente, Desvio Padrão) para resumir e comparar grupos, entendendo como o Pandas e o Seaborn são nossas ferramentas para executar esses cálculos e visualizações.

Por fim, investigamos a Correlação, aprendendo a quantificá-la com o Coeficiente de Pearson e internalizando a regra mais importante da análise de dados: correlação não implica causalidade. Agora que sabemos como descrever o que aconteceu no passado, estamos prontos para o próximo passo. O próximo capítulo nos ensinará a lidar com a incerteza e a decidir o que fazer a seguir, usando o poder da Probabilidade e da Inferência Estatística.

Capítulo 3

Da Incerteza à Decisão: Probabilidade e Inferência

Iniciando o diálogo...

No capítulo anterior, aprendemos a "olhar para o passado". Usamos a Estatística Descritiva e a Visualização para resumir e entender os dados que já coletamos. Vimos que a Ração A parecia mais consistente que a Ração B. Mas isso descreve apenas aqueles 30 peixes. Como podemos usar essa informação para tomar uma decisão futura? Como podemos ter certeza de que a diferença que vimos não foi "apenas sorte"?
Para responder a isso, precisamos de um novo conjunto de ferramentas. Precisamos da linguagem matemática para quantificar a incerteza e da estrutura formal para tomar decisões baseadas em evidências. Este é o mundo da Probabilidade e da Estatística Inferencial.

3.1 Probabilidade

Até agora, tratamos nossos dados como fatos concretos. Nós calculamos a média exata do `crescimento_kg` (5.01 kg para a Ração A) e a correlação exata ($r = 0.96$). Esse é o mundo da Estatística Descritiva: resumir o que já aconteceu. Mas a Ciência de Dados raramente se contenta em apenas descrever o passado. Nosso verdadeiro objetivo é usar o passado para tomar decisões sobre o futuro.

- Qual a chance do próximo peixe alimentado com Ração A crescer mais de 5kg?
- Qual a probabilidade de a Ração A ser realmente melhor que a Ração B, ou será que o resultado que vimos foi apenas sorte?

Para responder a essas perguntas, precisamos de uma nova linguagem: a Probabilidade. Ela é a ferramenta matemática que nos permite quantificar a incerteza e gerenciar o risco.

3.1.1 Variáveis Aleatórias

O primeiro conceito que precisamos é o de Variável Aleatória. Uma variável aleatória não é como uma variável normal de programação (como `x = 10`). É um conceito que descreve um processo ou evento futuro cujo resultado é incerto, mas que pertence a um conjunto de possibilidades.

- **Exemplo Discreto:** O resultado de jogar um dado. A variável aleatória "Dado" pode assumir os valores {1, 2, 3, 4, 5, 6}.
- **Exemplo Contínuo:** A temperatura de amanhã. A variável aleatória "Temperatura" pode assumir qualquer valor dentro de um intervalo (ex: 20°C, 20.1°C, 20.11°C...).

É aqui que tudo se conecta. Em Ciência de Dados, tratamos nossas colunas de dados como observações passadas de uma variável aleatória.

- A coluna `crescimento_kg` do nosso `df_peixes` é uma amostra de 30 observações da variável aleatória "Crescimento de um Peixe".
- Ao analisar a distribuição dessa coluna (com um histograma), estamos, na verdade, tentando estimar a "forma" da distribuição de probabilidade dessa variável aleatória.

Entender isso é o que nos permite usar os dados que temos para fazer previsões sobre os dados que não temos.

3.1.2 Distribuições Contínuas

No último tópico, vimos que existem variáveis aleatórias discretas (como um dado, com 6 resultados possíveis) e contínuas (como o `crescimento_kg` de um peixe, que pode ser 5.0, 5.1, 5.001, etc.). Descrever a probabilidade de um dado é fácil: a chance de sair "4" é 1/6. Mas como descrevemos a probabilidade de uma variável contínua? Qual é a chance de um peixe crescer exatamente 5.00000000... kg? A resposta é zero. Existe um número infinito de resultados possíveis, então a chance de acertar um valor exato é nula. Para variáveis contínuas, nunca perguntamos a probabilidade de um ponto, mas sim a probabilidade de um intervalo.

- Qual a chance do peixe crescer entre 4.5 kg e 5.5 kg?
- Qual a chance do pH da água ser maior que 7.0?

Para responder a isso, usamos uma Função Densidade de Probabilidade, ou *Probability Density Function*. A Função Densidade de Probabilidade é uma curva que descreve a "forma" da nossa variável aleatória. Ela tem duas regras principais:

1. Ela nunca pode ser negativa (não existe chance negativa).
2. A área total sob a curva inteira é exatamente 1 (representando 100% de chance de algum resultado acontecer).

A probabilidade de um evento acontecer dentro de um intervalo é, então, a área sob a curva dentro daquele intervalo. A probabilidade de a variável X cair entre os valores a e b é a integral (a área) da função de densidade $f(x)$ de a até b:

$$P(a \leq X \leq b) = \int_a^b f(x)dx$$

O Histograma que plotamos na seção 2.2.3 é a nossa melhor tentativa de "desenhar" a Função Densidade de Probabilidade da nossa variável `crescimento_kg`, usando nossos dados de amostra. A linha suave que o Seaborn desenhou (quando usamos `kde=True`) é uma estimativa estatística dessa curva $f(x)$. Ela nos mostrou que a Função Densidade de Probabilidade da "Ração A" se parece muito com uma forma específica, famosa e incrivelmente importante na estatística: a Distribuição Normal.

3.1.3 A Distribuição Normal

"Existe um 'padrão natural' para a distribuição de dados contínuos?" A Distribuição Normal (ou Curva de Sino / Curva de Gauss) é a distribuição de probabilidade mais importante. Muitos fenômenos do mundo real (altura de pessoas, erros de medição, e o crescimento dos peixes da Ração A) tendem a seguir esta distribuição.

Propriedades da Distribuição Normal

Uma Distribuição Normal é simétrica e definida por apenas dois parâmetros:

1. **Média (μ):** O centro exato da curva (onde está o pico). Na normal, a Média, a Mediana e a Moda são o mesmo valor.
2. **Desvio Padrão (σ):** A "largura" da curva. Um desvio padrão pequeno (como o da Ração A) resulta em uma curva alta e estreita.

Vamos plotar o histograma da Ração A novamente, mas desta vez sobrepondo a curva de uma Distribuição Normal "perfeita" que usa a média e o desvio padrão que calculamos.

Copie e Teste!

```
# --- 1. Preparar os dados da Ração A ---
crescimento_A = df_peixes[df_peixes['tipo_racao'] == 'Ração A']['
    crescimento_kg']

# --- 2. Calcular os parâmetros (Média e Desvio Padrão) ---
media_A = crescimento_A.mean()
std_A = crescimento_A.std()

print(f"--- Parâmetros da Ração A ---")
print(f"Média μ(): {media_A:.2f}")
print(f"Desvio Padrão σ(): {std_A:.2f}")

# --- 3. Plotar o Histograma e a Curva Normal Teórica ---
print("\n--- Gerando Gráfico de Comparação ---")
plt.figure(figsize=(10, 6))

# Plotar o Histograma dos nossos dados (stat='density' normaliza a
    área para 1)
sns.histplot(crescimento_A, kde=True, stat='density', label='Dados
    Reais (Histograma)')

# Gerar uma curva normal "perfeita" com a mesma média e desvio
    padrão
x = np.linspace(media_A - 3*std_A, media_A + 3*std_A, 100)
curva_normal = stats.norm.pdf(x, media_A, std_A)

# Plotar a curva teórica por cima
plt.plot(x, curva_normal, color='red', lw=3, linestyle='--', label
    ='Distribuição Normal Teórica')
```

```
# --- 4. Customização ---
plt.title('Distribuição da Ração A vs. Curva Normal')
plt.xlabel('Crescimento (kg)')
plt.ylabel('Densidade de Probabilidade')
plt.legend()
plt.show()
```

Tela do Terminal

```
--- Parâmetros da Ração A ---
Média (μ): 5.01
Desvio Padrão (σ): 0.50
```

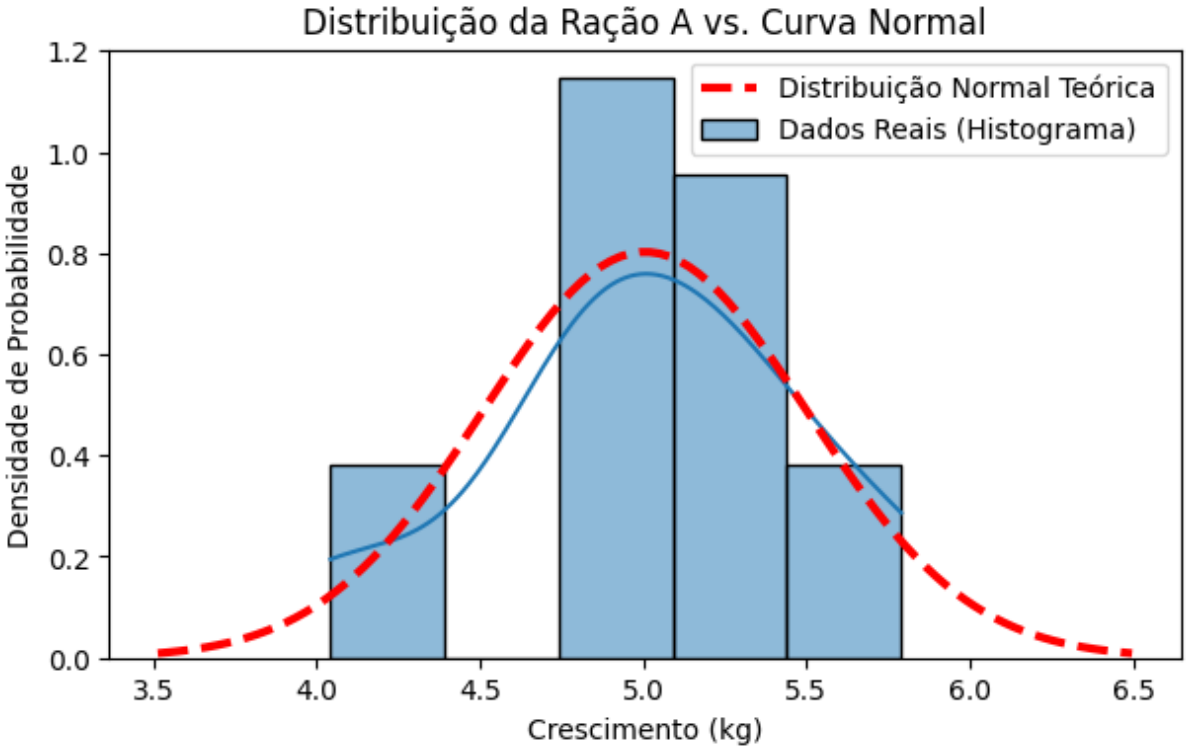

Figura 3.1: Comparação da distribuição da Ração A com a curva normal teórica.

Análise: O Desafio da Inferência com Dados Não-Normais

Como o gráfico mostra, nossos dados da Ração A seguem quase perfeitamente uma Distribuição Normal, o que nos permite usar todo o poder preditivo dessa distribuição.

- Mas e a Ração B? Ela não era normal por causa do *outlier*.
- Isso é um problema?
- E se nossos dados nunca forem normais?
- Como podemos fazer inferências?

A resposta é o conceito mais importante da estatística.

3.1.4 O Teorema do Limite Central (TLC)

"O que eu faço se meus dados não seguem uma Distribuição Normal?" Esta é a resposta para o nosso problema com a Ração B. O Teorema do Limite Central (TLC) afirma algo extraordinário, não importa o quão estranha ou assimétrica seja a distribuição dos seus dados originais (pode ser a Ração B, pode ser qualquer coisa). Se você tiver amostras suficientemente grandes (geralmente n > 30) desses dados, calcular a média de cada amostra, e depois plotar um histograma dessas médias, o resultado será aproximadamente uma Distribuição Normal.

Em outras palavras, os dados individuais podem ser assimétricos, mas a distribuição das médias das amostras ($\bar{x}$) sempre tenderá a ser normal. Vamos provar isso. Vamos criar uma população de dados totalmente "bagunçada" (usando uma distribuição exponencial) e ver o que acontece com as médias de suas amostras.

Copie e Teste!

```
# --- 1. Criar uma População "Mãe" Assimétrica ---
# Usaremos uma distribuição exponencial (muito assimétrica)
populacao_mae = np.random.exponential(scale=2, size=100_000)

# Plotar os dados originais
plt.figure(figsize=(10, 4))
plt.subplot(1, 2, 1) # Gráfico da esquerda
sns.histplot(populacao_mae, kde=True, bins=100)
plt.title('1. Distribuição da População "Mãe"\n(Assimétrica)')
plt.xlabel('Valor')
plt.ylabel('Frequência')

# --- 2. Simular a Coleta de Amostras e Calcular suas Médias ---
# Vamos simular 10.000 coletas, cada uma com 30 observações (n=30)
tamanho_amostra = 30
numero_de_amostras = 10_000
medias_das_amostras = []

for i in range(numero_de_amostras):
# Pega uma amostra aleatória de 30 valores
amostra = np.random.choice(populacao_mae, size=tamanho_amostra)
# Calcula a média da amostra e guarda
medias_das_amostras.append(np.mean(amostra))

# --- 3. Plotar o Histograma DAS MÉDIAS ---
plt.subplot(1, 2, 2) # Gráfico da direita
sns.histplot(medias_das_amostras, kde=True, bins=50, color='green'
   )
plt.title('2. Distribuição das MÉDIAS das Amostras\n(Mágica: É
   Normal!)')
plt.xlabel('Média da Amostra')
plt.ylabel('Frequência')

plt.tight_layout()
plt.show()
```

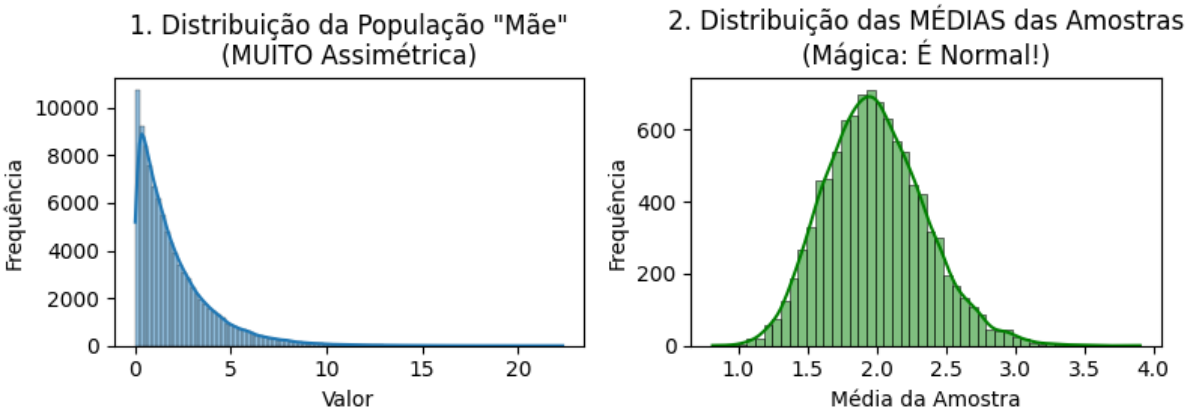

Figura 3.2: Ilustração do Teorema do Limite Central (TLC).

Análise: A Mágica do TLC

Acabamos de provar o Teorema do Limite Central.

1. Começamos com dados originais (Gráfico 1) que não tinham nada a ver com uma Distribuição Normal.

2. Mas a distribuição das médias de suas amostras (Gráfico 2) é perfeitamente normal.

Por que isso é tão importante? Porque o TLC nos dá luz verde para usar testes estatísticos (como o Teste T) que se baseiam na Distribuição Normal. Quando comparamos a média(Ração A) vs. média(Ração B), não estamos testando os dados individuais, estamos testando as médias. E graças ao TLC, sabemos que essas médias (assumindo N > 30) se comportam de maneira normal e previsível.

3.1.5 Dependência e Independência

"O resultado de um evento afeta a probabilidade de outro evento acontecer?"

- **Eventos Independentes:** O resultado de um **não afeta** a probabilidade do outro. (Ex: Jogar um dado duas vezes).
- **Eventos Dependentes:** O resultado de um **afeta** a probabilidade do outro. (Ex: Tirar uma carta de um baralho e *não* a colocar de volta).

No nosso experimento, nossa hipótese central é que o `crescimento_kg` depende do `tipo_racao`. Se os eventos fossem independentes, significaria que a ração não faz a menor diferença.

3.1.6 Probabilidade Condicional

"Qual é a probabilidade do evento A acontecer, dado que o evento B já aconteceu?" Este conceito nos permite atualizar nossas crenças à medida que recebemos novas informações. A probabilidade é denotada como $P(A|B)$, ou "a probabilidade de A, dado B".

Formalismo Matemático:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Onde:

- $P(A|B)$: A chance de A acontecer, sabendo que B aconteceu.
- $P(A \cap B)$: A chance de ambos A e B acontecerem juntos.
- $P(B)$: A chance total de B acontecer.

Caso Prático

Vamos calcular a Probabilidade Condicional no "Experimento do Peixe".

1. $P(A)$ = Probabilidade de um peixe qualquer ter saúde "Alta".
2. $P(A|B)$ = Probabilidade de um peixe ter saúde "Alta", dado que ele comeu "Ração A".

Copie e Teste!

```
# 1. Calcular a Probabilidade Total (P(Alta))
total_peixes = len(df_peixes)
peixes_saude_alta = df_peixes[df_peixes['avaliacao_saude'] == '
    Alta'].shape[0]
prob_alta_total = peixes_saude_alta / total_peixes
print(f"Probabilidade total de saúde 'Alta' (P(Alta)): {
    prob_alta_total:.2%}")

# 2. Calcular a Probabilidade Condicional P(Alta | Ração A)
# P(A|B) = P(A e B) / P(B)
# P(B): Probabilidade de ser Ração A
total_racao_A = df_peixes[df_peixes['tipo_racao'] == 'Ração A'].
    shape[0]
prob_racao_A = total_racao_A / total_peixes

# P(A e B): Probabilidade de ser 'Alta' E 'Ração A'
peixes_alta_e_A = df_peixes[(df_peixes['avaliacao_saude'] == 'Alta
    ') & (df_peixes['tipo_racao'] == 'Ração A')].shape[0]
prob_alta_e_A = peixes_alta_e_A / total_peixes

# Finalmente, P(A|B)
prob_alta_dado_A = prob_alta_e_A / prob_racao_A
print(f"Prob. de 'Alta' DADO 'Ração A' (P(Alta|Ração A)): {
    prob_alta_dado_A:.2%}")

# 3. A Maneira Intuitiva (e mais fácil) de Calcular P(A|B)
# "Reduzimos o universo" primeiro, filtrando apenas pela Ração A
df_racao_A = df_peixes[df_peixes['tipo_racao'] == 'Ração A']

# Calculamos a probabilidade de 'Alta' DENTRO desse novo universo
```

```
prob_alta_dado_A_intuitiva = df_racao_A[df_racao_A['
    avaliacao_saude'] == 'Alta'].shape[0] / len(df_racao_A)
print(f"Cálculo intuitivo P(Alta|Ração A): {
    prob_alta_dado_A_intuitiva:.2%}")
```

Tela do Terminal

```
Probabilidade total de saúde Alta - (P(Alta)): 33.33%
Prob. de Alta DADO Ração A - (P(Alta|Ração A)): 20.00%
Cálculo intuitivo - P(Alta|Ração A): 20.00%
```

A análise mostra que a probabilidade muda!

- A chance de um peixe aleatório ter saúde "Alta" é de 33.33%.
- Mas, no momento em que sabemos que ele comeu a "Ração A", nossa estimativa de probabilidade cai para 20.00%.

Isso prova que os eventos são dependentes. A nova informação (`tipo_racao`) alterou a probabilidade do resultado (`avaliacao_saude`).

3.1.7 Teorema de Bayes

"Eu observei um efeito. Qual a probabilidade de que ele tenha vindo de uma causa específica?" Na seção anterior, respondemos $P(A|B)$ ("Qual a chance de Alta, dado Ração A?"). O Teorema de Bayes nos permite inverter essa pergunta para $P(B|A)$: "Eu observei um peixe com saúde Alta. Qual a chance de ele ter vindo da Ração A?"

Formalismo Matemático:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Vamos traduzir:

- $P(A|B)$ (Posterior): O que queremos saber $\rightarrow P(\text{Ração A}|\text{Saúde Alta})$
- $P(B|A)$ (Likelihood): O que calculamos antes $\rightarrow P(\text{Saúde Alta}|\text{Ração A})$
- $P(A)$ (Prior): Nossa crença inicial $\rightarrow P(\text{Ração A})$
- $P(B)$ (Evidência): A probabilidade total da evidência $\rightarrow P(\text{Saúde Alta})$

Copie e Teste!

```
# --- 1. Coletar todas as probabilidades necessárias ---
# (Calculadas no bloco anterior e no value_counts)
prob_A = 0.50             # P(Ração A) - Nossos grupos são
    balanceados (15/30)
prob_B = 0.3333           # P(Saúde Alta) - (10/30)
prob_B_dado_A = 0.2000    # P(Saúde Alta | Ração A) - (3/15)
```

```
# --- 2. Aplicar o Teorema de Bayes ---
# P(A|B) = (P(B|A) * P(A)) / P(B)
prob_A_dado_B = (prob_B_dado_A * prob_A) / prob_B

print(f"--- Probabilidades Base (Inputs) ---")
print(f"P(Ração A) [Prior]:              {prob_A:.2%}")
print(f"P(Saúde Alta) [Evidência]:       {prob_B:.2%}")
print(f"P(Saúde Alta | Ração A)[Likelihood]: {prob_B_dado_A:.2%}")

print(f"\n--- Resultado (Posterior) ---")
print(f"P(Ração A | Saúde Alta): {prob_A_dado_B:.2%}")
```

Tela do Terminal

```
--- Probabilidades Base (Inputs) ---
P(Ração A) [Prior]: 50.00%
P(Saúde Alta) [Evidência]: 33.33%
P(Saúde Alta | Ração A)[Likelihood]: 20.00%

--- Resultado (Posterior) ---
P(Ração A | Saúde Alta): 30.00%
```

Análise: Aprendendo com a Evidência

- **Crença Inicial (Prior):** Antes de vermos o peixe, nossa chance de ele ser da "Ração A" era de 50%.
- **Nova Evidência:** Observamos que o peixe tem "Saúde Alta".
- **Crença Atualizada (Posterior):** Usando o Teorema de Bayes, atualizamos nossa crença. A probabilidade de o peixe ser da "Ração A" caiu de 50% para 30.00%.

Isso aconteceu porque, no nosso *dataset* (que foi gerado aleatoriamente), a Ração A produziu menos peixes com saúde "Alta" (20%) do que a média geral (33.33%). O Teorema de Bayes é a lógica matemática formal para "aprender com as evidências".

3.2 Estatística Inferencial

Nas seções anteriores, ficamos muito bons em descrever nossos dados. Sabemos que no nosso `df_peixes` de 30 peixes:

- A média de crescimento da Ração A foi 5.01 kg.
- A média de crescimento da Ração B (com *outlier*) foi 5.33 kg.
- A mediana da Ração B (sem *outlier*) foi 4.18 kg.

Mas isso descreve apenas o passado. Descreve apenas aqueles 30 peixes específicos. O tomador de decisão (nosso piscicultor) não quer saber o que aconteceu com aqueles 30 peixes. Ele quer saber o que fazer no futuro. A pergunta dele é: *"Para os próximos 10.000 peixes que vou criar, qual ração devo comprar?"* Para responder a isso, precisamos de um novo conjunto de ferramentas. Precisamos da Estatística Inferencial.

3.2.1 Amostra vs. População

Este é o conceito fundamental que separa a Estatística Descritiva da Inferencial.

- **População (O que queremos saber):** É o universo inteiro de interesse. São todos os peixes que poderiam ser alimentados com a Ração A no futuro. A média de crescimento desta população é o que realmente importa (chamamos de μ, a "média verdadeira"), mas é impossível de medir.
- **Amostra (O que temos):** É o subconjunto de dados que conseguimos coletar e analisar. Os nossos 15 peixes da Ração A são uma amostra. A média que calculamos (5.01 kg) é a média da amostra ($\bar{x}$).

A Inferência Estatística é o processo de usar os dados da nossa amostra (ex: $\bar{x} = 5.01$ kg) para fazer uma afirmação educada e com um nível de confiança sobre a população inteira (μ). A média da nossa amostra da Ração A foi 5.01 kg. A média (sem *outlier*) da Ração B foi 3.92 kg. A Ração A parece melhor. Mas será que é?

E se, por puro azar, nós pegamos os 15 melhores peixes para a Ração A e os 14 piores para a Ração B? Se repetíssemos o experimento, será que a Ração B não poderia se sair melhor? A Estatística Inferencial não elimina essa incerteza, mas nos dá as ferramentas para quantificá-la. *"Qual é a probabilidade de que a diferença que vimos (5.01 vs 3.92) seja real e não apenas um acaso da amostragem?"* Para responder a isso, usamos um framework formal chamado Teste Estatístico de Hipótese.

3.2.2 Teste Estatístico de Hipótese

"Como eu posso usar a probabilidade para tomar uma decisão sim ou não sobre meus dados?" O Teste de Hipótese é o "tribunal" da estatística. É um procedimento formal para tomarmos uma decisão entre duas afirmações concorrentes. Em um tribunal, o réu é "inocente até que se prove o contrário". Na estatística, a ideia é a mesma. Nós não tentamos provar que nossa teoria é verdadeira; nós tentamos provar que a "versão chata" do mundo (a de que "nada aconteceu") é falsa. Para fazer isso, sempre formulamos duas hipóteses opostas:

1. A Hipótese Nula (H_0)

- **O que é:** É a "hipótese do status quo", a afirmação cética, a suposição de "inocência". Ela sempre afirma que não há efeito ou não há diferença.
- **No Tribunal:** "O réu é inocente."
- **No Nosso Experimento:** "Não há diferença nenhuma entre a Ração A e a Ração B. A diferença que vimos foi apenas um acaso da amostragem."
- **Formalismo Matemático:** $H_0 : \mu_A = \mu_B$ (A média verdadeira da Ração A é igual à média verdadeira da Ração B).

2. A Hipótese Alternativa (H_a ou H_1)

- **O que é:** É a nossa "teoria". Ela afirma que existe um efeito real.
- **No Tribunal:** "O réu é culpado."
- **No Nosso Experimento:** "A Ração A é realmente melhor que a Ração B. A diferença que vimos é muito grande para ser só sorte."
- **Formalismo Matemático:** $H_a : \mu_A > \mu_B$ (A média verdadeira da Ração A é maior que a da Ração B).

A Lógica do Teste:

1. Nós assumimos que a Hipótese Nula (H_0) é verdadeira. (Assumimos que as rações são iguais).
2. Então, olhamos para os nossos dados (nossa "evidência").
3. Calculamos a probabilidade de ter obtido dados tão "extremos" (uma diferença tão grande) quanto os que coletamos, assumindo que H_0 é verdadeira.
4. Se essa probabilidade for muito, muito baixa, nós dizemos: "É improvável que isso tenha sido apenas sorte." E, nesse caso, nós rejeitamos a Hipótese Nula (H_0).

3.2.3 A Moeda Viciada

Vamos aplicar o framework do "tribunal estatístico". Você desconfia que a moeda de um colega é viciada para dar mais "Cara" do que "Coroa". Você joga a moeda 10 vezes. E você obtém 9 Caras e 1 Coroa. Vamos levar esta evidência ao tribunal:

1. Formulando as Hipóteses

- **Hipótese Nula (H_0):** "A moeda é justa. $P(\text{Cara}) = 0.5$. Os 9 Caras que vimos foram apenas um acaso extremo."
- **Hipótese Alternativa (H_a):** "A moeda é viciada para Cara. $P(\text{Cara}) > 0.5$. Os 9 Caras são a evidência."

2. O Processo Lógico

Nós assumimos que H_0 é verdadeira (a moeda é justa). A pergunta que a estatística faz é: *"Se a moeda é realmente justa, qual é a probabilidade de, em 10 lançamentos, obtermos um resultado tão extremo ou mais extremo do que 9 Caras?"*

- **Se a probabilidade for ALTA** (ex: 20%): Nós diríamos: *"É bem possível que tenha sido sorte. Nós falhamos em rejeitar a H_0."*
- **Se a probabilidade for MUITO BAIXA** (ex: 1%): Nós diríamos: *"A chance de isso acontecer por acaso é muito baixa. É mais provável que nossa suposição (a moeda é justa) esteja errada."* Nós rejeitamos a H_0.

Para sua informação, a probabilidade de obter 9 ou 10 caras em 10 lançamentos de uma moeda justa é de aproximadamente 1.07%.

3. Conclusão

Como 1.07% é uma probabilidade muito baixa, nós teríamos forte evidência estatística para rejeitar a Hipótese Nula e concluir que a moeda é, de fato, viciada. É exatamente isso que faremos com nossos peixes. Vamos calcular a probabilidade de ver a diferença de médias que vimos (5.01 vs 3.92) assumindo que as rações são iguais. Se essa probabilidade for muito baixa, rejeitaremos a H_0. Essa "probabilidade muito baixa" que usamos para tomar a decisão tem um nome especial.

3.2.4 P-value

No exemplo da moeda, chegamos à probabilidade de 1.07%. Esse número é o *p-value*, a probabilidade de obter uma evidência (dados) tão extrema, ou mais extrema, do que a que realmente observamos, assumindo que a Hipótese Nula (H_0) é verdadeira.

- **Em português simples:** "O p-value é a probabilidade de ser apenas sorte."
- **No exemplo da moeda:** $p - value = 1.07\%$. "Se a moeda for justa, a chance de você ter essa sorte de tirar 9 ou 10 Caras é de apenas 1.07%."

O Limiar da Decisão: Nível de Significância (α)

"Quão baixo" é "baixo o suficiente"? Para evitar subjetividade, a comunidade científica define um nível de significância (chamado de alpha, ou α) antes do experimento. O valor mais comum é $\alpha = 0.05$ (5%). Isso se torna nossa regra de decisão:

1. Se o p-value $\leq \alpha$ (ex: $0.01 \leq 0.05$): O resultado é estatisticamente significativo. A probabilidade de ser "apenas sorte" é tão baixa que nós Rejeitamos a Hipótese Nula (H_0).
2. Se o p-value > α (ex: 0.61 > 0.05): O resultado não é estatisticamente significativo. A probabilidade de ser "apenas sorte" é alta. Nós Falhamos em Rejeitar a Hipótese Nula (H_0).

Fique Alerta!

Este é o erro mais comum. O p-value **NÃO É** a "probabilidade da Hipótese Nula ser verdadeira".

- Um p-value de 0.01 **NÃO SIGNIFICA** que há 1% de chance da H_0 ser verdadeira.
- Ele é apenas a probabilidade da nossa evidência (nossos dados), assumindo que a H_0 já é verdadeira.

Vamos revisitar o tribunal: O p-value **NÃO** é a "chance do réu ser inocente". O p-value **É** a "chance da promotoria apresentar uma evidência TÃO forte, SE o réu for inocente". Se a chance for minúscula, o júri rejeita a "inocência" (H_0).

O Veredito do "Experimento do Peixe"

Vamos rodar o Teste T (a ferramenta estatística para comparar duas médias) para nossos dois cenários, usando $\alpha = 0.05$.

Copie e Teste!

```
# --- 1. Separar os grupos ---
grupo_A = df_peixes[df_peixes['tipo_racao'] == 'Ração A']['
    crescimento_kg']
grupo_B = df_peixes[df_peixes['tipo_racao'] == 'Ração B']['
    crescimento_kg']

# --- 2. Cenário 1: Teste T com o OUTLIER ---
# Usamos equal_var=False (Teste T de Welch), pois as dispersões
    são diferentes.
t_stat_com_outlier, p_value_com_outlier = stats.ttest_ind(grupo_A,
     grupo_B, equal_var=False)
print(f"--- Cenário 1 (Com Outlier) ---")
print(f"Valor-p (p-value): {p_value_com_outlier:.4f}")
if p_value_com_outlier <= 0.05:
    print("Veredito: Rejeitamos H0. As médias são estatisticamente
    diferentes.")
else:
    print("Veredito: Falhamos em Rejeitar H0. As médias NÃO são
    diferentes.")

print("\n" + "="*30 + "\n")

# --- 3. Cenário 2: Teste T SEM o OUTLIER ---
grupo_B_sem_outlier = grupo_B[grupo_B < 20] # Filtra o outlier
t_stat_sem_outlier, p_value_sem_outlier = stats.ttest_ind(grupo_A,
     grupo_B_sem_outlier, equal_var=False)
print(f"--- Cenário 2 (Sem Outlier) ---")
print(f"Valor-p (p-value): {p_value_sem_outlier:.4f}")
if p_value_sem_outlier <= 0.05:
    print("Veredito: Rejeitamos H0. As médias são estatisticamente
    diferentes.")
else:
    print("Veredito: Falhamos em Rejeitar H0. As médias NÃO são
    diferentes.")
```

Tela do Terminal

```
--- Cenário 1 (Com Outlier) ---
Valor-p (p-value): 0.8292
Veredito: Falhamos em Rejeitar H0. As médias NÃO são
diferentes.

--- Cenário 2 (Sem Outlier) ---
Valor-p (p-value): 0.0204
Veredito: Rejeitamos H0. As médias são estatisticamente
diferentes.
```

O Poder de uma Decisão Baseada em Dados

1. No Cenário 1, o p-value (0.8292 ou 82.9%) foi altíssimo. Isso significa que, se as rações fossem iguais, teríamos 83% de chance de obter essa diferença por acaso. Não temos evidência nenhuma para rejeitar a H_0.

2. No Cenário 2 (sem o *outlier*), o p-value (0.0204 ou 2%) foi baixíssimo, muito menor que nosso α de 0.05. A chance de ser "apenas sorte" é de 2%. A evidência é forte. Rejeitamos a H_0 e declaramos que a diferença é estatisticamente significativa.

O p-value nos deu o veredito final do "tribunal". O p-value é ótimo para uma decisão "sim" ou "não", mas ele não nos diz o tamanho da diferença.

3.2.5 Intervalo de Confiança

"Ok, eu sei que a Ração A é melhor. Mas quão melhor? Ela é 0.1kg melhor ou 2kg melhor?" O p-value da seção anterior nos deu uma resposta de "Sim" ou "Não": "Sim, a diferença que vimos é estatisticamente significativa" (Cenário 2). Mas isso não nos diz o *tamanho* do efeito. Para um tomador de decisão, saber a magnitude da diferença é crucial. É aqui que entra o Intervalo de Confiança (IC).

Um Intervalo de Confiança é um intervalo de valores (uma "faixa de estimativa") que, com um certo nível de confiança (geralmente 95%), acreditamos conter o parâmetro verdadeiro da população (a média μ). A interpretação de um "Intervalo de Confiança de 95%" é:

> "Se nós repetíssemos nosso experimento 100 vezes e calculássemos 100 intervalos, esperaríamos que 95 desses 100 intervalos *capturassem* a verdadeira média da população (μ)."

Na prática, usamos isso como uma medida de precisão da nossa estimativa:

- Um IC estreito (ex: [4.9 kg, 5.1 kg]) é ótimo. Significa que temos uma estimativa muito precisa.
- Um IC largo (ex: [2.0 kg, 8.0 kg]) não é tão útil. Significa que há muita incerteza.

Qual o Crescimento Real de Cada Ração?

Vamos calcular o Intervalo de Confiança de 95% para a média de cada ração (usando o grupo B limpo).

Copie e Teste!

```
# --- 1. Preparar os grupos (Ração A e Ração B limpa) ---
grupo_A = df_peixes[df_peixes['tipo_racao'] == 'Ração A']['
    crescimento_kg']
grupo_B_sem_outlier = df_peixes.loc[(df_peixes['tipo_racao'] == '
    Ração B') & (df_peixes['crescimento_kg'] < 20),'crescimento_kg'
    ]

# --- 2. Calcular o IC de 95% para a Média da Ração A ---
confianca = 0.95
graus_liberdade_A = len(grupo_A) - 1
```

```
media_A = grupo_A.mean()
erro_padrao_A = stats.sem(grupo_A) # sem = Standard Error of the
    Mean
ic_A = stats.t.interval(confianca, graus_liberdade_A, loc=media_A,
     scale=erro_padrao_A)

print(f"--- Ração A ---")
print(f"Média da Amostra: {media_A:.2f} kg")
print(f"Intervalo de Confiança (95%): [{ic_A[0]:.2f}, {ic_A[1]:.2f
    }]")

# --- 3. Calcular o IC de 95% para a Média da Ração B (limpa) ---
graus_liberdade_B = len(grupo_B_sem_outlier) - 1
media_B = grupo_B_sem_outlier.mean()
erro_padrao_B = stats.sem(grupo_B_sem_outlier)
ic_B = stats.t.interval(confianca, graus_liberdade_B, loc=media_B,
     scale=erro_padrao_B)

print(f"\n--- Ração B (Sem Outlier) ---")
print(f"Média da Amostra: {media_B:.2f} kg")
print(f"Intervalo de Confiança (95%): [{ic_B[0]:.2f}, {ic_B[1]:.2f
    }]")
```

Tela do Terminal

```
--- Ração A ---
Média da Amostra: 5.01 kg
Intervalo de Confiança (95%): [4.73, 5.28]

--- Ração B (Sem Outlier) ---
Média da Amostra: 3.92 kg
Intervalo de Confiança (95%): [3.06, 4.79]
```

Análise: Quantificando o Tamanho da Diferença

Esta saída é a resposta final para o tomador de decisão.

- O p-value (da seção anterior) nos disse: **A diferença é real?** (Sim, $p < 0.05$).
- O Intervalo de Confiança (acima) nos diz: **O quão grande é a diferença?**

Podemos afirmar com 95% de confiança que o crescimento verdadeiro da Ração A está em algum lugar entre 4.73 kg e 5.28 kg. Também podemos afirmar com 95% de confiança que o crescimento verdadeiro da Ração B (limpa) está em algum lugar entre 3.06 kg e 4.79 kg.

Fique Alerta!

Note que os dois intervalos mal se sobrepõem. O valor mais baixo do IC da Ração A (4.73 kg) é praticamente igual ao valor mais alto do IC da Ração B (4.79 kg). Essa falta de sobreposição dos intervalos de confiança é a confirmação visual do que o p-value nos disse numericamente: **a diferença entre os grupos é real e não apenas sorte!**

3.2.6 O Perigo do P-Hacking

"Se eu não encontrar um p-value baixo, posso 'ajustar' meus dados até encontrar um?" No nosso experimento, tínhamos uma justificativa honesta para remover o *outlier*. Mas, e se o p-value original fosse 0.06, muito próximo do nosso α de 0.05, mas ainda "falhando" o teste?

Você poderia ficar tentado a "ajudar" os dados: *"E se eu remover os 2 peixes com crescimento mais baixo?"* ou *"E se eu só analisar os peixes do* `mes_cultivo == 8`*?"* Se você rodar o Teste T repetidamente, mudando os filtros, uma hora, por pura sorte, você vai encontrar uma combinação que lhe dê um p-value < 0.05. Isso é P-Hacking (ou "torturar os dados até que eles confessem").

Fique Alerta!

Por que isso é tão perigoso e antiético?
O Teste de Hipótese só é válido se você definir sua hipótese e sua amostra **ANTES** de olhar os dados. Ao "caçar" um p-value baixo, você está quebrando a regra fundamental da estatística. Você está apenas medindo o "ruído" e a "sorte", e não um efeito real. Você está enganando a si mesmo e ao seu cliente. A remoção de *outliers* é válida, mas deve ser feita com transparência e uma justificativa forte, e não com o objetivo de forçar um resultado. Um p-value de 0.83 não é um "fracasso"; é uma descoberta que nos diz: "Com os dados que temos, não podemos concluir que uma ração é melhor que a outra."

3.2.7 O Teste A/B

"Eu tenho duas versões de algo (A e B). Qual delas tem o melhor desempenho na prática?" O Teste A/B não é uma nova técnica estatística. É o nome de mercado para a aplicação de um Teste de Hipótese (geralmente um Teste T) para comparar duas versões de um produto. O "Experimento do Peixe" que analisamos é um Teste A/B clássico.

- Uma empresa testa duas cores de botão "Comprar" (Versão A: verde, Versão B: azul).
- Um piscicultor testa duas rações (Ração A vs. Ração B).

O Fluxo de Trabalho de um Teste A/B

Vamos rodar o teste final, como um cientista de dados faria para apresentar uma recomendação.

1. **Definir as Hipóteses:**
   - H_0: As médias de crescimento das rações são iguais. ($\mu_A = \mu_B$)
   - H_a: As médias de crescimento são diferentes. ($\mu_A \neq \mu_B$)
2. **Definir o Nível de Significância (α):** $\alpha = 0.05$.
3. **Coletar Dados e Executar o Teste:** Usaremos os dados limpos (sem o *outlier*).

Copie e Teste!

```
# --- 1. Preparar os grupos para o Teste A/B ---
# Vamos usar os dados limpos (sem o outlier)
```

```
grupo_A = df_peixes[df_peixes['tipo_racao'] == 'Ração A']['
    crescimento_kg']
grupo_B_limpo = df_peixes[(df_peixes['tipo_racao'] == 'Ração B')
    &(df_peixes['crescimento_kg'] < 20)]['crescimento_kg']

print("--- Dados Prontos para o Teste A/B ---")
print(f"Média Ração A: {grupo_A.mean():.2f} kg (n={len(grupo_A)})"
    )
print(f"Média Ração B (limpa): {grupo_B_limpo.mean():.2f} kg (n={
    len(grupo_B_limpo)})")

# --- 2. Executar o Teste T ---
t_stat, p_value = stats.ttest_ind(grupo_A, grupo_B_limpo,
    equal_var=False)

print(f"\n--- Resultados do Teste ---")
print(f"Estatística T (t-statistic): {t_stat:.4f}")
print(f"Valor-p (p-value): {p_value:.4f}")
```

Tela do Terminal

```
--- Dados Prontos para o Teste A/B ---
Média Ração A: 5.01 kg (n=15)
Média Ração B (limpa): 3.92 kg (n=14)

--- Resultados do Teste ---
Estatística T (t-statistic): 2.5804
Valor-p (p-value): 0.0204
```

Análise: A Tomada de Decisão

1. **Interpretar o p-value:** Nosso p-value é 0.0204. Nosso α era 0.05.
2. **Decisão:** Como $0.0204 < 0.05$, o resultado é estatisticamente significativo.
3. **Veredito:** Nós Rejeitamos a Hipótese Nula (H_0).
4. **Tradução para o Negócio:** "A diferença que vimos no crescimento entre a Ração A (Média de 5.01 kg) e a Ração B (Média de 3.92 kg) não foi um acaso. Nossos testes mostram com alta confiança (98%) que a Ração A é estatisticamente superior."

3.2.8 A Inferência Bayesiana

"Eu vi uma nova evidência. Como isso atualiza minha crença na minha hipótese?" Tudo o que vimos até agora (p-values, ICs) pertence à escola de pensamento mais comum: a Inferência Frequentista. A lógica frequentista é um pouco contraintuitiva. Nós **NÃO** calculamos a probabilidade da hipótese. Nós calculamos a probabilidade dos nossos dados, assumindo que a hipótese é verdadeira (o p-value). A Inferência Bayesiana é uma escola de pensamento alternativa que nos permite responder à pergunta que o tomador de decisão realmente quer saber: *"Ok, mas qual é a probabilidade de que a Ração A seja realmente melhor que a Ração B?"*

A Lógica Bayesiana

Para um estatístico Bayesiano, a probabilidade é um grau de crença. O processo é um ciclo de atualização dessa crença usando o Teorema de Bayes (que vimos na seção 3.1.7):

1. Começamos com uma **Crença Inicial (Prior):** "Acho que há 50% de chance da Ração A ser melhor".
2. Coletamos **Evidências:** nossos dados.
3. Usamos o **Teorema de Bayes** para combinar o Prior com a Evidência.
4. O resultado é uma **Crença Atualizada (Posterior):** "Agora que vi os dados, tenho 98% de certeza de que a Ração A é melhor".

   Formalismo Matemático:

$$P(H_a|\text{Dados}) = \frac{P(\text{Dados}|H_a) \cdot P(H_a)}{P(\text{Dados})}$$

- **Vantagem:** A conclusão Bayesiana é intuitiva. Dizer "Há uma probabilidade de 98% de que a Ração A seja superior" é muito mais claro do que "Rejeitamos a hipótese nula com p=0.02".
- **Desvantagem:** Complexidade, a abordagem tem dois desafios:
  1. **Subjetividade do Prior:** Como definimos a "crença inicial" ($P(H_a)$)?
  2. **Complexidade Matemática:** Calcular $P(\text{Dados})$ é matematicamente extremamente difícil e, na prática, requer simulações computacionais pesadas (MCMC).

Para a maioria dos Testes A/B, a abordagem Frequentista (p-values) é a ferramenta "padrão": é rápida e universalmente entendida. É vital, no entanto, saber que a Inferência Bayesiana existe como uma alternativa poderosa para problemas mais complexos.

3.3 Aplicando seus conhecimentos

1. **Hipótese Nula e Alternativa:** Você é um cientista de dados em uma empresa de logística fluvial na Amazônia. A empresa testa um novo tipo de motor de barco (Motor B) que *promete* ser mais rápido que o motor atual (Motor A). Você coleta dados de 50 viagens de cada motor. Qual seria sua **Hipótese Nula (H_0)** e sua **Hipótese Alternativa (H_a)** para este teste?

2. **Interpretando o p-value:** Após rodar seu teste dos motores, você obtém um **p-value = 0.45**. Seu nível de significância (α) é 0.05. O que você decide (Rejeitar H_0 ou Falhar em Rejeitar H_0)? O que você diria ao seu gerente sobre o novo Motor B?

3. **Interpretando o p-value (Cenário 2):** Imagine que, em vez disso, você obteve um **p-value = 0.003**. O que você decide (Rejeitar H_0 ou Falhar em Rejeitar H_0)? O que você diria ao seu gerente?

4. **p-value vs. Intervalo de Confiança:** No Cenário 3 (p-value = 0.003), você rejeitou a H_0 e disse ao seu gerente que o Motor B é mais rápido. Ele então pergunta: "Ótimo! Mas quanto mais rápido? Ele economiza 5 minutos ou 5 horas?" Qual ferramenta estatística da Seção 3.2 você usaria para responder a essa pergunta sobre a magnitude da diferença?

3.4 Considerações deste Capítulo

Neste capítulo, construímos a ponte entre a descrição e a decisão. Começamos com a Probabilidade, a linguagem da incerteza, entendendo conceitos fundamentais como a Distribuição Normal e o poderoso Teorema do Limite Central (TLC). Entramos no "tribunal" da estatística: a Inferência. Aprendemos o fluxo de trabalho mais importante para um cientista de dados: o Teste de Hipótese. Vimos como formular a Hipótese Nula (H_0), usar o **p-value** para decidir se uma evidência é "apenas sorte" e, finalmente, usar Intervalos de Confiança para comunicar o tamanho de um efeito.

Completamos nossa análise do "Experimento do Peixe", provando estatisticamente (e não apenas visualmente) que a Ração A é, de fato, a escolha superior. Aprendemos a teoria e praticamos com um *dataset* fictício (os peixes). No próximo capítulo, vamos aplicar todo esse conhecimento em um cenário real, usando dados públicos para analisar o clima da Amazônia.

Capítulo 4

Estudo de Caso Real: Análise de Dados Climáticos e Agrícolas

Iniciando o diálogo...

Você já parou para pensar de onde vêm os dados sobre o clima? Quando assistimos à previsão do tempo ou vemos gráficos sobre mudanças climáticas, essas informações não surgem do nada. Elas são coletadas por diferentes fontes e analisadas para nos ajudar a entender padrões e fazer previsões mais precisas. Mas como isso acontece na prática? Neste capítulo, vamos aplicar tudo o que aprendemos. Usaremos os conceitos de Análise Exploratória e as ferramentas de limpeza para trabalhar com dados reais da Amazônia, focando em fontes climáticas e agrícolas.

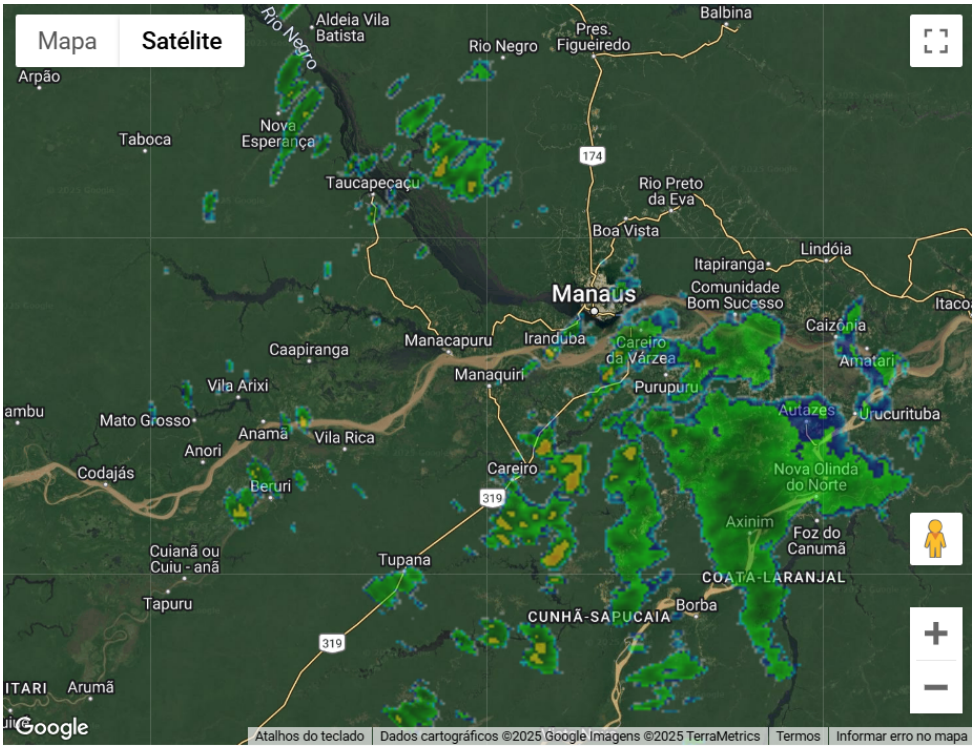

Figura 4.1: Imagem obtida em SOS Amazonia - SIPAM com a layer (camada) ativa: Fonte: Radar Manaus – SIPAM, data e hora 2025-02-17 15:48:00

Estações Meteorológicas e Radares:Já viu aquelas torres com sensores espalhados por cidades, rodovias ou áreas agrícolas? São as estações meteorológicas! Elas registram temperatura, umidade, velocidade do vento, pressão atmosférica e muito mais. No Brasil, o INMET (Instituto Nacional de Meteorologia) opera uma vasta rede de estações automáticas e convencionais, garantindo um fluxo contínuo de informações sobre o clima. Além das estações meteorológicas, os radares meteorológicos são essenciais para monitorar tempestades e padrões atmosféricos em tempo real. O SIPAM (Sistema de Proteção da Amazônia) utiliza radares para mapear chuvas intensas, ventos e outros fenômenos climáticos, ajudando a prever condições meteorológicas adversas e a planejar ações preventivas.

Figura 4.2: Exemplo de uma Estação Meteorológica Automática. Estes sensores em superfície são a principal fonte de dados locais de temperatura, umidade e vento.
Fonte: https://amazonasatual.com.br/estacao-meteorologica-em-manaus-emite-dados-a-cada-5-minutos/

Satélites Meteorológicos: Agora, imagine observar o clima de uma perspectiva bem mais ampla... do espaço! Satélites meteorológicos captam imagens e medições da atmosfera terrestre, ajudando a monitorar frentes frias, tempestades e até mesmo o desmatamento. Órgãos como o INPE (Instituto Nacional de Pesquisas Espaciais) fornecem imagens de satélite que auxiliam cientistas e governos a compreenderem mudanças climáticas em nível global. O oceano também influencia muito o clima, e para monitorá-lo, existem boias equipadas com sensores que coletam informações sobre temperatura da água, correntes marítimas e níveis de CO_2 (gás carbônico). Essas medições são essenciais para prever eventos como o El Niño e La Niña, que afetam o clima no mundo todo.

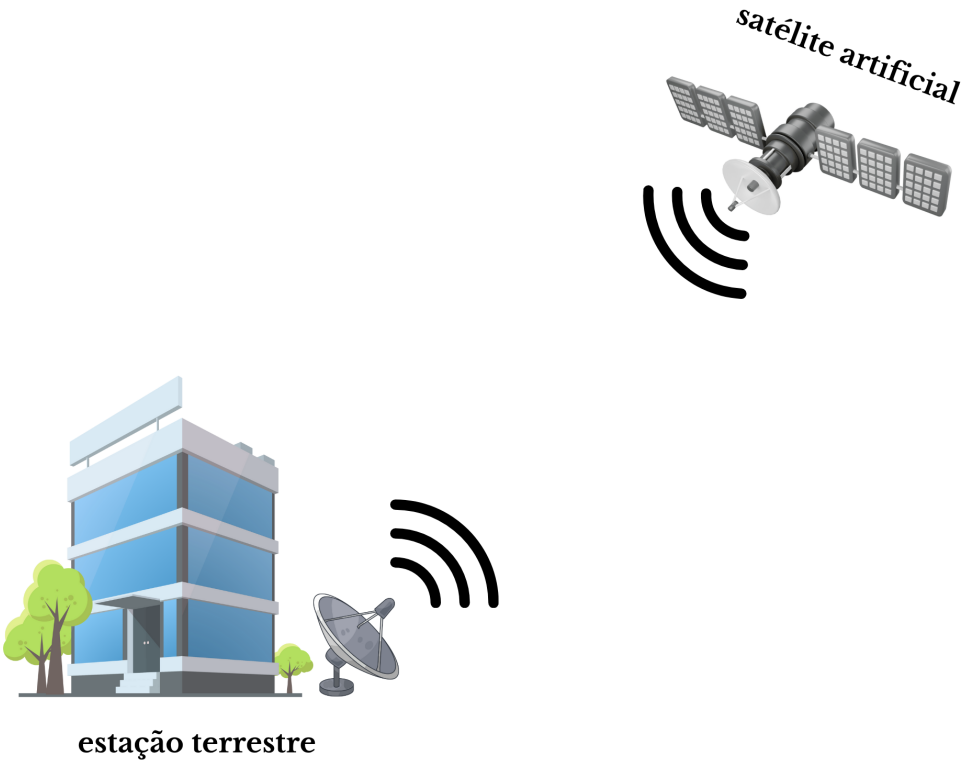

Figura 4.3: Representação artística de um satélite de observação da Terra.

E como esses dados chegam até nós? Muitos órgãos disponibilizam seus dados publicamente através de APIs (Interfaces de Programação de Aplicações) e bancos de dados online. Plataformas como o BDMEP do INMET oferecem acesso a registros históricos e informações em tempo real, permitindo que cientistas, empresas e até estudantes realizem suas próprias análises. E por que tudo isso é tão importante? Os dados climáticos ajudam a prever secas, chuvas intensas e outras variações meteorológicas que impactam desde a agricultura até a segurança da população. Com eles, podemos tomar decisões mais informadas, como planejar a irrigação de lavouras, evitar desperdícios de recursos naturais e até mitigar os impactos das mudanças climáticas. Agora que você já sabe de onde vêm os dados climáticos, que tal explorar algumas dessas fontes e começar a analisar padrões por conta própria?

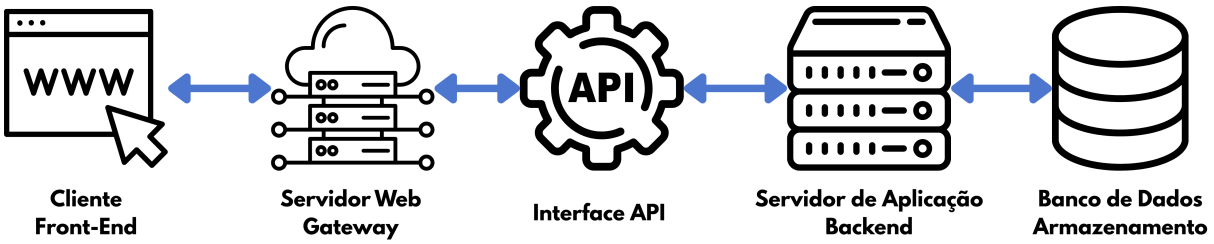

Figura 4.4: Diagrama de uma arquitetura web moderna, ilustrando como o Cliente se comunica com os recursos do Servidor de Aplicação e o Banco de Dados através de uma API.

Fique Alerta!

Para este estudo, vamos nos concentrar na prática com as estações automáticas, pois elas fornecem um volume maior de dados disponíveis, permitindo uma análise mais abrangente e detalhada.

4.1 Passo a Passo: Coletando Dados do INMET

Para acessar os dados meteorológicos, visite o site do INMET em https://portal.inmet.gov.br/, navegue até a seção "Dados Meteorológicos" e clique em "Banco de Dados Meteorológicos". Isso permitirá acesso ao portal em https://bdmep.inmet.gov.br, onde é possível consultar e baixar informações detalhadas sobre o clima. Ao entrar no Banco de Dados Meteorológicos, será necessário seguir algumas instruções, incluindo dois alertas importantes:

- Para baixar pacotes anuais de dados de todas as estações automáticas separadas por ano, pode-se clicar no link a seguir: https://portal.inmet.gov.br/dadoshistoricos.
- Para coleta de dados horários de curta duração (até 6 meses) de uma estação específica, utilizar a "Tabela de Dados de Estação" através do sistema TEMPO, acessível em https://-tempo.inmet.gov.br/TabelaEstacoes/A001, que nesse caso acessará a estação de Brasília com os dados atualizados do dia de hoje.
- E se desejar escolher uma estação específica, consultar a distribuição espacial das estações no "Mapa de Estações" disponível no endereço https://mapas.inmet.gov.br/.

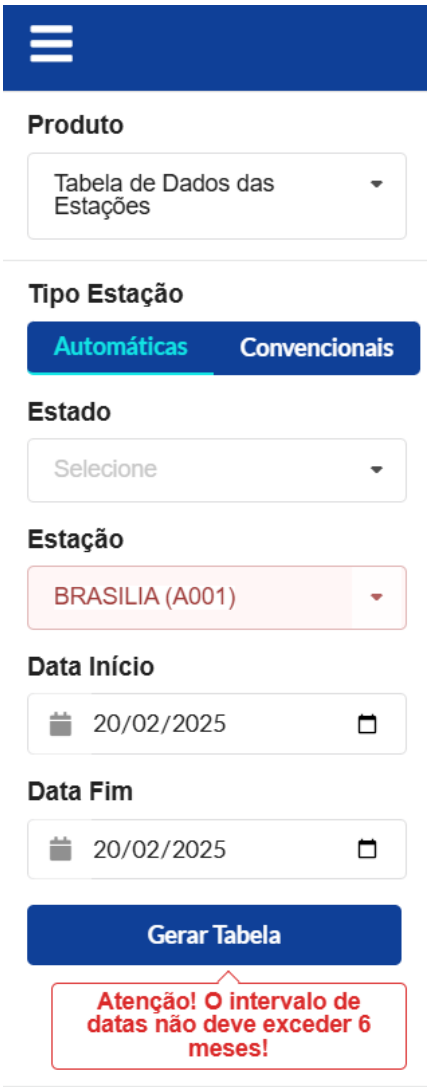

Figura 4.5: Interface do sistema TEMPO do INMET.

Ao acessar o sistema TEMPO no link https://tempo.inmet.gov.br/TabelaEstacoes/A001, você visualizará um menu. A partir dele, é possível:

- Selecionar o produto "Tabela de Dados das Estações".
- Escolher o tipo de estação (Automáticas ou Convencionais).
- Definir o Estado (UF) e a estação desejada.
- Especificar a data de início e a data de fim para a consulta.
- Clicar no botão "Gerar Tabela" para obter os dados.

É importante compreender que os dados exibidos na tela são aqueles disponíveis para filtragem na base de dados acessada. Essa interface facilita a visualização para usuários sem experiência com linguagens de banco de dados relacional como SQL. No entanto, para análises mais avançadas, é essencial ter conhecimento em Python para realizar filtragens de dados com base em critérios como data e hora.

No Capítulo 2, usamos o Pandas para limpar e agrupar dados. Aqui, faremos o mesmo. Os dados do INMET vêm em formato CSV, mas podem precisar de limpeza. O código a seguir usa `pd.read_csv` para carregar os dados.

Note, porém, algo crucial: para o computador, uma data como "20/02/2025" é apenas um texto (*string*). Para que o Python entenda que isso é um momento no tempo e nos permita filtrar por períodos (como "todos os dados de Fevereiro"), precisamos converter essa coluna de texto para um formato especial de *datetime*. Para isso, usamos a função `pd.to_datetime()`.

Copie e Teste!

```
import pandas as pd
# A biblioteca Pandas É fundamental para carregar e manipular
# dados de maneira eficiente.
# Ela permite a leitura de arquivos CSV e a aplicacao de filtros
# para analise de periodos especificos.
dados = pd.read_csv('dados_clima.csv', sep=';')

# Antes precisamos converter a coluna de data corretamente para a
# filtragem funcionar
dados["Data"] = pd.to_datetime(dados["Data"], dayfirst=True,
    errors='coerce')

# Filtrar os dados para um periodo especifico, nesse exemplo
# apenas no dia 20 de fevereiro de 2025,
# garantindo que apenas as informacoes desejadas sejam analisadas.
dados_filtrados = dados[(dados['Data'] >= '2025-02-20') & (dados['
    Data'] <= '2025-02-20')]

# Exibir as primeiras linhas do DataFrame para verificar se os
# dados foram carregados corretamente.
print(dados_filtrados.head())
```

No estado do Amazonas, destacamos algumas cidades que possuem monitoramento meteorológico automático, como Coari, Lábrea, Manacapuru, Manaus e São Gabriel da Cachoeira, que contam com estações do INMET.

- Coari: `https://tempo.inmet.gov.br/TabelaEstacoes/A117`
- Lábrea: `https://tempo.inmet.gov.br/TabelaEstacoes/A111`
- Manacapuru: `https://tempo.inmet.gov.br/TabelaEstacoes/A119`
- Manaus: `https://tempo.inmet.gov.br/TabelaEstacoes/A101`
- São Gabriel da Cachoeira: `https://tempo.inmet.gov.br/TabelaEstacoes/A134`

Em todos eles podem ser baixado o arquivo CSV, um formato de arquivo utilizado para armazenar dados estruturados de forma simples e eficiente. CSV significa "Comma-Separated Values" (Valores Separados por Vírgula), porém, dependendo da configuração regional, pode utilizar outros delimitadores, como ponto e vírgula (;). Abaixo temos um exemplo do conteúdo advindo dos dados exportados no formato .csv do sistema:

Tabela 4.1: Dados Meteorológicos (Tabela Transposta).

Parâmetro	**22/01/2025 (03:00)**	**22/01/2025 (04:00)**
Temp. Ins. (C)	25,2	25,1
Temp. Max. (C)	25,2	25,3
Temp. Min. (C)	25,1	25,1
Umi. Ins. (%)	90,0	91,0
Umi. Max. (%)	91,0	91,0
Umi. Min. (%)	90,0	89,0
Pto Orvalho Ins. (C)	23,4	23,5
Pto Orvalho Max. (C)	23,5	23,5
Pto Orvalho Min. (C)	23,3	23,2
Pressao Ins. (hPa)	1005,1	1004,8
Pressao Max. (hPa)	1005,1	1005,2
Pressao Min. (hPa)	1004,8	1004,8
Vel. Vento (m/s)	2,1	1,8
Dir. Vento (m/s)	85,0	76,0
Raj. Vento (m/s)	5,1	5,4
Radiacao (KJ/m²)	—	—
Chuva (mm)	0,0	0,0

O arquivo CSV bruto, como visto acima, não está pronto para análise. Esta é uma etapa clássica de **Limpeza de Dados** como visto no começo do livro, seção 1.2.3. Note que:

- Os números decimais usam vírgula ("25,2") e não ponto. O Python não entenderá isso como um número.
- Existem valores ausentes (células vazias, como em "Radiacao").
- Todas as colunas, exceto "Data" e "Hora", são lidas como texto (objeto) e precisam ser convertidas para números.

O script a seguir é uma rotina de limpeza completa. Ele carrega os dados, converte a coluna de data, e então itera por todas as colunas numéricas para trocar a "," por "." e convertê-las para um formato numérico. Finalmente, ele preenche os dados ausentes com a média da coluna, uma técnica comum de limpeza.

Copie e Teste!

```
import pandas as pd

# Carregar os dados do arquivo CSV
df = pd.read_csv('dados_clima.csv', sep=';', encoding='utf-8')

# Exibir os primeiros dados para checagem
print("\nDados brutos:")
print(df.head())

# Converter a coluna de data corretamente
df["Data"] = pd.to_datetime(df["Data"], dayfirst=True, errors='
   coerce')

# Lista de colunas numericas
colunas_numericas = [
    "Temp. Ins. (C)", "Temp. Max. (C)", "Temp. Min. (C)", "Umi.
   Ins. (%)","Umi. Max. (%)", "Umi. Min. (%)", "Pto Orvalho Ins. (
   C)","Pto Orvalho Max. (C)", "Pto Orvalho Min. (C)", "Pressao
   Ins. (hPa)","Pressao Max. (hPa)", "Pressao Min. (hPa)", "Vel.
   Vento (m/s)","Dir. Vento (m/s)", "Raj. Vento (m/s)", "Radiacao
   (KJ/m²)", "Chuva (mm)"
]

# Corrigir colunas numéricas substituindo ',' por '.'
for col in colunas_numericas:
    df[col] = df[col].astype(str).str.replace(',', '.')
    df[col] = pd.to_numeric(df[col], errors='coerce')

# Verificar tipos de dados
print("\nTipos de dados no DataFrame:")
print(df.dtypes)

# Estatisticas descritivas
print("\nEstatisticas basicas:")
print(df.describe())

# Identificar valores ausentes
print("\nValores ausentes por coluna:")
print(df.isnull().sum())

# Preenchendo valores ausentes com a media das colunas numericas
df.fillna(df.mean(numeric_only=True), inplace=True)
```

```
# Salvar os dados limpos em um novo arquivo CSV
df.to_csv('dados_clima_processados.csv', sep=';', index=False,
    encoding='utf-8')
# (É uma boa prática salvar o arquivo limpo em utf-8)

print("\nDados processados, salvo em dados_clima_processados.csv")
```

Fique Alerta!

Neste exemplo utilizamos os dados da estação de Manaus entre 01/01/2024 e 31/01/2024.

Com os dados limpos e processados, podemos iniciar a Análise Exploratória Visual. Como os dados climáticos são uma Série Temporal (dados medidos ao longo do tempo), o gráfico mais importante que aprendemos é o Gráfico de Linha. Ele nos permite ver a tendência e o comportamento das variáveis ao longo do período. O código a seguir usa o Seaborn (sns.lineplot) para plotar a variação de cada coluna numérica em nosso DataFrame limpo.

Copie e Teste!

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Carregar os dados do arquivo CSV JÁ PROCESSADO
df = pd.read_csv('dados_clima_processados.csv', sep=';', encoding=
    'utf-8', low_memory=False)

# Garantir que 'Data' seja datetime para o eixo X
df['Data'] = pd.to_datetime(df['Data'])

sns.set(style="whitegrid") # Definir o estilo aqui

# --- Gráfico 1: Figura 4.6 (Temperatura Instantânea) ---
plt.figure(figsize=(10, 5)) # Cria a FIGURA 1
sns.lineplot(data=df, x='Data', y='Temp. Ins. (C)')
plt.title('Temp. Ins. (C) ao Longo do Tempo')
plt.ylabel('Temp. Ins. (C)')
plt.xlabel('Data')
plt.xticks(rotation=45)
plt.tight_layout()

# --- Gráfico 2: Figura 4.7 (Temperatura Mínima) ---
plt.figure(figsize=(10, 5)) # Cria a FIGURA 2
sns.lineplot(data=df, x='Data', y='Temp. Min.(C)', color='orange')
plt.title('Temp. Min. (C) ao Longo do Tempo')
plt.ylabel('Temp. Min. (C)')
plt.xlabel('Data')
plt.xticks(rotation=45)
plt.tight_layout()
```

```
# --- Gráfico 3: Figura 4.8 (Temperatura Máxima) ---
plt.figure(figsize=(10, 5)) # Cria a FIGURA 3
sns.lineplot(data=df, x='Data', y='Temp. Max. (C)', color='red')
plt.title('Temp. Max. (C) ao Longo do Tempo')
plt.ylabel('Temp. Max. (C)')
plt.xlabel('Data')
plt.xticks(rotation=45)
plt.tight_layout()

# Mostrar os graficos
plt.show() # Exibe todas as 3 figuras
```

Os gráficos a seguir apresentam a temperatura (instantânea, mínima e máxima) da cidade de Manaus durante o período de janeiro de 2025.

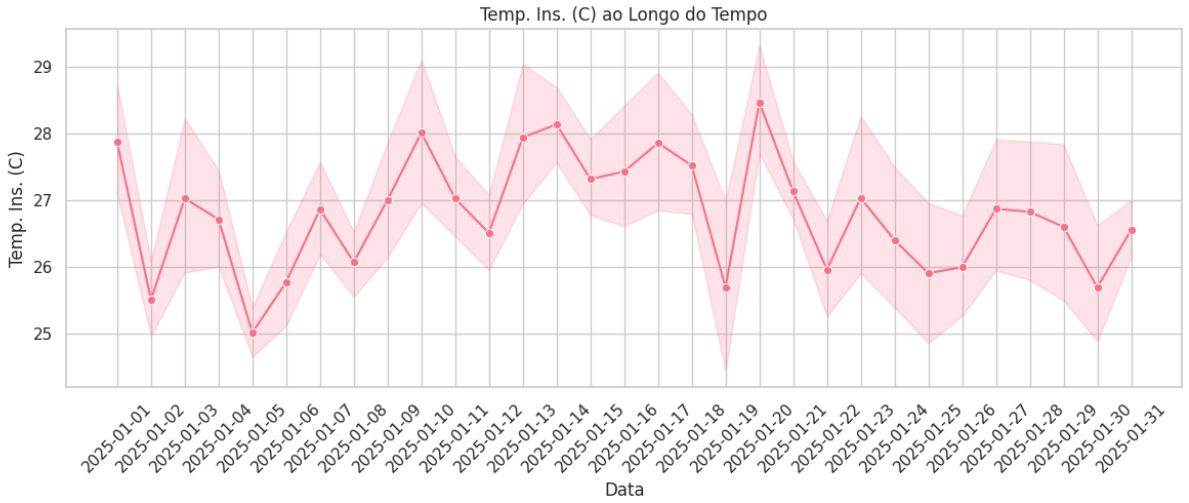

Figura 4.6: Gráficos de temperatura instantânea em Manaus (Janeiro/2025).

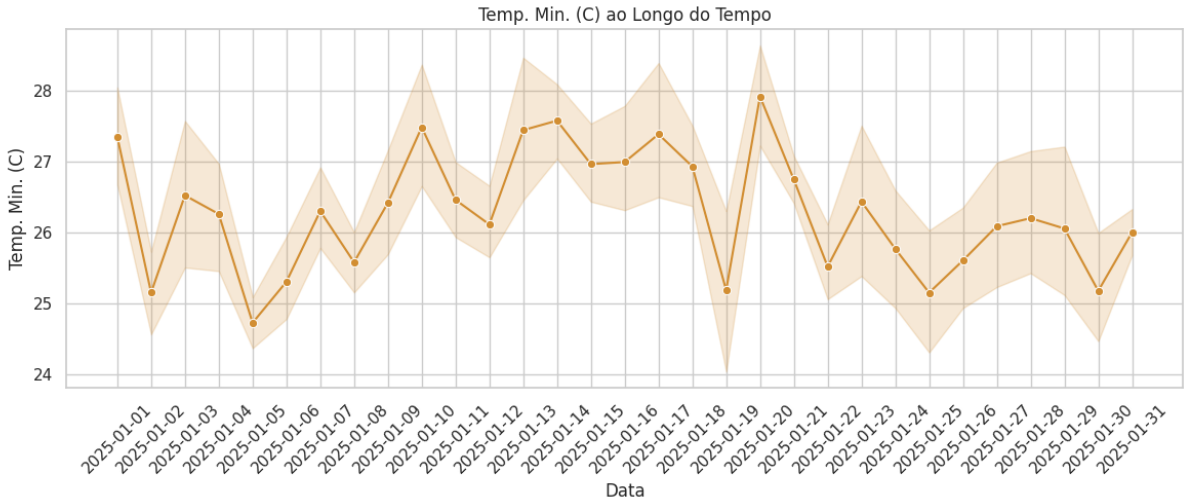

Figura 4.7: Gráficos de temperatura mínima em Manaus (Janeiro/2025).

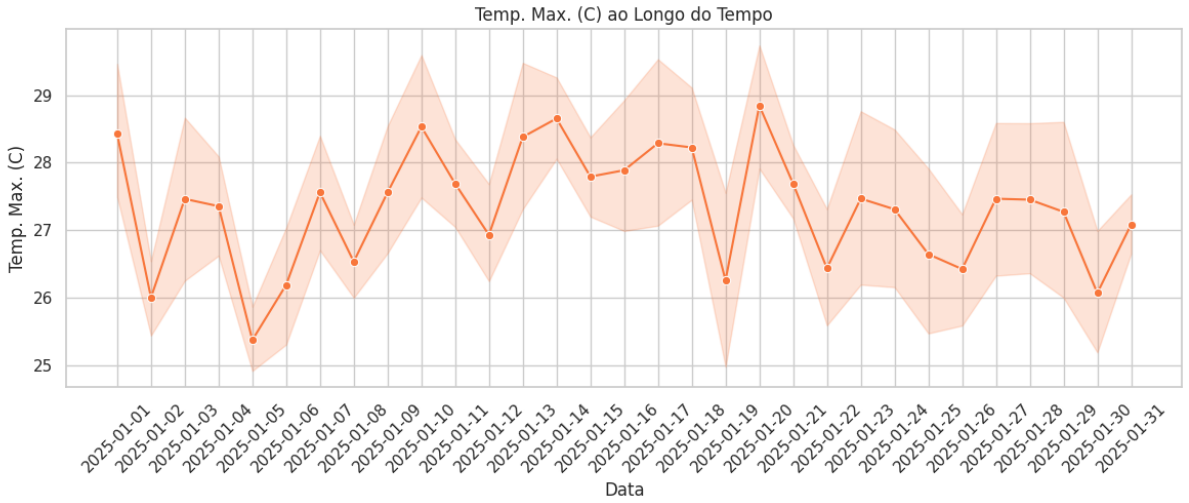

Figura 4.8: Gráficos de temperatura máxima em Manaus (Janeiro/2025).

4.2 Analisando Tendências de Temperatura (INMET)

Neste exercício, você irá coletar, processar e visualizar dados climáticos para analisar tendências de temperatura ao longo do tempo. O foco será identificar padrões sazonais e possíveis variações na temperatura média, máxima e mínima.

4.2.1 O que você deve fazer?

1. **Obter os Dados Climáticos:** Acesse o site do INMET e baixe os dados históricos de temperatura de uma estação meteorológica específica no formato CSV.
2. **Carregar os Dados no Python:** Utilize a biblioteca Pandas para ler o arquivo CSV.
3. **Processar e Limpar os Dados:** Converta as colunas de temperatura para valores numéricos e trate valores ausentes.
4. **Criar Gráficos para Análise:** Utilize a biblioteca Matplotlib para visualizar as variações de temperatura ao longo do tempo.

4.2.2 Temos algumas perguntas para você:

1. A temperatura máxima e mínima seguem um padrão consistente no período analisado?
2. Há picos ou quedas bruscas que possam indicar eventos climáticos atípicos?
3. Como a temperatura média diária varia ao longo do tempo?

4.2.3 Visualizando Sazonalidade com Mapas de Calor

Embora o gráfico de linha seja uma excelente ferramenta para identificar a tendência geral dos dados ao longo do tempo, ele se torna limitado quando nosso objetivo é analisar padrões sazonais complexos, como a variação da temperatura mês a mês, ao longo de vários

anos. Tentar sobrepor muitas linhas (uma para cada ano) em um único gráfico rapidamente o tornaria ilegível.

Para essa tarefa, o Mapa de Calor (*Heatmap*) é a visualização ideal. Ele nos permite representar os dados em uma grade bidimensional (uma matriz), onde uma dimensão é o Ano e a outra é o Mês. A intensidade da cor em cada célula da grade representará a magnitude da variável (neste caso, a temperatura média), facilitando a identificação imediata de padrões cíclicos, anomalias ou mudanças graduais ano a ano.

Preparação dos Dados para o Heatmap

Um heatmap não pode ser construído diretamente a partir dos dados "crus" (sejam eles horários ou diários). A visualização exige uma estrutura de dados específica: uma matriz onde cada célula `(Ano, Mes)` contenha um único valor agregado. O formato atual dos nossos dados é conhecido como formato longo (*long*), onde cada linha é uma observação (ex: `[Timestamp, Temperatura]`). Precisamos convertê-lo para um formato largo (*wide*) ou de matriz. Realizamos essa transformação em duas etapas principais, geralmente encadeadas em uma única linha de código:

- **1. Agregação (Agrupamento):** Primeiro, agrupamos todos os dados pertencentes ao mesmo **Ano** e **Mês**. Em seguida, calculamos uma métrica de resumo para cada um desses grupos. Para a temperatura, a média é a escolha mais comum.
  - **Código:** `.groupby(['Ano', 'Mes']).mean()`
  - **Resultado:** Uma Série (ou DataFrame) indexada por `(Ano, Mes)`, onde cada par tem um valor de temperatura média.
- **2. Pivotagem (Reshape):** O resultado anterior ainda está em formato "longo"(um índice multinível). Precisamos "pivotar" ou "desempilhar" (unstack) a tabela. Esta operação move um dos níveis do índice (neste caso, o `Mes`) das linhas para as colunas.
  - **Código:** `.unstack()`
  - **Resultado:** A estrutura de matriz final que o heatmap necessita: os Anos se tornam as linhas, os Meses se tornam as colunas, e os valores dentro da matriz são as temperaturas médias correspondentes.

Com os dados agora formatados como uma matriz (Ano $\times$ Mês), podemos finalmente plotar o mapa de calor.

Copie e Teste!

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Carregar os dados
dados = pd.read_csv('dados_clima.csv', sep=';', decimal=',')

# Converter a coluna de data para o formato datetime
dados['Data'] = pd.to_datetime(dados['Data'], format='%d/%m/%Y')
```

```
# Criar colunas de Ano e Mes para agregacao dos dados
dados['Ano'] = dados['Data'].dt.year
dados['Mes'] = dados['Data'].dt.month

# Preenchendo valores ausentes com a media das colunas numericas
dados.fillna(dados.mean(numeric_only=True), inplace=True)

# Selecionar a coluna de temperatura a ser analisada
coluna_temperatura = "Temp. Ins. (C)"

# Converter para valores numericos, lidando com possiveis erros
dados[coluna_temperatura] = pd.to_numeric(dados[coluna_temperatura
    ], errors='coerce')

# Agrupar os dados e calcular a media mensal da temperatura
heatmap_data = dados.groupby(['Ano', 'Mes'])[coluna_temperatura].
    mean().unstack()

# Criar o mapa de calor utilizando Seaborn
plt.figure(figsize=(12, 6))
sns.heatmap(heatmap_data, cmap="coolwarm", annot=True, fmt=".1f",
    linewidths=0.5)
plt.title("Variacao da Temperatura ao Longo dos Meses")
plt.xlabel("Mes")
plt.ylabel("Ano")
plt.show()
```

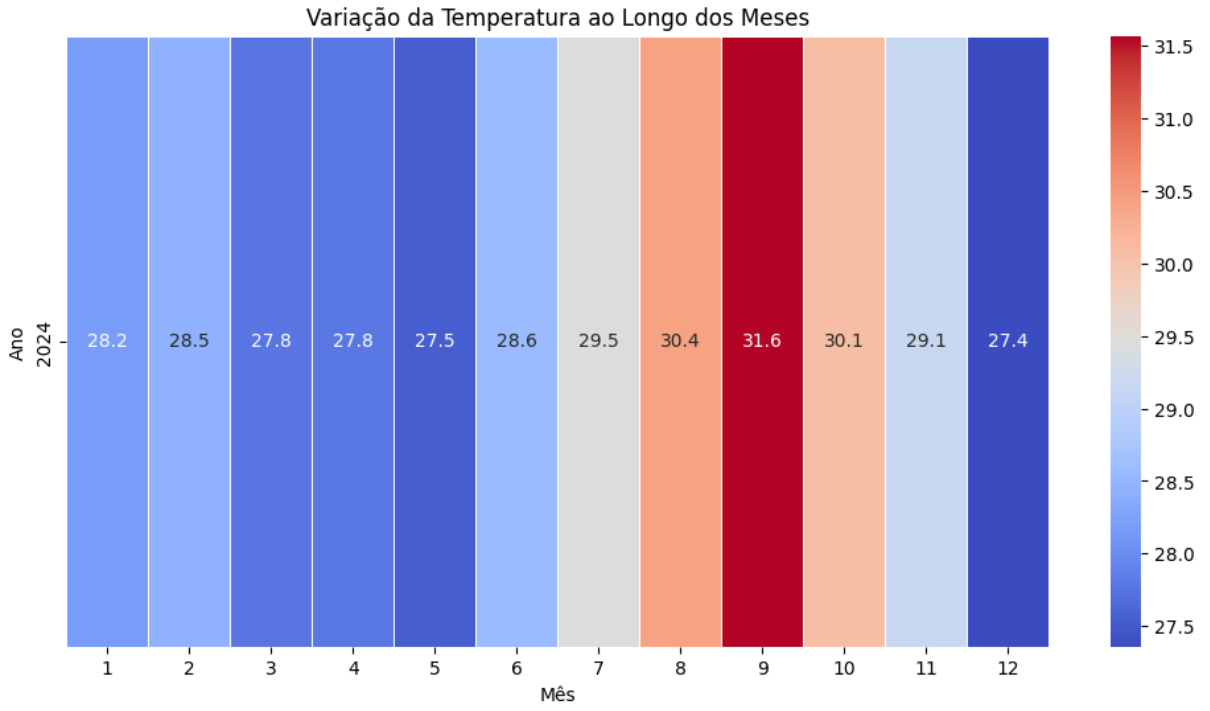

Figura 4.9: Heatmap da Variação da Temperatura ao Longo dos Meses.

Nos gráficos de linha, você pode notar que a temperatura "pula" muito de um dia para o outro. Esse "ruído" de curto prazo pode dificultar a visualização da tendência principal. Para "suavizar" o gráfico, usamos a técnica de **Média Móvel**. O comando `.rolling(window=7).mean()` do Pandas calcula, para cada dia, a média dos últimos 7 dias. Isso remove o "ruído" e revela a tendência de longo prazo de forma muito mais clara.

Copie e Teste!

```
# Aplicar uma media movel de 7 dias para suavizacao dos dados
dados['Media Movel (7 dias)'] = dados[coluna_temperatura].rolling(
    window=7).mean()

# Criar um grafico de tendencia com a media movel
plt.figure(figsize=(15, 5))
plt.plot(dados['Data'], dados[coluna_temperatura], label='
    Temperatura Instantanea', color='blue', alpha=0.5)
plt.plot(dados['Data'], dados['Media Movel (7 dias)'], label='
    Media Movel (7 dias)', color='red', linewidth=2)

# Personalizar o grafico
plt.xlabel('Data')
plt.ylabel('Temperatura (°C)')
plt.title('Tendencia de Temperatura com Media Movel')
plt.legend()
plt.grid(True)
plt.show()
```

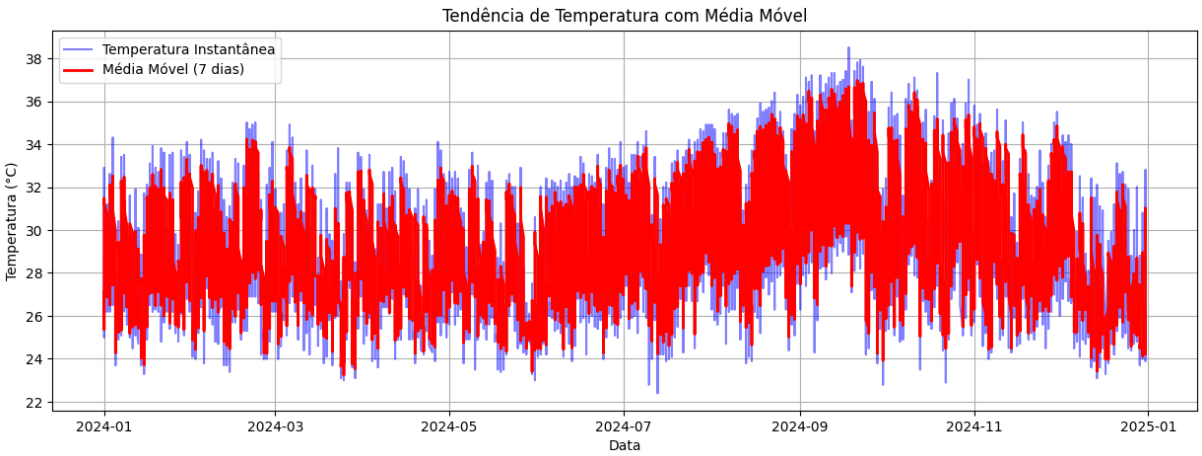

Figura 4.10: Gráfico de Tendência de Temperatura com Média Móvel.

4.3 Estudo de Caso Real: Produção de Sementes (SIGEF)

Nesta seção, aplicaremos as técnicas estatísticas e de visualização aprendidas em um contexto prático e de alta relevância: a análise da produção de sementes no Brasil. O agronegócio é um pilar da economia nacional, e entender seus dados é fundamental para a tomada de decisão, tanto no setor privado quanto no público.

Este estudo de caso oferece uma oportunidade valiosa para conectar a teoria da ciência de dados com problemas reais. Utilizaremos um *dataset* governamental autêntico, extraído diretamente do Sistema de Gestão de Fiscalização (SIGEF) do Ministério da Agricultura, Pecuária e Abastecimento (MAPA).

Fique Alerta!

Todos os dados utilizados nesse exercício foram extraídos do link abaixo:
`https://dados.agricultura.gov.br/sv/dataset/dados-referentes-ao-controle-da-producao-de-sementes-sigef`

4.3.1 Objetivos da Análise

Mais do que apenas aplicar funções, nosso objetivo é extrair *insights* estratégicos deste conjunto de dados. Buscaremos responder a perguntas-chave que um gestor ou analista de políticas públicas faria, tais como:

- **Tendências Temporais:** Como a área plantada e a produção evoluíram ao longo das safras? Existem tendências de crescimento ou retração?
- **Distribuição Geográfica:** Quais estados e municípios são os pilares da produção de sementes no país? Como essa produção está distribuída?
- **Análise de Precisão:** Qual é a relação entre a *área estimada* antes do plantio e a *área efetivamente plantada*? Isso pode indicar o nível de precisão das previsões do setor.

4.3.2 Roteiro Metodológico

Para responder a essas perguntas, executaremos um roteiro metodológico que consolida as habilidades desenvolvidas nos capítulos anteriores. O processo será dividido em etapas:

1. **Carga e Preparação dos Dados:** Carregar o arquivo `.csv` usando `pd.read_csv` e realizar uma inspeção inicial.
2. **Agregação e Sumarização:** Agregar os dados em níveis relevantes (por safra, por estado, etc.), combinando `.groupby()` com `.sum()` ou `.mean()`
3. **Tendências:** Aplicar `sns.lineplot` para analisar a evolução temporal da produção.
4. **Comparações:** Empregar `sns.barplot` para comparar a produção entre diferentes estados ou municípios.
5. **Relações:** Utilizar `sns.scatterplot` para investigar a correlação entre a área estimada e a área plantada.

Copie e Teste!

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Carregar o dataset
file_path = "sigefdeclaracaoareaproducao.csv"
df = pd.read_csv(file_path, encoding="utf8", delimiter=";")

# Converter coluna de data
if 'DATAPLANTIO' in df.columns:
    df['DATAPLANTIO'] = pd.to_datetime(df['DATAPLANTIO'], errors='
   coerce')

# Agrupar por periodo e somar a area plantada
area_por_safra = df.groupby('PERIODO')['AREAPLANTADA'].sum()

# Grafico de evolucao da area plantada ao longo das safras
plt.figure(figsize=(10,5))
sns.lineplot(x=area_por_safra.index, y=area_por_safra.values,
   marker='o')
plt.xticks(rotation=45)
plt.xlabel("Safra")
plt.ylabel("Area Plantada (ha)")
plt.title("Evolucao da Area Plantada por Safra")
plt.grid()
plt.show()

# Distribuicao da Area Plantada por Estado (UF)
plt.figure(figsize=(12,6))
area_por_estado = df.groupby('UF')['AREAPLANTADA'].sum().
   sort_values(ascending=False)
sns.barplot(x=area_por_estado.index, y=area_por_estado.values)
plt.xlabel("Estado (UF)")
plt.ylabel("Area Plantada (ha)")
plt.title("Area Plantada por Estado")
plt.xticks(rotation=90)
plt.show()

# Top 10 municipios com maior quantidade reservada de sementes
plt.figure(figsize=(12,6))
top_municipios = df.groupby('MUNICIPIO')['QUANTRESERVADA'].sum().
   sort_values(ascending=False).head(10)
sns.barplot(y=top_municipios.index, x=top_municipios.values)
plt.xlabel("Quantidade Reservada (kg)")
plt.ylabel("Municipio")
plt.title("Top 10 Municipios com Maior Quantidade de Sementes
   Reservadas")
plt.show()

# Comparacao entre Area Estimada e Area Plantada
plt.figure(figsize=(8,6))
sns.scatterplot(x=df['AREAPLANTADA'], y=df['AREAESTIMADA'])
plt.xlabel("Area Plantada (ha)")
plt.ylabel("Area Estimada (ha)")
```

```
plt.title("Relacao entre Area Estimada e Plantada")
plt.grid()
plt.show()
```

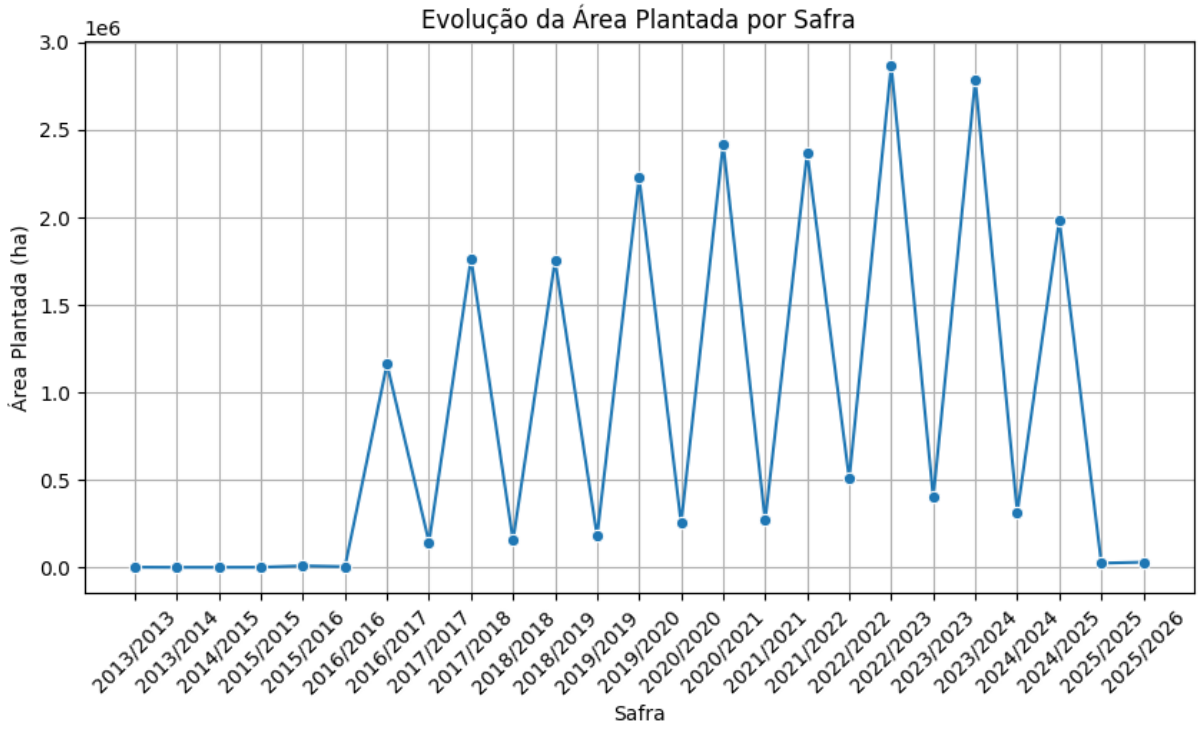

Figura 4.11: Evolução temporal da área plantada por safra.

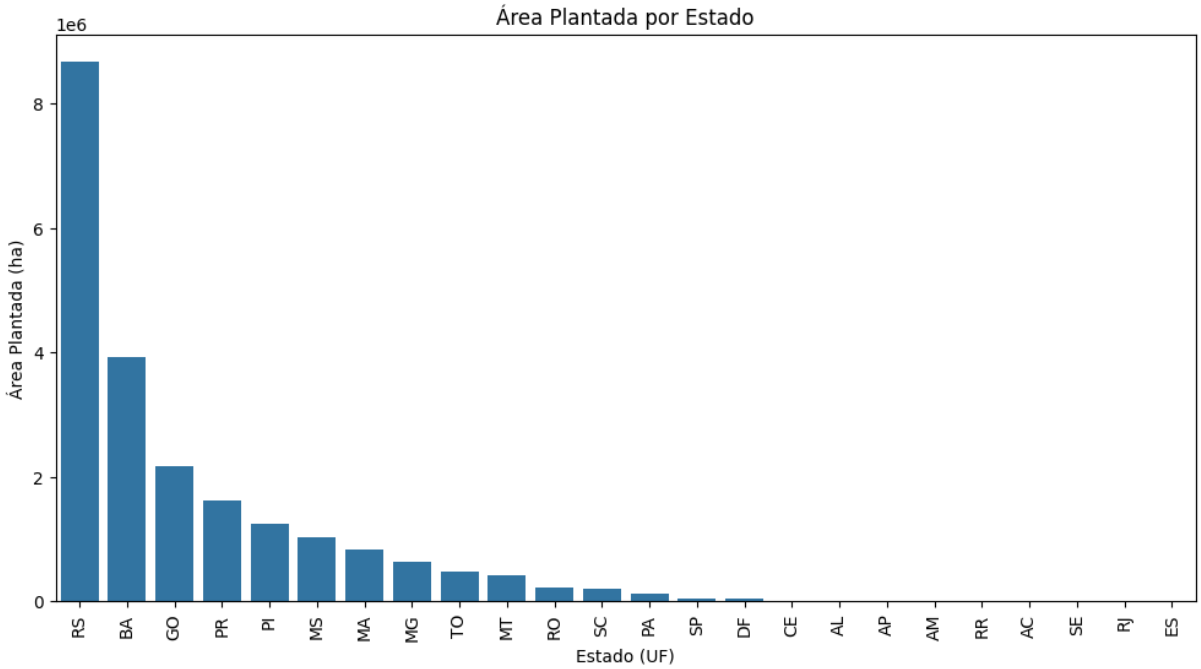

Figura 4.12: Distribuição da área plantada por estado (UF).

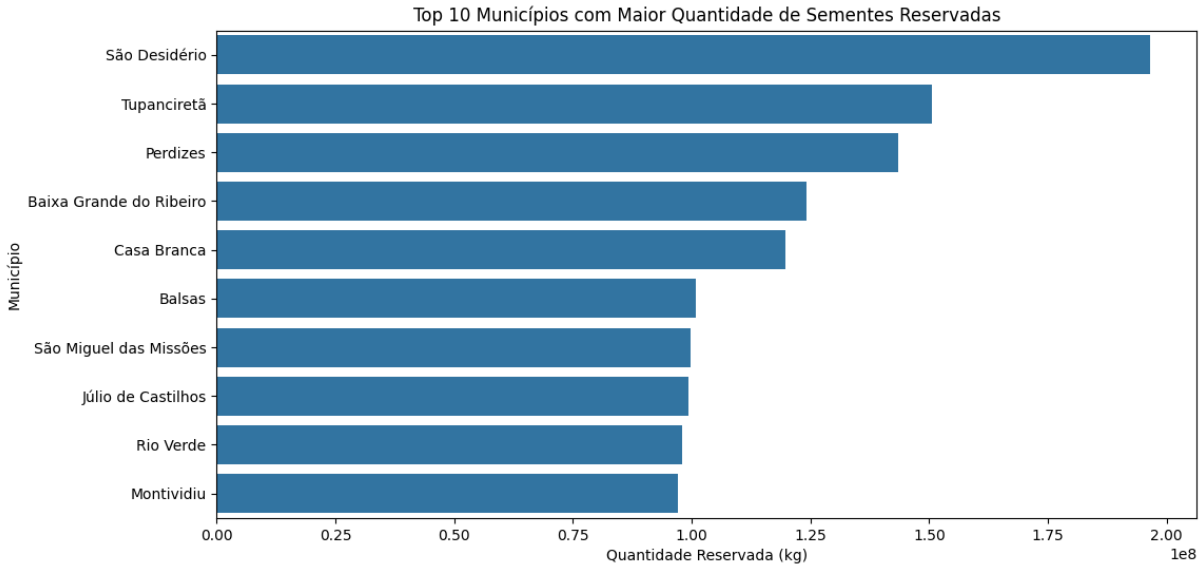

Figura 4.13: Lista dos 10 municípios com maior quantidade de sementes reservadas.

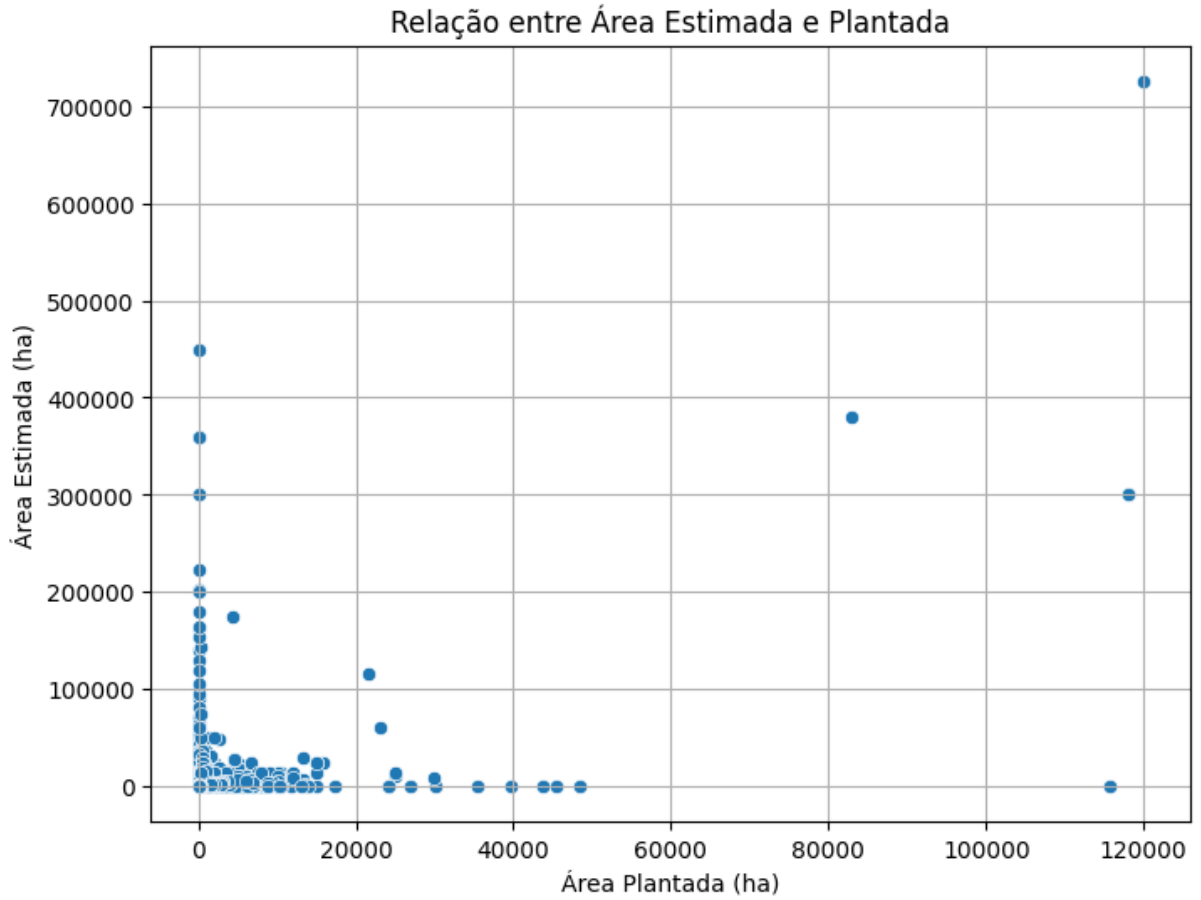

Figura 4.14: Diagrama de dispersão da relação entre a área estimada e a área plantada.

Convidamos você a ir além dos gráficos apresentados e aprofundar sua investigação. Sugestões de novas análises:

- Tendência de crescimento da produção por estado.

- Análise específica por cultura.
- Mapeamento geográfico da produção.

AGORA É COM VOCÊ! O QUE MAIS OS DADOS PODEM REVELAR?

4.4 Contexto Adicional: Dados na Agricultura Sustentável

As seções a seguir fornecem contexto sobre como a análise de dados, similar à que fizemos com os dados do INMET, é crucial para a sustentabilidade. Elas servem de inspiração para os desafios finais.

4.4.1 Como Dados Auxiliam na Economia de Água e Energia?

Você já parou para pensar em como usamos água e energia no dia a dia? Muitas vezes, esses recursos são consumidos de forma excessiva ou ineficiente, e é aí que os dados entram em ação! Com a análise de informações, podemos entender melhor como utilizamos esses recursos e encontrar maneiras de economizar sem comprometer a qualidade de vida ou a produção agrícola, por exemplo. Mas como isso funciona na prática?

Imagine que você tem sensores espalhados por uma fazenda ou até mesmo em uma cidade inteira. Esses sensores coletam informações sobre consumo de água e energia a todo momento. Se houver um uso excessivo em determinado horário ou local, os dados podem indicar isso instantaneamente, permitindo que ajustes sejam feitos na hora. E se pudéssemos prever o futuro? Bom, os dados podem nos ajudar nisso! Com base no histórico de consumo e em fatores como temperatura e umidade, modelos preditivos conseguem estimar quando a demanda por água ou energia será maior.

Fique Alerta!

Da próxima vez que você ligar a luz ou abrir a torneira, pense no poder que os dados têm para transformar a forma como utilizamos esses recursos!

4.4.2 Eficiência no uso de recursos agrícolas

A Ciência de Dados tem sido uma grande aliada do setor agrícola, ajudando produtores a tomar decisões mais inteligentes e sustentáveis. Imagine um agricultor que precisa irrigar sua plantação, mas não tem certeza de quanta água é realmente necessária. Antes, ele poderia simplesmente seguir um cronograma fixo de irrigação, desperdiçando água em dias chuvosos ou não irrigando o suficiente em períodos mais secos.

Mas agora, sensores de umidade do solo e dados climáticos são coletados em tempo real. Com essa informação, sistemas inteligentes ajustam automaticamente a irrigação para fornecer apenas a quantidade de água necessária. Além disso, imagens de satélite podem ajudar agricultores a aplicar fertilizantes apenas onde é realmente necessário. O que todos esses casos mostram? Que a tecnologia e a análise de dados estão transformando a forma como usamos os recursos naturais na agricultura.

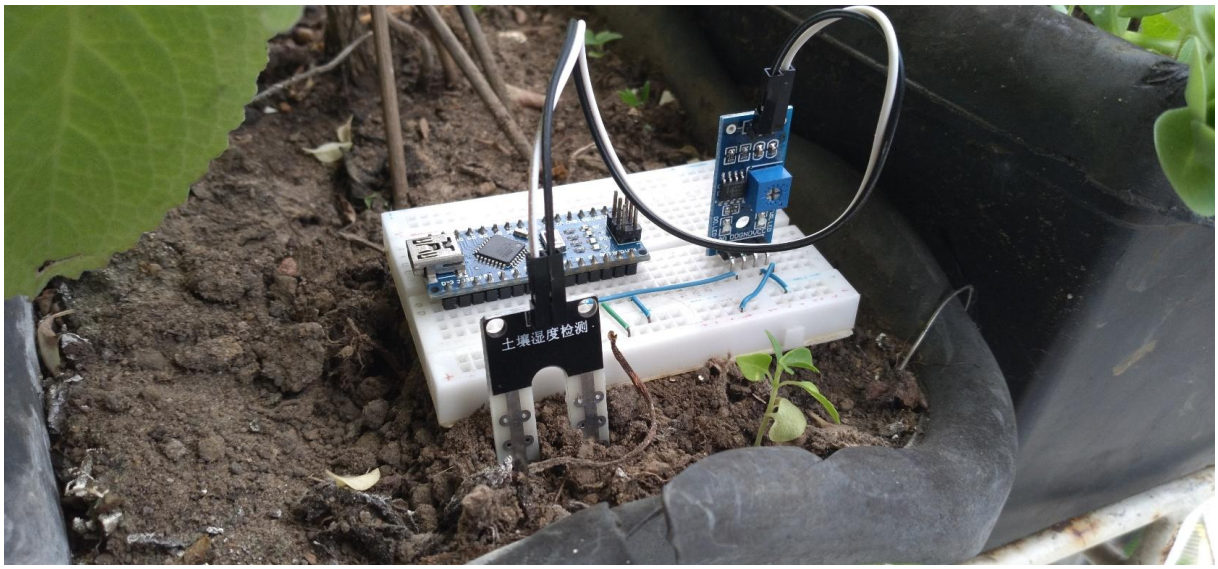

Figura 4.15: Sensor de umidade do solo com Arduino.
Fonte: https://elcereza.com/sensor-de-umidade-do-solo/

4.5 Desafios Adicionais com Datasets Simulados

Os estudos de caso anteriores usaram dados reais. Agora, é sua vez de praticar com *datasets* simulados, criados especificamente para testar suas habilidades de análise exploratória.

4.5.1 Desafio 1: Planejamento de Irrigação

Este exercício tem como propósito explorar o uso de Python e Ciência de Dados para auxiliar no planejamento eficiente da irrigação. Através da análise de informações como umidade do solo, temperatura e padrões climáticos, você aprenderá a identificar os momentos ideais para irrigação, evitando desperdícios e garantindo uma melhor produtividade agrícola. O *dataset* contém informações essenciais para o planejamento da irrigação, incluindo o tipo de plantio, a umidade do solo, a temperatura ambiente e um indicador binário que informa se a irrigação foi acionada ou não.

Copie e Teste!

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Carregar os dados do arquivo CSV
dados = pd.read_csv('data.csv')

# Verificar estatisticas basicas
dados.describe()

# Criar um grafico de dispersao para visualizar a relacao entre
   umidade e temperatura
plt.figure(figsize=(10, 5))
```

```
sns.scatterplot(x=dados['umidade'], y=dados['temperatura'], hue=
    dados['bombear_agua'], palette='coolwarm')
plt.xlabel("Umidade do Solo")
plt.ylabel("Temperatura")
plt.title("Relacao entre Umidade, Temperatura e Necessidade de
    Irrigacao")
plt.show()

# Criar um histograma para visualizar a distribuicao da umidade
dados['umidade'].hist(bins=20, edgecolor='black', alpha=0.7)
plt.xlabel("Umidade do Solo")
plt.ylabel("Frequencia")
plt.title("Distribuicao da Umidade do Solo")
plt.show()

# Criar um grafico de linha para visualizar a tendencia da umidade
    do solo ao longo do tempo
plt.figure(figsize=(12, 5))
plt.plot(dados.index, dados['umidade'], label='Umidade do Solo',
    color='blue')
plt.axhline(y=600, color='red', linestyle='--', label='Limite
    Critico de Umidade')
plt.xlabel("Indice do Registro")
plt.ylabel("Umidade do Solo")
plt.title("Tendencia da Umidade do Solo ao Longo do Tempo")
plt.legend()
plt.grid(True)
plt.show()
```

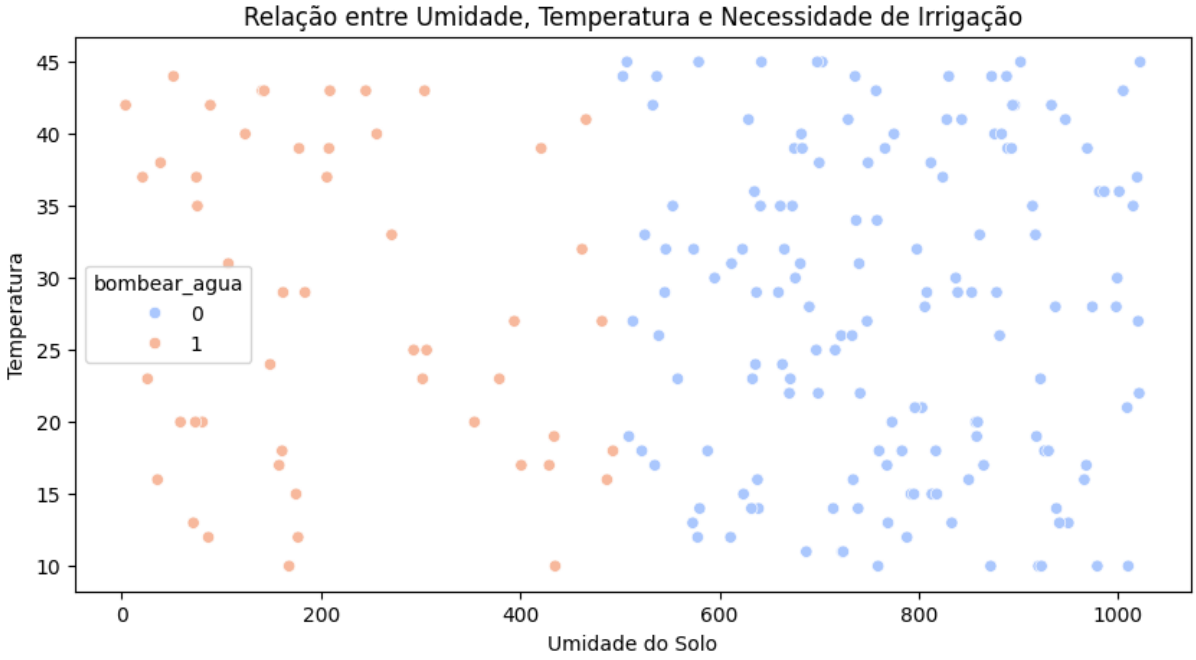

Figura 4.16: Relação entre a umidade do solo, a temperatura e a necessidade de irrigação.

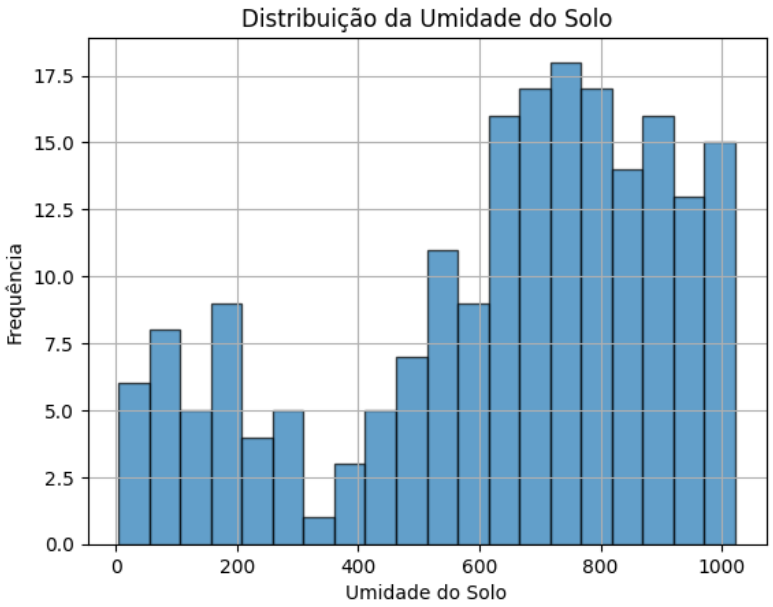

Figura 4.17: Distribuição da frequência dos níveis de umidade do solo.

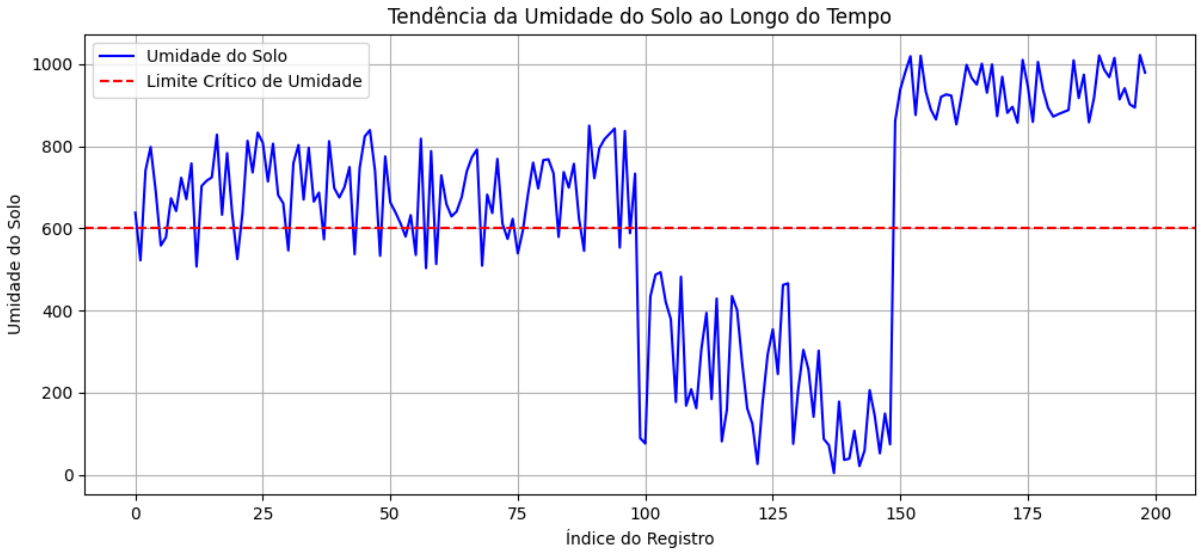

Figura 4.18: Tendência da umidade do solo ao longo do tempo, com o limite crítico.

Temos algumas perguntas para você:

1. Qual a relação entre umidade do solo e temperatura? Os dados mostram algum padrão? Em que situações a irrigação foi acionada?
2. Como a variação da umidade influencia a decisão de irrigação? Há um limite claro onde a irrigação se torna necessária?
3. Se você fosse um agricultor, como utilizaria essas informações na sua plantação para tornar a irrigação mais eficiente e sustentável?
4. Como a Ciência de Dados pode contribuir para a agricultura moderna, que outros desafios agrícolas poderiam ser resolvidos com a análise de dados?

4.5.2 Desafio 2: Análise Agronômica e Produtividade

Aqui o objetivo é analisar quais tipos de solo apresentam maior rendimento para cada cultura e identificar padrões que possam otimizar o uso da terra para maximizar a produtividade.

A escolha do solo adequado para cada cultura é um dos fatores mais importantes para garantir uma produção agrícola eficiente. Com o auxílio da Ciência de Dados, podemos analisar características do solo e da produtividade para identificar padrões e tomar decisões mais assertivas no manejo das lavouras.

Copie e Teste!

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Carregar os dados do CSV
dados = pd.read_csv('dados_agronomicos.csv')

# Exibir as primeiras linhas do dataset
print(dados.head())

# Verificar estatisticas basicas
print("\n\n\nEstatisticas basicas do dataset\n\n", dados.describe
    ())

# Identificar o melhor tipo de solo para cada cultura com base na
    media de produtividade
melhores_solos = dados.groupby(['cultura_plantada', 'tipo_solo'])[
    'produtividade'].mean().reset_index()
melhor_solo_por_cultura = melhores_solos.loc[melhores_solos.
    groupby('cultura_plantada')['produtividade'].idxmax()]

print("\nMelhor tipo de solo para cada cultura:")
print(melhor_solo_por_cultura)

# Criar uma matriz com a contagem de culturas por tipo de solo
tabela_heatmap = dados.pivot_table(index="tipo_solo", columns="
    cultura_plantada", aggfunc="size", fill_value=0)

# Criar heatmap
plt.figure(figsize=(8, 5))
sns.heatmap(tabela_heatmap, annot=True, cmap="viridis", fmt="d")

# Configurar titulo e rotulos
plt.xlabel("Cultura Plantada")
plt.ylabel("Tipo de Solo")
plt.title("Frequencia das Culturas por Tipo de Solo")
plt.show()
```

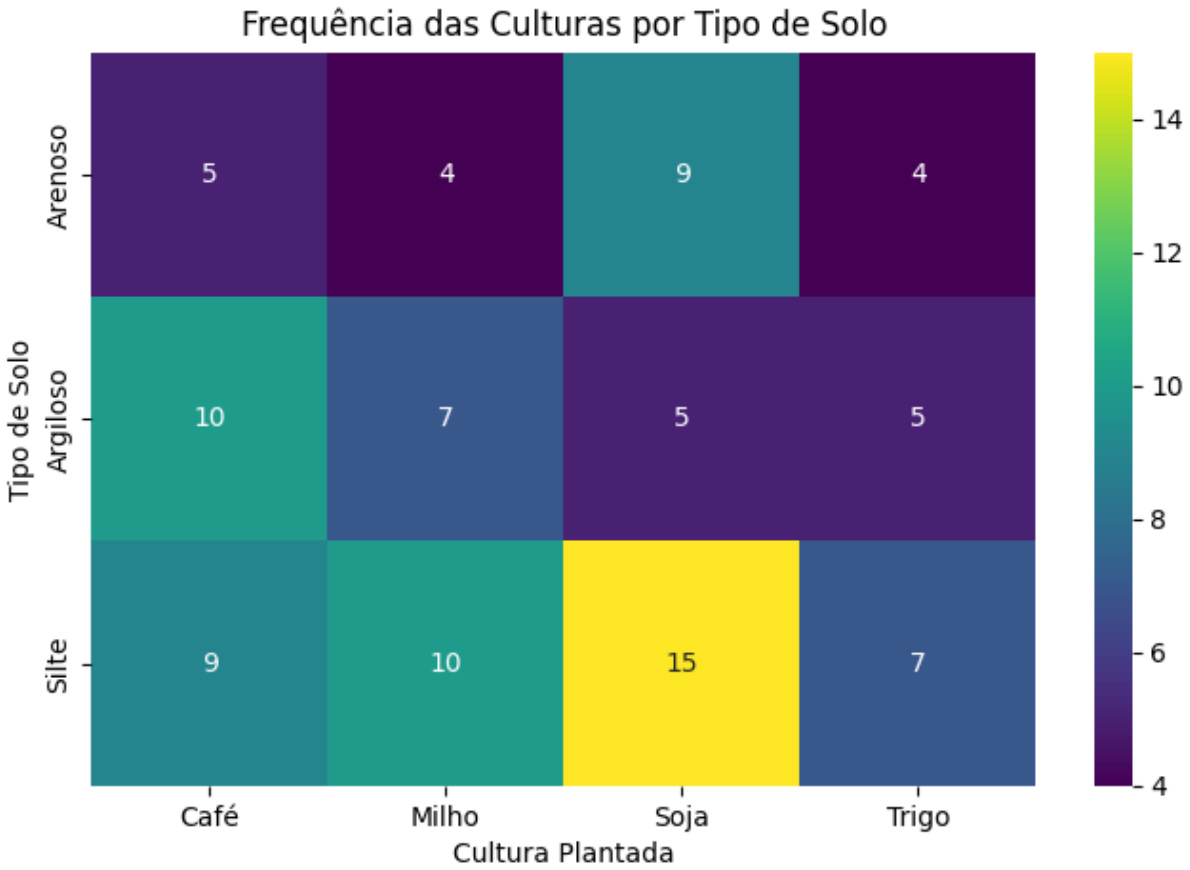

Figura 4.19: Heatmap da Frequência das Culturas por Tipo de Solo.

O que esse gráfico nos mostra?

- Quais solos são mais usados para determinadas culturas.
- Se existe um solo dominante para uma cultura específica.
- A diversidade de cultivos dentro de cada solo.

Fique Alerta!

Este é apenas um exemplo didático. Os dados apresentados neste estudo são fictícios e foram gerados apenas para fins educacionais. Eles não representam valores reais de produtividade agrícola nem condições reais do solo.

Capítulo 5

Estudo de Caso sobre Classificação do PIB Brasileiro

Iniciando o diálogo...

Bem-vindo(a), estudante! Nos capítulos anteriores, você montou sua caixa de ferramentas para manipulação e análise de dados com Python e Pandas. Agora, é hora de aplicar esse conhecimento em um dos desafios mais complexos e impactantes do mundo real: a análise de ciclos econômicos.

Neste capítulo, você aprenderá a usar dados para entender e prever o comportamento da economia de um país. Assumiremos o papel de um cientista de dados encarregado de analisar a saúde econômica do Brasil, utilizando indicadores macroeconômicos reais para tentar antecipar as tendências do Produto Interno Bruto (PIB).

Vamos mergulhar em um projeto completo que abordará desde a coleta automatizada de dados via API do Banco Central até as técnicas de tratamento e normalização de séries temporais. Ao final, você terá uma compreensão prática de como transformar dados econômicos brutos em informações valiosas e acionáveis, preparando o terreno para futuras análises preditivas.

5.1 Contextualização do Problema

5.1.1 Ciclos Econômicos e o Produto Interno Bruto

A economia de um país não cresce de forma linear. Ela se move em ciclos, alternando entre períodos de expansão (crescimento da atividade) e contração ou recessão (desaceleração da produção). Compreender em que fase do ciclo uma economia se encontra e para qual direção ela aponta é uma tarefa crucial para governos, empresas e investidores [6].

A principal métrica para medir a saúde econômica é o Produto Interno Bruto (PIB), já que sua variação é o indicador definitivo do crescimento ou retração da economia. Prever o PIB se baseia na análise de indicadores antecedentes, variáveis que tendem a mudar antes do próprio PIB, fornecendo pistas sobre o futuro [26].

5.1.2 Arquitetura do Projeto

Para enfrentar este desafio de forma estruturada, nosso projeto seguirá uma arquitetura modular, dividida em quatro camadas lógicas, conforme ilustrado na Figura 5.1. Este fluxograma

servirá como um mapa, guiando-nos através de cada etapa do processo.

- **Aquisição de Dados:** Etapa onde os dados brutos são coletados de diversas fontes.
- **Preparação de Dados:** O coração do nosso trabalho, onde os dados são limpos, transformados e preparados para a análise.
- **Classificação:** Etapa onde são realizadas as previsões através da aplicação de modelos de Machine Learning.
- **Validação:** Etapa onde é avaliada a performance preditiva dos modelos.

Neste livro, nosso foco principal é a coleta pré-processamento dos dados referentes as duas primeiras camadas, que são a base de qualquer projeto de ciência de dados bem-sucedido.

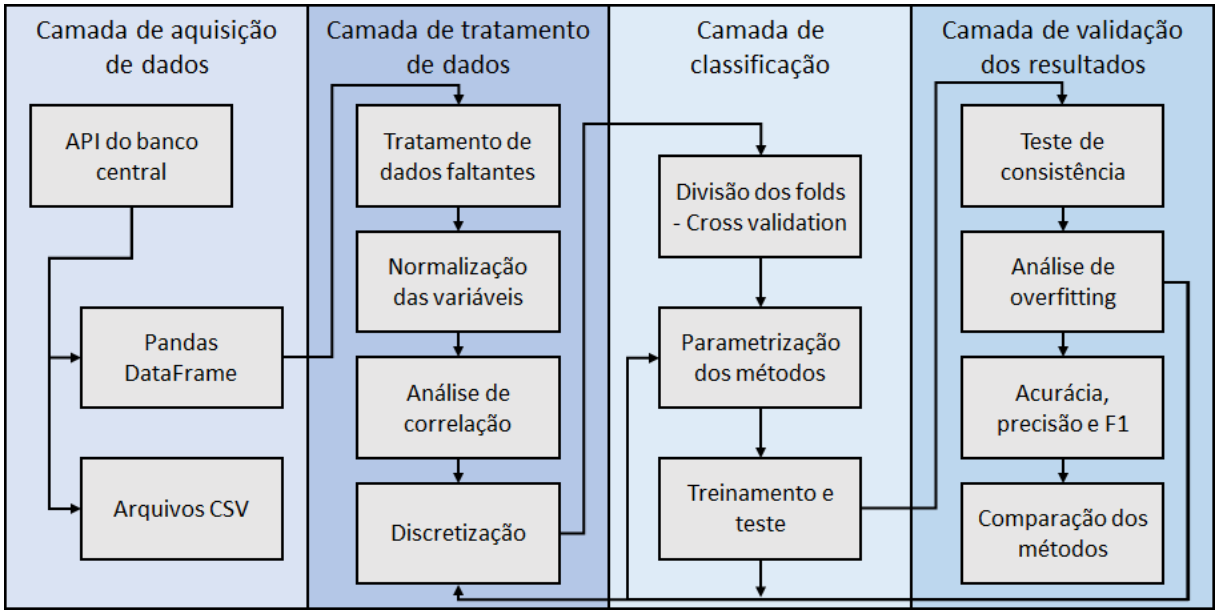

Figura 5.1: Arquitetura do Projeto de Análise de Ciclos Econômicos.
Fonte: O próprio autor

5.2 Coleta de Dados: A Investigação Inicial

5.2.1 A Motivação por Trás das Variáveis

A escolha das variáveis explicativas é um passo fundamental em qualquer modelo preditivo. Em vez de uma seleção aleatória, o processo deve ser guiado pelo conhecimento do domínio — neste caso, a teoria econômica. A análise de ciclos econômicos possui uma vasta literatura dedicada a identificar variáveis que servem como ”sinais”que antecedem os movimentos da economia como um todo [6, 7].

Inspirado em estudos de referência, foi realizada uma investigação na vasta base de dados do **Sistema Gerenciador de Séries Temporais (SGS) do Banco Central do Brasil** para compilar um conjunto de potenciais indicadores. Após um processo de filtragem para garantir a consistência da granularidade temporal e a relevância metodológica (discutido na próxima seção), chegamos ao conjunto final de 16 indicadores detalhados na Tabela 5.1.

Tabela 5.1: Indicadores Macroeconômicos Selecionados

Sigla	Indicador	Categoria	Fonte
PIB	PIB mensal - Valores correntes (R$ milhões)	Preço	BCB-DEPEC
IPA	Índice de Preços por Atacado-Mercado	Preço	FGV
IPEM	Indicadores da produção - Extrativa mineral	Atividade	IBGE
IPIT	Indicadores da produção - Indústria de transformação	Atividade	IBGE
IPBC	Indicadores da produção - Bens de capital	Atividade	IBGE
IPBCD	Indicadores da produção - Bens de consumo duráveis	Atividade	IBGE
IVVV	Índice volume de vendas no varejo - Automóveis, motocicletas, partes e peças	Atividade	IBGE
VVCCL	Vendas de veículos - Comerciais leves	Atividade	FENABRAVE
VVCC	Vendas de veículos - Caminhões	Atividade	FENABRAVE
IEF	Índice de Expectativas Futuras	Confiança	Fecomercio
ICC	Índice de Confiança do Consumidor	Confiança	CNI
Spub	Saldos das operações de crédito (público)	Monetário	BCB-DSTAT
Spriv	Saldos das operações de crédito (privado)	Monetário	BCB-DSTAT
M1	Meios de pagamento - M1	Monetário	BCB-DSTAT
M2	Meios de pagamento amplos - M2	Monetário	BCB-DSTAT

Indicadores de Atividade Econômica (O ”Termômetro”da Economia)

Este é o grupo de indicadores mais direto. Eles medem a produção física e as vendas, funcionando como um termômetro em tempo real da saúde dos principais setores da economia.

Produção Industrial (IPEM, IPIT, IPBC, IPBCD): A indústria é um pilar do PIB. Uma queda nos indicadores de produção, seja na extração mineral (IPEM), na transformação (IPIT) ou na fabricação de bens de capital (IPBC) e de consumo (IPBCD), é um dos sinais mais claros de que a atividade econômica está desacelerando. A produção de bens de capital (máquinas e equipamentos) é especialmente importante, pois reflete o investimento das empresas no futuro.

Vendas no Varejo (IVVV, VVCCL, VVCC): Se a indústria é a oferta, o varejo é a demanda. A queda nas vendas de veículos (IVVV, VVCCL, VVCC) é um forte indicador antecedente, pois a compra de bens de alto valor é uma das primeiras a ser adiada pelas famílias e empresas em momentos de incerteza econômica.

Indicadores de Confiança (O ”Barômetro”das Expectativas)

Diferente dos indicadores de atividade, que medem o que já aconteceu, os índices de confiança medem o que as pessoas e empresas esperam que vá acontecer. Eles funcionam como um barômetro do sentimento econômico.

Confiança do Consumidor e do Empresário (ICC, ICEI, IEF): Se os consumidores (ICC) estão pessimistas, eles tendem a poupar mais e consumir menos. Se os empresários (ICEI, IEF) estão com baixa expectativa, eles adiam contratações e novos investimentos. Juntos, esses índices capturam a disposição para gastar e investir na economia, que são os principais motores do PIB. Uma queda na confiança hoje geralmente precede uma queda na atividade econômica amanhã.

Indicadores Monetários e de Crédito (O "Lubrificante"da Economia)

Este grupo de variáveis mede a quantidade de dinheiro e crédito disponíveis na economia. Pense neles como o óleo que lubrifica o motor econômico.

Meios de Pagamento (M1, M2 - Nova Metodologia): O M1 representa o dinheiro mais líquido (papel-moeda e depósitos à vista), enquanto o M2 inclui também investimentos de alta liquidez. Um aumento nesses agregados pode indicar maior capacidade de consumo e investimento. Uma contração súbita pode ser sinal de uma crise de liquidez.

Operações de Crédito (Spub, Spriv): O volume de crédito concedido por bancos públicos (Spub) e privados (Spriv) é um indicador vital. O crédito financia desde a compra de uma geladeira até a construção de uma nova fábrica. Uma expansão do crédito tende a impulsionar o PIB, enquanto uma restrição (bancos emprestando menos) pode frear a economia.

Indicadores de Preços (O Sinalizador de Inflação)

Índice de Preços por Atacado (IPA-M): Este índice mede a variação dos preços no "atacado", ou seja, antes de chegarem ao consumidor final. Ele é um importante indicador antecedente da inflação. Se os preços para os produtores estão subindo, é provável que esse aumento seja repassado aos consumidores no futuro (pressionando o IPCA). Uma alta no IPA pode levar o Banco Central a aumentar as taxas de juros para controlar a inflação, uma ação que deliberadamente desacelera a atividade econômica e, consequentemente, o PIB.

5.2.2 O Desafio da Coleta Manual

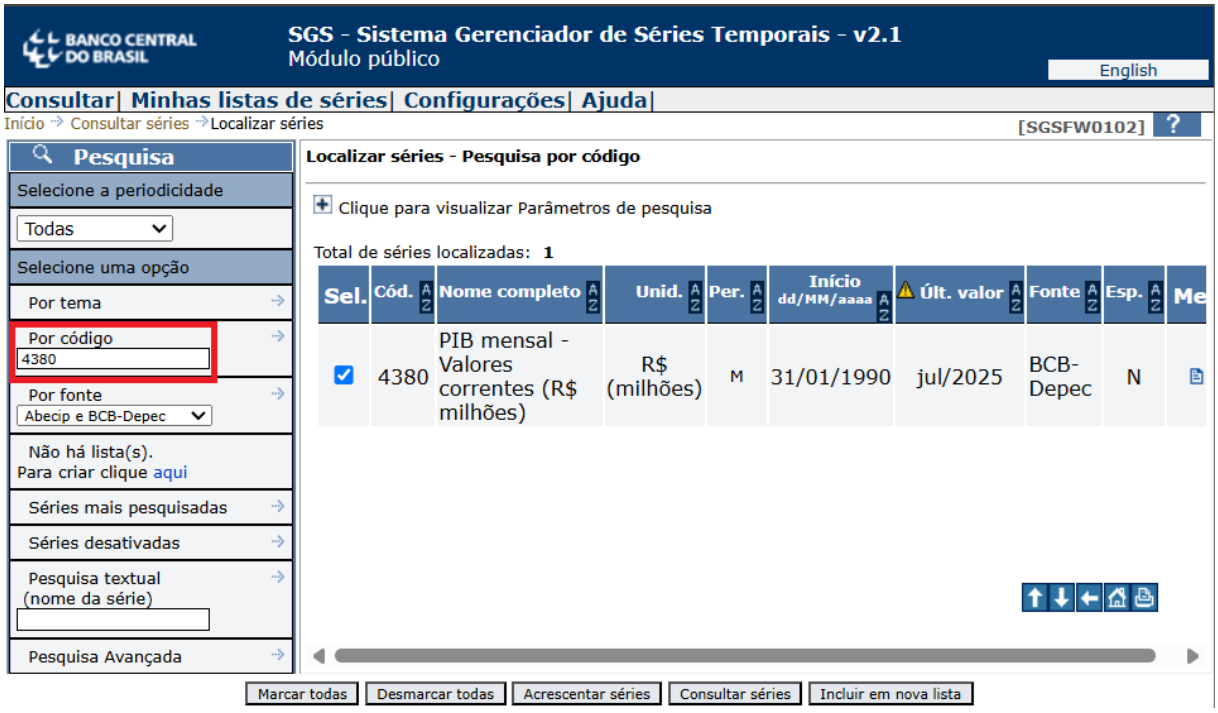

Figura 5.2: Interface de busca do Sistema Gerenciador de Séries Temporais (SGS).
Fonte: O próprio autor

O primeiro passo para construir nosso dataset é obter os dados brutos. A fonte primária e oficial para indicadores macroeconômicos no Brasil é o Sistema Gerenciador de Séries Temporais (SGS) do Banco Central, um portal público que permite a consulta a milhares de séries históricas.

A abordagem mais direta, e o ponto de partida de muitos analistas, é a coleta manual. O processo envolve navegar pela interface do sistema, buscar cada indicador individualmente, configurar os parâmetros da consulta e, finalmente, exportar os dados. A Figura 5.2 ilustra a interface de busca do SGS, onde o usuário pode pesquisar por código ou nome para encontrar a série temporal de interesse.

Uma vez que a série é encontrada e o período desejado é configurado, a plataforma apresenta os dados brutos que podem ser vizualizados de forma tabular ou gráfica na prórpia plataforma, permitindo uma análise prévia das informações. Para uma análise para detalhada em ferramenta externa, existe a opção de exportá-los para um arquivo, geralmente no formato CSV (Valores Separados por Vírgulas) ou XLSX (planilha Excel), como destacado na Figura 5.3.

Figura 5.3: Visualização e exportação de uma série temporal no portal SGS.
Fonte: O próprio autor

Este processo, embora funcional para uma única consulta, revela rapidamente suas limitações quando aplicado a um projeto de maior escala. Para o nosso estudo que utiliza diversos indicadores distintos o fluxo de trabalho manual seria o seguinte:

1. Repetir o processo de busca, configuração e download diversas vezes, uma para cada indicador.

2. Gerenciar cada arquivos CSV ou XLSX separados em uma pasta.

3. Abrir cada um desses arquivos e copiar manualmente os dados para uma única planilha mestra, tomando extremo cuidado para alinhar corretamente as datas de cada série.

As desvantagens desta abordagem são evidentes:

- **Processo Lento e Repetitivo:** A tarefa de baixar e consolidar os dados consome um tempo considerável que poderia ser dedicado à análise.
- **Alto Risco de Erro Humano:** A consolidação manual de dados (copiar e colar) é uma das maiores fontes de erros em planilhas, podendo invalidar completamente a análise.
- **Baixa Reprodutibilidade:** Se precisarmos atualizar nosso estudo com dados de um novo mês, todo o processo manual teria que ser refeito do zero.

Para superar esses desafios, a abordagem de um cientista de dados é criar uma solução automatizada, reprodutível e eficiente. Vamos verificar como automatizar esse processo utilizando a API do Banco Central e a biblioteca Pandas.

5.2.3 A Solução Automatizada com API

O Banco Central do Brasil disponibiliza uma API (Interface de Programação de Aplicações), que permite que nossos scripts acessem o Sistema Gerenciador de Séries Temporais (SGS) diretamente para consultar e baixar dados.

Utilizando a biblioteca Pandas, podemos criar uma função em Python para realizar essa tarefa. A função a seguir encapsula toda a lógica necessária para se conectar à API, solicitar uma série temporal específica pelo seu código, filtrar o período desejado e retornar os dados já estruturados em um DataFrame.

```
def consulta_bc(codigo_bcb, data_inicial, data_final):
    url = f'http://api.bcb.gov.br/dados/serie/bcdata.sgs.{
   codigo_bcb}/dados?formato=json'

    df = pd.read_json(url)
    df['data'] = pd.to_datetime(df['data'], dayfirst=True)
    periodo = (df['data'] >= data_inicial) & (df['data'] <=
   data_final)
    df = df.loc[periodo]
    df.set_index('data', inplace=True)

    return df
```

Antes de analisarmos a lógica interna, é fundamental entender a ”interface”da função, ou seja, os dados que ela recebe (argumentos) e o que ela devolve (retorno).

- **Argumentos (o que a função recebe):**
  - `codigo_bcb`: Um número inteiro que representa o código único da série no sistema SGS. Cada indicador (PIB, IPCA, etc.) possui seu próprio código.
  - `data_inicial`: Uma string de texto no formato ’AAAA-MM-DD’ que define o início do período de consulta.
  - `data_final`: Uma string de texto no formato ’AAAA-MM-DD’ que define o fim do período de consulta.

- **Retorno (o que a função devolve):**
  - A função retorna um DataFrame do Pandas. Este DataFrame contém a série temporal solicitada, com as datas como índice e os valores do indicador em uma única coluna.

A elegância desta função está em sua simplicidade e no poder das ferramentas que utiliza. Vamos analisar suas etapas principais:

1. **Construção da URL:** A primeira linha usa uma f-string para construir dinamicamente o endereço da API, inserindo o `codigo_bcb` do indicador que queremos consultar.
2. **Leitura Direta com Pandas:** Em vez de tratar a resposta da API manualmente, a função `pd.read_json(url)` do Pandas se conecta à internet, baixa os dados em formato JSON e os converte instantaneamente em um DataFrame.
3. **Conversão de Datas:** `pd.to_datetime(df['data'], dayfirst=True)` é um passo crucial de limpeza. Ela garante que a coluna de datas seja tratada como um objeto temporal, e não como texto, o que é essencial para qualquer análise de séries temporais. O argumento `dayfirst=True` especifica o formato de data utiizado no Brasil (DD/MM/AAAA).
4. **Filtragem por Período:** A função cria uma máscara booleana para selecionar apenas as linhas cujo índice de data esteja dentro do período de interesse, intervalo especificado por `data_inicial` e `data_final`.
5. **Indexação por Data:** Por fim, `df.set_index('data', inplace=True)` define a coluna de datas como o índice do DataFrame. Esta é uma convenção e boa prática em análise de séries temporais com Pandas, pois otimiza e simplifica futuras manipulações e visualizações baseadas no tempo.

Com a função `consulta_bc` pronta, podemos agora iterar sobre nossa lista de indicadores para construir nosso dataset completo. Para automatizar o preenchimento dos argumentos da função, definimos o período de análise através de variáveis e criamos um DataFrame inicialmente vazio, cujas colunas serão populadas com os dados referentes a cada indicador selecionado.

```
data_inicial = '2002-01-01'
data_final = '2025-09-01'

base = pd.DataFrame()

base['PIB'] = consulta_bc(4380, data_inicial, data_final)['valor']
base['IBOV']= consulta_bc(7849, data_inicial, data_final)['valor']
base['IPA'] = consulta_bc(7450, data_inicial, data_final)['valor']
# ... (e assim por diante para todos os indicadores) ...
```

Esta abordagem programática busca resolver os problemas do método manual:

- **Velocidade:** reduz para segundos o que levaria minutos ou horas
- **Confiabilidade:** elimina o risco de erro humano
- **Reprodutibilidade:** permite que qualquer pessoa sem experiencia execute a coleta

5.2.4 Refinando o Dataset: Critérios de Seleção e Limpeza

Ter uma lista de potenciais indicadores é apenas o começo. Um dos passos mais importantes na preparação de dados é garantir a **consistência e a integridade** do conjunto de dados final. Antes de prosseguir para a análise, a lista inicial de variáveis foi submetida a um rigoroso processo de refinamento, onde cada série foi avaliada com base em critérios técnicos. As seguintes decisões foram tomadas:

- **Harmonização da Granularidade Temporal:** A análise de múltiplas séries temporais exige que todas possuam a mesma frequência. Na lista inicial, o "Índice de Confiança do Empresário Industrial"era uma série **trimestral**, enquanto todos os outros indicadores eram **mensais**. Incluir esta variável exigiria a conversão de todas as 15 séries mensais para trimestrais, um processo que resultaria em uma perda drástica de mais de 60% dos pontos de dados. Considerando que as séries já eram relativamente curtas, a decisão foi **remover o indicador trimestral** para preservar a riqueza dos dados mensais.
- **Atualização de Metodologia (Dados Descontinuados):** Séries temporais podem sofrer alterações metodológicas ou serem descontinuadas. Foi o caso das séries originais de "Meios de Pagamento"(M1 e M2), que foram oficialmente descontinuadas em 2018. Para garantir a relevância da análise, estas foram **substituídas pelas novas séries metodológicas** correspondentes.
- **Integridade da Série Histórica:** Uma variável pode ser teoricamente relevante, mas praticamente inviável se seus dados forem incompletos. A série do **IBOV** (Valor das empresas listadas na Bovespa) foi **removida em uma etapa posterior** da análise ao se constatar que estava descontinuada e incompleta no período final do estudo. Este é um exemplo prático de como a qualidade dos dados se sobrepõe à relevância teórica.
- **Definindo a Janela Temporal Comum:** Para garantir que todas as séries sejam comparáveis desde o primeiro ponto, é necessário definir uma data de início comum. Analisando a Tabela 5.1, nota-se que um conjunto significativo de indicadores de produção do IBGE inicia sua série histórica em Janeiro de 2002. Portanto, para assegurar que não haveria dados faltantes no início do período, **a data de início do nosso estudo foi definida como janeiro de 2002**.

Este processo de refinamento é fundamental e ilustra a natureza investigativa do trabalho com dados, garantindo que a base final seja coesa, íntegra e pronta para a próxima etapa: a análise exploratória.

5.3 Análise Exploratória e Transformação dos Dados

Com o nosso DataFrame `base` consolidado, o próximo passo em qualquer projeto de análise é a **Análise Exploratória de Dados (EDA)**. O objetivo desta etapa é "sentir"os dados, entender suas características e, principalmente, identificar problemas que precisam ser corrigidos antes de qualquer modelagem.

5.3.1 O Diagnóstico do Problema de Escala

A primeira e mais intuitiva etapa da EDA é visualizar as séries temporais. Um gráfico simples pode revelar muito sobre a estrutura e os desafios ocultos em um conjunto de dados.

```
fig, ax = plt.subplots(figsize=(10, 5))
base.plot(ax=ax)

ax.set_title('Indicadores Macroeconômicos', fontsize=16)
ax.set_xlabel('Ano', fontsize=12)
ax.set_ylabel('Valor', fontsize=12)
ax.legend(loc='upper left', frameon=True)

plt.show()
```

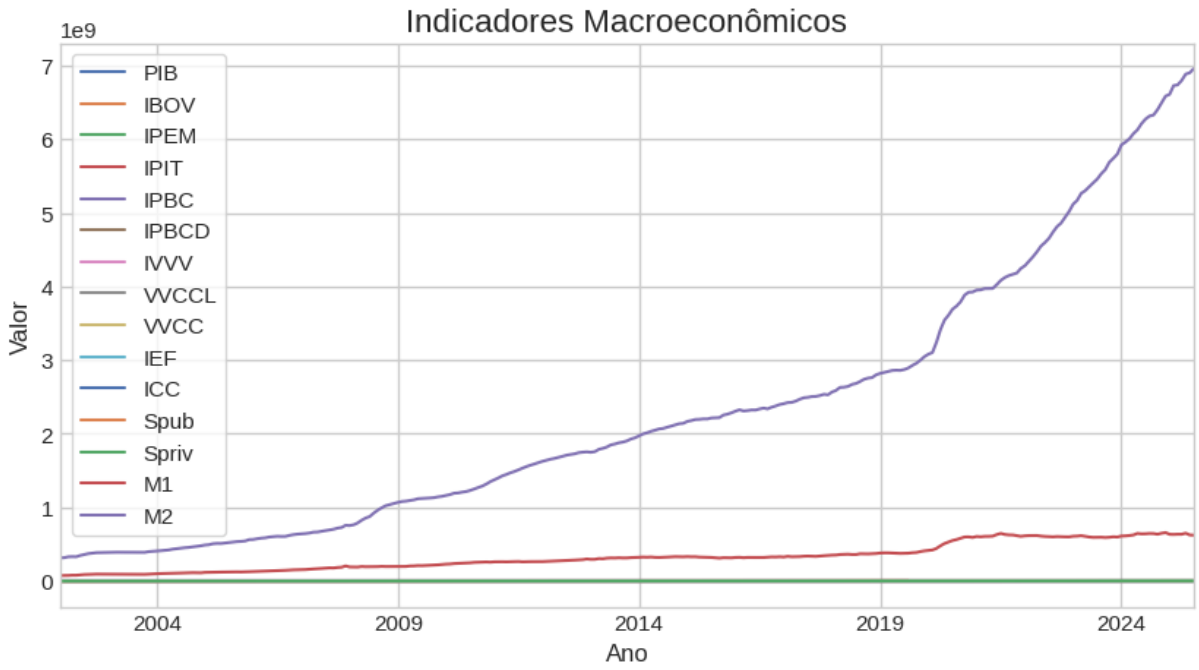

Figura 5.4: Comportamento temporal dos indicadores em escala linear.

O gráfico 5.4 revela um problema clássico em dados macroeconômicos, conhecido como a discrepância de escalas. Variáveis monetárias como `M1` e `M2`, cujos valores estão na casa dos milhões, dominam completamente o eixo vertical. Como resultado, as outras variáveis, que são índices ou variações, aparecem como uma linha achatada e indistinguível próxima ao zero.

```
fig, ax = plt.subplots(figsize=(9, 5))
base.plot(ax=ax)

ax.set_title('Indicadores Macroeconômicos', fontsize=16)
ax.set_xlabel('Ano', fontsize=12)
ax.set_ylabel('Valor (escala logarítmica)', fontsize=12)
ax.set_yscale('log')
ax.legend(loc='upper left', frameon=True, bbox_to_anchor=(1, 1))

plt.show()
```

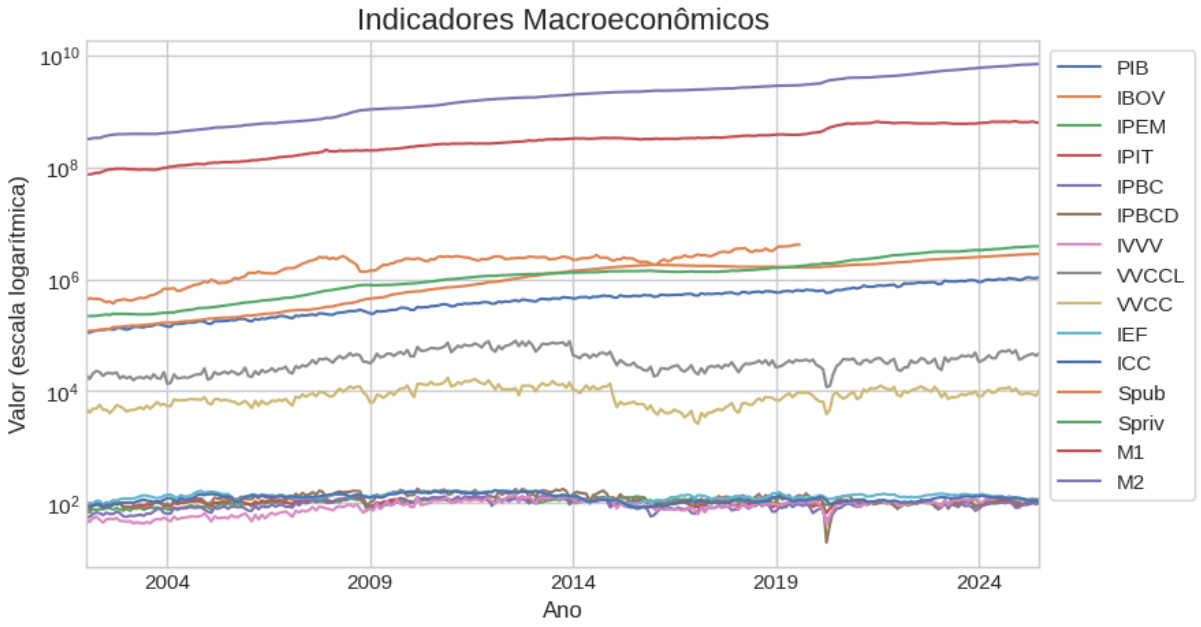

Figura 5.5: Comportamento temporal dos indicadores em escala logarítmica.

Uma técnica comum para visualizar dados com diferentes ordens de magnitude é usar uma escala logarítmica. Através da Figura 5.5 podemos distinguir claramente o comportamento de todas as variáveis, que se agrupam em diferentes ordens de grandeza, refletindo suas distintas unidades de medida:

- **1ª Ordem (10^8 a 10^{10}):** As variáveis `M1` e `M2` (Meios de Pagamento).
- **2ª Ordem (10^5 a 10^7):** O `PIB`, o `IBOV` e os saldos de crédito (`Spub`, `Spriv`).
- **3ª Ordem (10^4):** As vendas de veículos (`VVCC`, `VVCCL`).
- **4ª Ordem (10^2):** O restante dos indicadores, que são majoritariamente números-índice.

Essa visualização deixa claro que estamos comparando "laranjas com melancias". O problema, no entanto, não é apenas visual, já que ao calcularmos a matriz de correlação de Pearson dos dados brutos, a distorção se torna estatisticamente evidente.

Antes de analisar a matriz, é importante entender o que ela representa. A **matriz de correlação** é uma tabela que mostra o **coeficiente de correlação de Pearson** entre todas as combinações de variáveis de um dataset. Este coeficiente é um número que varia de -1 a +1:

- **+1: Correlação positiva perfeita.** Quando uma variável sobe, a outra sobe na mesma proporção.
- **0: Nenhuma correlação linear.** As variáveis não se relacionam, ou seja, o crescimento de uma não diz nada sobre o crescimento da outra.
- **-1: Correlação negativa perfeita.** Quando uma variável sobe, a outra desce na mesma proporção.

Em um **heatmap** (mapa de calor) como o da Figura 5.6, cores quentes (vermelho/laranja) indicam correlação positiva, cores frias (azul) indicam correlação negativa, e cores neutras (próximas ao branco) indicam baixa correlação.

```
pearson_base = base.corr()

plt.figure(figsize=(12, 10))
sns.heatmap(pearson_base, annot=True, cmap='coolwarm', fmt=".2f",
   linewidths=.5)
plt.title('Mapa de Calor dos Indicadores', fontsize=16)
plt.show()
```

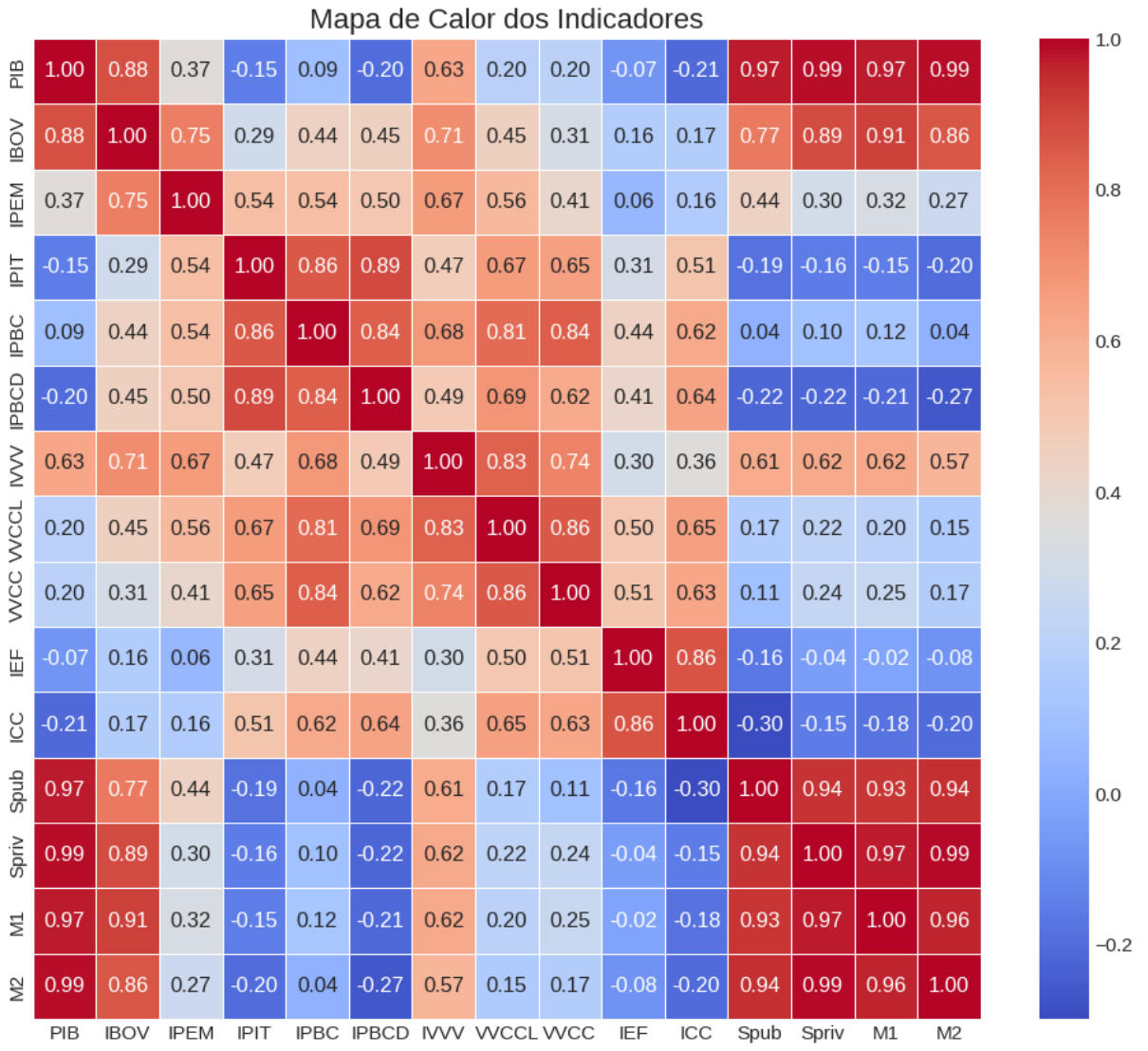

Figura 5.6: Matriz de correlação de Pearson dos indicadores

O mapa de calor da Figura 5.6 sugere correlações altíssimas (cores quentes intensas) entre o PIB e as variáveis monetárias. No entanto, esta é uma correlação espúria. Ela existe não por uma relação econômica direta, mas porque todas essas séries compartilham uma mesma tendência de crescimento ao longo do tempo e possuem ordens de magnitude semelhantes. Para construir um modelo preditivo robusto, precisamos isolar as relações de curto prazo, e para isso, a normalização dos dados é indispensável.

5.3.2 Normalização das Variáveis

A análise dos dados brutos deixou claro que as diferentes escalas e as tendências de longo prazo distorcem a análise de correlação e visualização. Para que as variáveis se tornem comparáveis e para que possamos focar nas dinâmicas de curto prazo — que são mais relevantes para prever o próximo passo da economia — precisamos normalizar os dados.

A inspiração para a solução veio de um dos próprios indicadores, o **IPA (Índice de Preços por Atacado)**, que já é naturalmente medido em variação percentual. Esta unidade tem a vantagem de ser adimensional e de refletir a *mudança* relativa de um período para o outro, em vez de seu nível absoluto. Decidiu-se, portanto, aplicar essa mesma transformação a todas as outras séries.

Para esta tarefa, a biblioteca Pandas oferece o método `.pct_change()`. Com uma única linha de código, ele calcula a variação percentual de cada elemento em relação ao elemento anterior da série. O indicador IPA já é um dado de variação percentual, portanto essa variável não é incluida diretamente na construção do DataFrame `base`. Para incluí-lo no dataframe já normalizado, é importante se atentar a unidade de medida, já que o método `.pct_change()` retorna um valor decimal, ou seja, é preciso normalizar a variável IPA transformando de porcentagem para decimal.

```
# 1. Calcular a variação percentual para todas as colunas
base_retorno = base.pct_change()

# 2. Inserir a coluna 'IPA' (dividida por 100 para normalizar)
base_retorno.insert(loc=1, column='IPA', value=base_IPA/100)

# 3. Tratar a primeira linha (NaN)
base_retorno.iloc[0] = 0

# 4. Remover a coluna 'IBOV' incompleta
base_retorno = base_retorno.drop(['IBOV'], axis=1)
```

Desvendando o Processo:

- `.pct_change()`: A função percorre cada coluna e, para cada linha, calcula:

$$\frac{valor\_atual - valor\_anterior}{valor\_anterior}$$

  O resultado é uma nova série de dados que representa a taxa de crescimento de cada indicador em relação a observação (mês) anterior.

- **Tratando o `NaN` inicial:** A primeira linha de `base_retorno` se torna `NaN` (Nulo), pois não há um ”mês anterior”para o cálculo da variação da primeira observação. A decisão de preencher com `0` é uma convenção comum, assumindo que não houve variação no ponto de partida da nossa análise.

- **Remoção Final:** A remoção (`.drop()`) da coluna `IBOV` aqui é o passo final de limpeza, garantindo que apenas séries completas e íntegras prossigam para a modelagem.

Com esta transformação, nosso DataFrame deixa de representar os valores absolutos dos indicadores e passa a representar suas **taxas de variação mensais**. Agora, a pergunta que analisaremos não é mais ”quão grande é a produção industrial?”, mas sim ”a produção industrial está crescendo ou diminuindo, e com que intensidade?”.

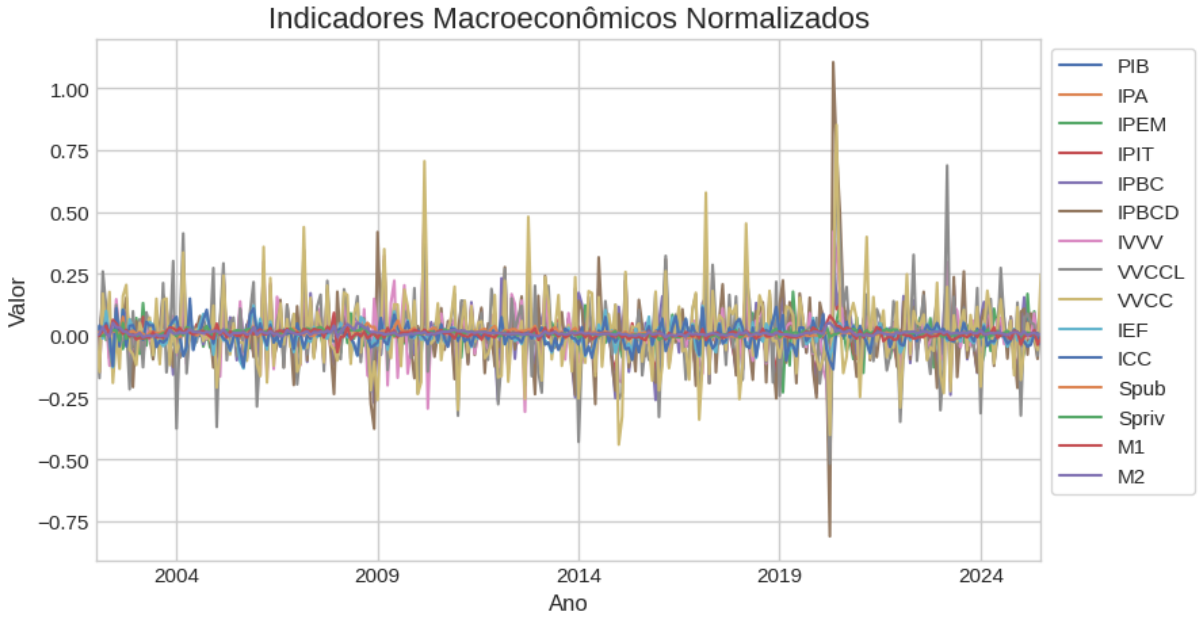

Figura 5.7: Comportamento temporal dos indicadores normalizados

É possível verificar na Figura 5.7 que todas as séries agora oscilam em torno de um eixo central em zero e suas variações são diretamente comparáveis. Onde antes tínhamos um gráfico ilegível, agora temos uma visualização clara da volatilidade de cada indicador. O excesso de informações ainda torna o gráfico pouco informativo mas suficiente para garantir que os dados agora são comparáveis

Analisando o novo mapa de calor, a figura 5.8 revela um cenário muito mais realista e útil para a análise. As correlações espúrias e infladas desapareceram. Agora podemos observar que:

- Indicadores de Atividade Econômica, como IPIT (Indústria de Transformação) e VVCC (Vendas de Caminhões), mantêm a correlação positiva mais forte com a variação do PIB.
- Indicadores de Confiança (IEF, ICC) possuem uma correlação positiva, porém mais moderada.
- Os Indicadores Monetários, que antes pareciam dominantes, agora mostram uma correlação bem mais fraca, confirmando que sua relação anterior era um artefato da escala e da tendência.

Isso nos oferece insights muito mais plausíveis do ponto de vista econômico, podendo notar alguns padrões interessantes:

- **Alta Correlação Intra-Categoria:** Como esperado, muitos indicadores dentro da mesma categoria estão fortemente correlacionados entre si. Por exemplo, os diversos índices de produção industrial (IPIT, IPBC, IPBCD) e de vendas de veículos (IVVV, VVCCL, VVCC) mostram uma forte correlação positiva. Isso faz sentido, pois eles medem facetas diferentes da mesma dimensão da economia: a atividade do setor privado.

- **A Dinâmica dos Indicadores Monetários:** Curiosamente, a sua observação está correta: essa alta correlação intra-grupo não se repete com a mesma intensidade para os indicadores monetários (Spub, Spriv, M1, M2). Eles mostram uma correlação mais fraca entre si. Isso sugere que eles capturam dinâmicas diferentes. Por exemplo, o crédito ao setor público (Spub) e ao setor privado (Spriv) podem não se mover em perfeita sincronia, refletindo diferentes políticas de crédito ou apetites de risco.

- **Relação com o PIB**: Mais importante, agora podemos ver as correlações mais genuínas com a variação do PIB. Indicadores de Atividade como IPIT e VVCC mantêm a correlação mais forte, confirmando seu papel como importantes termômetros da economia. Em contraste, os indicadores monetários mostram uma relação bem mais fraca, validando nossa hipótese de que a correlação anterior era espúria."

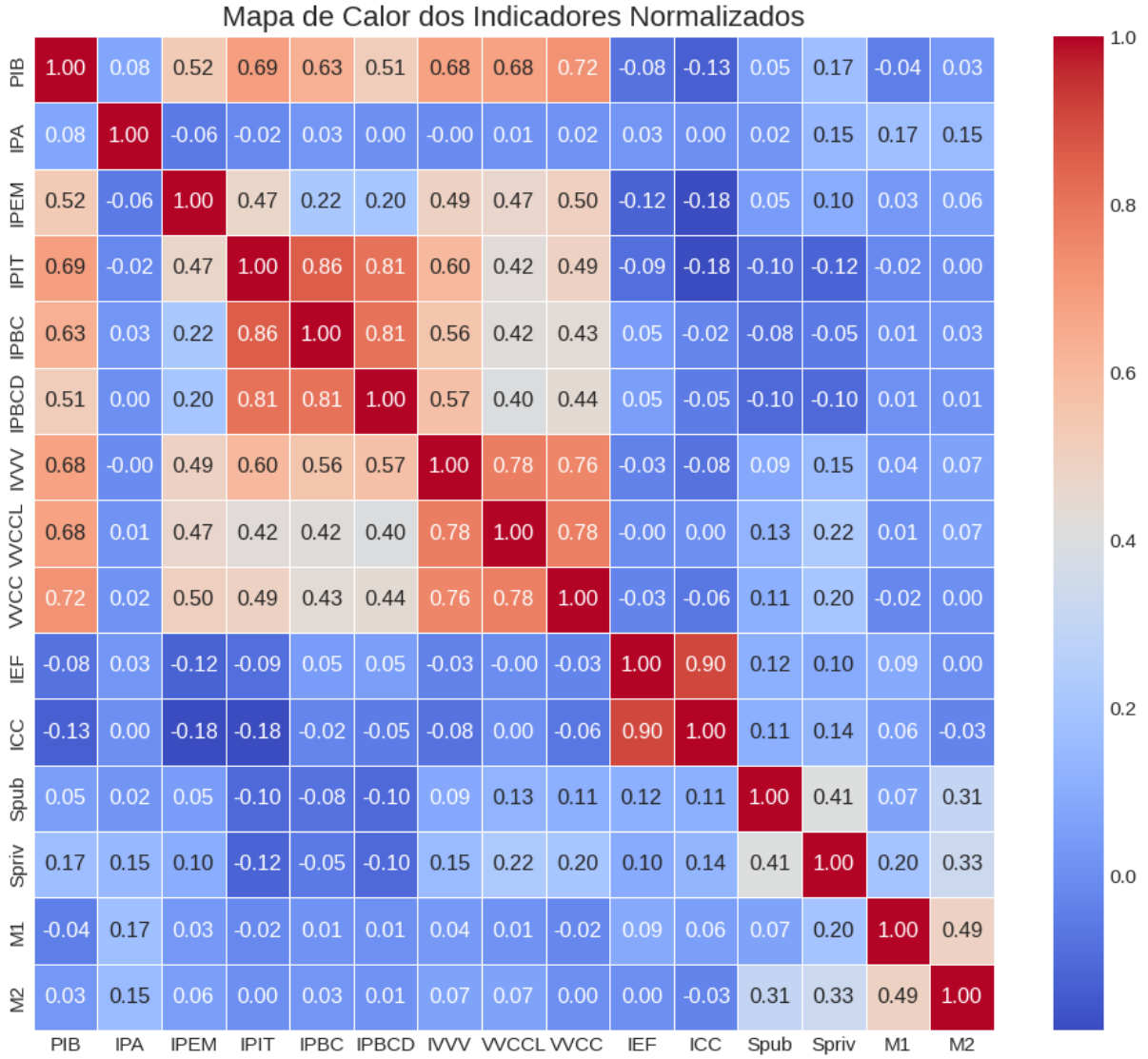

Figura 5.8: Matriz de correlação de Pearson dos indicadores normalizados

Com os dados devidamente transformados e validados, concluímos a etapa de preparação e estamos prontos para a engenharia de features da nossa variável-alvo.

Conhecendo um pouco mais!

Além da variação percentual mensal `.pct_change()`, que é ideal para analisar a volatilidade e as mudanças de curto prazo, outra técnica comum em análise financeira e econômica é o cálculo do retorno acumulado, utilizando a função `.cumprod()`.

```
base_retorno_acumulado = (1 + base_retorno).cumprod()
```

Esta transformação responde a uma pergunta diferente: ”Se tivéssemos investido R$1,00 em cada um desses indicadores no início do período, qual teria sido a evolução desse valor ao longo do tempo?”. O resultado é um índice que começa em 1 e mostra o crescimento (ou queda) acumulado de cada série.

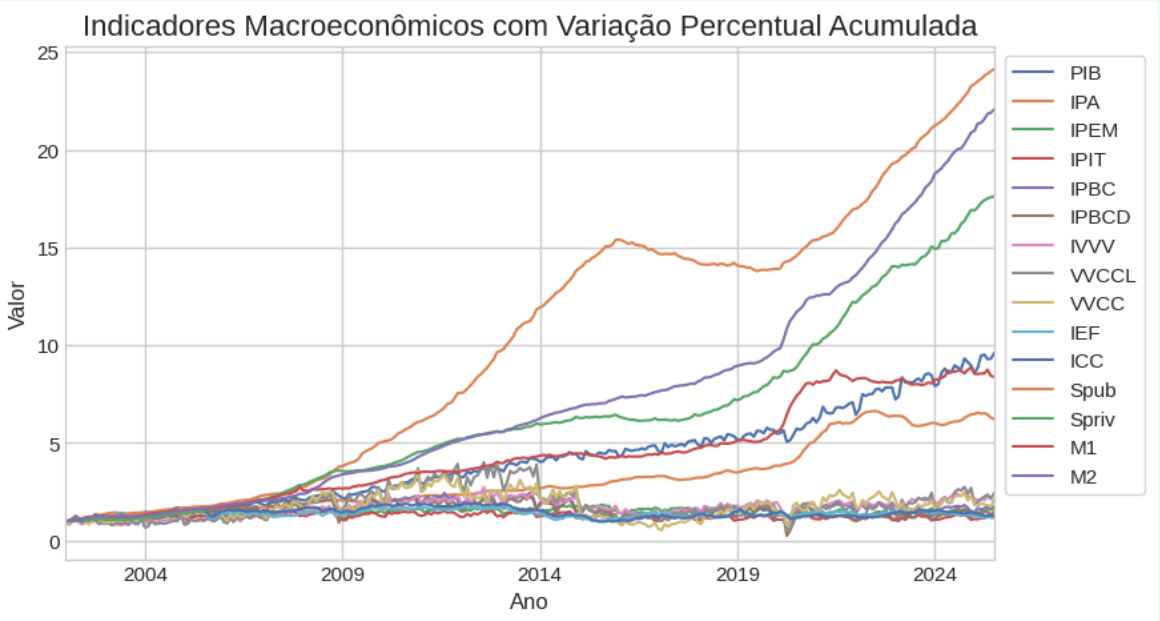

Figura 5.9: Comportamento temporal dos indicadores com variação percentual acumulada

Embora seja uma visualização poderosa para entender o crescimento de longo prazo, para o nosso objetivo específico de prever a variação do mês seguinte, a abordagem com `.pct_change()` é mais direta e informativa, pois foca nas dinâmicas de curto prazo. Por essa razão, seguiremos com o DataFrame `base_retorno` em nossas próximas análises.

5.4 Engenharia de Features: A Discretização das Variáveis

Após normalizarmos as séries temporais, enfrentamos uma questão central para o nosso objetivo: o que significa ”prever o comportamento do PIB”? A variação percentual que calculamos é uma variável praticamente contínua, mas um tomador de decisão (um economista, um investidor) muitas vezes necessita de indicadores categóricos, que no nosso contexto podem ser interpretados como ”Alta”, ”Queda”ou ”Estabilidade”do indicador.

Para alinhar a análise a este objetivo prático, precisamos transformar a variável-alvo contínua (a variação do PIB) em um conjunto de classes discretas. Este processo, conhecido

como discretização ou binning, é uma etapa crucial de engenharia de features que prepara os dados para modelos de classificação.

Antes de definirmos qualquer regra de discretização, o primeiro passo é entender a natureza da variável que queremos transformar. Uma análise estatística prévia da distribuição da variação percentual do `PIB` nos dará a linha de base para avaliarmos o impacto de nossas futuras transformações. Podemos iniciar com um resumo através do método `.describe()`.

```
print(base_retorno['PIB'].describe())
```

```
count    283.000000
mean       0.008767
std        0.038995
min       -0.105067
25%       -0.014812
50%        0.004924
75%        0.030188
max        0.134265
Name: PIB, dtype: float64
```

A tabela de saída nos fornece métricas precisas, como a média (`mean`) de aproximadamente **0.88%** e a mediana (`50%`) de **0.49%**. Notamos também a presença de eventos extremos, com uma queda máxima (`min`) de **10.5%** e uma alta recorde (`max`) de **13.4%** em um único mês. Para uma compreensão mais intuitiva da distribuição e dos outliers, a visualização gráfica torna-se uma etapa indispensável.

```
# Criando uma figura com dois subplots, lado a lado
fig, axes = plt.subplots(1, 2, figsize=(10, 4))

# Título geral para a figura inteira
fig.suptitle('Análise da Distribuição da Variação Mensal do PIB',
   fontsize=16)

# Gráfico 1: Histograma
sns.histplot(data=base_retorno, x='PIB', kde=True, ax=axes[0])
axes[0].set_title('Histograma da Distribuição')
axes[0].set_xlabel('Variação Percentual')
axes[0].set_ylabel('Frequência')
axes[0].xaxis.set_major_formatter(PercentFormatter(1, decimals=0))

# Gráfico 2: Boxplot
sns.boxplot(data=base_retorno, x='PIB', ax=axes[1])
axes[1].set_title('Boxplot da Dispersão e Outliers')
axes[1].set_xlabel('Variação Percentual')
axes[1].xaxis.set_major_formatter(PercentFormatter(1, decimals=1))

plt.show()
```

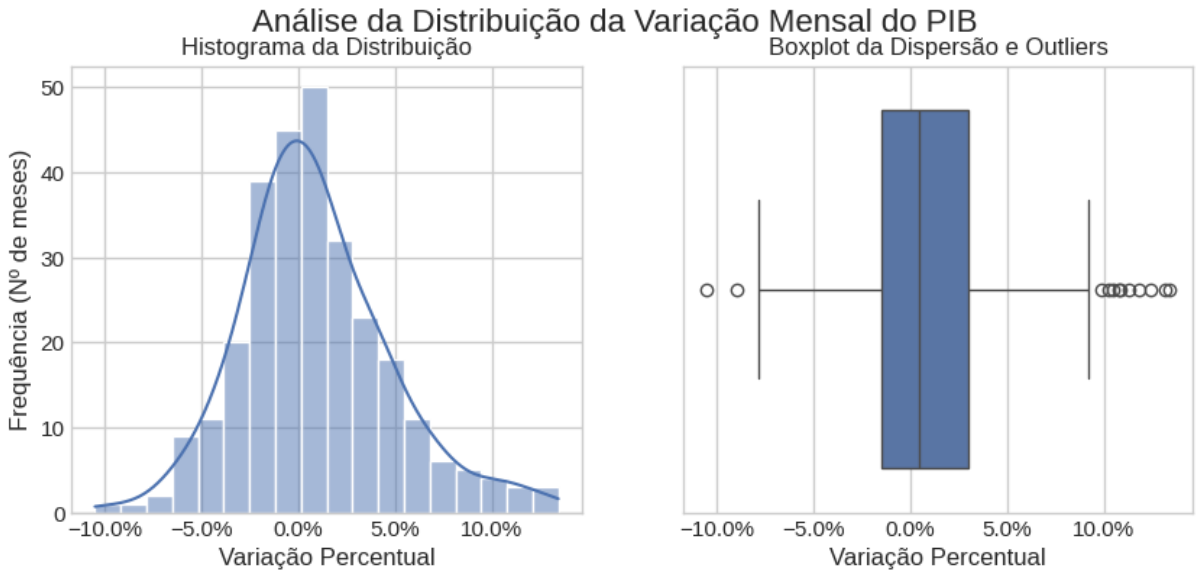

Figura 5.10: Distribuição estatística da variação mensal do PIB.

Analisando a figura 5.10 o histograma confirma que a distribuição se assemelha a uma curva normal (”formato de sino”), com a maioria das variações mensais concentradas em torno da média. O boxplot é ainda mais revelador: ele não só mostra a dispersão dos dados — com 50% dos meses variando entre -1.48% (25%) e +3.01% (75%) — mas também destaca claramente a presença de múltiplos outliers. São esses pontos fora dos ”bigodes”do gráfico que representam os meses de variações extremas. É com base nesta distribuição e na existência desses eventos significativos que definiremos nossas regras de discretização na próxima seção.

5.4.1 Discretização Binária e Suas Limitações

A forma mais simples e intuitiva de converter a variação contínua do PIB em classes é através de uma discretização binária. A pergunta fundamental que um tomador de decisão faz é se a economia cresceu ou encolheu. Essa abordagem traduz diretamente essa questão em duas categorias distintas, servindo como um excelente ponto de partida.

$$\begin{cases} x \mapsto 0, & \text{if } \Delta x < 0 \\ x \mapsto 1, & \text{if } \Delta x \geq 0. \end{cases}$$

Qualquer variação percentual mensal do PIB maior ou igual a zero é rotulada como **”Alta”(classe 1)**, e qualquer variação negativa é rotulada como **”Queda”(classe 0)**. Existem diferentes maneiras de implementar essa transformação, como a função `numpy.where()` que aplica seleção condicional de forma vetorizada sem precisar de loops explicitos.

```
# np.where(condição, valor_se_verdadeiro, valor_se_falso)
base_binario['PIB'] = np.where(base_binario['PIB'] >= 0, 1, 0)
```

Após a transformação, é crucial analisar a distribuição das novas classes. Como a classe ”Alta”tem o valor 1 e ”Queda”o valor 0, a média (`mean`) de 0.5618 nos informa diretamente a proporção da classe majoritária. Isso significa que aproximadamente 56% dos meses no período analisado foram de alta, enquanto 44% foram de queda.

```
count    283.000000
mean       0.561837
std        0.497040
min        0.000000
25%        0.000000
50%        1.000000
75%        1.000000
max        1.000000
Name: PIB, dtype: float64
```

Observando a descrição estatística das classes, conseguimos perceber previamente que o diagrama de BoxPlot não é muito informativo quando temos poucas classes.

Como a classe ”Alta”tem o valor 1 e ”Queda”o valor 0, a média (`mean`) de 0.5618 nos informa diretamente a proporção da classe majoritária. Isso significa que aproximadamente 56% dos meses no período analisado foram de alta, enquanto 44% foram de queda. O histograma apresentado no gráfico 5.11 ilustra explcitamente esse desbalanceamento de classes.

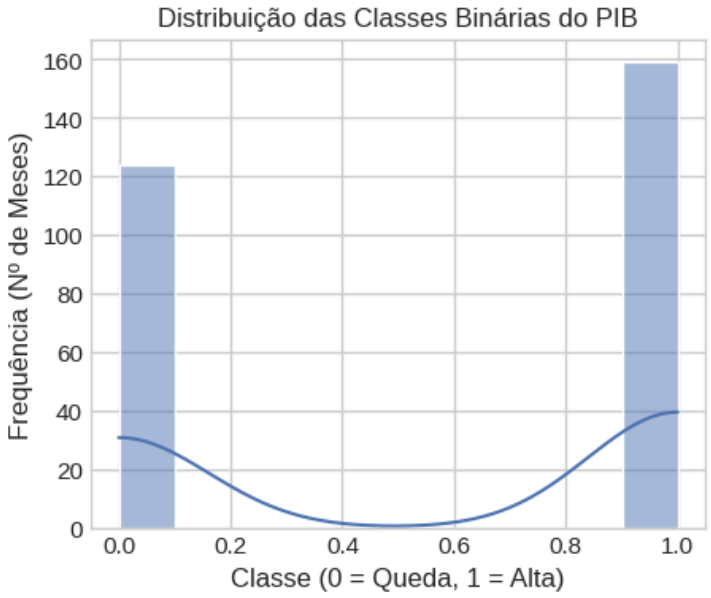

Figura 5.11: Distribuição das classes após a discretização binária.

Analisando de forma geral essa primeira abordagem baseada em apenas 2 classes, podemos perceber algumas vantagens e limitações:

- **Vantagem:** A principal vantagem é a **simplicidade**. O critério é fácil de entender e implementar, e os resultados iniciais superam uma previsão aleatória.
- **Limitação:** A abordagem é conceitualmente **simplista**. Ela trata uma variação de +0.01% da mesma forma que uma de +5.0%. Para um tomador de decisão, a **magnitude** da variação é crucial. O objetivo não é prever qualquer ruído em torno de zero, mas sim identificar **movimentos relevantes**.

Para superar essa limitação, precisamos de uma abordagem que consiga isolar esses movimentos significativos, o que nos leva à um critério de discretização baseada em estatística.

5.4.2 Discretização Baseada em 3 Classes

A principal limitação da abordagem binária é sua incapacidade de distinguir movimentos relevantes de pequenas flutuações. Para um tomador de decisão, é crucial filtrar os ”ruídos”e focar nos sinais de tendência. Para isso, precisamos de um critério que defina uma ”zona de normalidade”ou ”estabilidade”.

$$\begin{cases} x \mapsto -1, & \text{if } \Delta x \leq \mu_x - \sigma_x; \\ x \mapsto 0, & \text{if } \mu_x - \sigma_x < \Delta x < \mu_x + \sigma_x; \\ x \mapsto 1, & \text{if } \Delta x \geq \mu_x + \sigma_x. \end{cases}$$

A abordagem estatística utiliza a média (μ) e o desvio padrão (σ) da variação do PIB. Com base nas propriedades da **distribuição normal**, sabemos que aproximadamente **68%** das observações se concentram no intervalo entre $\mu - \sigma$ e $\mu + \sigma$. Usamos este intervalo para definir nossa classe de ”Estabilidade”, como ilustra a Figura 5.12.

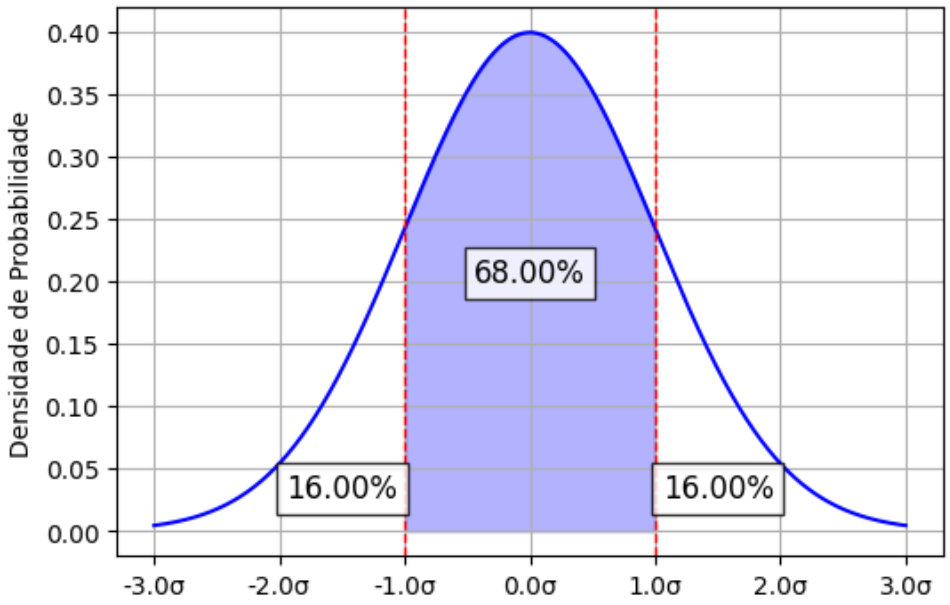

Figura 5.12: Distribuição normal com os limiares de $\pm 1\sigma$, definindo as 3 classes de análise

Para calcular os limiares que separam as classes, podemos utilizar métodos diretos como `numpy.mean()` (média) e `numpy.std()` (desvio padrão). No processo de rotulagem podemos utilizar o método `numpy.select()` que consegue lidar com multiplas condicionais a partir de uma lista.

```
alta = base_3classes['PIB'].mean() + base_3classes['PIB'].std()
queda = base_3classes['PIB'].mean() - base_3classes['PIB'].std()

condicoes = [
    base_3classes['PIB'] >= alta,
    base_3classes['PIB'] <= queda]
classes = [1, -1] # 1:Alta, -1:Queda, 0:Estabilidade

base_3classes['PIB'] = np.select(condicoes, classes, default=0)
```

A análise da nova variável nos mostra como os 283 meses do nosso estudo foram distribuídos entre as três classes.

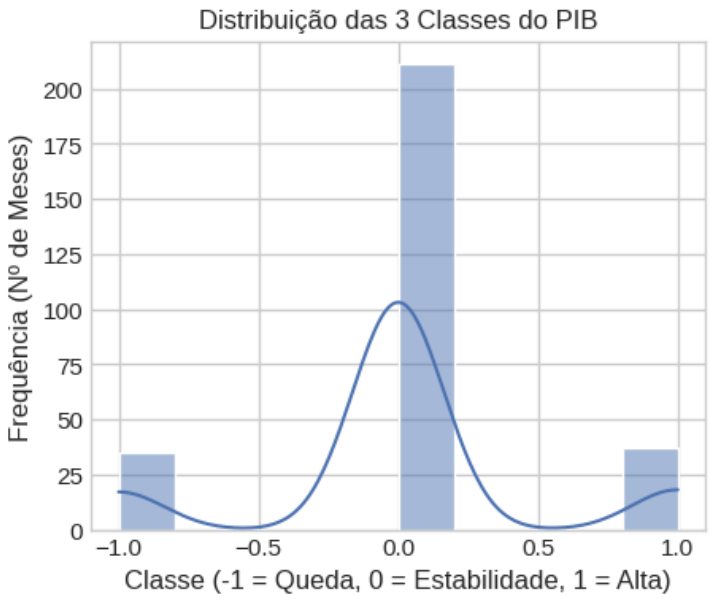

Figura 5.13: Distribuição das 3 classes do PIB após a discretização.

O histograma 5.13 ilustra perfeitamente o resultado da nossa regra estatística. A classe **"Estabilidade"(0)** é o evento mais frequente, concentrando a maioria dos meses onde a variação do PIB foi considerada "normal"ou dentro do esperado. As classes **"Queda"(-1)** e **"Alta"(1)** são menos frequentes, capturando os eventos de cauda da distribuição. Elas representam os meses em que a economia se desviou significativamente da sua tendência média, sinalizando movimentos economicamente relevantes.

Apesar de corrigir o problema conceitual da discretização binária em relação ao ruído gerado por pequenas variações, essa abordagem também tem suas limitações:

- **Vantagem:** Esta abordagem é estatisticamente robusta e muito mais útil para a tomada de decisão, já que filtra os ruídos e permite que a análise e os futuros modelos de machine learning se concentrem nos eventos que realmente sinalizam uma mudança de ciclo econômico.
- **Limitação:** Embora superior, esta abordagem ainda agrupa altas moderadas com altas muito fortes na mesma classe. Para uma análise mais granular que busque diferenciar a intensidade dos movimentos, uma modelagem que considera mais classes é necessária.

5.4.3 Discretização Baseada em 5 Classes

Para uma análise mais granular, a abordagem anterior pode ser extendida a 5 classes, permitindo diferenciar não apenas a direção do ciclo econômico, mas também sua **intensidade**.

$$\begin{cases} x \mapsto -2, & \text{if } \Delta x \leq \mu_x - 2 \cdot \sigma_x; \\ x \mapsto -1, & \text{if } \mu_x - 2 \cdot \sigma_x < \Delta x \leq \mu_x - \sigma_x; \\ x \mapsto 0, & \text{if } \mu_x - \sigma_x < \Delta x < \mu_x + \sigma_x; \\ x \mapsto 1, & \text{if } \mu_x + \sigma_x \leq \Delta x < \mu_x + 2 \cdot \sigma_x; \\ x \mapsto 2, & \text{if } \Delta x \geq \mu_x + 2 \cdot \sigma_x. \end{cases}$$

O raciocínio aplicado é o mesmo, mas agora considerando novos limiares em **dois desvios padrão ($\pm 2\sigma$)** distantes da média para isolar os eventos extremos, como mostra a Figura 5.14.

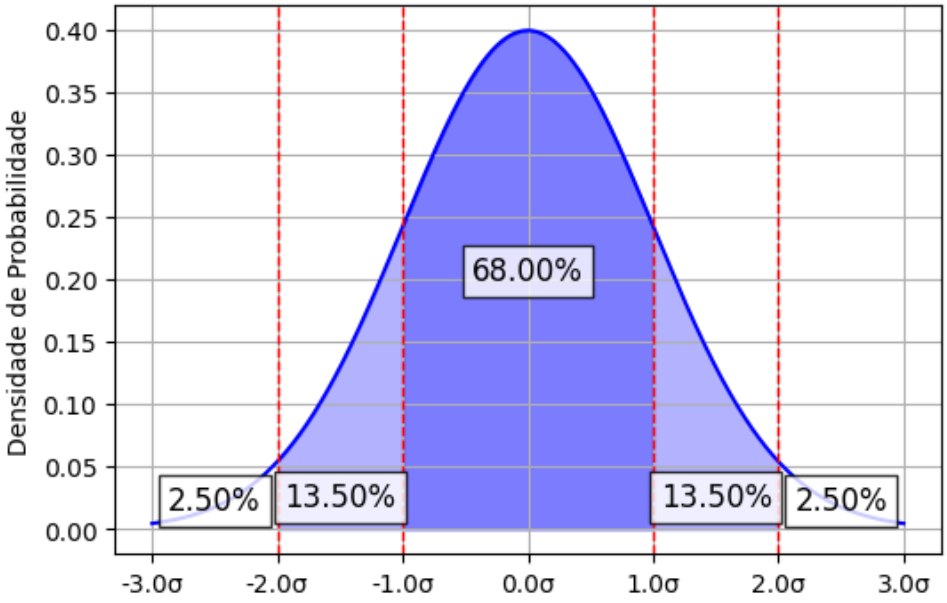

Figura 5.14: Discretização em 5 classes, com limiares em $\pm 1\sigma$ e $\pm 2\sigma$.

Essa segmentação cria as classes ”Queda Forte”(-2), ”Queda Moderada”(-1), ”Estabilidade”(0), ”Alta Moderada”(1) e ”Alta Forte”(2), no entanto é importante salientar que variações superiores a $\pm 2\sigma$ podem considerados eventos relativamente raros, especialmente em sistemas pouco voláteis. Portanto, é importante investigar a distribuição das classes para verificar se esses limiares sugeridos são uma representação adequada para o fenômeno estudado. Investigando a distribuição das 5 classes para a variável PIB, a Figura 5.15 mostra um desbalanceamento entre as classes, já que a ”Queda Forte”é muito mais incomum que a ”Alta Forte”.

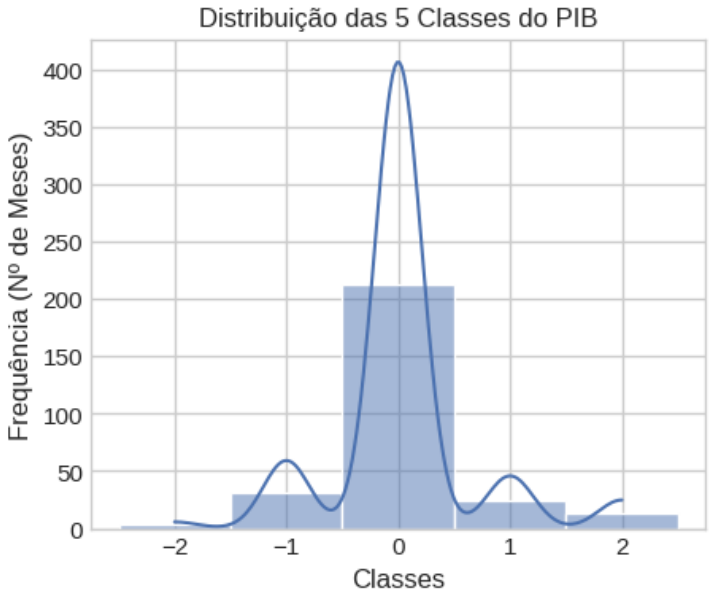

Figura 5.15: Distribuição das 5 classes do PIB após a discretização.

- **Vantagem:** A abordagem sugerida possuí a capacidade de **diferenciar a magnitude** dos eventos econômicos, oferecendo insights potencialmente mais ricos para a tomada de decisão.
- **Limitação e Ponto de Reflexão:** O histograma demonstra que as classes extremais (-2 e 2) são eventos raros, acentuando o **desbalanceamento de classes**, um desafio para o treinamento de modelos. Adicionalmente, é crucial refletir se esta regra, ao ser aplicada a variáveis menos voláteis, poderia resultar em **classes vazias**, um ponto de atenção que deve ser investigado durante o pré-processamento para evitar problemas de víes nos modelos de previsão.

Depois de analisar diversas abordagens, é possível observar que cada umas das estratégias apresentadas tem seus pontos fortes e fracos, podendo ou não ser aceitável a depender do contexto. Verificamos uma construção gradual onde se sugere a solução mais simples possível, identifica suas limitações e (se necessário) aplica melhorias. Modelos mais sofisticados podem corriger alguns problemas e criar outros, portanto, é fundamental sempre fazer um balanço entre efetividade e interpretabilidade.

5.5 Análise Complementar e Boas Práticas com Dados

Ao longo deste capitulo, seguimos o fluxo de tarefas essencial para coleta e tratamento de um dado relativamente comportado, separando as etapas e tentando não se aprofundar em detalhes muito específicos. No entanto, a análise de dados pode ser muito vasta, já que temos diferentes metodologias e ferramentas para abordar os problemas, buscando garantir a integridade e automação das análises. Esta seção aprofunda alguns conceitos e ferramentas de boas práticas essenciais no arsenal de um cientista de dados.

5.5.1 Backup e Reprodutbilidade de Dados

Em um projeto de ciência de dados, é considerada uma boa prática criar ”fotografias”ou backups dos seus datasets em diferentes estágios. Essa prática, conhecida como versionamento de dados, é crucial para garantir a reprodutibilidade da análise — a capacidade de obter exatamente os mesmos resultados ao executar o mesmo código no futuro. Na arquitetura do nosso projeto representada na Figure 5.1, o bloco ”Arquivos CSV”representa esses pontos de salvamento.

Fontes de dados ”vivas”, como APIs, podem sofrer alterações. Dados econômicos, em particular, frequentemente passam por revisões retroativas ou correções pela fonte original. Se realizarmos a coleta hoje e novamente daqui a um mês, sutis diferenças nos dados históricos podem surgir. Para evitar que essas mudanças inesperadas afetem nossa análise, salvamos uma cópia local dos dados brutos assim que os coletamos.

```
base.to_csv('dados.csv', index=True, sep=';', encoding='utf-8')
```

O método `.to_csv()` é usado para exportar o conteúdo de um DataFrame para um arquivo de texto no formato CSV. Embora simples, seus parâmetros nos dão controle total sobre o arquivo de saída.

- `'dados_brutos.csv'`: O primeiro argumento é o caminho e o nome do arquivo que será criado.
- `index=True`: Este é um parâmetro crucial. Se True (o padrão), ele salva a coluna de índice do DataFrame no arquivo. Se False, o índice é descartado. A escolha depende se o seu índice contém informações importantes (como datas, que é o nosso caso).
- `sep=';'`: Define o separador (ou delimitador) que será usado para separar os valores em cada linha, onde o padrão é a vírgula (,).
- `encoding='utf-8'`: Especifica a codificação de caracteres do arquivo. Usar 'utf-8' é uma boa prática universal, pois garante que caracteres especiais e acentos (comuns na língua portuguesa) sejam salvos e lidos corretamente em qualquer sistema.

Após etapas relevante de processamento é recomendável salvar o DataFrame atualizado, criando uma espécie de "checkpoint"que poupa o retrabalho de re-executar todas as etapas de limpeza a cada nova sessão de análise. De maneira análoga, processos que envolver várias etapas podem ser modularizados e trabalhados em paralelo, desde que as informações possam fluir atualizadas entre as camadas. Para carregar os dados um arquivo salvo localmente em formato CSV para um DataFrame Pandas, podemos utilizar o método `pd.read_csv()`.

```
base_retorno = pd.read_csv('dados_normalizados.csv', index_col='
    data', parse_dates=True, sep=';')
```

- `'backup/dados_normalizados.csv'`: O caminho para o arquivo a ser lido.
- `index_col='data'`: Informa ao Pandas que a coluna chamada 'data' no arquivo CSV deve ser usada como o índice do nosso novo DataFrame. Isso é fundamental para restaurar a estrutura original da nossa série temporal.
- `parse_dates=True`: Um parâmetro extremamente útil. Ele instrui o Pandas a tentar converter o índice (ou outras colunas, se especificadas) para o formato de data/hora (datetime), o que é essencial para nossas análises temporais.
- `sep=';'`: Assim como no salvamento, precisamos informar ao Pandas qual é o separador utilizado no arquivo para que ele possa dividir as colunas corretamente.

Embora o formato CSV seja universal, a biblioteca Pandas é equipada para ler e escrever dados em diversos outros formatos (Excel, JSON, SQL, Parquet, Feather, etc...), cada um com suas próprias vantagens. Conhecer essas alternativas é crucial para escolher a ferramenta certa para cada necessidade do projeto.

Quando estamos trabalhando com uma instalação local do Python, os arquivos são salvos e/ou carregados no próprio disco, a partir do diretório onde o script está armazenado, ou ainda definindo o caminho manualmente na função. Por outro lado, quando estamos trabalhando em um ambiente em nuvem (como o Google Colab), muitas vezes é necessária conectar-se a um diretório externo para garantir a persistencia dos dados.

Quando estamos trabalhando com uma instalação local do Python, os arquivos são salvos e/ou carregados no próprio disco, a partir do diretório onde o script está armazenado, ou ainda definindo o caminho manualmente na função. Por outro lado, quando estamos trabalhando em um ambiente em nuvem (como o Google Colab), muitas vezes é necessária conectar-se a um diretório externo para garantir a persistencia dos dados.

Fique Alerta!

O arquivo salvo no ambiente padrão do Google Colab é temporário. Se sua sessão for desconectada por inatividade ou se você fechar a aba, a máquina virtual é desligada e **todos os arquivos salvos nela são permanentemente apagados**. A estratégia de salvar localmente no Colab só é segura para uso imediato dentro da mesma sessão.

5.5.2 Análise de Distribuições Assimétricas: O Caso da Variável `M2`

Uma análise exploratória completa não se encerra na visão geral do dataset. É crucial investigar a distribuição de variáveis individuais, especialmente aquelas que se comportam de maneira atípica. Ao contrário do PIB, cuja variação percentual se mostrou razoavelmente simétrica, o agregado monetário `M2` apresenta um comportamento bastante assimétrico, conforme podemos observar na Figura 5.16.

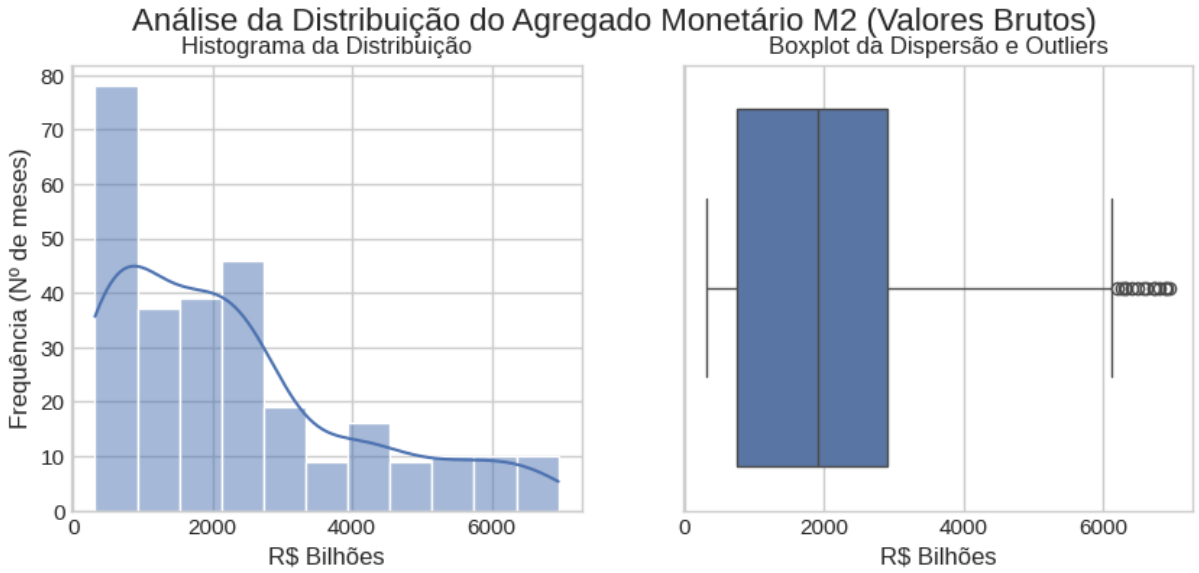

Figura 5.16: Análise da distribuição do agregado monetário M2 (valores brutos).

O histograma (esquerda) mostra uma forte assimetria, com uma longa cauda à direita. O boxplot (direita) confirma isso, identificando muitas observações como "outliers". Este padrão é característico de muitas séries temporais econômicas, como a base monetária de um país, que em valores nominais tendem a crescer continuamente ao longo do tempo, caracterizando uma série com forte tendência, ou não-estacionária. Utilizar uma variável com uma distribuição tão enviesada diretamente em uma análise de correlação ou ainda em um modelo de predição pode lhe atribuir um peso desproporcional e levar a conclusões equivocadas.

A transformação em variação percentual (`.pct_change()`) é uma das técnicas para tratar séries com fortes tendências. O seu objetivo principal não é criar uma distribuição perfeitamente simétrica, mas sim remover a tendência e tornar a série aproximadamente estacionária. Ao normalizar as variáveis e alterar o foco da análise dos níveis absolutos da `M2` para suas taxas de crescimento mensais, temos como consequencia dados cujas propriedades estatísticas são muito mais estáveis, como pode ser observado na Figura 5.17.

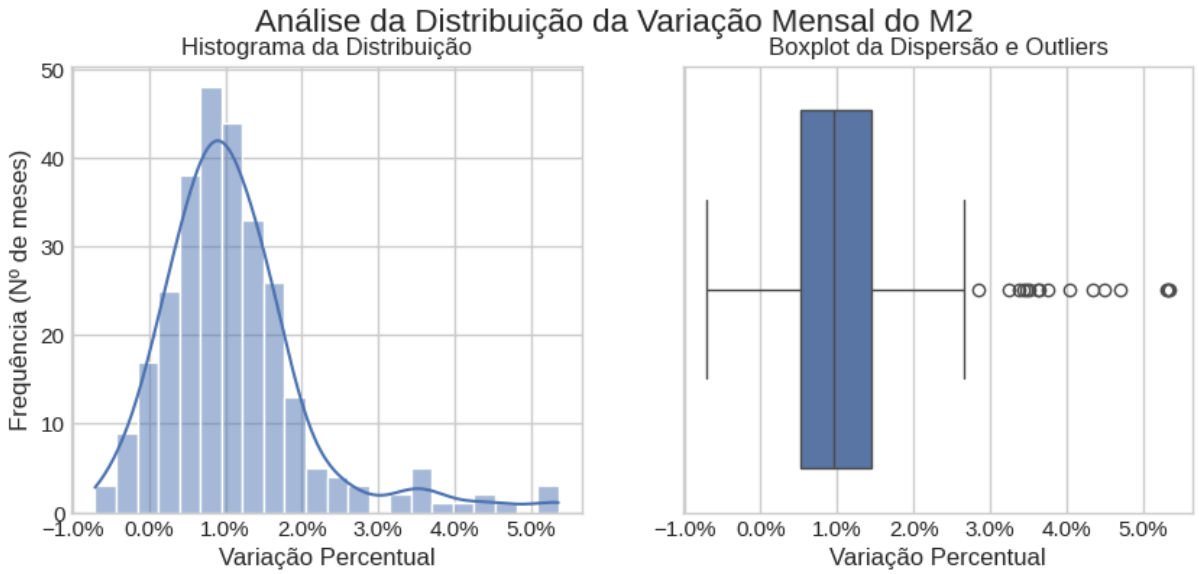

Figura 5.17: Análise da distribuição da variação mensal do M2 (após normalização).

- A média ainda é ligeiramente positiva, o que faz sentido economicamente, pois a base monetária tende a se expandir mais do que se contrair. O importante é que a média agora é estável ao longo do tempo, em vez de crescente.
- A distribuição ainda possui uma leve assimetria e ”caudas pesadas”(outliers), características comuns em dados financeiros e econômicos.

No entanto, a série transformada é bem superior à original para fins de modelagem. A transformação foi bem-sucedida não por ter criado uma curva Gaussiana perfeita, mas por ter convertido uma série não-estacionária, com tendência e variância crescentes, em uma série aproximadamente estacionária, cujas flutuações ocorrem em torno de uma média constante. É essa estabilidade que a torna uma variável válida e muito mais confiável para ser comparada com as outras e utilizada como feature em um modelo preditivo.

A verdadeira importância de entender a distribuição de uma variável se revela quando aplicamos etapas de engenharia de features, como a discretização. Vimos que a `M2` original possui uma forte assimetria. Embora a transformação `.pct_change()` tenha tornado a série mais estacionária, a natureza intrínseca da variável (poucas quedas acentuadas) ainda persiste e tem um impacto direto quando consideramos o efeito da discretização. Considerando o modelo baseado em 5 classes conforme Figura 5.18, podemos observar:

- **Desbalanceamento Extremo de Classes:** A classe ”Estabilidade”(0) domina esmagadoramente a distribuição. As classes de variação moderada (-1 e 1) são muito menos frequentes, e as de variação extrema (2) são raríssimas. Um modelo de Machine Learning treinado com dados tão desbalanceados terá uma forte tendência a prever sempre a classe majoritária, ignorando os eventos raros, que são justamente os mais importantes de se prever.
- **O Problema da Classe Vazia:** A observação mais crítica é a ausência completa da classe -2”(Queda Forte). Isso ocorre porque, no período analisado, a variação mensal da M2 nunca foi negativa o suficiente para cruzar o limiar de $\mu - 2\sigma$.

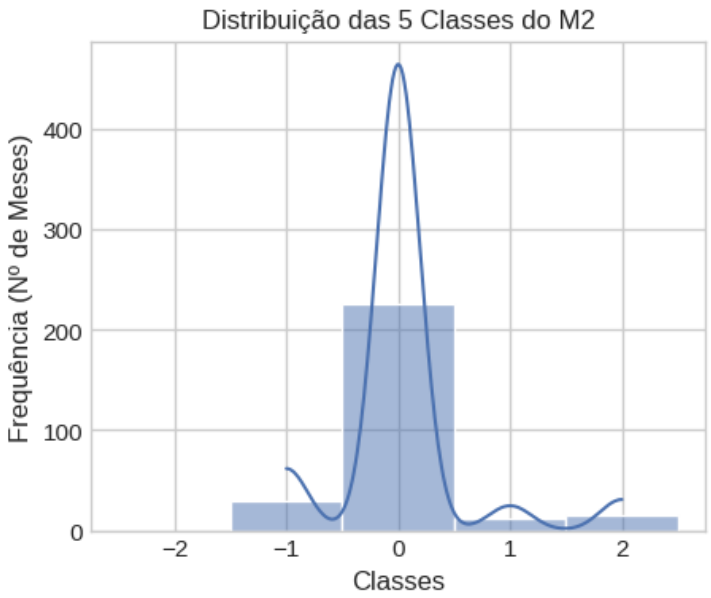

Figura 5.18: Distribuição das 5 classes para a variável `M2` normalizada.

Este é um exemplo prático e fundamental do dia a dia de um cientista de dados. A mesma regra de engenharia de features, quando aplicada a variáveis com distribuições diferentes, pode gerar resultados drasticamente distintos. A presença de classes vazias em uma variável explicativa é um ”sinal de alerta”. Ela nos informa que, para esta variável específica (`M2`), o modelo não terá nenhum exemplo de como ela se comporta durante eventos de ”Queda Forte”, o que pode limitar seu poder preditivo em cenários de crise.

Esta análise não invalida a variável, mas nos força a prosseguir com um entendimento mais profundo de suas limitações, um conhecimento crucial para interpretar corretamente os resultados do modelo de Machine Learning.

5.5.3 Técnicas Alternativas de Normalização: Z-score e Min-Max

Em nosso estudo de caso foi utilizada a **variação percentual (`.pct_change()`)** para transformar os dados afim de torna-los comparáveis. Existem diversas técnicas para essa finalidade, mas quando se trata da preparação de dados para a maioria dos algoritmos de Machine Learning, destacam-se a **Padronização (Z-score)** e a **Normalização Min-Max**.

O objetivo dessas técnicas é colocar todas as variáveis (features) em uma escala comum, o que é fundamental para o bom desempenho de muitos algoritmos sensíveis à escala dos dados, como redes neurais e SVMs.

A **Padronização (Z-score)** também conhecida como `StandardScaler` , é a técnica mais robusta e utilizada. Ela transforma os dados para que a nova distribuição tenha uma **média de 0** e um **desvio padrão de 1**. Sua principal vantagem é a robustez a outliers. Para cada valor x na série, o Z-score é calculado através da expressão:

$$Z = \frac{(x - \mu)}{\sigma}$$

onde μ é a média e σ é o desvio padrão da série original.

A **Normalização Min-Max** chamada de `MinMaxScaler`, redimensiona os dados para um intervalo fixo, geralmente entre **0 e 1**. É útil para algoritmos que exigem dados em um

intervalo específico, mas é sensível a outliers. Para cada valor x, o novo valor x' é calculado como:

$$x' = \frac{(x - \min(X))}{(\max(X) - \min(X))}$$

onde $\min(X)$ e $\max(X)$ são os valores mínimo e máximo da série original.

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler

valores_zscore = StandardScaler().fit_transform(base)
base_zscore = pd.DataFrame(valores_zscore,
                           columns=base.columns,
                           index=base.index)

valores_minmax = MinMaxScaler().fit_transform(base)
base_minmax = pd.DataFrame(valores_minmax,
                           columns=base.columns,
                           index=base.index)
```

5.5.4 Codificação de Variáveis Categóricas (One-Hot Encoding)

Quando foi realizada a etapa de discretização da variável PIB, algumas classes numéricas (-1, 0, 1) foram criadas para representar o comportamento da variável. Muitos algoritmos de inferência estatística ou Machine Learning podem interpretar isso de forma equivocada, assumindo uma falsa relação ordinal e de magnitude. O modelo pode aprender, por exemplo, que a classe ”Alta”(1) é matematicamente ”maior”que a ”Estabilidade”(0), ou ainda que Queda (-1) + Alta (1) = Estabilidade (0), o que não faz sentido economicamente.

Para remover essa ambiguidade, é considerada uma boa prática utilizar o conceito **One-Hot Encoding**, uma técnica que transforma uma única coluna contendo `N` categorias em `N` novas colunas, uma coluna para cada categoria. Cada uma dessas novas colunas é binária (`0` ou `1`), indicando a presença (`1`) ou ausência (`0`) daquela categoria na observação original. No Pandas, a maneira mais fácil de aplicar esta técnica é com a função `pd.get_dummies()`.

```
pd.get_dummies(base_3classes["PIB"], prefix='PIB', dtype=int)
```

	PIB			PIB_-1	PIB_0	PIB_1
data			data			
2017-01-01	-1		2017-01-01	1	0	0
2017-02-01	0	⇒	2017-02-01	0	1	0
2017-03-01	1		2017-03-01	0	0	1
2017-04-01	-1		2017-04-01	1	0	0
2017-05-01	0		2017-05-01	0	1	0

A coluna original foi transformada em três novas colunas, eliminando qualquer relação de ordem entre as classes e representando cada categoria de forma independente. No entanto, em certos casos a criação de muitas colunas pode causar um problema conhecido como ”maldição da dimensionalidade”, tornando o modelo computacionalmente caro e lento, além de aumentar a tendência a overfitting e multicolinearidade. Métodos de inferência lineares ou baseados

em distância tendem a sofrer influencia mais significativa desse tipo de abordagem, como a Regressão Logística, K-Vizinhos Próximos, Máquina de Vetores de Suporte e Redes Neurais.

Em contrapartida, modelos baseados em árvores não são muito impactados devido a sua característica inerentemente não-linear, cuja tomada de decisão é baseada em cortes sequênciais nos dados. A distinção entre categorias não se relaciona com a magnitude, tornando métodos como Árvores de Decisão, Floresta Aleatória e eXtreme Gradient Boost, menos sensíveis a relação ordinal e alta dimensionalidade.

Conhecendo um pouco mais!

A biblioteca `ydata-profiling` é uma ferramenta poderosa para analisar um DataFrame, criando automaticamente um relatório HTML interativo que compila informações sobre as variáveis e suas relações. Por não se tratar de uma biblioteca nativa do Python, é necessário que seja instalada no sistema antes de utiliza pela primera vez .

```
pip install ydata-profiling
```

O relatório contém um resumo das estatísticas gerais do dataset, bem como das variáveis individuais. Além disso é possível verificar de forma interativa os diagramas de dispersão das variáveis 2 a 2 bem como o mapa de calor, os dados faltantes e amostras do dataset. O relatório pode ser incorporado dentro do próprio notebook ou exportado em HTML.

```
from ydata_profiling import ProfileReport

relatorio = ProfileReport(base_retorno, title="Relatório de
   Análise", html={'style':{'full_width':True}})

relatorio.to_notebook_iframe() # Visualização no notebook
relatorio.to_notebook_to_file() # Exportação para arquivo HTML
```

Por ser uma biblioteca externa é preciso se atentar a sua atualização e manutenção, especialmente quando se utiliza um ambiente em nuvem cuja instalação deve ser repetida sempre que uma nova sessão é iniciada. Embora seja uma ferramenta que acelera e padroniza as análises, é crucial dominar as etapas intermediárias manualmente para conseguir realizar a exploração manualmente em caso de necessidade.

5.6 Aplicando seus Conhecimentos

Agora que você acompanhou todo o fluxo de trabalho de um estudo de caso real, é hora de colocar a mão na massa! Os exercícios a seguir formam um projeto completo, projetado para que você possa praticar, aprofundar os conceitos e desenvolver seu senso crítico como cientista de dados.

Exercício 1: Validação da Coleta e Expansão do Dataset

a) Acesse o portal do SGS do Banco Central e utilize a Tabela 5.1 como guia para pesquisar os indicadores e seus respectivos códigos.

b) Utilizando a função `consulta_bc`, faça a importação dos dados e compare através da API e compare com os valores exibidos na tabela do portal do SGS. Eles são consistentes?

c) **Desafio:** Pesquise no portal por outros indicadores de seu interese, que acredite serem relevantes para prever o PIB (como por exemplo emprego, juros ou inflação).

Exercício 2: Comparando Técnicas de Normalização

a) Normalize as variáveis utilizando a **Padronização (Z-score)** e a **Normalização Min-Max**, verifique o comportamento temporal (gráfico) e o mapa de calor.

b) Considerando o nosso objetivo (analisar o comportamento do ciclo econômico), qual das três normalizações você considera a mais adequada e por quê? A resposta está relacionada ao conceito de estacionariedade.

c) **Desafio:** Inclua os novos indicadores escolhidos no Exercício 1, verifique se o método de normalização se mantém viável e justifique.

Exercício 3: Implementação da Discretização com 5 Classes

a) Para cada uma das variáveis explicativas, faça uma análise estatística utilizando funções e gráficos. Os indicadores escolhidos são comparáveis ou suas distribuições possuem comportamento muito distinto?

b) Aplique a regra de discretização de 5 classes (com os limiares de $\mu \pm 1\sigma$ e $\mu \pm 2\sigma$) a todas as variáveis. Verifique se alguma delas resultou em classes vazias.

c) **Desafio:** Crie uma função que receba um DataFrame e um dicionário com as regras de discretização e retorne o DataFrame inteiro com todas as colunas discretizadas de uma só vez.

Exercício 4: Investigando Intervalos de Discretização

a) Modifique a regra de discretização de 5 classes, alterando os limiares ($\mu \pm 0.6745\sigma$ e $\mu \pm 1.645\sigma$) e compare o balanceamento das classes resultante em relação a abordagem "Clássica".

b) Quais as vantagens e limitações de usar intervalos mais curtos em relação aos intervalos padrão da abordagem Clássica? Pense no trade-off entre o número de observações na classe "Estabilidade"e a sensibilidade do modelo para capturar movimentos menos intensos.

c) **Desafio:** Proponha uma regra de discretização baseada em quartis ou outro critério interesse, e justifique por que ela poderia ser interessante para este problema.

Exercício 5: Construindo um Modelo Preditivo

a) Com os dados devidamente preparados, separe as variáveis explicativas (`X`) da variável-alvo (`y`) e construa os conjuntos de treinamento, validação e teste.

b) Utilize algum algoritmo de classifição para estimar o `y` e avalie sua performance.

c) Compare a performance dos diferentes cenários de preparação de dados que foram testadas nos exercícios anteriores:

- As diferentes estratégias de normalização discutidas como variação percentual, `Z-score` e `Min-Max`.
- Os diferentes intervalos de discretização considerando 3 ou 5 classes com os limiares clássicos ou modificações nos intervalos.
- A abordagem baseada em classes numéricas e o `One-Hot Encoding`.

Fique Alerta!

Os exercicios sugeridos seguem uma arquitetura estruturada, onde cada parâmetro cria um cenário de análise único. Após prototipar a solução básica, é sugerido criar uma metodologia para comparar os inúmeros cenários propostos, facilitando também uma ánalise qualitativa. O objetivo maior não é necessariamente encontrar a melhor solução, mas compreender o processo de análise e organização baseado no método científico, conseguindo utilizar dados para a tomada de decisão. Como fonte de inspiração e até mesmo validação, podemos citar [20, 2, 3, 22, 21], trabalhos que reforçam o processo de melhoria contínua baseada na observação e reflexão.

Apêndice A

O Motor por Trás dos Cálculos: Álgebra Linear com NumPy

Nas seções anteriores, usamos o Pandas para organizar e o Matplotlib para visualizar nossos dados. Agora, vamos dar um passo atrás e entender o "motor" que faz tudo isso funcionar de forma eficiente: a biblioteca NumPy (*Numerical Python*). A Álgebra Linear é o ramo da matemática que lida com vetores e matrizes (que podemos pensar como listas de números e tabelas de números). Em ciência de dados, nossos *datasets* são exatamente isso:

- Uma coluna (como `crescimento_kg`) é um vetor.
- Um *dataset* inteiro (com várias colunas numéricas) é uma matriz.

O NumPy é a biblioteca que permite ao Python realizar operações matemáticas nessas estruturas de forma incrivelmente rápida. Na verdade, a biblioteca Pandas que tanto usamos é, em grande parte, construída sobre o NumPy.

A.1 Operações Fundamentais com Arrays

A.1.1 Criando Arrays

A forma mais comum de criar um array é a partir de uma lista Python, usando a função `np.array()`.

Copie e Teste!

```
# 1. Criando um array simples (vetor)
lista_simples = [10, 20, 30, 40]
array_simples = np.array(lista_simples)

print(f"Array simples: {array_simples}")
print(f"Tipo do array: {type(array_simples)}")

# 2. Criando um array de 2 dimensões (matriz)
lista_de_listas = [[1, 2, 3], [4, 5, 6]]
array_2d = np.array(lista_de_listas)

print(f"\nArray de 2D (Matriz):\n{array_2d}")
```

```
print(f"Formato (shape) da matriz: {array_2d.shape}") # (linhas,
    colunas)
```

Tela do Terminal

```
Array simples: [10 20 30 40]
Tipo do array: <class 'numpy.ndarray'>
Array de 2D (Matriz):
[[1 2 3]
[4 5 6]]
Formato (shape) da matriz: (2, 3)
```

A.1.2 Operações Vetorizadas

Aqui está a principal diferença entre um *array* NumPy e uma *lista* Python. Quando você usa operadores matemáticos (+, *) em listas, o Python as "concatena". Quando você usa os mesmos operadores em arrays NumPy, ele realiza a operação matemática elemento por elemento.

Copie e Teste!

```
# --- Dados para o exemplo ---
array_a = np.array([10, 20, 30])
array_b = np.array([1, 2, 3])

lista_a = [10, 20, 30]
lista_b = [1, 2, 3]

# --- 1. Adição ---
print("--- Adição (+) ---")
print(f"Com Listas Python: {lista_a + lista_b}  (Concatenou!)")
print(f"Com Arrays NumPy:  {array_a + array_b}  (Somou elemento
    por elemento!)")

# --- 2. Multiplicação por um número (Escalar) ---
print("\n--- Multiplicação por 2 (*) ---")
print(f"Com Listas Python: {lista_a * 2}  (Repetiu a lista!)")
print(f"Com Arrays NumPy:  {array_a * 2}  (Multiplicou cada
    elemento!)")

# --- 3. Multiplicação entre arrays ---
print("\n--- Multiplicação (Array * Array) ---")
print(f"Com Arrays NumPy:  {array_a * array_b}  (Multiplicou
    elemento por elemento!)")
```

Tela do Terminal

```
--- Adição (+) ---
Com Listas Python: [10, 20, 30, 1, 2, 3] (Concatenou!)
Com Arrays NumPy: [11 22 33] (Somou elemento por elemento!)

--- Multiplicação por 2 (*) ---
Com Listas Python: [10, 20, 30, 10, 20, 30] (Repetiu a
lista!)
Com Arrays NumPy: [20 40 60] (Multiplicou cada elemento!)

--- Multiplicação (Array * Array) ---
Com Arrays NumPy: [10 40 90] (Multiplicou elemento por
elemento!)
```

A.1.3 Análise da Vetorização

Os resultados do código acima são a demonstração prática da vetorização:

- `lista_a + lista_b` resultou em `[10, 20, 30, 1, 2, 3]`.
- `array_a + array_b` resultou em `[11 22 33]`.

O NumPy entende que você quer aplicar a operação matemática em todos os elementos da estrutura de uma só vez, sem a necessidade de um *loop* `for`. É exatamente esse comportamento que o Pandas utiliza quando você faz `df['crescimento_kg'] * 100` ou `df['coluna_A'] + df['coluna_B']`.

A.2 Fatiamento e Indexação de Arrays (Slicing)

"Como eu acesso um elemento específico ou um subconjunto de dados (um 'pedaço' do meu array)?" Agora que sabemos criar *arrays* e operar neles, precisamos aprender a selecionar os dados que nos interessam. Assim como as listas Python, os *arrays* NumPy podem ser "fatiados" (*sliced*). A sintaxe é muito similar, mas com o poder da vetorização. A sintaxe básica de fatiamento é `array[inicio:fim:passo]`.

- `inicio`: O índice onde o "corte" começa (inclusivo).
- `fim`: O índice onde o "corte" termina (exclusivo).
- `passo`: De quantos em quantos elementos pular (opcional).

Copie e Teste!

```python
# --- 1. Fatiamento Básico (1D) ---
print("--- Fatiamento em Array 1D ---")
array_teste = np.arange(10) # Gera um array de 0 a 9: [0, 1, 2, 3,
    4, 5, 6, 7, 8, 9]
print(f"Array original: {array_teste}")
```

```python
# Pega os elementos do índice 2 até o índice 5 (sem incluir o 5)
print(f"Índices [2:5]:  {array_teste[2:5]}")

# Pega do início até o índice 4 (sem incluir o 4)
print(f"Índices [:4]:   {array_teste[:4]}")

# Pega do índice 6 até o final
print(f"Índices [6:]:   {array_teste[6:]}")

# Pega o array inteiro, pulando de 2 em 2
print(f"Índices [::2]:  {array_teste[::2]}")

# --- 2. Indexação e Fatiamento em Matrizes (2D) ---
print("\n--- Fatiamento em Matriz 2D ---")
array_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(f"Matriz original:\n{array_2d}")

# A sintaxe é [linha, coluna]
# Pega o elemento na linha 0, coluna 2
print(f"\nElemento [0, 2]: {array_2d[0, 2]}")

# Pega a linha inteira de índice 1
print(f"Linha inteira [1, :]: {array_2d[1, :]}") # O ':' significa
    'todos os elementos'

# Pega a coluna inteira de índice 0
print(f"Coluna inteira [:, 0]: {array_2d[:, 0]}")

# Pega um "bloco" (linhas 0 e 1, colunas 1 e 2)
print(f"Bloco [0:2, 1:3]:\n{array_2d[0:2, 1:3]}")
```

Tela do Terminal

```
--- Fatiamento em Array 1D ---
Array original: [0 1 2 3 4 5 6 7 8 9]
Índices [2:5]: [2 3 4]
Índices [:4]: [0 1 2 3]
Índices [6:]: [6 7 8 9]
Índices [::2]: [0 2 4 6 8]

--- Fatiamento em Matriz 2D ---
Matriz original:
[[1 2 3]
[4 5 6]
[7 8 9]]

Elemento [0, 2]: 3
Linha inteira [1, :]: [4 5 6]
Coluna inteira [:, 0]: [1 4 7]
```

```
Bloco [0:2, 1:3]:
[[2 3]
[5 6]]
```

A.2.1 Análise do Fatiamento

A sintaxe de fatiamento é a linguagem que usamos para consultar nossos *arrays* e matrizes. Por que isso é importante para a Ciência de Dados?

1. **Base para o Pandas:** A forma como selecionamos dados no Pandas (`.iloc`, `.loc`) é diretamente inspirada nesta sintaxe de fatiamento do NumPy.

2. **Preparação para Machine Learning:** Quando formos para o *Machine Learning*, nossos dados serão uma grande matriz. Usaremos essa sintaxe de fatiamento o tempo todo para separar nossas "features" (as colunas de entrada, `X`) do nosso "alvo" (a coluna que queremos prever, `y`).

Com isso, fechamos nossa introdução prática ao NumPy. Entendemos por que ele é rápido (vetorização) e como usá-lo para criar, operar e fatiar *arrays*.

A.3 Conectando a Teoria à Prática

O objetivo deste apêndice não foi oferecer um curso exaustivo sobre o tema, mas sim garantir que você, leitor, possua as ferramentas conceituais e notacionais necessárias para navegar pelo conteúdo principal deste livro. Vimos os conceitos fundamentais da Álgebra Linear que servem como alicerce para as técnicas discutidas neste livro. Partimos das definições básicas de vetores e matrizes, exploramos suas operações aritméticas e compreendemos seu significado geométrico. A Álgebra Linear é a "linguagem" na qual grande parte da ciência de dados, da física moderna, da computação gráfica e da engenharia é expressa.

Referências Bibliográficas

[1] ALMEIDA, Marcus. **Pandas Python**: o que é, para que serve e como instalar. Alura, 2023. Disponível em: `https://www.alura.com.br/artigos/pandas-o-que-e-para-que-serve-como-instalar`. Acesso em: 23 fev. 2025.

[2] ARAUJO, Antonio Marcos Teixeira de; PALHARES JÚNIOR, Eduardo. **O efeito da discretização na classificação: um estudo comparativo de técnicas de aprendizagem supervisionada para caracterização de variáveis econômicas**. 2024. Trabalho de Conclusão de Curso (Pós-Graduação em Aprendizado de Máquina) - Projeto Aranouá, Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, Manaus, 2024.

[3] ARAUJO, Luiz Eduardo Santos de; PALHARES JÚNIOR, Eduardo. **Discretização e seu efeito na classificação: um estudo comparativo de intervalos não-usuais de discretização para caracterização de variáveis econômicas**. 2024. Trabalho de Conclusão de Curso (Pós-Graduação em Aprendizado de Máquina) - Projeto Aranouá, Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, Manaus, 2024.

[4] BEHRMAN, Kennedy R. **Fundamentos de Ciência de Dados**. Porto Alegre: Bookman, 2022.

[5] BRUCE, Peter; BRUCE, Andrew; GEDECK, Peter. **Estatísticas práticas para cientistas de dados**: mais de 50 conceitos essenciais usando R e Python. 2. ed. Sebastopol: O'Reilly Media, 2020.

[6] BURNS, A. F.; MITCHELL, W. C. **Measuring Business Cycles**. National Bureau of Economic Research, 1946.

[7] ESTRELLA, A.; MISHKIN, F. S. Predicting U.S. Recessions: Financial Variables as Leading Indicators. **Review of Economics and Statistics**, vol. 80, no. 1, 1998, pp. 45-61.

[8] GÉRON, Aurélien. **Aprenda Machine Learning com Scikit-Learn, Keras e TensorFlow**. 3. ed. Sebastopol: O'Reilly Media, 2023.

[9] GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. Cambridge: MIT Press, 2016. Disponível em: `https://www.deeplearningbook.org`.

[10] HAN, J.; KAMBER, M. **Data Mining**: Concepts and Techniques. 3. ed. San Francisco: Morgan Kaufmann, 2011.

[11] IDRIS, Ivan. **Python Data Analysis**. Birmingham: Packt Publishing, 2014.

[12] MARQUES, Bruno Torres; MARQUES, Leonardo Torres. **Aprendizado de máquina: uma abordagem para descoberta de conhecimento**. [S.l.]: Novas Edições Acadêmicas, 2022.

[13] MATTHES, Eric. **Curso intensivo de Python**: uma introdução prática e baseada em projetos à programação. São Paulo: Novatec Editora, 2023.

[14] MCKINNEY, Wes. **Python para análise de dados**. São Paulo: Novatec Editora, 2018.

[15] MITCHELL, Tom M. **Machine Learning**. New York: WCB/McGraw-Hill, 1997.

[16] MUELLER, John Paul. **Aprendizado de máquina para leigos**. Rio de Janeiro: Alta Books, 2019.

[17] NG, Andrew. **Machine Learning Yearning: technical strategy for AI engineers, in the era of deep learning**. [S.l.]: deeplearning.ai project, 2018. Disponível em: `https://info.deeplearning.ai/machine-learning-yearning-book/`.

[18] NG, Andrew. **Machine Learning Specialization**. [Curso online]. [S.l.]: DeepLearning.AI; Stanford Online, 2022. Disponível em: `https://www.coursera.org/specializations/machine-learning-introduction`.

[19] O'NEIL, Cathy; SCHUTT, Rachel. **Fazendo ciência de dados**: conversa direta do Frontline. 1. ed. Sebastopol: O'Reilly Media, 2013.

[20] PALHARES JÚNIOR, Eduardo.; ARAUJO, Antônio Marcos Teixeira de ; SOUZA, Adriano Honorato de ; SILVA, Noam Gadelha da ; SOUZA, Wenndisson da Silva. ENSEMBLE OF MACHINE LEARNING APPLIED TO ECONOMIC CYCLES ANALYSIS: A COMPARATIVE STUDY USING ANTECEDENT MACROECONOMIC INDICATORS FOR BRAZILIAN GDP PREDICTION CLASSIFICATION. In: **II Conferência Internacional de Políticas Públicas e Ciência de Dados**, 2024, Curitiba. Anais da II Conferência Internacional de Políticas Públicas e Ciência de Dados, 2024, ISBN 978-65-272-0661-3. Disponível em: https://www.even3.com.br. Acesso em: 22 set. 2025

[21] PALHARES JÚNIOR, Eduardo ; PENACHI, Rian ; SOUZA, Adriano Honorato de ; SILVA, Nivaldo Rodrigues da; SOUZA, Wenndisson da Silva ; CARDOSO, Edgard, Golçalves. The effect of discretization on classification: a comparative study of machine learning methods applied to unusual discretization intervals for the characterization and prediction of economic variables. In: **III Conferência Internacional de Políticas Públicas e Ciência de Dados**, Aveiro, 2025.

[22] PALHARES JÚNIOR, Eduardo.; ARAUJO, Antônio Marcos Teixeira de ; SOUZA, Adriano Honorato de ; SILVA, Noam Gadelha da ; SOUZA, Wenndisson da Silva. ENSEMBLE OF MACHINE LEARNING APPLIED TO ECONOMIC CYCLES ANALYSIS: A COMPARATIVE STUDY USING ANTECEDENT MACROECONOMIC INDICATORS FOR BRAZILIAN GDP PREDICTION CLASSIFICATION. **REVISTA BRASILEIRA DE PLANEJAMENTO E DESENVOLVIMENTO**, ISSN: 2317-2363, Curitiba, no prelo, 2025.

[23] PROVOST, Foster; FAWCETT, Tom. **Data Science for Business**: what you need to know about data mining and data-analytic thinking. 1. ed. Sebastopol: O'Reilly Media, 2013.

[24] RASCHKA, Sebastian. **Python Machine Learning: machine learning e deep learning com Python, Scikit-learn e TensorFlow 2**. São Paulo: Novatec, 2021.

[25] RUSSELL, Stuart; NORVIG, Peter. **Artificial Intelligence: a modern approach**. 3. ed. Upper Saddle River: Prentice Hall, 2010.

[26] STOCK, J. H.; WATSON, M. W. Business cycle fluctuations in US macroeconomic time series. **Handbook of Macroeconomics**, vol. 1, 1999, pp. 3–64.

[27] VASILIEV, Yuli. **Python para ciência de dados**: uma introdução prática. 1. ed. São Paulo: Novatec Editora, 2023.

FOTO DE NEMUEL SERETI

ISBN: 978-65-01-80960-1

CITHA

Capacitação e Interiorização em
Tecnologias Habilitadoras na Amazônia