Probabilidade e Inferência Estatística para Ciência de Dados

Magno T. F. Severino

2025-10-18

Índice

Introdução

Notas de aula de um curso de Probabilidade e Inferência para Ciência de Dados. Baseado nas notas de aula do curso MAE5702 do professor Felipe de Queiroz, a quem agradeço pela gentil disponibilização do material.

Parte I

Módulo Probabilidade

TODO: descrever essa parte

1 Introdução: Da Teoria à Prática Analítica

Imagine que você trabalha como Cientista de Dados em uma empresa de streaming de música. Uma pergunta fundamental do negócio é: "Quais são os nossos perfis de usuários? Quem são os ouvintes leais e quem são os esporádicos?". Para responder a isso, você tem acesso aos dados de login de cada usuário, mês a mês.

Como poderíamos definir matematicamente o que significa ser um "ouvinte leal"? Seria alguém que logou todos os meses? Ou alguém que, a partir de certo ponto, nunca mais deixou de logar? E o "ouvinte esporádico"? Seria aquele que, mesmo que desapareça por alguns meses, sempre acaba voltando?

Para responder a essas perguntas de forma precisa e rigorosa, precisamos de uma linguagem formal. Essa linguagem é a **Teoria dos Conjuntos**. Nesta aula, vamos construir o alicerce matemático que nos permitirá não apenas estruturar nosso pensamento analítico, mas também desenvolver as ferramentas para analisar o comportamento de sistemas que evoluem ao longo do tempo. Cada definição e proposição que veremos é um passo em direção à solução do nosso problema.

1.1 A Linguagem dos Dados: Conjuntos e Eventos

Para analisar dados, primeiro precisamos defini-los. A teoria dos conjuntos nos fornece o vocabulário fundamental para isso.

Definição 1.1 (Conjunto)**.** Um conjunto Ω é uma coleção de objetos distintos, que serão denotados por ω.

- $\omega \in \Omega$ (elemento ω pertence ao conjunto Ω).
- $\omega \notin \Omega$ (elemento ω não pertence ao conjunto Ω).

  Perspectiva de Data Science: Pense no conjunto universal Ω como todo o seu **universo de dados**, ou **espaço amostral**. Cada elemento ω é uma **unidade observacional**: um cliente, uma transação, um produto. Por exemplo, Ω pode ser "o conjunto de todos os usuários da nossa plataforma".

Definição 1.2 (Subconjunto)**.** Dizemos que A é um subconjunto de Ω, ou que A está contido em Ω, e denotamos por $A \subseteq \Omega$, se $\forall \omega \in A \rightarrow \omega \in \Omega$.

Perspectiva de Data Science: Um subconjunto é um **segmento de interesse** ou um **evento** dentro do seu universo de dados, geralmente obtido através de um filtro ou consulta.

- Se Ω é o conjunto de todos os usuários, um subconjunto A pode ser: $A = \{\text{usuários do plano Premium}\}$.
- Outro subconjunto B poderia ser: $B = \{\text{usuários que ouviram mais de 100 horas de música no último mês}\}$.

1.2 A Gramática da Análise: Operações com Conjuntos

Com nossos segmentos definidos, precisamos de uma forma de combiná-los e compará-los. As operações com conjuntos são a "gramática" que nos permite realizar análises complexas.

Sejam A, A_1, A_2,... subconjuntos de Ω. Temos as seguintes operações:

1. **Complementar de A**: $A^c = \{\omega \in \Omega : \omega \notin A\}$.
2. **União**: $\bigcup_{i=1}^{n} A_i = \{\omega \in \Omega : \omega \in A_i \text{ para ao menos um } i = 1, 2, ..., n\}$.
3. **Intersecção**: $\bigcap_{i=1}^{n} A_i = \{\omega \in \Omega : \omega \in A_i, \forall i = 1, ..., n\}$.
4. **Diferença**: $A_1 - A_2 = \{\omega \in \Omega : \omega \in A_1, \omega \notin A_2\} = A_1 \cap A_2^c$.
5. **Diferença simétrica**: $A_1 \Delta A_2 = (A_1 - A_2) \cup (A_2 - A_1) = (A_1 \cap A_2^c) \cup (A_1^c \cap A_2)$.

- **Conjunto vazio ($\emptyset$)**: não contém nenhum elemento.

Perspectiva de Data Science: Cada operação corresponde diretamente a uma operação lógica em uma consulta de dados:

- **Intersecção ($A \cap B$)** é a lógica **E (AND)**. Ex: "Usuários do plano Premium **E** que ouviram mais de 100 horas".
- **União ($A \cup B$)** é a lógica **OU (OR)**. Ex: "Usuários do plano Premium **OU** que ouviram mais de 100 horas".
- **Complementar (A^c)** é a lógica **NÃO (NOT)**. Ex: "Usuários que **NÃO** são do plano Premium".

Definição 1.3 (Relações entre Conjuntos)**.**

- Dizemos que A_1 e A_2 são **disjuntos** se $A_1 \cap A_2 = \emptyset$.
- Dizemos que $A_1 = A_2$ se $A_1 \subseteq A_2$ e $A_2 \subseteq A_1$.
- Dizemos que $A_1, A_2, ...$ são **mutuamente disjuntos** se $A_i \cap A_j = \emptyset$, $\forall i \neq j$.

Proposição 1.1 (Lei de De Morgan)**.** *Sejam $A_1, A_2, ...$ subconjuntos de Ω. Então:*

a) $(\bigcup_{i=1}^{\infty} A_i)^c = \bigcap_{i=1}^{\infty} A_i^c$

b) $(\bigcap_{i=1}^{\infty} A_i)^c = \bigcup_{i=1}^{\infty} A_i^c$

Nota: As Leis de De Morgan são extremamente úteis para simplificar consultas lógicas complexas. A negação de uma condição "OU" ampla é o mesmo que exigir que todas as condições "E" individuais sejam falsas.

Demonstração (a):

Precisamos mostrar que $(\bigcup_{i=1}^{\infty} A_i)^c \subseteq \bigcap_{i=1}^{\infty} A_i^c$ e $\bigcap_{i=1}^{\infty} A_i^c \subseteq (\bigcup_{i=1}^{\infty} A_i)^c$.

Parte 1: $(\bigcup_{i=1}^{\infty} A_i)^c \subseteq \bigcap_{i=1}^{\infty} A_i^c$

1. Tome $\omega \in (\bigcup_{i=1}^{\infty} A_i)^c$
2. $\Rightarrow \omega \notin \bigcup_{i=1}^{\infty} A_i$ (Por definição de complementar)
3. $\Rightarrow \omega \notin A_i, \forall i = 1, 2, ...$ (Se não está na união, não está em nenhum conjunto)
4. $\Rightarrow \omega \in A_i^c, \forall i = 1, 2, ...$ (Por definição de complementar)
5. $\Rightarrow \omega \in \bigcap_{i=1}^{\infty} A_i^c$ (Se pertence a todos os complementares, pertence à intersecção deles)

Parte 2: $\bigcap_{i=1}^{\infty} A_i^c \subseteq (\bigcup_{i=1}^{\infty} A_i)^c$

1. Tome $\omega \in \bigcap_{i=1}^{\infty} A_i^c$
2. $\Rightarrow \omega \in A_i^c, \forall i = 1, 2, ...$ (Por definição de intersecção)
3. $\Rightarrow \omega \notin A_i, \forall i = 1, 2, ...$ (Por definição de complementar)
4. $\Rightarrow \omega \notin \bigcup_{i=1}^{\infty} A_i$ (Se não está em nenhum conjunto, não está na união)
5. $\Rightarrow \omega \in (\bigcup_{i=1}^{\infty} A_i)^c$ (Por definição de complementar)

1.3 Análise Dinâmica: Sequências de Conjuntos

Agora, voltamos ao nosso problema original: analisar o comportamento dos usuários *ao longo do tempo*. Para isso, introduzimos o conceito de sequências de conjuntos.

Definição 1.4 (Sequência Monótona)**.** Uma sequência $\{A_n\}_{n\geq 1}$ é dita ser **monótona** se:

i) $A_1 \subseteq A_2 \subseteq A_3 \subseteq ...$ (isto é, A_n é não decrescente, denotado por $A_n \uparrow$).
ii) $A_1 \supseteq A_2 \supseteq A_3 \supseteq ...$ (isto é, A_n é não crescente, denotado por $A_n \downarrow$).

O limite de uma sequência monótona é denotado por:

i) se $A_n \uparrow$, $\lim_{n\to\infty} A_n = \bigcup_{i=1}^{\infty} A_i$.
ii) se $A_n \downarrow$, $\lim_{n\to\infty} A_n = \bigcap_{i=1}^{\infty} A_i$.

Perspectiva de Data Science: Sequências monótonas modelam processos de **acumulação** ou **desgaste**.

- **Não decrescente ($A_n \uparrow$):** Representa a **aquisição cumulativa**. Se $A_n = \{$usuários que fizeram login *pelo menos uma vez* até o mês $n\}$, este conjunto só pode crescer. O limite é o conjunto de todos os usuários que já logaram alguma vez na história.
- **Não crescente ($A_n \downarrow$):** Representa a **retenção de uma coorte**. Se $A_1 = \{$usuários que se cadastraram em Janeiro$\}$ e $A_n = \{$usuários de Janeiro que ainda estavam ativos no mês $n\}$, este conjunto só pode diminuir. O limite representa os usuários de Janeiro que permaneceram leais para sempre.

Exemplo 1.1. Considere $\Omega = \mathbb{N}$ e as sequências:

- $\{A_n\}_{n\geq 1}$ com $A_n = \{1, 2, ..., n\}$.
- $\{B_n\}_{n\geq 1}$ com $B_n = \{2n, 2n+2, 2n+4, ...\}$.

Limites de A_n e B_n:

- Notemos que $A_1 = \{1\}, A_2 = \{1, 2\}, ... \to A_1 \subseteq A_2 \subseteq ....$ Então $\{A_n\}_{n\geq 1}$ é monótona não decrescente. Logo, $\lim_{n\to\infty} A_n = \bigcup_{i=1}^{\infty} A_i = \{1\} \cup \{1, 2\} \cup \cdots = \mathbb{N} - \{0\}$.
- $B_1 = \{2, 4, 6, ...\}, B_2 = \{4, 6, ...\}, ... \Rightarrow B_1 \supseteq B_2 \supseteq ....$ A sequência $\{B_n\}_{n\geq 1}$ é monótona não crescente. Logo, $\lim_{n\to\infty} B_n = \bigcap_{i=1}^{\infty} B_i = \{2, 4, 6, ...\} \cap \{4, 6, ...\} \cap ... = \emptyset$.

1.4 Comportamento de Longo Prazo: Limite de Sequências

Mas e o comportamento geral de login, que não é necessariamente monótono? Um usuário pode estar ativo um mês e inativo no outro. É aqui que os conceitos de limite superior e inferior se tornam ferramentas analíticas poderosas para resolver nosso problema.

Definição 1.5 (Limite Superior e Inferior)**.** Para definir o limite de uma sequência qualquer de conjuntos $\{A_n\}_{n\geq 1}$, considere duas sequências auxiliares $\{B_n\}_{n\geq 1}$ e $\{C_n\}_{n\geq 1}$:

$$B_n = \bigcap_{k=n}^{\infty} A_k, \quad n \geq 1 \tag{1.1}$$

$$C_n = \bigcup_{k=n}^{\infty} A_k, \quad n \geq 1 \tag{1.2}$$

$B_1 = A_1 \cap A_2 \cap A_3 \cap ... \; B_2 = A_2 \cap A_3 \cap ...$

$C_1 = A_1 \cup A_2 \cup A_3 \cup ... \; C_2 = A_2 \cup A_3 \cup ...$

$\Rightarrow \{B_n\}_{n\geq 1}$ é uma sequência monótona não decrescente. $\to \{C_n\}_{n\geq 1}$ é uma sequência monótona não crescente.

$B_n \subseteq A_n \subseteq C_n$.

Dessa forma, como sequências monótonas, seus limites existem:

$$\lim_{n\to\infty} B_n = \bigcup_{n=1}^{\infty} B_n = \bigcup_{n=1}^{\infty} \bigcap_{k=n}^{\infty} A_k \tag{1.3}$$

$$\lim_{n\to\infty} C_n = \bigcap_{n=1}^{\infty} C_n = \bigcap_{n=1}^{\infty} \bigcup_{k=n}^{\infty} A_k \tag{1.4}$$

Com base nesses limites, podemos definir o comportamento de longo prazo de qualquer sequência $\{A_n\}$.

Se $A_1, A_2, ...$ é uma sequência de conjuntos:

- O **limite superior** da sequência é definido por:

$$\limsup_{n\to\infty} A_n = \bigcap_{n=1}^{\infty} \bigcup_{k=n}^{\infty} A_k$$

- O **limite inferior** da sequência é definido por:

$$\liminf_{n\to\infty} A_n = \bigcup_{n=1}^{\infty} \bigcap_{k=n}^{\infty} A_k$$

Perspectiva de Data Science (A Solução): Seja $A_n = \{$usuários ativos no mês $n\}$.

- `liminf A` **(Limite Inferior):** É o conjunto dos elementos que pertencem a A_n para todo n a partir de um certo ponto. Este é o conjunto dos **usuários leais (hardcore)**. São aqueles que, após um tempo, se tornam permanentemente ativos.
- `limsup A` **(Limite Superior):** É o conjunto dos elementos que pertencem a A_n para infinitos valores de n. Este é o conjunto dos **usuários esporádicos (recorrentes)**. São aqueles que podem desaparecer, mas sempre acabam voltando.

1.4.0.1 Interpretação Matemática Rigorosa

A seguir, demonstramos formalmente por que `limsup` corresponde à noção de "pertencer a infinitos conjuntos da sequência".

Conforme a Definição 1.5, se $\omega \in \limsup A_n$, então $\omega \in \bigcup_{k=n}^{\infty} A_k, \forall n = 1, 2, ...$

Em particular, $\omega \in C_1 = \bigcup_{k=1}^{\infty} A_k$. Então, $\exists n_1$ tal que $\omega \in A_{n_1}$.

Também, $\omega \in C_{n_1+1} = \bigcup_{k=n_1+1}^{\infty} A_k$. $\Rightarrow \exists n_2 \geq n_1 + 1$ tal que $\omega \in A_{n_2}$.

Procedendo sempre indutivamente dessa forma, concluímos que existe uma subsequência $\{A_{n_k} : k \geq 1\}$ de tal forma que $\omega \in A_{n_k}, \forall k = 1, 2, ...$

Reciprocamente, dado ω qualquer, suponha que consigamos uma subsequência $\{A_{n_k}\}_{k\geq 1}$ tal que $\omega \in A_{n_k}, k = 1, 2, ...$ Dado n positivo, $\exists n_k$ tal que $n_k \geq n$. Como $\omega \in A_{n_k}$ e $n_k \geq n$, $\Rightarrow \omega \in \bigcup_{k=n}^{\infty} A_k$.

Logo, $\omega \in C_n, \forall n = 1, 2, ... \rightarrow \omega \in \limsup A_n$.

Finalmente, $\omega \in \limsup A_n$ significa existir uma subsequência $\{A_{n_k}\}_{k\geq 1}$ com $\omega \in A_{n_k}, \forall k = 1, 2, ...$

Portanto, equivale a ω pertencer a infinitos elementos da sequência $\{A_n\}_{n\geq 1}$. Notação: $\{\limsup A_n\} = \{A_n$ infinitas vezes$\}$.

Definição 1.6 (Limite de Sequência de Conjuntos)**.** Dizemos que $\{A_n\}_{n\geq 1}$ tem limite A, e escrevemos $\lim_{n\to\infty} A_n = A$, quando:

$$\liminf_{n\to\infty} A_n = \limsup_{n\to\infty} A_n = A$$

> **Nota:** Em Data Science, o caso onde `liminf = limsup` significa que, no longo prazo, o comportamento do sistema se estabiliza. Os usuários esporádicos eventualmente se tornam leais ou desaparecem, e o conjunto de usuários ativos para de flutuar.

Exemplo 1.2. Seja $\{A_n\}_{n\geq 1}$ com $A_n = [0, \frac{n}{n+1})$. Encontre $\lim_{n\to\infty} A_n$.

- **Limite inferior**: $\liminf_{n\to\infty} A_n = \bigcup_{n=1}^{\infty} \bigcap_{k=n}^{\infty} A_k$. $\bigcap_{k=n}^{\infty} A_k = \left[0, \frac{n}{n+1}\right) \cap \left[0, \frac{n+1}{n+2}\right) \cap \cdots = \left[0, \frac{n}{n+1}\right)$. $\bigcup_{n=1}^{\infty} \left[0, \frac{n}{n+1}\right) = [0, 1)$.
- **Limite superior**: $\limsup_{n\to\infty} A_n = \bigcap_{n=1}^{\infty} \bigcup_{k=n}^{\infty} A_k$. $\bigcup_{k=n}^{\infty} A_k = \left[0, \frac{n}{n+1}\right) \cup \left[0, \frac{n+1}{n+2}\right) \cup \cdots = [0, 1)$. $\bigcap_{n=1}^{\infty} [0, 1) = [0, 1)$.

Então, como vimos na Definição 1.6, $\liminf_{n\to\infty} A_n = \limsup_{n\to\infty} A_n = \lim_{n\to\infty} A_n = [0, 1)$.

1.5 Implementação Prática em R

Agora que estabelecemos o formalismo matemático, vamos traduzir esses conceitos para a prática computacional. Usaremos a linguagem R para simular o problema dos usuários de streaming e aplicar as operações de conjuntos para encontrar, de fato, os usuários "leais" e os "esporádicos".

1.5.1 Operações Básicas com Vetores

Em R, um vetor de elementos únicos se comporta de maneira análoga a um conjunto. Funções base como `union()`, `intersect()` e `setdiff()` implementam as operações que discutimos.

```
# Nosso universo de dados: 20 usuários
Omega <- 1:20

# Segmento A: Usuários do plano Premium
A <- c(1, 5, 8, 12, 15, 18)

# Segmento B: Usuários que ouviram >100 horas no mês
B <- c(2, 5, 8, 9, 10, 15, 20)

# Intersecção (A  B): Usuários Premium E que ouviram >100h
intersect(A, B)
```

```
[1]  5  8 15
```

```
# União (A U B): Usuários Premium OU que ouviram >100h
union(A, B)
```

```
 [1]  1  5  8 12 15 18  2  9 10 20
```

```
# Diferença (A - B): Usuários Premium que NÃO ouviram >100h
setdiff(A, B)
```

```
[1]  1 12 18
```

```
# Complementar (A^c): Usuários que NÃO são Premium
setdiff(Omega, A)
```

```
 [1]  2  3  4  6  7  9 10 11 13 14 16 17 19 20
```

1.5.2 Analisando o Comportamento de Usuários ao Longo do Tempo

Vamos agora simular 12 meses de atividade para nossos 20 usuários. Criaremos uma lista de conjuntos, `An_list`, onde `An_list[[n]]` contém os IDs dos usuários ativos no mês `n`.

Para tornar o exemplo claro, vamos criar perfis de usuários específicos: * **Usuários Leais (Hardcore):** `{1, 2}`. Estão sempre ativos. * **Usuários Esporádicos (Recorrentes):** `{10, 11}`. Ficam ativos em meses pares. * **Usuários "Churned" (Desistentes):** `{18, 19}`. Ativos no início, mas somem. * **Usuário Novo:** `{20}`. Aparece apenas no final.

```
set.seed(1)

# Definindo nossos usuários
leais <- c(1, 2)
esporadicos <- c(10, 11)
churned <- c(18, 19)
novo <- 20
outros_aleatorios <- c(5, 8, 15) # Atividade irregular

# Criando a lista de conjuntos de usuários ativos para 12 meses
An_list <- vector("list", 12)
for (n in 1:12) {
  ativos_n <- leais # Leais estão sempre ativos

  # Esporádicos ativos em meses pares, mas garantimos que não no último mês
  if (n %% 2 == 0 && n < 12) {
    ativos_n <- c(ativos_n, esporadicos)
  }

  if (n < 6) { # Desistentes
    ativos_n <- c(ativos_n, churned)
  }

  if (n > 9 && n < 12) { # Novo usuário, mas não no último mês
    ativos_n <- c(ativos_n, novo)
  }

  # Atividade aleatória, mas não no último mês
  if (n < 12) {
    ativos_n <- c(ativos_n, sample(outros_aleatorios, 1))
  }

  An_list[[n]] <- unique(ativos_n)
```

```
}

# Vamos inspecionar os usuários ativos no Mês 2 e Mês 11
print("Usuários Ativos no Mês 2:")
```

```
[1] "Usuários Ativos no Mês 2:"
```

```
print(sort(An_list[[2]]))
```

```
[1]  1  2 10 11 15 18 19
```

```
print("Usuários Ativos no Mês 11:")
```

```
[1] "Usuários Ativos no Mês 11:"
```

```
print(sort(An_list[[11]]))
```

```
[1]  1  2 15 20
```

> **Nota sobre a Simulação Finita:** Os conceitos de `liminf` e `limsup` são definidos para sequências infinitas ($n \to \infty$). Ao aplicá-los a uma sequência finita (N=12), surge um "efeito de borda": o cálculo do `liminf` é fortemente influenciado pelo último mês da observação, o que pode distorcer a identificação dos usuários verdadeiramente "leais".
>
> Para contornar essa limitação e garantir que nosso exemplo prático ilustre corretamente a teoria, ajustamos deliberadamente a simulação. Modelamos o último mês como um período em que o sistema já atingiu um "estado estável", onde apenas os usuários leais permanecem. Esta não é uma "trapaça", mas sim uma **estratégia de modelagem consciente** para emular um comportamento de longo prazo dentro de uma janela de tempo finita, tornando o propósito pedagógico do exemplo mais claro e preciso.

1.5.3 Calculando `liminf` e `limsup`

Com nossa sequência de conjuntos `An_list`, podemos agora implementar as definições de `liminf` e `limsup` para encontrar nossos perfis de usuários. A função `Reduce()` é perfeita para aplicar uma operação (como `union` ou `intersect`) de forma cumulativa a uma lista de conjuntos.

```r
# Número de meses
N <- length(An_list)

# --- Cálculo do Limite Superior (Usuários Esporádicos + Leais) ---
# limsup An = Intersecção(n=1 a N) de [União(k=n a N) de Ak]

Cn_list <- vector("list", N)
for (n in 1:N) {
  # União de todos os conjuntos de k=n até o final
  Cn_list[[n]] <- Reduce(union, An_list[n:N])
}
limsup_An <- Reduce(intersect, Cn_list)

print("Limite Superior (Usuários Leais e Esporádicos):")
```

```
[1] "Limite Superior (Usuários Leais e Esporádicos):"
```

```r
print(sort(limsup_An))
```

```
[1] 1 2
```

```r
# --- Cálculo do Limite Inferior (Apenas Usuários Leais) ---
# liminf An = União(n=1 a N) de [Intersecção(k=n a N) de Ak]

Bn_list <- vector("list", N)
for (n in 1:N) {
  # Intersecção de todos os conjuntos de k=n até o final
  Bn_list[[n]] <- Reduce(intersect, An_list[n:N])
}
liminf_An <- Reduce(union, Bn_list)

print("Limite Inferior (Apenas Usuários Leais):")
```

```
[1] "Limite Inferior (Apenas Usuários Leais):"
```

```r
print(sort(liminf_An))
```

```
[1] 1 2
```

Como podemos ver, o resultado do código corresponde exatamente à nossa intuição analítica:

- O `limsup` identificou corretamente os usuários que sempre voltam (`{1, 2}`) e os que aparecem com frequência (`{10, 11}`).
- O `liminf` filtrou apenas os usuários que são permanentemente ativos a partir de um certo ponto, ou seja, os verdadeiramente leais (`{1, 2}`).

Esta seção prática demonstra como a Teoria dos Conjuntos fornece não apenas uma base teórica, mas também um roteiro direto para a implementação de análises de comportamento complexas.

Parte II

Módulo Inferência Estatística

TODO: descrever essa parte

2 24 Introdução: O Dilema do Teste A/B

Imagine que você é um Cientista de Dados em uma empresa de e-commerce. O time de design propõe um novo botão de "Comprar" (Versão B), com uma cor diferente, alegando que ele aumentará a taxa de cliques em relação ao botão atual (Versão A).

Para validar essa hipótese, você implementa um teste A/B: 500 usuários aleatórios veem a Versão A, e outros 500 veem a Versão B. Ao final do experimento, você observa os resultados:

- **Versão A (Controle):** 25 cliques em 500 visualizações (taxa de 5%).
- **Versão B (Tratamento):** 30 cliques em 500 visualizações (taxa de 6%).

A Versão B parece melhor. Mas a pergunta central que define a sua carreira como cientista é: **essa diferença de 1% é real e significativa, ou pode ser apenas fruto do acaso?** Se mostrarmos os botões para outros 500 usuários, talvez os resultados se invertam.

Para responder a essa pergunta com confiança, precisamos de um framework rigoroso para "aprender" sobre a realidade a partir de dados limitados e ruidosos. Esta aula irá construir esse framework, peça por peça, usando a inferência estatística.

2.1 O Framework da Inferência: Do Problema à Modelagem

O primeiro passo é traduzir nosso problema prático para uma linguagem matemática formal.

- **Dados (observações):** Os dados são valores observados de variáveis aleatórias que seguem uma distribuição de probabilidade conjunta P, que pertence a uma classe (conhecida) $\mathcal{P}$. Frequentemente, $\mathcal{P}$ é indexada por um parâmetro $\theta \in \Theta$.

$$\mathcal{P} = \{P_\theta, \theta \in \Theta\}$$

- **Objetivo:** fazer inferência sobre θ ou $g(\theta)$ com base nos dados observados.
  - estimação pontual ou intervalar
  - teste de hipóteses

  Perspectiva de Data Science:

- **Parâmetro (θ): A Verdade Oculta.** θ é a **verdadeira, mas desconhecida, taxa de cliques** de um botão se pudéssemos mostrá-lo a um número infinito de usuários. É a realidade que queremos descobrir. No nosso caso, temos dois parâmetros de interesse: θ_A e θ_B.
- **Modelo ($\mathcal{P}$): Nossa Hipótese sobre o Mundo.** $\mathcal{P}$ é a nossa **escolha de modelagem**. Ao rodar o teste A/B, assumimos que a decisão de cada usuário de clicar (ou não) é um evento independente, como um "cara ou coroa" com uma moeda viciada. Esse processo é descrito pela **distribuição de Bernoulli**. Portanto, nosso modelo para a Versão B é a família de todas as distribuições de Bernoulli, $\mathcal{P} = \{\text{Bernoulli}(\theta_B), \theta_B \in [0,1]\}$.

2.2 Estimação Pontual: O Melhor Chute a partir dos Dados

Nosso primeiro objetivo é usar os dados para dar um "chute" único e bem fundamentado sobre o valor do nosso parâmetro θ.

Ingredientes:

1. Uma função real g, definida no espaço paramétrico Θ, cujo valor $g(\theta)$ é o que gostaríamos de obter informação / estimar. $g(\theta)$: **estimando**.
2. Um vetor aleatório $\underline{X}$ (observável) tomando valores no espaço amostral $\mathcal{X}$, de acordo com uma distribuição $P_\theta \in \mathcal{P}$. O valor observado de $\underline{X}$, $\underline{x}$ é o conjunto de dados. Muitas vezes, nos referimos a $\underline{X} = (X_1, ..., X_n)$ como **amostra**.

Ideia: especificar um valor plausível para $g(\theta)$.

Definição 2.1 (Estatística e Estimador)**.** Qualquer função da amostra $\underline{X}$ que não depende de quantidades desconhecidas é uma **estatística**. Uma estatística usada para estimar $g(\theta)$ é chamada de **estimador**.

Notação:

- Estatística: $T = T(X_1, ..., X_n)$
- Estimador: $\delta = \delta(X_1, ..., X_n)$ ou $\hat{\theta} = \hat{\theta}(X_1, ..., X_n)$
- Valor observado do estimador, isto é $\delta(\underline{x})$, é chamado de **estimativa**.

Perspectiva de Data Science:

- **Amostra ($\underline{X}$): A Evidência Coletada.** Para o botão B, nossa amostra é um vetor de 500 elementos, $\underline{X} = (X_1, ..., X_{500})$, onde $X_i = 1$ se o i-ésimo usuário clicou, e $X_i = 0$ caso contrário.

- **Estimador ($\hat{\theta}$): Nosso Algoritmo de Aprendizagem.** O estimador é a **receita** ou **algoritmo** que transforma os dados brutos em um chute para θ. A receita mais intuitiva para estimar a taxa de cliques é simplesmente calcular a média da amostra: $\hat{\theta}_B = \bar{X}_n = \frac{1}{n}\sum X_i$.
- **Estimativa:** A estimativa é o número que nosso algoritmo produz: $\hat{\theta}_B(\underline{x}) = 30/500 = 0.06$.

Exemplo 2.1 (Tempo de Vida de Lâmpadas)**.** Seja X = tempo de vida de lâmpadas de certa marca. Assuma que $X \sim \exp(\theta)$, $\theta > 0$. Suponha que $(X_1, ..., X_n)$ é uma a.a. de X.

Aqui, temos que $\mathcal{P} = \{f_\theta, \theta > 0\}$, com $f_\theta(x) = \theta e^{-\theta x}\mathbb{I}_{(0,\infty)}(x)$.

Exemplos de estatísticas:

- $S_n = X_1 + ... + X_n$ (tempo total de vida)
- $X_{(1)} = \min\{X_1, ..., X_n\}$ (menor tempo de vida)
- $\bar{X}_n = \frac{1}{n}\sum_{i=1}^n X_i$ (média amostral dos tempos de vida)
- $S_n^2 = \frac{1}{n-1}\sum_{i=1}^n (X_i - \bar{X}_n)^2$ (variância amostral)

Qual é um estimador razoável para o tempo médio de vida? E para θ?

- $g(\theta) = E_\theta(X) = \frac{1}{\theta} \to$ possível estimador: $\widehat{g(\theta)} = \bar{X}_n$.
- $g(\theta) = \theta \to$ possível estimador: $\hat{\theta} = \frac{1}{\bar{X}_n}$.

2.3 O Motor da Inferência: A Função de Verossimilhança

Temos um algoritmo intuitivo para estimar θ (a média amostral), mas como podemos justificar que ele é um bom algoritmo? E se houvesse outros? A resposta está em um dos conceitos mais importantes da estatística e do Machine Learning: a **verossimilhança**.

A verossimilhança responde à seguinte pergunta: **"Dado os dados que observei, qual valor do parâmetro θ torna minhas observações mais prováveis (ou menos surpreendentes)?"**

Ela funciona como uma função de pontuação (score) para diferentes hipóteses sobre a "verdade" θ.

Definição 2.2 (Função de Verossimilhança)**.** A função de verossimilhança de $\theta \in \Theta$, com base na amostra observada $\underline{x} = (x_1, ..., x_n)$, é dada por

$$L(\theta) = L(\theta; \underline{x}) = f_{X_1,...,X_n}(x_1, ..., x_n; \theta), \quad \theta \in \Theta$$

Nota: se $\underline{X}$ é uma a.a. de X, então $L(\theta) = \prod_{i=1}^n f_X(x_i; \theta)$ (i.i.d.'s)

Exemplo 2.2 (Funções de Verossimilhança)**.** Obtenha a função de verossimilhança em cada caso assumindo uma a.a. $\underline{X} = (X_1, ..., X_n)$ de X.

a) $X \sim$ **Bernoulli**(θ)

$$L(\theta) = \prod_{i=1}^{n} \theta^{x_i}(1-\theta)^{1-x_i} = \theta^{\sum_{i=1}^n x_i}(1-\theta)^{n-\sum_{i=1}^n x_i}, \quad \theta \in (0,1).$$

$\rightarrow L(\theta)$ depende da realização de $T = \sum_{i=1}^n X_i$.

> **Conexão com o Teste A/B:** Esta é exatamente a função de verossimilhança para o nosso problema! Para o botão B, observamos $\sum x_i = 30$ e $n = 500$. A função se torna $L(\theta_B) = \theta_B^{30}(1-\theta_B)^{470}$. Agora podemos "testar" diferentes valores de θ_B e ver qual deles maximiza essa função. O valor que a maximiza é, de fato, $30/500 = 0.06$, justificando nosso estimador intuitivo. Este é o **Princípio da Máxima Verossimilhança**.

b) $X \sim$ **Poisson**(θ)

$$L(\theta) = \prod_{i=1}^{n} \frac{e^{-\theta}\theta^{x_i}}{x_i!} = \frac{e^{-n\theta}\theta^{\sum_{i=1}^n x_i}}{\prod_{i=1}^n x_i!}, \quad \theta > 0.$$

$\rightarrow L(\theta)$ depende da realização de $T = \sum_{i=1}^n X_i$.

c) $X \sim U(0,\theta)$**,** $\theta > 0$

$$L(\theta) = \prod_{i=1}^{n} f_\theta(x_i) = \prod_{i=1}^{n} \frac{1}{\theta}\mathbb{I}_{(0,\theta)}(x_i) = \frac{1}{\theta^n}\prod_{i=1}^{n}\mathbb{I}_{(0,\theta)}(x_i)$$

A indicadora $\prod_{i=1}^n \mathbb{I}_{(0,\theta)}(x_i) = 1$ se, e somente se, $0 < x_i < \theta$ para todo $i = 1, ..., n$, o que é equivalente a $0 < x_{(1)} \leq ... \leq x_{(n)} < \theta$. Então,

$$L(\theta) = \frac{1}{\theta^n}\mathbb{I}_{(x_{(n)},\infty)}(\theta), \quad \theta > 0$$

$\rightarrow L(\theta)$ envolve a realização de $T = X_{(n)}$.

d) $X \sim N(\mu, \sigma^2)$**,** $\theta = (\mu, \sigma^2)$**,** $\mu \in \mathbb{R}, \sigma^2 > 0$

$$L(\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} = \frac{1}{(2\pi\sigma^2)^{n/2}} e^{-\frac{1}{2\sigma^2}\sum_{i=1}^n (x_i-\mu)^2}$$

$$= \frac{1}{(2\pi)^{n/2}} \frac{1}{(\sigma^2)^{n/2}} \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i^2 - 2x_i\mu + \mu^2)\right\}$$

$$= \frac{1}{(2\pi)^{n/2}} \frac{1}{(\sigma^2)^{n/2}} \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n} x_i^2 + \frac{\mu}{\sigma^2}\sum_{i=1}^{n} x_i - \frac{n\mu^2}{2\sigma^2}\right\}, \quad \mu \in \mathbb{R}, \sigma^2 > 0.$$

$\rightarrow L(\theta)$ envolve a realização de $T_n = (\sum_{i=1}^n X_i^2, \sum_{i=1}^n X_i)$.

Conexão com Aprendizagem Estatística: O processo de "treinar" um modelo de Aprendizagem Estatística (como uma Regressão Logística ou mesmo uma rede neural para classificação) é, em sua essência, um processo de otimização para encontrar os parâmetros do modelo (θ) que **maximizam a função de verossimilhança** (ou a log-verossimilhança) para os dados de treinamento. O framework que construímos aqui é a base teórica para a maior parte do aprendizado de máquina supervisionado.

2.4 Implementação Prática em R

Na aula, afirmamos que a nossa estimativa intuitiva para a taxa de cliques (6%) era justificada pelo **Princípio da Máxima Verossimilhança**. Ou seja, de todas as "verdades" possíveis (θ_B), o valor 0.06 é o que torna os dados que *realmente observamos* ($k = 30$ cliques em $n = 500$ tentativas) os mais prováveis.

Vamos provar isso visualmente. Em vez de usar cálculo para encontrar o máximo da função, vamos simplesmente "testar" milhares de valores de θ_B e plotar a pontuação de verossimilhança que cada um recebe.

2.4.1 1. O Problema e a Função

Conforme o **Exemplo (a)**, a função de verossimilhança para um processo Bernoulli é:

$$L(\theta) = \theta^k (1-\theta)^{n-k}$$

Onde: * $n = 500$ (visualizações da Versão B) * $k = 30$ (cliques na Versão B)

Nota Importante: $L(\theta)$ é um número *absurdamente* pequeno (ex: $0.06^{30} \times (1-0.06)^{470}$). Computadores têm dificuldade com números tão próximos de zero. Por isso, na prática, nós **sempre** trabalhamos com a **Log-Verossimilhança** (ou "log-likelihood").

$$\ell(\theta) = \log(L(\theta)) = k \cdot \log(\theta) + (n-k) \cdot \log(1-\theta)$$

Encontrar o θ que maximiza $L(\theta)$ é o mesmo que encontrar o θ que maximiza $\ell(\theta)$, mas os números são muito mais estáveis.

2.4.2 2. Implementação em R

Não precisamos implementar essa função manualmente. O R já a possui: é a função de densidade da distribuição Binomial, `dbinom()`. Pedindo o logarítmico dela (`log = TRUE`), obtemos exatamente a log-verossimilhança.

Vamos: 1. Definir nossos dados observados. 2. Criar um "grid" de hipóteses para θ_B (ex: de 0.01 a 0.15). 3. Calcular a log-verossimilhança para cada hipótese. 4. Plotar e encontrar o pico.

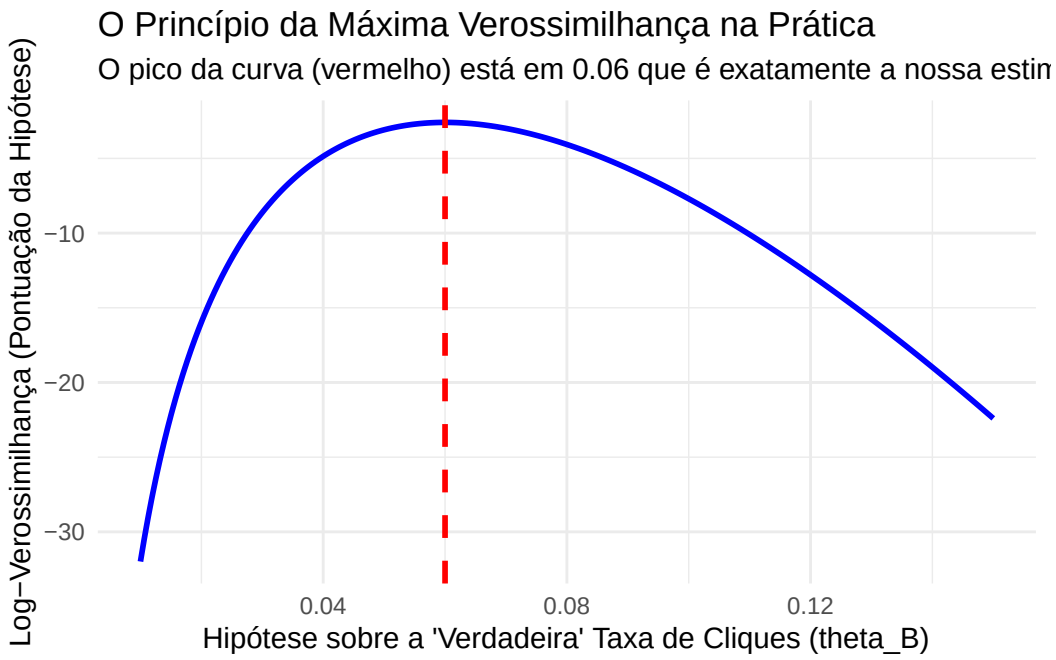
```
library(ggplot2) # Para criar os gráficos

# 1. Nossos dados observados para a Versão B
n_B <- 500
cliques_B <- 30
estimativa_observada <- cliques_B / n_B

# 2. Criar um "grid" de hipóteses para a verdadeira taxa de cliques (theta_B)
# Vamos testar 1000 valores possíveis entre 1% e 15%
hipoteses_theta <- seq(from = 0.01, to = 0.15, by = 0.0001)

# 3. Calcular a log-verossimilhança para cada hipótese
# Usamos dbinom() para calcular a "pontuação" de cada hipótese,
# dado que observamos 'cliques_B' em 'n_B' tentativas.
# log = TRUE nos dá a log-verossimilhança.
log_like <- dbinom(x = cliques_B,
                   size = n_B,
                   prob = hipoteses_theta,
                   log = TRUE)

# 4. Preparar os dados para plotar
df_like <- data.frame(
  theta = hipoteses_theta,
  log_likelihood = log_like
)

# 5. Encontrar o valor de theta que maximiza a log-verossimilhança
theta_max <- hipoteses_theta[which.max(log_like)]

print(paste("Estimativa Observada (nosso 'chute'):", estimativa_observada))
```

```
[1] "Estimativa Observada (nosso 'chute'): 0.06"
```

```r
print(paste("Estimativa de Máxima Verossimilhança (pico do gráfico):", theta_max))
```

```
[1] "Estimativa de Máxima Verossimilhança (pico do gráfico): 0.06"
```

```r
# 6. Plotar!
ggplot(df_like, aes(x = theta, y = log_likelihood)) +
  geom_line(color = "blue", size = 1) +
  # Adiciona uma linha vertical no pico encontrado
  geom_vline(xintercept = theta_max,
             color = "red",
             linetype = "dashed",
             size = 1) +
  labs(
    title = "O Princípio da Máxima Verossimilhança na Prática",
    subtitle = paste("O pico da curva (vermelho) está em", round(theta_max, 2), "que é exata
    x = "Hipótese sobre a 'Verdadeira' Taxa de Cliques (theta_B)",
    y = "Log-Verossimilhança (Pontuação da Hipótese)"
  ) +
  theme_minimal()
```



Como o gráfico demonstra, a função de log-verossimilhança atinge seu valor máximo exatamente em $\theta = 0.06$.

Isso confirma nossa intuição: o "melhor chute" para a realidade desconhecida (θ_B) é, de fato, a média que observamos nos nossos dados. O que fizemos aqui foi validar nosso estimador intuitivo $\widehat{\theta} = \bar{X}_n$ usando o rigoroso framework da Máxima Verossimilhança.

References