

Simulação Estocástica

Thiago Rodrigo Ramos

29 de outubro de 2025

Sumário

1	Introdução	7
1.1	Um conselho: a importância de ser ruim antes de ser bom	7
2	Elementos básicos de probabilidade	9
2.1	Axiomas da probabilidade	9
2.1.1	Probabilidade condicional e independência	9
2.2	Variáveis aleatórias	10
2.3	Valor esperado	11
2.4	Variância	12
2.4.1	Covariância	12
2.5	Desigualdades básicas de concentração	14
2.6	Teoremas assintóticos	14
3	Variáveis discretas e como simulá-las	17
3.1	Variáveis com suporte finito	19
3.2	Bernoulli	20
3.3	Distribuição binomial	21
3.3.1	Simulando via Bernoullis	21
3.3.2	Simulando via identidade recursiva	22
3.3.3	Aspectos computacionais	23
3.3.4	Número médio de passos em algoritmos de inversão recursiva	24
3.4	Distribuição geométrica	25
3.4.1	Simulando via Bernoullis	25
3.4.2	Simulando a geométrica via inversão	26
3.5	Distribuição de Poisson	27
3.5.1	Simulação a Poisson via inversão e recursão	28
3.5.2	Algoritmo melhorado	29
3.5.3	Relação com a binomial	30
3.6	Distribuição binomial negativa	31
3.6.1	Simulando via Bernoullis	32
3.6.2	Simulando via soma de geométricas	32
3.6.3	Simulando via inversão recursiva	33
3.6.4	Por que o nome “Binomial Negativa”?	34

3.7	Distribuição hipergeométrica	34
3.7.1	Simulando a Hipergeométrica	35
4	Variáveis contínuas e como simulá-las	37
4.1	Método da Inversão	37
4.1.1	Distribuição exponencial	38
4.2	Método da rejeição-aceitação	41
4.2.1	Distribuição normal	44
4.3	Distribuição Gamma	47
4.3.1	Simulando quando α é inteiro	48
4.3.2	Simulando quando $\alpha > 1$ via aceitação–rejeição com Exponencial	48
4.3.3	Simulando quando $\alpha < 1$	50
4.4	Distribuição Beta	51
4.4.1	Simulando a Beta via aceitação–rejeição com proposta uniforme	52
4.4.2	Simulando a Beta via Gammas independentes	53
4.5	Transformações de Variáveis Aleatórias	53
4.5.1	Geração de Normais via Método de Box–Muller	54
4.5.2	Geração da normal bivariada	56
4.5.3	Distribuição Qui-quadrado	59
4.5.4	Simulando a distribuição t de Student	61
5	Simulação via Monte Carlo	63
5.1	Estimando médias	63
5.1.1	Exemplos	63
5.2	Intervalos de Confiança	66
6	Redução de variância	69
6.1	Uso de variáveis antitéticas	69
6.2	O uso de variáveis de controle	74
6.3	Redução de Variância por Condicionamento	77
6.4	Amostragem por importância	81
6.4.1	Densidades Inclinadas (Tilted Densities)	82
7	Cadeias de Markov e MCMC	85
7.1	Cadeias de Markov (Resumo)	86
7.1.1	Classificação dos estados	89
7.1.2	Distribuição estacionária	92
7.1.3	Reversibilidade	96
7.2	Markov Chain Monte Carlo	99
7.3	Algoritmo de Metropolis–Hastings	100
7.4	Amostragem de Gibbs	105

<i>SUMÁRIO</i>	5
8 Processos de Difusão	111
8.1 Movimento Browniano e SDE	111
8.2 Equação de Fokker–Planck	114
8.3 Amostragem de Langevin	115
8.4 Denoising Score Matching	117
8.4.1 Estimando o score na prática	119
9 Estratégias para acelerar códigos em Python	123
9.1 Profiling com <code>cProfile</code>	123
9.2 Paralelização com <code>joblib.Parallel</code>	126
9.3 Compilação Just-In-Time com Numba	128
9.4 Paralelismo simples em Bash	131

Capítulo 1

Introdução

1.1 Um conselho: a importância de ser ruim antes de ser bom

É natural que, quando começamos a fazer algo, a gente faça essa coisa muito malfeita ou cheia de defeitos. Isso é comum em qualquer processo de aprendizagem, e sempre foi assim, desde o início dos tempos.

Quando comecei a programar em Python, muita coisa sobre a linguagem eu aprendi por conta própria, apesar de já ter feito alguns cursos básicos em C. Programei de forma amadora em Python por muitos anos, até que, no doutorado, precisei aprender a programar de forma mais organizada e profissional. Lembro que, nessa época, um amigo da pós-graduação me apresentou ao "submundo da programação". Foi aí que aprendi muito do que sei hoje sobre terminal do Linux, Git, e foi também quando comecei a usar o Vim.

Uma das coisas que esse amigo me mostrou foi o Pylint, que nada mais é do que um verificador de bugs e qualidade de código para Python. O Pylint é bem rigoroso na análise, e ainda te dá, ao final, uma nota que vai até 10. Nessa fase, apesar de já ter evoluído bastante, meus códigos ainda recebiam notas por volta de 6 ou 7. Resolvi então rodar o Pylint nos meus códigos antigos pra ter uma noção de quão ruins eles eram — e a nota final foi -900. Pois é, existe um limite superior para o quão bem você consegue fazer algo, mas aparentemente o fundo do poço é infinito.

O que eu queria mostrar com essa história é que faz parte do processo de aprendizado ser ruim no começo e melhorar com o tempo. Falo isso porque, hoje em dia, com o crescimento dos LLMs, a gente fica tentado a pular essa etapa de errar muito até acertar, e ir direto pra fase em que escrevemos códigos limpos, bem comentados, identados e organizados. Mas não se enganem: apesar da aparência profissional, depender de LLMs pra escrever tudo atrapalha justamente essa parte essencial de aprender errando.

Neste curso, vários exercícios envolvem escrever códigos em Python. Meu conselho é: não tenham vergonha de errar, de escrever soluções ruins ou confusas. Isso é absolutamente normal. Vocês estão aqui para evoluir — e errar faz parte do processo.

Capítulo 2

Elementos básicos de probabilidade

2.1 Axiomas da probabilidade

Um *espaço de probabilidade* é uma tupla composta por três elementos: o *espaço amostral*, o *conjunto de eventos* e uma *distribuição de probabilidade*:

- **Espaço amostral Ω** : Ω é o conjunto de todos os eventos elementares ou resultados possíveis de um experimento. Por exemplo, ao lançar um dado, $\Omega = \{1, 2, 3, 4, 5, 6\}$.
- **Conjunto de eventos \mathcal{F}** : \mathcal{F} é uma σ -álgebra, ou seja, um conjunto de subconjuntos de Ω que contém Ω e é fechado sob complementação e união enumerável (e, consequentemente, também sob interseção enumerável). Um exemplo de evento é: “o dado mostra um número ímpar”.
- **Distribuição de probabilidade \mathbb{P}** : \mathbb{P} é uma função que associa a cada evento de \mathcal{F} um número em $[0, 1]$, tal que $\mathbb{P}[\Omega] = 1$, $\mathbb{P}[\emptyset] = 0$ e, para eventos mutuamente exclusivos A_1, \dots, A_n , temos:

$$\mathbb{P}[A_1 \cup \dots \cup A_n] = \sum_{i=1}^n \mathbb{P}[A_i].$$

A distribuição de probabilidade discreta associada ao lançamento de um dado justo pode ser definida como $\mathbb{P}[A_i] = 1/6$ para $i \in \{1, \dots, 6\}$, onde A_i é o evento “o dado mostra o valor i ”.

2.1.1 Probabilidade condicional e independência

A probabilidade condicional do evento A dado o evento B é definida como a razão entre a probabilidade da interseção $A \cap B$ e a probabilidade de B , desde que $\mathbb{P}[B] \neq 0$:

$$\mathbb{P}[A | B] = \frac{\mathbb{P}[A \cap B]}{\mathbb{P}[B]}.$$

Dois eventos A e B são ditos independentes quando a probabilidade conjunta $\mathbb{P}[A \cap B]$ pode ser fatorada como o produto $\mathbb{P}[A]\mathbb{P}[B]$:

$$\mathbb{P}[A \cap B] = \mathbb{P}[A]\mathbb{P}[B].$$

De forma equivalente, a independência entre A e B pode ser expressa afirmando que $\mathbb{P}[A | B] = \mathbb{P}[A]$, sempre que $\mathbb{P}[B] \neq 0$.

Além disso, uma sequência de variáveis aleatórias é dita *i.i.d.* (independentes e identicamente distribuídas) quando todas as variáveis da sequência são mutuamente independentes e seguem a mesma distribuição de probabilidade.

Seguem algumas propriedades importantes:

$$\mathbb{P}[A \cup B] = \mathbb{P}[A] + \mathbb{P}[B] - \mathbb{P}[A \cap B] \quad (\text{regra da soma})$$

$$\mathbb{P}\left[\bigcup_{i=1}^n A_i\right] \leq \sum_{i=1}^n \mathbb{P}[A_i] \quad (\text{desigualdade da união})$$

$$\mathbb{P}[A | B] = \frac{\mathbb{P}[B | A]\mathbb{P}[A]}{\mathbb{P}[B]} \quad (\text{fórmula de Bayes})$$

$$\mathbb{P}\left[\bigcap_{i=1}^n A_i\right] = \mathbb{P}[A_1]\mathbb{P}[A_2 | A_1] \cdots \mathbb{P}\left[A_n | \bigcap_{i=1}^{n-1} A_i\right] \quad (\text{regra da cadeia})$$

Exercício 1. Prove os resultados acima.

2.2 Variáveis aleatórias

Uma *variável aleatória* X é uma função mensurável $X : \Omega \rightarrow \mathbb{R}$, ou seja, tal que, para qualquer intervalo $I \subset \mathbb{R}$, o conjunto

$$\{\omega \in \Omega : X(\omega) \in I\}$$

pertence à σ -álgebra de eventos.

No caso discreto, a *função de massa de probabilidade* de X é dada por

$$x \mapsto \mathbb{P}[X = x].$$

Quando a distribuição de X é *absolutamente contínua*, existe uma *função densidade de probabilidade* f tal que, para todo $a, b \in \mathbb{R}$,

$$\mathbb{P}[a \leq X \leq b] = \int_a^b f(x) dx.$$

A função f é chamada *função densidade de probabilidade* da variável aleatória X . A relação entre a *função de distribuição acumulada* $F(\cdot)$ e a densidade $f(\cdot)$ é

$$F(a) = \mathbb{P}\{X \leq a\} = \int_{-\infty}^a f(x) dx.$$

Derivando ambos os lados, obtemos

$$\frac{d}{da} F(a) = f(a),$$

ou seja, a densidade é a derivada da função de distribuição acumulada.

Uma interpretação mais intuitiva de f pode ser obtida observando que, para $\varepsilon > 0$ pequeno,

$$\mathbb{P}\left(a - \frac{\varepsilon}{2} < X < a + \frac{\varepsilon}{2}\right) = \int_{a-\varepsilon/2}^{a+\varepsilon/2} f(x) dx \approx \varepsilon f(a).$$

Assim, $f(a)$ quantifica a probabilidade de X assumir valores próximos de a .

Em muitos contextos, o interesse recai não apenas sobre variáveis aleatórias individuais, mas também sobre o relacionamento entre duas ou mais variáveis. Para descrever a dependência entre X e Y , definimos a *função de distribuição acumulada conjunta* como

$$F(x, y) = \mathbb{P}\{X \leq x, Y \leq y\},$$

que fornece a probabilidade de X ser menor ou igual a x e, simultaneamente, Y ser menor ou igual a y .

Se X e Y forem variáveis aleatórias discretas, a *função de massa de probabilidade conjunta* é

$$p(x, y) = \mathbb{P}\{X = x, Y = y\}.$$

Se forem *conjuntamente contínuas*, existe uma *função densidade de probabilidade conjunta* $f(x, y)$ tal que, para quaisquer conjuntos $C, D \subset \mathbb{R}$,

$$\mathbb{P}\{X \in C, Y \in D\} = \iint_{x \in C, y \in D} f(x, y) dx dy.$$

As variáveis X e Y são *independentes* se, para quaisquer $C, D \subset \mathbb{R}$,

$$\mathbb{P}\{X \in C, Y \in D\} = \mathbb{P}\{X \in C\} \mathbb{P}\{Y \in D\}.$$

De forma intuitiva, isso significa que conhecer o valor de uma das variáveis não altera a distribuição da outra.

No caso discreto, X e Y são independentes se, e somente se, para todo x, y ,

$$\mathbb{P}\{X = x, Y = y\} = \mathbb{P}\{X = x\} \mathbb{P}\{Y = y\}.$$

Se forem conjuntamente contínuas, a independência é equivalente a

$$f(x, y) = f_X(x) f_Y(y), \quad \forall x, y,$$

onde f_X e f_Y são as densidades marginais de X e Y , respectivamente.

2.3 Valor esperado

A esperança ou valor esperado de uma variável aleatória X é denotada por $\mathbb{E}[X]$ e, no caso discreto, é definida como

$$\mathbb{E}[X] = \sum_x x \mathbb{P}[X = x].$$

Exemplo 1. Se I é uma variável aleatória indicadora do evento A , isto é,

$$I = \begin{cases} 1, & \text{se } A \text{ ocorre,} \\ 0, & \text{se } A \text{ não ocorre,} \end{cases}$$

então

$$\mathbb{E}[I] = 1 \cdot \mathbb{P}(A) + 0 \cdot \mathbb{P}(A^c) = \mathbb{P}(A).$$

Portanto, a esperança de uma variável indicadora de um evento A é exatamente a probabilidade de que A ocorra.

No caso contínuo, quando X possui uma função densidade de probabilidade $f(x)$, a esperança é dada por

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} xf(x) dx.$$

Além disso, dado uma função qualquer g , temos que:

$$\mathbb{E}[g(X)] = \int_{-\infty}^{\infty} g(x)f(x) dx.$$

Uma propriedade fundamental da esperança é sua linearidade. Isto é, para quaisquer variáveis aleatórias X e Y e constantes $a, b \in \mathbb{R}$, temos:

$$\mathbb{E}[aX + bY] = a\mathbb{E}[X] + b\mathbb{E}[Y].$$

2.4 Variância

A variância de uma variável aleatória X é denotada por $\text{Var}[X]$ e definida como

$$\text{Var}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2].$$

O desvio padrão de X é denotado por σ_X e definido como

$$\sigma_X = \sqrt{\text{Var}[X]}.$$

Para qualquer variável aleatória X e qualquer constante $a \in \mathbb{R}$, as seguintes propriedades básicas são válidas:

$$\text{Var}[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2,$$

$$\text{Var}[aX] = a^2 \text{Var}[X].$$

Além disso, se X e Y forem independentes, então

$$\text{Var}[X + Y] = \text{Var}[X] + \text{Var}[Y].$$

Exercício 2. Prove os resultados anteriores.

2.4.1 Covariância

A covariância entre duas variáveis aleatórias X e Y é denotada por $\text{Cov}(X, Y)$ e definida por

$$\text{Cov}(X, Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])].$$

Exercício 3. Prove que

$$\text{Cov}(X, Y) = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y].$$

Dizemos que X e Y são *não correlacionadas* quando $\text{Cov}(X, Y) = 0$. Se X e Y forem independentes, então certamente são não correlacionadas, mas a recíproca nem sempre é verdadeira.

Exercício 4. Seja X uniforme no intervalo $[-1, 1]$ e seja $Y = X^2$. Mostre que $\text{Cov}(X, Y) = 0$ mas X, Y não são independentes.

Observação 1. Considere uma variável aleatória contínua X centrada em zero, ou seja, $\mathbb{E}[X] = 0$, com densidade de probabilidade par e definida em um intervalo do tipo $(-a, a)$, com $a > 0$. Seja $Y = g(X)$ para uma função g . A questão é: para quais funções $g(X)$ temos $\text{Cov}(X, g(X)) = 0$?

Sabemos que

$$\text{Cov}(X, g(X)) = \mathbb{E}[Xg(X)] - \mathbb{E}[X]\mathbb{E}[g(X)].$$

Como $\mathbb{E}[X] = 0$, segue que $\text{Cov}(X, g(X)) = \mathbb{E}[Xg(X)]$. Denotando a densidade de X por $f(x)$, temos

$$\text{Cov}(X, g(X)) = \int_{-a}^a xg(x)f(x)dx.$$

Uma maneira de garantir que $\text{Cov}(X, g(X)) = 0$ é exigir que $g(x)$ seja uma função par. Assim, $xg(x)f(x)$ será uma função ímpar e a integral em $(-a, a)$ se anulará, ou seja,

$$\int_{-a}^a xg(x)f(x)dx = 0.$$

Portanto, $\text{Cov}(X, f(X)) = 0$ e como $Y = g(X)$, teremos que ambas são dependentes.

Dessa forma, podemos concluir que a distribuição precisa de X não afeta a condição, desde que $p(x)$ seja simétrica em torno da origem. Qualquer função par $f(\cdot)$ satisfará $\text{Cov}(X, f(X)) = 0$.

A covariância é uma forma bilinear simétrica e semi-definida positiva, com as seguintes propriedades:

- **Simetria:** $\text{Cov}(X, Y) = \text{Cov}(Y, X)$ para quaisquer variáveis X e Y .
- **Bilinearidade:** $\text{Cov}(X + X', Y) = \text{Cov}(X, Y) + \text{Cov}(X', Y)$ e $\text{Cov}(aX, Y) = a\text{Cov}(X, Y)$ para qualquer $a \in \mathbb{R}$.
- **Semi-definida positiva:** $\text{Cov}(X, X) = \text{Var}[X] \geq 0$ para qualquer variável X .

Além disso, vale a desigualdade de Cauchy-Schwarz, que afirma que para variáveis X e Y com variância finita,

$$|\text{Cov}(X, Y)| \leq \sqrt{\text{Var}[X] \text{Var}[Y]}.$$

Perceba a semelhança do resultado acima com a desigualdade de Cauchy-Schwarz!

Exercício 5. Prove os resultados acima.

A matriz de covariância de um vetor de variáveis aleatórias $\mathbf{X} = (X_1, \dots, X_p)$ é a matriz em $\mathbb{R}^{n \times n}$ denotada por $\mathbf{C}(\mathbf{X})$ e definida por

$$\mathbf{C}(\mathbf{X}) = \mathbb{E} \left[(\mathbf{X} - \mathbb{E}[\mathbf{X}])(\mathbf{X} - \mathbb{E}[\mathbf{X}])^\top \right].$$

Portanto, $\mathbf{C}(\mathbf{X})$ é a matriz cujos elementos são $\text{Cov}(X_i, X_j)$. Além disso, é imediato mostrar que

$$\mathbf{C}(\mathbf{X}) = \mathbb{E}[\mathbf{X}\mathbf{X}^\top] - \mathbb{E}[\mathbf{X}]\mathbb{E}[\mathbf{X}]^\top.$$

2.5 Desigualdades básicas de concentração

Nesta seção, apresentamos duas desigualdades fundamentais que estabelecem limites superiores para a probabilidade de uma variável aleatória assumir valores distantes de sua média. Tais resultados são amplamente utilizados em probabilidade, estatística e teoria da informação para analisar o comportamento de caudas de distribuições.

A primeira delas é a *Desigualdade de Markov*, que fornece um limite simples para variáveis aleatórias não-negativas em função apenas de sua esperança.

Teorema 1 (Desigualdade de Markov). *Seja X uma variável aleatória não-negativa ($X \geq 0$ quase certamente) com valor esperado $\mathbb{E}[X] < \infty$. Então, para todo $t > 0$, temos:*

$$\mathbb{P}(X \geq t) \leq \frac{\mathbb{E}[X]}{t}.$$

Exercício 6. Prove a desigualdade de Markov. **Dica:** use o fato de que $\frac{x}{t} \geq \mathbb{I}\{x \geq t\}$.

A próxima desigualdade é um refinamento da anterior. Conhecida como *Desigualdade de Chebyshev*, ela aplica a desigualdade de Markov à variável aleatória $(X - \mu)^2$ e relaciona o desvio da média com a variância da distribuição.

Teorema 2 (Desigualdade de Chebyshev). *Seja X uma variável aleatória com valor esperado $\mu = \mathbb{E}[X]$ e variância finita $\text{Var}(X) = \sigma^2$. Então, para todo $\varepsilon > 0$, vale:*

$$\mathbb{P}(|X - \mu| \geq \varepsilon) \leq \frac{\sigma^2}{\varepsilon^2}.$$

Exercício 7. Prove a desigualdade de Chebyshev a partir da desigualdade de Markov aplicada a $(X - \mu)^2$.

2.6 Teoremas assintóticos

Em muitas aplicações de probabilidade e estatística, estamos interessados no comportamento de sequências de variáveis aleatórias quando o número de observações tende ao infinito. Os *teoremas assintóticos* fornecem resultados fundamentais que descrevem como certos estimadores ou somas de variáveis aleatórias se comportam no limite, ou seja, quando o tamanho da amostra n cresce indefinidamente.

Teorema 3 (Lei Fraca dos Grandes Números). *Seja $(X_n)_{n \in \mathbb{N}}$ uma sequência de variáveis aleatórias independentes, todas com a mesma esperança μ e variância $\sigma^2 < \infty$. Definindo a média amostral por*

$$\bar{X}_n = \frac{1}{n} \sum_{i=1}^n X_i,$$

então, para qualquer $\varepsilon > 0$,

$$\lim_{n \rightarrow \infty} \mathbb{P}(|\bar{X}_n - \mu| \geq \varepsilon) = 0.$$

Exercício 8. Prove a Lei Fraca dos Grandes números utilizando a desigualdade de Chebyshev.

Teorema 4 (Teorema Central do Limite). *Seja X_1, \dots, X_n uma sequência de variáveis aleatórias i.i.d. com esperança μ , variância σ^2 e momento de ordem 3 finito. Definimos a média amostral como*

$$\bar{X}_n = \frac{1}{n} \sum_{i=1}^n X_i.$$

Então,

$$\frac{\sqrt{n}(\bar{X}_n - \mu)}{\sigma} \xrightarrow{d} N(0, 1).$$

Demonstração. Suponha, sem perda de generalidade, que $\mu = 0$ e $\sigma = 1$. Defina

$$A_n = \frac{1}{\sqrt{n}} \sum_{i=1}^n X_i \quad \text{e} \quad B_n = \frac{1}{\sqrt{n}} \sum_{i=1}^n N_i,$$

onde $N_i \stackrel{\text{i.i.d.}}{\sim} N(0, 1)$ independentes de tudo. Note que $B_n \sim N(0, 1)$ para todo n .

Para provar que $A_n \xrightarrow{d} N(0, 1)$, é suficiente mostrar que, para qualquer função de teste f suave e com crescimento controlado,

$$\mathbb{E}[f(A_n)] - \mathbb{E}[f(B_n)] \longrightarrow 0.$$

Passo 1: Construção telescópica. Considere as variáveis intermediárias

$$\begin{aligned} C_n^{(0)} &= \frac{1}{\sqrt{n}}(X_1 + X_2 + \dots + X_n), \\ C_n^{(1)} &= \frac{1}{\sqrt{n}}(N_1 + X_2 + \dots + X_n), \\ C_n^{(2)} &= \frac{1}{\sqrt{n}}(N_1 + N_2 + X_3 + \dots + X_n), \\ &\vdots \\ C_n^{(n)} &= \frac{1}{\sqrt{n}}(N_1 + N_2 + \dots + N_n). \end{aligned}$$

Claramente, $C_n^{(0)} = A_n$ e $C_n^{(n)} = B_n$.

Assim,

$$\begin{aligned} \mathbb{E}[f(A_n)] - \mathbb{E}[f(B_n)] &= \mathbb{E}[f(C_n^{(0)})] - \mathbb{E}[f(C_n^{(n)})] \\ &= \sum_{k=1}^n \Delta_k, \end{aligned}$$

onde

$$\Delta_k := \mathbb{E}[f(C_n^{(k-1)}) - f(C_n^{(k)})].$$

Passo 2: Isolando o termo que difere. Entre $C_n^{(k)}$ e $C_n^{(k-1)}$, o único termo diferente é o k -ésimo. Definamos

$$D_n^{(k)} = \frac{1}{\sqrt{n}}(N_1 + \dots + N_{k-1} + 0 + X_{k+1} + \dots + X_n),$$

isto é, a parte comum entre $C_n^{(k)}$ e $C_n^{(k-1)}$, mas com o k -ésimo termo anulado.

Assim,

$$C_n^{(k)} = D_n^{(k)} + \frac{N_k}{\sqrt{n}}, \quad C_n^{(k-1)} = D_n^{(k)} + \frac{X_k}{\sqrt{n}}.$$

Portanto,

$$\Delta_k = \mathbb{E} \left[f \left(D_n^{(k)} + \frac{X_k}{\sqrt{n}} \right) - f \left(D_n^{(k)} + \frac{N_k}{\sqrt{n}} \right) \right].$$

Passo 3: Expansão de Taylor condicional. Fixe $D_n^{(k)} = d$. Aplicando Taylor em torno de d , temos:

$$\begin{aligned} f \left(d + \frac{X_k}{\sqrt{n}} \right) &= f(d) + \frac{X_k}{\sqrt{n}} f'(d) + \frac{X_k^2}{2n} f''(d) + \frac{X_k^3}{6n^{3/2}} f^{(3)}(d + \xi_X), \\ f \left(d + \frac{N_k}{\sqrt{n}} \right) &= f(d) + \frac{N_k}{\sqrt{n}} f'(d) + \frac{N_k^2}{2n} f''(d) + \frac{N_k^3}{6n^{3/2}} f^{(3)}(d + \xi_N), \end{aligned}$$

para alguns ξ_X, ξ_N entre 0 e X_k/\sqrt{n} ou N_k/\sqrt{n} .

Subtraindo,

$$f \left(d + \frac{X_k}{\sqrt{n}} \right) - f \left(d + \frac{N_k}{\sqrt{n}} \right) = \frac{1}{\sqrt{n}} f'(d)(X_k - N_k) + \frac{1}{2n} f''(d)(X_k^2 - N_k^2) + R_k(d),$$

onde

$$R_k(d) = \frac{1}{6n^{3/2}} \left(X_k^3 f^{(3)}(d + \xi_X) - N_k^3 f^{(3)}(d + \xi_N) \right).$$

Passo 4: Tomando esperança condicional. Voltamos para

$$\Delta_k = \mathbb{E} \left[f \left(D_n^{(k)} + \frac{X_k}{\sqrt{n}} \right) - f \left(D_n^{(k)} + \frac{N_k}{\sqrt{n}} \right) \right].$$

Usando a decomposição anterior e condicionando em $D_n^{(k)}$, temos:

$$\begin{aligned} \Delta_k &= \mathbb{E} \left[\frac{1}{\sqrt{n}} f'(D_n^{(k)})(X_k - N_k) \right] \\ &\quad + \mathbb{E} \left[\frac{1}{2n} f''(D_n^{(k)})(X_k^2 - N_k^2) \right] \\ &\quad + \mathbb{E}[R_k]. \end{aligned}$$

Agora, como X_k e N_k são independentes de $D_n^{(k)}$, obtemos:

$$\begin{aligned} \mathbb{E}[f'(D_n^{(k)})(X_k - N_k)] &= \mathbb{E}[f'(D_n^{(k)})] \cdot (\mathbb{E}[X_k] - \mathbb{E}[N_k]) = 0, \\ \mathbb{E}[f''(D_n^{(k)})(X_k^2 - N_k^2)] &= \mathbb{E}[f''(D_n^{(k)})] \cdot (\mathbb{E}[X_k^2] - \mathbb{E}[N_k^2]) = 0. \end{aligned}$$

Portanto, só resta

$$\Delta_k = \mathbb{E}[R_k].$$

Passo 5: Controle do resto. Do termo R_k , temos

$$|R_k| \leq \frac{1}{6n^{3/2}} \left(|X_k|^3 \sup |f^{(3)}| + |N_k|^3 \sup |f^{(3)}| \right).$$

Tomando esperança,

$$|\mathbb{E}[R_k]| \leq \frac{C}{n^{3/2}} (\mathbb{E}[|X_1|^3] + \mathbb{E}[|N_1|^3]),$$

onde $C = \frac{1}{6} \sup |f^{(3)}|$.

Somando sobre k ,

$$\left| \sum_{k=1}^n \Delta_k \right| \leq n \cdot \frac{C}{n^{3/2}} (\mathbb{E}[|X_1|^3] + \mathbb{E}[|N_1|^3]) = O\left(\frac{1}{\sqrt{n}}\right) \rightarrow 0.$$

Logo,

$$\mathbb{E}[f(A_n)] - \mathbb{E}[f(B_n)] \rightarrow 0,$$

e como $B_n \sim N(0, 1)$ para todo n , segue que $A_n \xrightarrow{d} N(0, 1)$. \square

Capítulo 3

Variáveis discretas e como simulá-las

O ponto de partida do nosso curso será sempre o mesmo: só podemos utilizar variáveis uniformes para gerar todas as demais distribuições. Ou seja, assumimos que temos disponível uma variável aleatória

$$U \sim \text{Uniforme}(0, 1),$$

e a partir dela construiremos algoritmos para simular outras variáveis.

A propriedade fundamental dessa variável é:

$$\mathbb{P}(a < U < b) = b - a, \quad 0 \leq a < b \leq 1.$$

Isto é, a probabilidade de U cair em um subintervalo do intervalo $(0, 1)$ é igual ao comprimento desse subintervalo.

Exercício 9. Seja $U \sim \text{Uniforme}(0, 1)$. Mostre que, para quaisquer números $0 \leq a < b \leq 1$,

$$\mathbb{P}(a < U < b) = b - a.$$

Para variáveis discretas, essa ideia pode ser usada da seguinte forma: suponha que X assuma valores x_1, x_2, \dots, x_m com probabilidades p_1, p_2, \dots, p_m , onde

$$p_k = \mathbb{P}(X = x_k), \quad p_k \geq 0, \quad \sum_{k=1}^m p_k = 1.$$

Definimos as probabilidades acumuladas

$$F_k = \sum_{i=1}^k p_i, \quad k = 1, \dots, m.$$

Então, o algoritmo de simulação é:

1. Gerar $U \sim \text{Uniforme}(0, 1)$;
2. Encontrar o menor índice k tal que $U \leq F_k$;
3. Retornar $X = x_k$.

A propriedade $\mathbb{P}(a < U < b) = b - a$ garante que

$$\mathbb{P}(X = x_k) = p_k.$$

De forma intuitiva, dividimos o intervalo $(0, 1)$ em subintervalos consecutivos de comprimentos p_k . Ao sortearmos $U \sim \text{Uniforme}(0, 1)$, o valor de X será aquele correspondente ao subintervalo no qual U cair. Esse procedimento é conhecido como *método da inversão* para variáveis discretas.

A Figura 3.1 ilustra esse processo para uma variável Bernoulli.

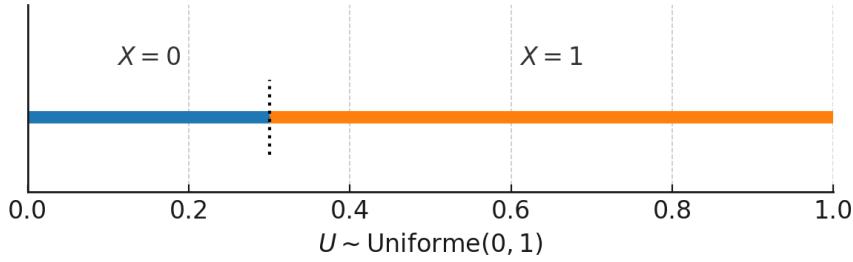


Figura 3.1: Particionamento do intervalo $(0, 1)$ para simular uma variável Bernoulli com $p = 0.7$. Sorteia-se $U \sim \text{Uniforme}(0, 1)$; se U cair na região azul, definimos $X = 0$, e caso contrário, $X = 1$.

A mesma ideia se aplica quando o conjunto de valores possíveis de X é infinito (ou muito grande). Nesse caso, o intervalo $(0, 1)$ é partitionado em uma sequência de subintervalos, cada um correspondente a um valor de X , como ilustrado na Figura 3.2.

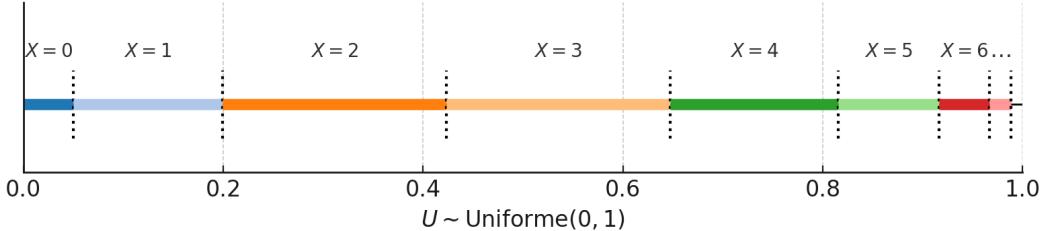


Figura 3.2: Particionamento do intervalo $(0, 1)$ para simular uma variável discreta com suporte infinito.

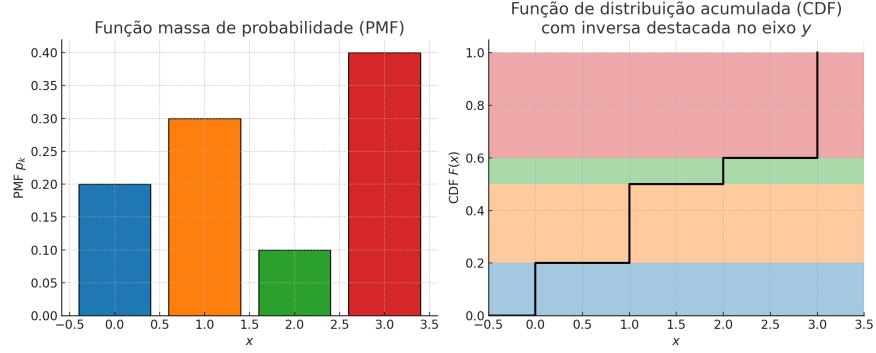
O nome *método da inversão* vem do fato de que a simulação utiliza a *função de distribuição acumulada* (CDF) e sua *inversa generalizada*. Seja X uma variável aleatória com CDF $F(x)$. Então, se $U \sim \text{Uniforme}(0, 1)$, vale que

$$X = F^{-1}(U),$$

onde a inversa generalizada é definida por

$$F^{-1}(u) = \min\{x : F(x) \geq u\}, \quad 0 < u < 1.$$

No caso discreto, isto corresponde exatamente ao passo do algoritmo em que escolhemos o menor k tal que $U \leq F_k$. Ou seja, sorteamos U , e depois “invertemos” a CDF para recuperar uma realização de X na sua escala original.



Esse procedimento pode parecer um pouco abstrato neste momento, já que a noção de inversa de uma função acumulada fica mais clara quando lidamos com variáveis contínuas. Por isso, retornaremos a esse método mais adiante, ao estudarmos a simulação de variáveis contínuas via inversão. Antes, porém, vale formalizar essa ideia de maneira geral.

Exercício 10. Seja X uma variável aleatória com função de distribuição acumulada F_X . Considere $U \sim \text{Uniforme}(0, 1)$ e defina

$$Y = F_X^{-1}(U), \quad \text{onde } F_X^{-1}(u) = \min\{x : F_X(x) \geq u\}.$$

Prove que Y tem a mesma distribuição que X .

Esse resultado mostra que, a partir de uma variável uniforme, podemos simular qualquer outra distribuição usando a CDF e sua inversa generalizada. Com essa ferramenta em mãos, passamos agora ao estudo de algumas distribuições discretas fundamentais, que servirão de exemplo concreto dessa ideia.

3.1 Variáveis com suporte finito

Comecemos com o caso em que X assume um número finito de valores x_1, x_2, \dots, x_m , cada um com probabilidade $p_j = \mathbb{P}(X = x_j)$.

Por exemplo, suponha que

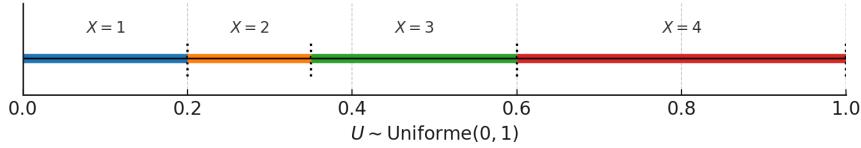
$$p_1 = 0.20, \quad p_2 = 0.15, \quad p_3 = 0.25, \quad p_4 = 0.40.$$

Uma maneira direta de simular X é gerar $U \sim \text{Uniforme}(0, 1)$ e aplicar:

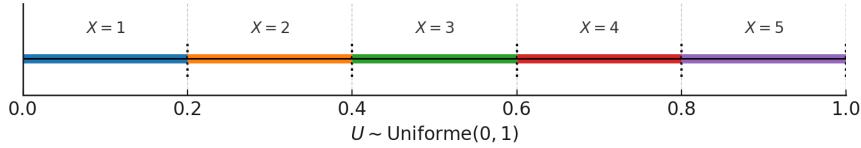
- Se $U < 0.20$, definir $X = 1$ e pare;
- Se $U < 0.35$, definir $X = 2$ e pare;
- Se $U < 0.60$, definir $X = 3$ e pare;
- Caso contrário, definir $X = 4$.

Embora possamos reordenar os testes para tornar a verificação mais eficiente, a ideia central permanece a mesma: dividir o intervalo $(0, 1)$ em partes de comprimentos p_j e identificar onde U caiu.

Exemplo: suporte finito com probabilidades distintas



Exemplo: uniforme discreta em $\{1, \dots, 5\}$



De forma geral, se X é uma variável com suporte finito $S = \{x_1, x_2, \dots, x_m\}$, sua distribuição é completamente determinada pela função de probabilidade

$$p_X(x_k) = \mathbb{P}(X = x_k), \quad x_k \in S,$$

a qual satisfaz

$$p_X(k) \geq 0 \quad \text{para todo } k \in S, \quad \sum_{k \in S} p_X(k) = 1.$$

Exemplo 2. Seja $S = \{x_1, x_2, \dots, x_K\}$ um conjunto de K valores distintos. Dizemos que X tem distribuição uniforme discreta em S quando

$$p_X(x_i) = \frac{1}{K}, \quad i = 1, 2, \dots, K.$$

Nesse caso, cada valor é igualmente provável e temos

$$\sum_{i=1}^K p_X(x_i) = \sum_{i=1}^K \frac{1}{K} = 1.$$

Um caso especial é a *uniforme discreta* nos inteiros $1, 2, \dots, n$, em que

$$\mathbb{P}(X = j) = \frac{1}{n}, \quad j = 1, \dots, n.$$

Neste cenário, o método se torna extremamente simples: basta gerar $U \sim \text{Uniforme}(0, 1)$ e definir

$$X = \lfloor nU \rfloor + 1,$$

onde $\lfloor x \rfloor$ indica a parte inteira de x (maior inteiro menor ou igual a x).

De fato, $X = j$ se e somente se $j - 1 \leq nU < j$, o que ocorre com probabilidade $\frac{1}{n}$. Variáveis uniformes discretas são particularmente importantes em simulação, pois permitem gerar inteiros equiprováveis de forma extremamente eficiente.

3.2 Bernoulli

A distribuição de Bernoulli modela experimentos com dois resultados possíveis, tipicamente denominados “sucesso” (valor 1) e “fracasso” (valor 0). Dizemos que $X \sim \text{Bernoulli}(p)$ se

$$\mathbb{P}(X = 1) = p \quad \text{e} \quad \mathbb{P}(X = 0) = 1 - p,$$

onde $0 \leq p \leq 1$ representa a probabilidade de sucesso.

A função de probabilidade (pmf) pode ser escrita de forma compacta como

$$p_X(k) = p^k(1-p)^{1-k}, \quad k \in \{0, 1\}.$$

As principais características dessa distribuição são:

$$\mathbb{E}[X] = p, \quad \text{Var}(X) = p(1-p).$$

Exercício 11. Prove as propriedades acima, isto é, calcule a esperança e a variância de uma variável Bernoulli.

No contexto de simulação, a Bernoulli é um caso particular da uniforme discreta em $\{0, 1\}$ com probabilidades $1 - p$ e p , respectivamente. O algoritmo é simples: sorteamos $U \sim \text{Uniforme}(0, 1)$ e definimos

$$X = \begin{cases} 1, & \text{se } U \leq p, \\ 0, & \text{caso contrário.} \end{cases}$$

Exercício 12. Mostre que o procedimento acima gera corretamente uma variável Bernoulli, isto é, verifique que $\mathbb{P}(X = 1) = p$ e $\mathbb{P}(X = 0) = 1 - p$.

3.3 Distribuição binomial

A distribuição binomial modela o número de sucessos em n repetições independentes de um experimento de Bernoulli com probabilidade de sucesso p .

Sejam X_1, X_2, \dots, X_n variáveis aleatórias independentes, todas com distribuição Bernoulli(p). Definimos

$$X = \sum_{i=1}^n X_i.$$

Nesse caso, dizemos que $X \sim \text{Binomial}(n, p)$, cuja função de probabilidade é

$$\mathbb{P}(X = k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad k = 0, 1, \dots, n.$$

As principais propriedades são:

$$\mathbb{E}[X] = np, \quad \text{Var}(X) = np(1-p).$$

Exercício 13. Prove as propriedades acima.

3.3.1 Simulando via Bernoullis

Uma forma simples e direta de simular uma variável aleatória binomial é a partir de variáveis de Bernoulli independentes.

Recorde que se $X \sim \text{Binomial}(n, p)$, então X pode ser escrito como

$$X = \sum_{i=1}^n B_i,$$

onde B_1, B_2, \dots, B_n são variáveis independentes e identicamente distribuídas, cada uma com

$$B_i \sim \text{Bernoulli}(p).$$

Assim, o algoritmo de simulação da binomial segue naturalmente:

1. Para $i = 1, \dots, n$, gerar $B_i \sim \text{Bernoulli}(p)$;
2. Retornar $X = \sum_{i=1}^n B_i$.

Em outras palavras, uma variável binomial conta o número de sucessos em n tentativas independentes, cada uma com probabilidade de sucesso p . Portanto, simular uma binomial se reduz a repetir n vezes o procedimento de simulação da Bernoulli e somar os resultados.

3.3.2 Simulando via identidade recursiva

Uma alternativa mais eficiente utiliza o *método da inversão*, aproveitando a identidade recursiva da função massa de probabilidade da Binomial.

Se $X \sim \text{Binomial}(n, p)$, então

$$\mathbb{P}(X = i) = \binom{n}{i} p^i (1-p)^{n-i}, \quad i = 0, 1, \dots, n.$$

Essas probabilidades satisfazem uma relação de recorrência simples. De fato, começando em

$$\mathbb{P}(X = i + 1) = \binom{n}{i+1} p^{i+1} (1-p)^{n-i-1},$$

observamos que ¹

$$\binom{n}{i+1} = \frac{n!}{(i+1)! (n-i-1)!} = \frac{n-i}{i+1} \frac{n!}{i! (n-i)!} = \frac{n-i}{i+1} \binom{n}{i}.$$

Substituindo essa relação,

$$\mathbb{P}(X = i + 1) = \frac{n-i}{i+1} \binom{n}{i} p^{i+1} (1-p)^{n-i-1}.$$

Reorganizando,

$$\mathbb{P}(X = i + 1) = \frac{n-i}{i+1} \cdot \frac{p}{1-p} \mathbb{P}(X = i).$$

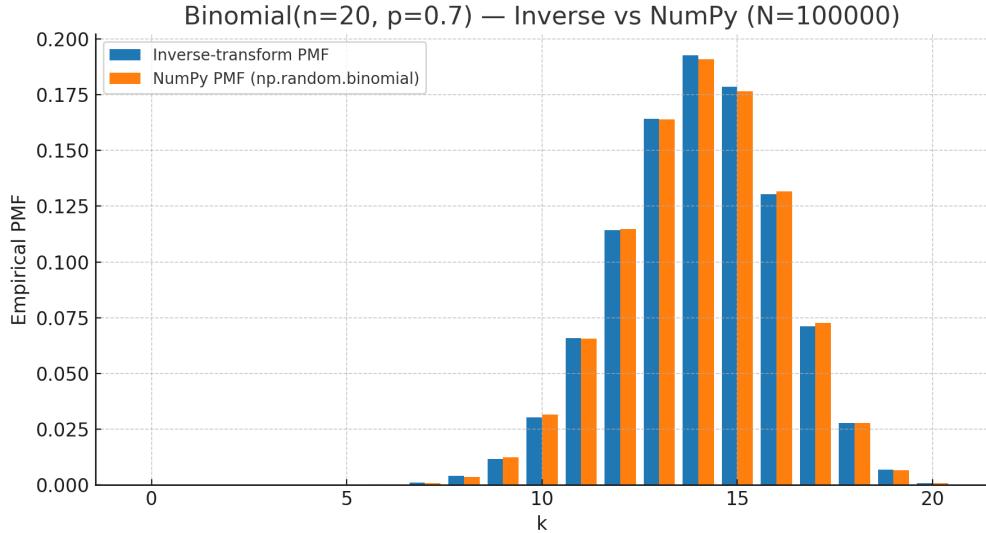
Assim, conhecendo $\mathbb{P}(X = 0) = (1-p)^n$, podemos calcular $\mathbb{P}(X = 1), \mathbb{P}(X = 2), \dots$ de forma recursiva, sem reavaliar coeficientes binomiais nem potências.

Isso leva ao seguinte algoritmo de simulação via inversão:

1. Gerar $U \sim \text{Uniforme}(0, 1)$;

¹Por exemplo, se $n = 10$, $i = 6$ e $i + 1 = 7$, então

$$\frac{10!}{7! 3!} = \frac{10! \cdot 4}{7 \cdot 6! \cdot 4 \cdot 3!} = \frac{10!}{6! 4!} \frac{4}{7}$$



2. Inicializar o índice $i = 0$, a probabilidade atual $p_i = (1 - p)^n$ e a soma acumulada $F = p_i$;
3. Enquanto $U > F$, atualizar

$$p_{i+1} = \frac{n-i}{i+1} \cdot \frac{p}{1-p} p_i, \quad i \leftarrow i + 1, \quad F \leftarrow F + p_i;$$

4. Retornar $X = i$.

Esse procedimento verifica primeiro se $X = 0$, depois se $X = 1$, e assim por diante, até encontrar o valor de X sorteado. Em média, o número de passos necessários é aproximadamente $1 + np$, o que pode ser bem mais eficiente do que gerar n variáveis de Bernoulli quando n é grande.

Exemplo 3. Considere $n = 5$ e $p = 0.3$. Temos $\mathbb{P}(X = 0) = (1 - 0.3)^5 = 0.16807$. Suponha que geramos $U = 0.4$. Como $U > 0.16807$, passamos ao próximo valor:

$$p_1 = \frac{5-0}{1} \cdot \frac{0.3}{0.7} \cdot 0.16807 \approx 0.36015, \quad F = 0.16807 + 0.36015 = 0.52822.$$

Agora $U = 0.4 < F$, logo o algoritmo retorna $X = 1$.

Portanto, neste caso específico, o sorteio resultou em exatamente um sucesso entre as cinco tentativas.

3.3.3 Aspectos computacionais

A escolha do método para simular variáveis binomiais tem implicações diretas em termos de eficiência. Dois fatores fundamentais influenciam o desempenho: o número de tentativas n e a probabilidade de sucesso p .

No método da soma de Bernoullis, o custo de cada amostra é proporcional a n , já que é necessário realizar n sorteios independentes. Esse custo não depende do valor de p : tanto para valores pequenos quanto grandes de p , o algoritmo precisa sempre gerar todas as n Bernoullis.

Já no método da inversão recursiva, o número médio de passos é da ordem de $1 + np$, pois o procedimento acumula probabilidades até ultrapassar o valor sorteado U . Quando p é pequeno,

o valor típico da variável X também é pequeno, e o algoritmo tende a parar cedo, podendo ser competitivo em relação à soma de Bernoullis. Por outro lado, quando p é moderado ou grande, o valor esperado np cresce e, com ele, o número de passos, tornando a inversão significativamente mais lenta.

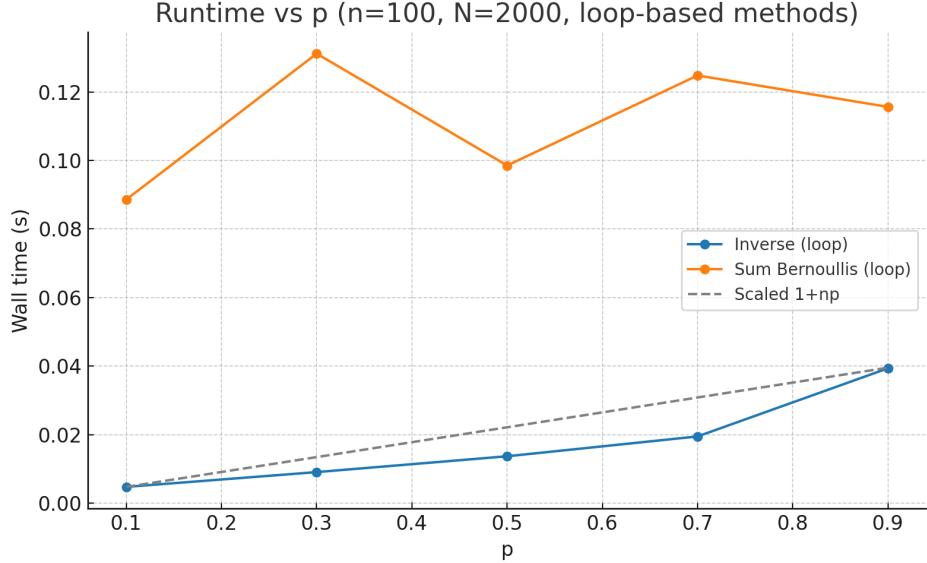


Figura 3.3: Comparaçāo de tempo de execuāo (em segundos) entre o mētodo da inversāo recursiva e a soma de Bernoullis para $n = 100$ e $N = 2000$ amostras, variando p .

A Figura 3.3 ilustra essa comparação em implementações com *loops* explícitos, para $n = 100$ e diferentes valores de p . Enquanto o tempo da soma de Bernoullis cresce linearmente apenas com n e não é afetado por p , o tempo do mētodo da inversāo cresce proporcionalmente a np , aumentando de forma acentuada à medida que p se aproxima de 1. Na prātica, bibliotecas como NumPy utilizam algoritmos especializados para a binomial, ainda mais rápidos do que ambos os mētodes discutidos aqui, de modo que a utilidade principal desses algoritmos é didática e comparativa, permitindo compreender os diferentes custos computacionais associados a cada abordagem.

3.3.4 Número médio de passos em algoritmos de inversāo recursiva

Nos algoritmos recursivos de inversāo, a lógica é sempre a mesma: dado um nūmero aleatório $U \sim \text{Uniforme}(0,1)$, acumulamos as probabilidades da distribuição até que a soma ultrapasse U . O valor de X sorteado é exatamente o índice k em que essa condição se verifica pela primeira vez.

Assim, se o valor sorteado é $X = k$, o algoritmo precisou verificar todos os valores $0, 1, 2, \dots, k - 1$ e só entāo aceitou k . Isso significa que o nūmero total de passos é

$$S = k + 1.$$

Como X é a variável aleatória que estamos simulando, temos

$$\mathbb{E}[S] = \mathbb{E}[X + 1] = \mathbb{E}[X] + 1.$$

Esse resultado é geral para qualquer algoritmo de inversão recursiva que inicie a busca no valor mínimo do suporte e avance de forma sequencial. No caso da binomial $X \sim \text{Bin}(n, p)$, por exemplo, o número esperado de passos é

$$\mathbb{E}[S] = 1 + np,$$

uma vez que $\mathbb{E}[X] = np$.

Portanto, o custo médio do algoritmo está diretamente ligado ao valor esperado da distribuição sorteada: distribuições concentradas em valores pequenos produzem simulações muito rápidas, enquanto distribuições centradas em valores grandes exigem proporcionalmente mais passos.

3.4 Distribuição geométrica

A distribuição geométrica modela o número de ensaios de Bernoulli até a ocorrência do primeiro sucesso. Seja p a probabilidade de sucesso em cada tentativa, com $0 < p \leq 1$. Definimos X como o número de ensaios necessários até o primeiro sucesso. Dizemos que $X \sim \text{Geom}(p)$ se

$$\mathbb{P}(X = k) = (1 - p)^{k-1}p, \quad k = 1, 2, 3, \dots$$

Nesse caso:

$$\mathbb{E}[X] = \frac{1}{p}, \quad \text{Var}(X) = \frac{1-p}{p^2}.$$

Exercício 14. Prove que a função de probabilidade acima satisfaz $\sum_{k=1}^{\infty} \mathbb{P}(X = k) = 1$.

Exercício 15. Prove as propriedades acima.

3.4.1 Simulando via Bernoullis

A distribuição geométrica modela o número de tentativas até a ocorrência do primeiro sucesso, em uma sequência de experimentos de Bernoulli independentes com probabilidade $p \in (0, 1)$ de sucesso. Essa definição leva naturalmente a um algoritmo de simulação baseado em Bernoullis. A ideia é repetir experimentos de Bernoulli até obter sucesso pela primeira vez:

1. Inicializar o contador $X \leftarrow 1$;
2. Gerar $B \sim \text{Bernoulli}(p)$;
3. Enquanto $B = 0$, repetir:
 - $X \leftarrow X + 1$;
 - Gerar novo $B \sim \text{Bernoulli}(p)$;
4. Retornar X .

Note que esse procedimento reflete exatamente a definição da variável: contar quantas tentativas são necessárias até que ocorra o primeiro sucesso.

Exemplo 4. Se $p = 0.3$, então a probabilidade de obter um sucesso logo na primeira tentativa é 0.3. Se a primeira tentativa falha, a segunda terá chance 0.3, e assim por diante.

Suponha que, ao simular, os primeiros valores de Bernoulli gerados foram 0, 0, 1. Isso indica duas falhas seguidas e um sucesso na terceira tentativa. Portanto, o algoritmo retorna $X = 3$.

3.4.2 Simulando a geométrica via inversão

A função de distribuição acumulada é

$$\mathbb{P}(X \leq j) = 1 - \mathbb{P}(X > j) = 1 - \mathbb{P}(\text{primeiras } j \text{ tentativas são falhas}) = 1 - (1 - p)^j.$$

Assim, podemos usar o método da inversão para gerar X . Seja $U \sim \text{Uniforme}(0,1)$. Definimos $X = j$ se

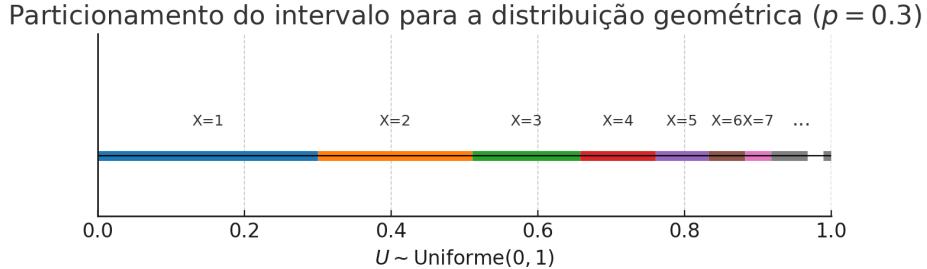
$$1 - (1 - p)^{j-1} \leq U < 1 - (1 - p)^j,$$

ou seja,

$$(1 - p)^j < 1 - U \leq (1 - p)^{j-1},$$

o que equivale a

$$X = \min\{j : (1 - p)^j < 1 - U\}.$$



Como $0 < 1 - p < 1$, temos $\log(1 - p) < 0$. Aplicando logaritmos,

$$(1 - p)^j < 1 - U \iff j \log(1 - p) < \log(1 - U).$$

Logo,

$$X = \min \left\{ j : j > \frac{\log(1 - U)}{\log(1 - p)} \right\}.$$

Portanto, obtemos a fórmula fechada

$$X = \left\lfloor \frac{\log(1 - U)}{\log(1 - p)} \right\rfloor + 1.$$

Como $1 - U \sim \text{Uniforme}(0,1)$, podemos substituir $1 - U$ por U sem perda de generalidade, resultando em

$$X = \left\lfloor \frac{\log(U)}{\log(1 - p)} \right\rfloor + 1.$$

Exemplo 5. Considere uma variável geométrica $X \sim \text{Geom}(p)$ com $p = 0.3$ e seja $U = 0.52$ uma realização de uma variável uniforme $(0, 1)$. Usando a fórmula fechada da inversão, temos

$$X = \left\lfloor \frac{\log(1 - U)}{\log(1 - p)} \right\rfloor + 1,$$

e, substituindo os valores, obtemos

$$X = \left\lfloor \frac{\log(0.48)}{\log(0.7)} \right\rfloor + 1 \approx \lfloor 2.06 \rfloor + 1 = 3.$$

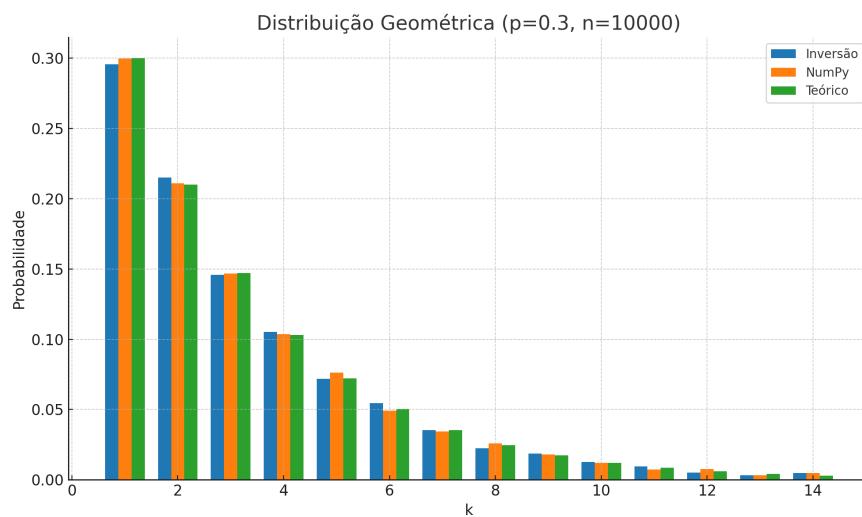
Outra forma é aplicar a inversão direta da CDF, que é dada por

$$F(j) = \mathbb{P}(X \leq j) = 1 - (1 - p)^j.$$

Procuramos o menor j tal que $F(j) \geq U$. Para $p = 0.3$, temos

$$F(1) = 0.3, \quad F(2) = 0.51, \quad F(3) \approx 0.657.$$

Como $F(2) = 0.51 < U = 0.52$ mas $F(3) = 0.657 \geq 0.52$, o menor j que satisfaz é $j = 3$.



3.5 Distribuição de Poisson

A distribuição de Poisson modela o número de ocorrências de um evento em um intervalo fixo de tempo ou espaço, assumindo que tais ocorrências sejam raras e independentes.

Dizemos que $X \sim \text{Poisson}(\lambda)$ se sua função de probabilidade for

$$\mathbb{P}(X = k) = \frac{e^{-\lambda} \lambda^k}{k!}, \quad k = 0, 1, 2, \dots,$$

onde $\lambda > 0$ representa a taxa média de ocorrências no intervalo considerado.

A média e a variância são dadas por

$$\mathbb{E}[X] = \lambda, \quad \text{Var}(X) = \lambda.$$

Exercício 16. Prove que $\sum_{k=0}^{\infty} \mathbb{P}(X = k) = 1$.

Exercício 17. Prove as propriedades acima. Dica: para a variância, calcule primeiro $\mathbb{E}[X(X - 1)]$ e use o fato de que

$$\text{Var}(X) = \mathbb{E}[X(X - 1)] + \mathbb{E}[X] - (\mathbb{E}[X])^2.$$

3.5.1 Simulação a Poisson via inversão e recursão

Seja $X \sim \text{Poisson}(\lambda)$. A função de probabilidade é

$$\mathbb{P}(X = i) = e^{-\lambda} \frac{\lambda^i}{i!}, \quad i = 0, 1, 2, \dots$$

e essa expressão satisfaz a relação de recorrência

$$\mathbb{P}(X = i + 1) = \frac{\lambda}{i + 1} \mathbb{P}(X = i).$$

Para ver isso, basta observar que $\mathbb{P}(X = i + 1) = e^{-\lambda} \frac{\lambda^{i+1}}{(i+1)!}$. Separando um fator $\lambda/(i+1)$, obtemos $\mathbb{P}(X = i + 1) = \frac{\lambda}{i+1} e^{-\lambda} \frac{\lambda^i}{i!}$, que nada mais é do que $\frac{\lambda}{i+1} \mathbb{P}(X = i)$. Assim, conhecendo $p_0 = \mathbb{P}(X = 0) = e^{-\lambda}$, é possível calcular recursivamente $p_1 = \lambda p_0$, depois $p_2 = (\lambda/2)p_1$, e assim sucessivamente. Esse raciocínio evita a recomputação de fatoriais a cada passo e fornece um procedimento numericamente mais estável.

O algoritmo clássico para gerar uma variável de Poisson com parâmetro λ funciona da seguinte maneira:

1. Gerar $U \sim \text{Uniforme}(0, 1)$;
2. Inicializar $i = 0$, $p_0 = e^{-\lambda}$ e $F = p_0$;
3. Enquanto $U > F$, atualizar

$$i \leftarrow i + 1, \quad p_i \leftarrow \frac{\lambda}{i} p_{i-1}, \quad F \leftarrow F + p_i;$$

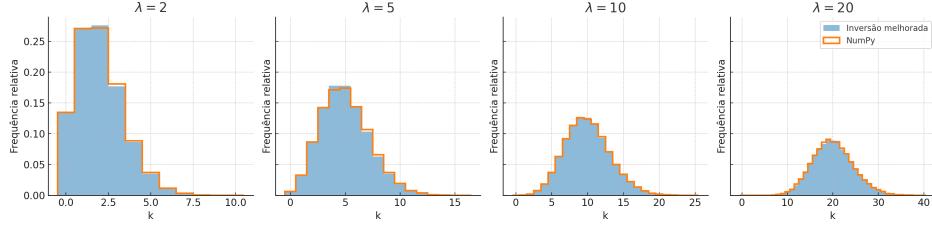
4. Retornar $X = i$.

Esse procedimento verifica primeiro se $X = 0$, depois se $X = 1$, e assim por diante, até encontrar o valor sorteado. O número médio de passos necessários é $1 + \lambda$, de modo que o algoritmo é eficiente para λ pequeno, mas se torna custoso para valores grandes de λ .

Exemplo 6. Considere $\lambda = 3$ e suponha que o número aleatório gerado seja $U = 0.35$.

- Primeiro, calculamos $p_0 = e^{-3} \approx 0.0498$ e $F = p_0 \approx 0.0498$. Como $U = 0.35 > F$, avançamos para o próximo valor.
- Calculamos $p_1 = \frac{3}{1} p_0 \approx 0.1494$ e atualizamos $F \approx 0.0498 + 0.1494 = 0.1992$. Ainda temos $U = 0.35 > F$, logo seguimos adiante.
- Agora $p_2 = \frac{3}{2} p_1 \approx 0.2240$ e $F \approx 0.1992 + 0.2240 = 0.4232$. Como $U = 0.35 < F$, o algoritmo para aqui e retornamos $X = 2$.

Portanto, neste exemplo, o valor simulado da variável aleatória foi $X = 2$.



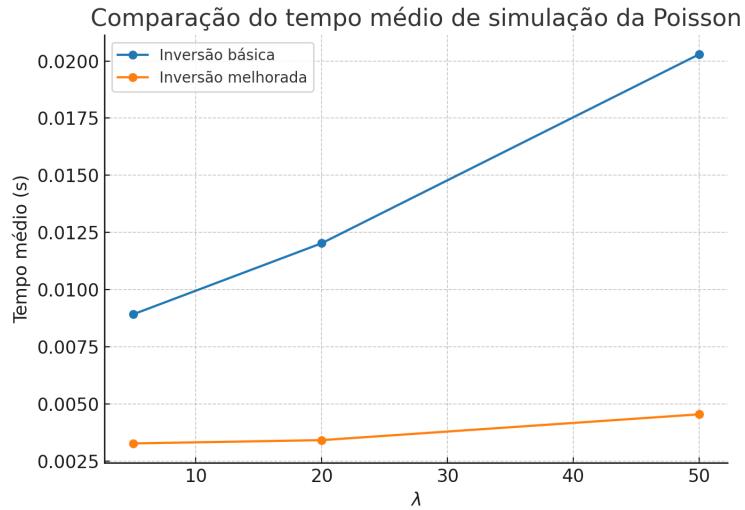
3.5.2 Algoritmo melhorado

Uma forma mais eficiente de implementar o método é iniciar a busca em torno do valor mais provável da variável, que está próximo de λ . Seja $m = \lfloor \lambda \rfloor$. Calcula-se a probabilidade acumulada

$$F(m) = \mathbb{P}(X \leq m),$$

usando a recorrência das probabilidades. Em seguida, gera-se $U \sim \text{Uniforme}(0,1)$ e procede-se assim:

- se $U < F(m)$, faz-se a busca recursiva para baixo ($m - 1, m - 2, \dots$);
- se $U \geq F(m)$, faz-se a busca recursiva para cima ($m + 1, m + 2, \dots$).



Note que a mesma identidade recursiva que relaciona $\mathbb{P}(X = i + 1)$ a $\mathbb{P}(X = i)$ também pode ser escrita no sentido inverso:

$$\mathbb{P}(X = i) = \frac{i+1}{\lambda} \mathbb{P}(X = i + 1).$$

Assim, a partir de $p_m = \mathbb{P}(X = m)$, é possível atualizar as probabilidades tanto para cima quanto para baixo, sem necessidade de recalcular fatoriais. Isso garante que a busca em torno de m seja realizada de forma eficiente, explorando o valor sorteado em ambas as direções.

Neste caso, o número de passos não depende mais diretamente de X , mas sim da distância entre X e λ : para localizar o valor sorteado, precisamos primeiro verificar m , e depois avançar $|X - m|$ passos adicionais. Assim, o número de passos é

$$T = 1 + |X - m|.$$

Como $m \approx \lambda$, o custo médio pode ser aproximado por

$$\mathbb{E}[T] = 1 + \mathbb{E}[|X - \lambda|].$$

3.5.3 Relação com a binomial

A distribuição de Poisson pode ser vista como um caso limite da distribuição binomial.

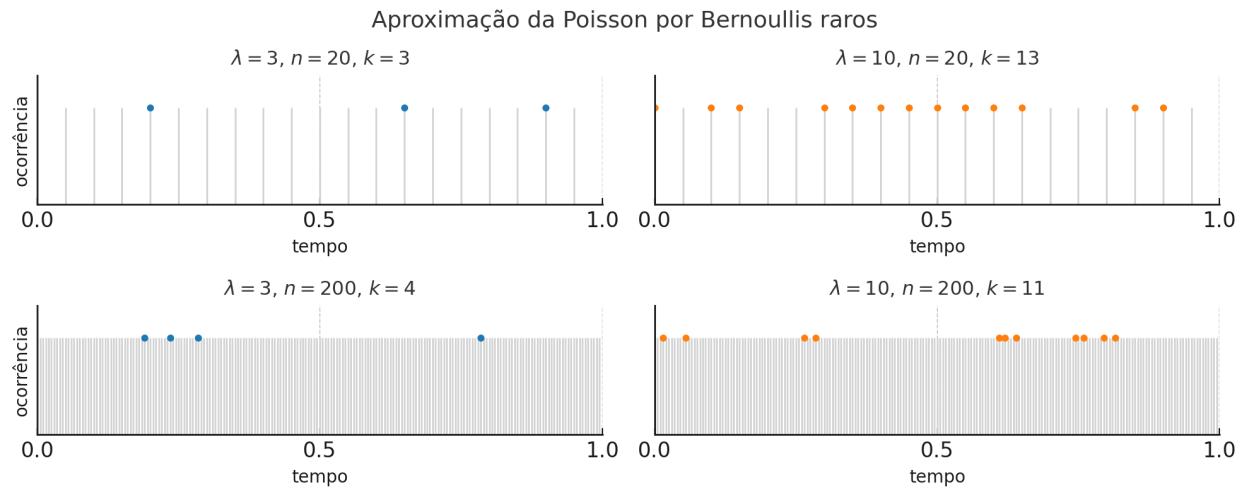
Seja $X \sim \text{Binomial}(n, p)$, que modela o número de sucessos em n tentativas independentes, cada uma com probabilidade p de sucesso. Suponha agora que

$$n \rightarrow \infty, \quad p \rightarrow 0, \quad \text{de modo que } \lambda = np \text{ permaneça constante.}$$

Nesse regime, dizemos que a binomial entra no chamado limite de Poisson, e temos

$$\text{Binomial}(n, p) \xrightarrow{d} \text{Poisson}(\lambda).$$

A função de probabilidade da binomial é



$$\mathbb{P}(X = k) = \binom{n}{k} p^k (1-p)^{n-k}.$$

Substituindo $p = \lambda/n$, obtemos

$$\mathbb{P}(X = k) = \binom{n}{k} \left(\frac{\lambda}{n}\right)^k \left(1 - \frac{\lambda}{n}\right)^{n-k}.$$

Para analisar o limite, consideramos cada fator separadamente. O coeficiente binomial pode ser escrito como

$$\binom{n}{k} = \frac{n(n-1) \cdots (n-k+1)}{k!}.$$

Dividindo numerador e denominador por n^k , temos

$$\binom{n}{k} = \frac{n^k}{k!} \cdot \frac{n(n-1) \cdots (n-k+1)}{n^k} = \frac{n^k}{k!} \cdot \frac{n}{n} \frac{n-1}{n} \cdots \frac{n-k+1}{n}.$$

Cada termo do produto no numerador pode ser escrito como

$$\frac{n-j}{n} = 1 - \frac{j}{n}, \quad j = 0, 1, \dots, k-1,$$

e portanto

$$\binom{n}{k} = \frac{n^k}{k!} \cdot \left(1 - \frac{1}{n}\right) \left(1 - \frac{2}{n}\right) \cdots \left(1 - \frac{k-1}{n}\right).$$

Quando $n \rightarrow \infty$, cada termo do produto tende a 1, de modo que

$$\binom{n}{k} \sim \frac{n^k}{k!}.$$

Além disso,

$$\left(1 - \frac{\lambda}{n}\right)^n \longrightarrow e^{-\lambda}, \quad \left(1 - \frac{\lambda}{n}\right)^{-k} \longrightarrow 1$$

Juntando os três resultados, obtemos

$$\mathbb{P}(X = k) \sim \frac{n^k}{k!} \left(\frac{\lambda}{n}\right)^k e^{-\lambda} \cdot 1 = \frac{\lambda^k}{k!} e^{-\lambda}.$$

Portanto,

$$\lim_{n \rightarrow \infty} \mathbb{P}(X = k) = \frac{e^{-\lambda} \lambda^k}{k!},$$

que é exatamente a função de massa de probabilidade da distribuição Poisson(λ).

3.6 Distribuição binomial negativa

Seja X o número de ensaios necessários para obter um total de r sucessos, considerando que cada ensaio é independente e resulta em sucesso com probabilidade p . Nesse caso, dizemos que X segue uma distribuição *binomial negativa* (também chamada *Pascal*) com parâmetros p e r .

Sua função de probabilidade é dada por:

$$\mathbb{P}(X = n) = \binom{n-1}{r-1} p^r (1-p)^{n-r}, \quad n = r, r+1, r+2, \dots$$

Essa fórmula é justificada pelo fato de que, para que sejam necessários exatamente n ensaios para obter r sucessos, os primeiros $n-1$ ensaios devem conter exatamente $r-1$ sucessos — o que ocorre com probabilidade

$$\binom{n-1}{r-1} p^{r-1} (1-p)^{n-r}$$

— e, em seguida, o n -ésimo ensaio deve ser um sucesso, com probabilidade p .

Seja X_i , $i = 1, \dots, r$, o número de ensaios necessários após o $(i-1)$ -ésimo sucesso para obter o i -ésimo sucesso. É fácil ver que X_1, X_2, \dots, X_r são variáveis aleatórias independentes com distribuição Geom(p). Assim, como

$$X = X_1 + X_2 + \cdots + X_r,$$

temos, usando os resultados da distribuição geométrica:

$$\mathbb{E}[X] = \sum_{i=1}^r \mathbb{E}[X_i] = \frac{r}{p}, \quad \text{Var}(X) = \sum_{i=1}^r \text{Var}(X_i) = \frac{r(1-p)}{p^2}.$$

Exercício 18. Prove que a função de probabilidade acima é válida, isto é, que $\sum_{n=r}^{\infty} \mathbb{P}(X = n) = 1$.

Exercício 19. Prove as fórmulas da média e variância usando o fato de que X é a soma de r variáveis independentes com distribuição Geom(p).

3.6.1 Simulando via Bernoullis

Uma forma direta de gerar uma variável Binomial Negativa é simular sucessivos ensaios de Bernoulli(p) até obter o r -ésimo sucesso.

De fato, por definição, X representa o número total de ensaios necessários até a ocorrência de r sucessos. Assim, o algoritmo pode ser descrito da seguinte forma:

1. Inicializar $n = 0$ (contador de ensaios) e $s = 0$ (contador de sucessos);
2. Enquanto $s < r$:
 - (a) Gerar $B \sim \text{Bernoulli}(p)$;
 - (b) Atualizar $n \leftarrow n + 1$;
 - (c) Se $B = 1$, atualizar $s \leftarrow s + 1$;
3. Retornar $X = n$.

Esse método é conceitualmente simples e corresponde exatamente à definição da distribuição Binomial Negativa. No entanto, quando p é pequeno e r é grande, o número esperado de ensaios $\mathbb{E}[X] = r/p$ pode ser elevado, tornando o algoritmo computacionalmente mais custoso.

3.6.2 Simulando via soma de geométricas

Recorde que se $X \sim \text{NegBin}(r, p)$, então X pode ser decomposto como

$$X = X_1 + X_2 + \cdots + X_r,$$

onde X_1, \dots, X_r são variáveis independentes com

$$X_i \sim \text{Geom}(p), \quad i = 1, \dots, r,$$

no suporte $\{1, 2, \dots\}$ (número de ensaios até o primeiro sucesso).

Assim, podemos simular uma Binomial Negativa somando r geométricas independentes, cada uma gerada via o método da inversão:

$$X_i = \left\lfloor \frac{\log(1 - U_i)}{\log(1 - p)} \right\rfloor + 1, \quad U_i \sim \text{Uniforme}(0, 1).$$

O algoritmo de simulação segue:

1. Para $i = 1, \dots, r$, gerar $X_i \sim \text{Geom}(p)$ via inversão;
2. Retornar $X = \sum_{i=1}^r X_i$.

3.6.3 Simulando via inversão recursiva

Outra forma de simular a Binomial Negativa é aplicar diretamente o método da inversão, aproveitando a relação de recorrência da sua função de probabilidade.

Se $X \sim \text{NegBin}(r, p)$, então

$$\mathbb{P}(X = n) = \binom{n-1}{r-1} p^r (1-p)^{n-r}, \quad n = r, r+1, r+2, \dots$$

Essas probabilidades satisfazem a seguinte relação recursiva:

$$\frac{\mathbb{P}(X = n+1)}{\mathbb{P}(X = n)} = \frac{n}{n-r+1} (1-p).$$

Exercício 20. Prove a identidade recursiva acima.

Portanto, conhecendo $\mathbb{P}(X = r) = p^r$, podemos calcular recursivamente as demais probabilidades. Isso leva ao seguinte algoritmo:

1. Gerar $U \sim \text{Uniforme}(0,1)$;
2. Inicializar $n = r$, $p_n = p^r$, $F = p_n$;
3. Enquanto $U > F$, atualizar

$$p_{n+1} = p_n \cdot \frac{n}{n-r+1} (1-p), \quad n \leftarrow n+1, \quad F \leftarrow F + p_{n+1};$$

4. Retornar $X = n$.

O número esperado de passos T no método recursivo não coincide diretamente com $\mathbb{E}[X]$, pois o algoritmo já inicia em $n = r$, que é o menor valor possível para a variável $X \sim \text{NegBin}(r, p)$.

De fato, se o sorteio resultar em $X = n$, o número de passos dados pelo algoritmo é

$$T = (n - r) + 1,$$

pois começamos verificando o valor $n = r$ (primeiro passo) e avançamos até alcançar n .

Assim, em termos de valor esperado,

$$\mathbb{E}[T] = \mathbb{E}[X - r] + 1 = \frac{r}{p} - r + 1.$$

Esse termo $-r$ aparece porque, embora $\mathbb{E}[X] = r/p$, o procedimento de inversão não percorre todos os valores desde 0, mas já parte de r .

Quando p é pequeno, $\mathbb{E}[T]$ pode ainda ser bastante grande, tornando o método recursivo lento. Nessas situações, a versão ingênua baseada na soma de geométricas pode ser mais eficiente na prática.

3.6.4 Por que o nome “Binomial Negativa”?

O nome Binomial Negativa tem origem na conexão com a expansão binomial para expoentes negativos. Para um inteiro $n \geq 0$ e $p + q = 1$, o teorema binomial fornece

$$1 = (p + q)^n = \sum_{k=0}^n \binom{n}{k} p^k q^{n-k}.$$

Essa identidade estende-se a expoentes reais (expansão binomial generalizada):

$$(1 - p)^{-\alpha} = \sum_{k=0}^{\infty} \binom{\alpha + k - 1}{k} p^k, \quad |p| < 1.$$

Tomando $\alpha = r \in \{1, 2, \dots\}$,

$$(1 - p)^{-r} = \sum_{k=0}^{\infty} \binom{r + k - 1}{k} p^k.$$

Os coeficientes $\binom{r+k-1}{k}$ são precisamente os que aparecem na parametrização da Binomial Negativa em termos do número de falhas k antes do r -ésimo sucesso:

$$\mathbb{P}(Y = k) = \binom{r + k - 1}{k} p^r (1 - p)^k, \quad k = 0, 1, 2, \dots$$

Para ver a equivalência com a forma escrita em função do número total de ensaios n , detalhamos a reparametrização. Defina $n = r + k$ (isto é, $k = n - r$). Então

$$\binom{r + k - 1}{k} = \frac{(r + k - 1)!}{k! (r - 1)!} = \frac{(n - 1)!}{(n - r)! (r - 1)!} = \binom{n - 1}{n - r}.$$

Pela simetria binomial, $\binom{a}{b} = \binom{a}{a-b}$; aplicando com $a = n - 1$ e $b = n - r$ obtemos

$$\binom{n - 1}{n - r} = \binom{n - 1}{(n - 1) - (n - r)} = \binom{n - 1}{r - 1}.$$

Substituindo $k = n - r$ em $\mathbb{P}(Y = k)$ e usando as igualdades acima,

$$\begin{aligned} \mathbb{P}(X = n) &= \mathbb{P}(Y = n - r) = \binom{r + (n - r) - 1}{n - r} p^r (1 - p)^{n-r} \\ &= \binom{n - 1}{r - 1} p^r (1 - p)^{n-r}, \quad n = r, r + 1, \dots \end{aligned}$$

Mostramos, assim, passo a passo, que as duas formas da PMF — em função de k (falhas) ou de n (ensaios) — são exatamente equivalentes; trata-se apenas de uma reparametrização.

3.7 Distribuição hipergeométrica

A distribuição hipergeométrica modela experimentos de seleção *sem reposição* a partir de uma população finita contendo dois tipos de elementos. Por exemplo, suponha uma urna com $N + M$ bolas, das quais N são claras e M são escuras. Retiramos, de forma aleatória e sem reposição, uma amostra de tamanho n . Seja X o número de bolas claras na amostra.

Nesse caso, cada subconjunto de tamanho n é igualmente provável, e a probabilidade de observar exatamente k bolas claras é

$$\mathbb{P}(X = k) = \frac{\binom{N}{k} \binom{M}{n-k}}{\binom{N+M}{n}}, \quad \max(0, n-M) \leq k \leq \min(n, N).$$

Dizemos então que $X \sim \text{Hipergeom}(N, M, n)$.

As principais propriedades dessa distribuição são:

$$\mathbb{E}[X] = n \cdot \frac{N}{N+M}, \quad \text{Var}(X) = n \cdot \frac{N}{N+M} \cdot \frac{M}{N+M} \cdot \frac{N+M-n}{N+M-1}.$$

Exercício 21. Prove que a função de probabilidade acima é válida, isto é, que

$$\sum_{k=\max(0, n-M)}^{\min(n, N)} \mathbb{P}(X = k) = 1.$$

Exercício 22. Prove as fórmulas da média e variância acima. Dica: considere o sorteio sequencial das n bolas e defina X_i como a variável indicadora do evento “a i -ésima bola é clara”. Para a variância, use a decomposição

$$\text{Var}(X) = \sum_{i=1}^n \text{Var}(X_i) + 2 \sum_{1 \leq i < j \leq n} \text{Cov}(X_i, X_j).$$

3.7.1 Simulando a Hipergeométrica

Uma maneira natural de simular $X \sim \text{Hipergeom}(N, K, n)$ é reproduzir o sorteio sem reposição. Basta imaginar uma população com K sucessos e $N - K$ fracassos, e retirar n elementos dela. O valor de X será o número de sucessos observados. Isso leva ao seguinte algoritmo:

1. Construir a população formada por K uns (sucessos) e $N - K$ zeros (fracassos);
2. Sortear n elementos dessa população sem reposição;
3. Definir X como a soma dos elementos sorteados;
4. Retornar X .

Para realizar o sorteio sem reposição, podemos usar um procedimento eficiente baseado no *embaralhamento parcial de Fisher-Yates*. A ideia é que não precisamos embaralhar toda a população, apenas selecionar n elementos distintos de forma aleatória. O algoritmo funciona assim:

1. Coloque os elementos da população em um vetor de tamanho N ;
2. Para cada posição $i = 1, 2, \dots, n$:
 - (a) Sorteie um índice j uniformemente entre i e N ;
 - (b) Troque os elementos das posições i e j .
3. Os n primeiros elementos do vetor agora constituem a amostra sem reposição.

Esse método garante que cada subconjunto de tamanho n tem a mesma probabilidade de ser escolhido, e é mais eficiente do que embaralhar toda a população.

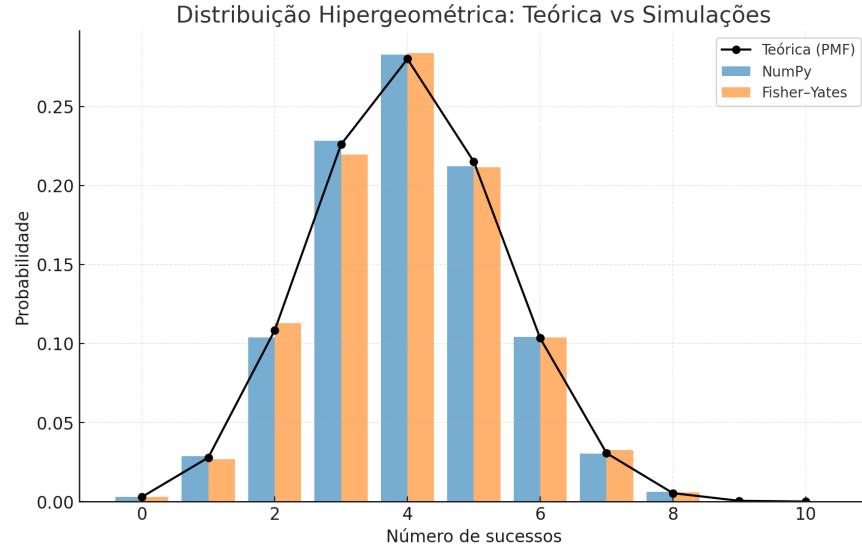


Figura 3.4: Distribuição Hipergeométrica com parâmetros $N = 50$ (população total), $K = 20$ (número de sucessos) e $n = 10$ (tamanho da amostra).

Tabela de referência

Distribuição	Técnica utilizada	Dica/Obs
Suporte finito	Inversão simples	Separar em intervalos
Bernoulli	Inversão simples	Caso particular do suporte finito com $m = 2$
Binomial	Soma de Bernoullis ou inversão recursiva	Relação de recorrência evita coeficientes binomiais
Poisson	Inversão recursiva	Relação $p_{i+1} = \frac{\lambda}{i+1} p_i$ evita fatoriais
Geométrica	Inversão direta (CDF)	Retornar $X = \lfloor \frac{\log(1-U)}{\log(1-p)} \rfloor + 1$
Binomial negativa	Soma de geométricas ou inversão recursiva	Soma de r geométricas independentes
Hipergeométrica	Sorteio sem reposição (Fisher-Yates)	Selecionar n elementos distintos e contar sucessos

Capítulo 4

Variáveis contínuas e como simulá-las

Nosso objetivo agora é estudar algoritmos para simular variáveis aleatórias contínuas, isto é, variáveis cuja distribuição é descrita por uma função densidade de probabilidade.

Como no caso discreto, o ponto de partida será sempre o mesmo: assumimos que temos acesso a uma variável

$$U \sim \text{Uniforme}(0, 1),$$

e construiremos a partir dela procedimentos para gerar amostras de outras distribuições.

A principal diferença em relação ao caso discreto é que, para variáveis contínuas, muitas vezes **não é possível** escrever a função de distribuição acumulada (CDF) de forma explícita, ou mesmo obter sua inversa em forma fechada. Com isso, diversos métodos alternativos são necessários.

Neste capítulo, organizamos os métodos de simulação em três grandes grupos:

- **Métodos por inversão:** funcionam diretamente a partir da CDF da distribuição;
- **Métodos por rejeição ou aceitação:** baseiam-se em gerar propostas e aceitar com certa probabilidade;
- **Métodos por transformação:** aplicam funções determinísticas a variáveis já conhecidas.

4.1 Método da Inversão

O método da inversão é uma das formas mais diretas de simular variáveis aleatórias contínuas.

A ideia central é simples: se conhecemos a função de distribuição acumulada (CDF) F de uma variável contínua X , e se essa função é estritamente crescente, então podemos inverter F e definir

$$X = F^{-1}(U), \quad \text{com } U \sim \text{Uniforme}(0, 1).$$

Exemplo 7. Prove o fato acima.

Esse procedimento garante que X terá exatamente a distribuição desejada, pois a probabilidade de X cair em qualquer intervalo será proporcional ao comprimento correspondente no domínio de U .

Esse método é particularmente útil quando a inversa de F pode ser escrita de forma explícita, como ocorre com as distribuições Exponencial, Uniforme e Pareto, por exemplo.

4.1.1 Distribuição exponencial

A distribuição exponencial modela o tempo de espera até a ocorrência de um evento em um processo de Poisson, isto é, um processo no qual eventos ocorrem de forma contínua e independente, a uma taxa constante. Seja $\lambda > 0$ a taxa de ocorrência dos eventos. Dizemos que $X \sim \text{Exponencial}(\lambda)$ se

$$f_X(x) = \lambda e^{-\lambda x}, \quad x \geq 0.$$

A função de distribuição acumulada (CDF) é dada por:

$$F_X(x) = \mathbb{P}(X \leq x) = 1 - e^{-\lambda x}, \quad x \geq 0.$$

As principais características da distribuição são:

$$\mathbb{E}[X] = \frac{1}{\lambda}, \quad \text{Var}(X) = \frac{1}{\lambda^2}.$$

Exercício 23. Verifique que $f_X(x)$ é uma densidade de probabilidade, isto é, $\int_0^\infty f_X(x) dx = 1$.

Exercício 24. Prove as expressões da média e variância acima.

A distribuição exponencial é um exemplo clássico onde o método da inversão pode ser aplicado diretamente. Sabemos que a CDF é

$$F(x) = 1 - e^{-\lambda x},$$

e queremos encontrar sua inversa, portanto, basta resolver a equação $F(x) = U$, com $U \sim \text{Uniforme}(0,1)$. Isso nos leva a:

$$1 - e^{-\lambda x} = U \quad \Rightarrow \quad x = -\frac{1}{\lambda} \log(1 - U).$$

Como $1 - U \sim \text{Uniforme}(0,1)$, podemos reescrever de forma equivalente:

$$X = -\frac{1}{\lambda} \log(U), \quad \text{com } U \sim \text{Uniforme}(0,1).$$

Assim, o algoritmo para simular uma variável $X \sim \text{Exponencial}(\lambda)$ via inversão é:

1. Gerar $U \sim \text{Uniforme}(0,1)$;
2. Calcular $X = -\frac{1}{\lambda} \log(U)$;
3. Retornar X .

Relação com a distribuição geométrica

A distribuição exponencial pode ser vista como o análogo contínuo da distribuição geométrica.

Na distribuição geométrica, $X \sim \text{Geom}(p)$, interpretamos X como o número de tentativas independentes até a ocorrência do primeiro sucesso, em uma sequência de ensaios de Bernoulli com probabilidade p de sucesso.

A distribuição exponencial, por sua vez, modela o tempo contínuo até a ocorrência de um evento, sob uma taxa constante $\lambda > 0$. Embora uma seja discreta e a outra contínua, existe uma relação direta entre essas duas distribuições, que pode ser formalizada por um limite.

Seja $X_n \sim \text{Geom}(p_n)$, com $p_n = \lambda/n$, e defina a variável reescalada

$$T_n = \frac{X_n}{n}.$$

A variável T_n representa o tempo até o primeiro sucesso quando fazemos n tentativas por unidade de tempo, cada uma com probabilidade de sucesso $p_n = \lambda/n$. À medida que $n \rightarrow \infty$, as tentativas se tornam mais frequentes e individualmente menos prováveis, mas o número esperado de sucessos por unidade de tempo permanece constante: $n \cdot p_n = \lambda$.

Vamos mostrar que T_n converge em distribuição para uma variável exponencial de parâmetro λ . De fato, temos:

$$\mathbb{P}(T_n > t) = \mathbb{P}\left(\frac{X_n}{n} > t\right) = \mathbb{P}(X_n > \lfloor nt \rfloor).$$

Como X_n é geométrica com parâmetro $p_n = \lambda/n$, segue que:

$$\mathbb{P}(X_n > k) = (1 - p_n)^k, \quad \text{logo} \quad \mathbb{P}(T_n > t) = \left(1 - \frac{\lambda}{n}\right)^{\lfloor nt \rfloor}.$$

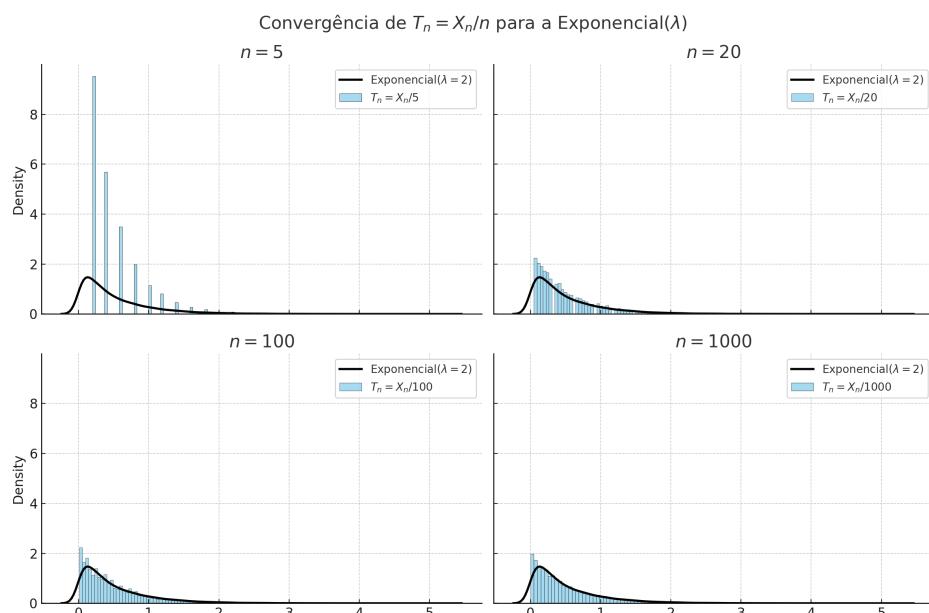
Quando $n \rightarrow \infty$, vale que $\lfloor nt \rfloor \sim nt$, e obtemos:

$$\left(1 - \frac{\lambda}{n}\right)^{nt} \longrightarrow e^{-\lambda t}.$$

Portanto,

$$\mathbb{P}(T_n \leq t) \rightarrow 1 - e^{-\lambda t},$$

que é a função de distribuição acumulada da exponencial $\text{Exp}(\lambda)$. Isso conclui a demonstração da convergência.



Essa convergência tem uma interpretação intuitiva. Inicialmente, a variável X_n conta o número de tentativas até o sucesso. Se cada tentativa leva um tempo fixo de $1/n$ segundos, então o tempo total até o sucesso é $T_n = X_n/n$.

A divisão por n serve justamente para transformar o número de tentativas em tempo contínuo. Por exemplo, se cada tentativa leva 0.01 segundo e o sucesso ocorre na 17ª tentativa, então o tempo até o sucesso foi $17 \times 0.01 = 0.17$ segundos.

À medida que n cresce, as tentativas são feitas cada vez mais rapidamente (a cada $1/n$ unidades de tempo), e a chance de sucesso em cada uma cai proporcionalmente ($p_n = \lambda/n$). O resultado final é que o tempo total até o sucesso — T_n — se aproxima de uma variável contínua exponencial com taxa λ .

Essa relação também pode ser observada diretamente nas fórmulas de inversão utilizadas para simulação.

Seja $U \sim \text{Uniforme}(0, 1)$. A inversão da CDF da exponencial dá:

$$T = -\frac{1}{\lambda} \ln(U).$$

Já no caso da geométrica $X_n \sim \text{Geom}(p_n)$, a fórmula de inversão baseada na CDF discreta é:

$$X_n = \left\lceil \frac{\ln(U)}{\ln(1 - p_n)} \right\rceil, \quad \text{com } p_n = \frac{\lambda}{n}.$$

Dividindo por n , temos:

$$T_n = \frac{X_n}{n} \approx \frac{1}{n} \cdot \frac{\ln(U)}{\ln(1 - \lambda/n)}.$$

Sabemos que para n grande,

$$\ln(1 - \lambda/n) \approx -\frac{\lambda}{n},$$

então:

$$T_n \approx -\frac{1}{\lambda} \ln(U),$$

o que mostra que, no limite, a fórmula de simulação da geométrica reescalada **tende para a fórmula da exponencial**.

Relação com a Poisson

A distribuição exponencial pode ser entendida como o análogo contínuo da distribuição geométrica, e sua relação com a distribuição de Poisson surge naturalmente ao considerarmos divisões finas de um intervalo fixo em pequenos subintervalos com experimentos de Bernoulli raros.

Considere o intervalo de tempo $[0, 1]$ dividido em n subintervalos de comprimento $1/n$. Em cada subintervalo, ocorre um evento (ou sucesso) com probabilidade $p_n = \lambda/n$, de forma independente. Este é exatamente o modelo da variável binomial

$$X_n \sim \text{Binomial}(n, \lambda/n),$$

que conta o número total de eventos no intervalo. Sabemos que, quando $n \rightarrow \infty$,

$$X_n \xrightarrow{d} \text{Poisson}(\lambda).$$

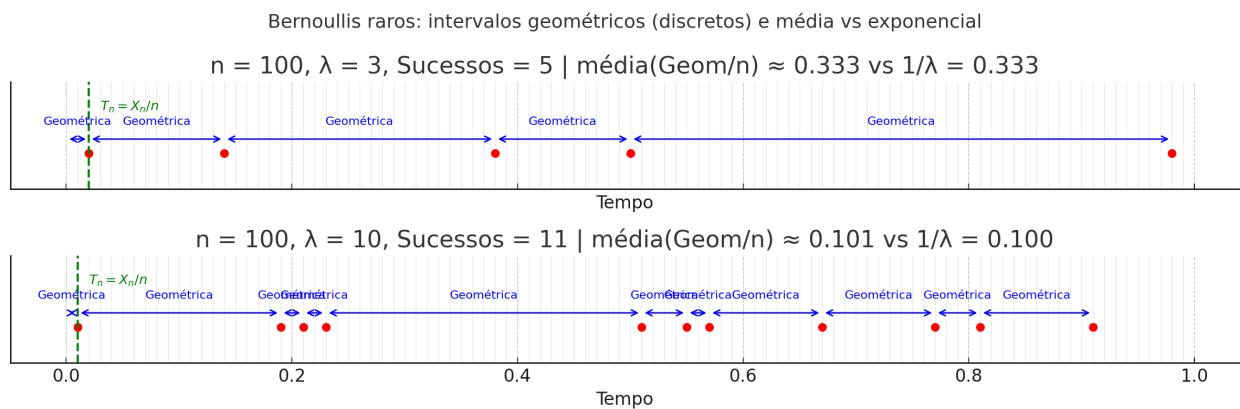
Por outro lado, podemos perguntar: quanto tempo leva até o primeiro evento acontecer? A resposta a essa pergunta leva à distribuição exponencial.

Seja $X_n \sim \text{Geom}(p_n)$ com $p_n = \lambda/n$, modelando o número de subintervalos até o primeiro sucesso. O tempo contínuo correspondente é então

$$T_n = \frac{X_n}{n}.$$

Como visto anteriormente, temos

$$T_n \xrightarrow{d} \text{Exponencial}(\lambda).$$



Portanto, podemos pensar nessas distribuições da seguinte forma:

- A distribuição *Poisson* modela o número total de eventos no intervalo.
- A distribuição *geométrica* modela a posição discreta do primeiro sucesso.
- A distribuição *exponencial* modela o tempo contínuo até o primeiro evento.

4.2 Método da rejeição-aceitação

Embora o método da inversão funcione muito bem para distribuições cuja função de distribuição acumulada (CDF) possa ser invertida de forma analítica ou computacionalmente eficiente, ele se torna inviável em casos como o da distribuição normal padrão. A função de distribuição acumulada da normal, denotada por $\Phi(x)$, não possui inversa em forma fechada, o que impede a aplicação direta da fórmula $X = \Phi^{-1}(U)$. Embora existam aproximações numéricas para Φ^{-1} , elas podem ser computacionalmente custosas ou introduzir erros de arredondamento. Nesses casos, recorre-se a métodos alternativos que não exigem a inversão da CDF — como o método da rejeição-aceitação.

O método de aceitação-rejeição é uma técnica geral para gerar variáveis aleatórias com uma dada densidade $f(x)$, partindo de uma densidade auxiliar $g(x)$ mais simples, da qual é fácil simular. A ideia central é gerar candidatos a partir de g e aceitá-los com uma certa probabilidade que depende da razão $f(x)/g(x)$.

Suponha que desejamos gerar uma variável aleatória X com densidade alvo $f(x)$, mas não dispomos de um método direto para isso. Por outro lado, assumimos que sabemos simular uma variável Y com densidade auxiliar $g(x)$, e que existe uma constante $c > 0$ tal que

$$\frac{f(x)}{g(x)} \leq c \quad \text{para todo } x.$$

Essa condição garante que a função f está sempre abaixo da curva cg , ou seja, $f(x) \leq cg(x)$ para todo x . Além disso, isso assegura que a razão $\frac{f(x)}{cg(x)}$ está sempre entre 0 e 1, podendo ser interpretada como uma probabilidade de aceitação. Note que, ao integrar ambos os lados da desigualdade $f(x) \leq cg(x)$, obtemos $\int f(x) dx \leq \int cg(x) dx$, ou seja, $1 \leq c$, e portanto $\frac{1}{c} \leq 1$.

O procedimento do método de rejeição-aceitação é o seguinte:

1. Gere um candidato $Y \sim g$.
2. Gere um número aleatório $U \sim \text{Uniforme}(0, 1)$, independente de Y .
3. Se $U < \frac{f(Y)}{cg(Y)}$, aceite Y como amostra e retorne $X = Y$.
4. Caso contrário, rejeite Y e retorne ao passo 1.

Para entender por que o método de rejeição-aceitação funciona, vamos construir uma intuição passo a passo com um exemplo concreto. Suponha que queremos gerar uma variável aleatória $X \sim f$, com densidade definida por

$$f(x) = 20x(1-x)^3, \quad x \in [0, 1].$$

Essa é uma densidade válida sobre o intervalo $[0, 1]$, mas sua função de distribuição acumulada $F(x)$ não possui inversa em forma fechada já que envolve resolver uma equação polinomial de grau 5, o que inviabiliza o uso direto do método da inversão. Por isso, recorremos ao método de rejeição.

Nesse caso, utilizamos como densidade auxiliar a uniforme $g(x) = 1$ sobre $[0, 1]$, que é fácil de simular. O procedimento funciona da seguinte forma:

1. Escolhemos uma constante $c > 0$ tal que $f(x) \leq cg(x)$ para todo $x \in [0, 1]$. Como $g(x) = 1$, essa condição se torna $f(x) \leq c$. Para garantir isso, basta determinar o valor máximo da função $f(x)$ no intervalo $[0, 1]$, o que pode ser feito derivando:

$$f(x) = 20x(1-x)^3 = 20x - 60x^2 + 60x^3 - 20x^4,$$

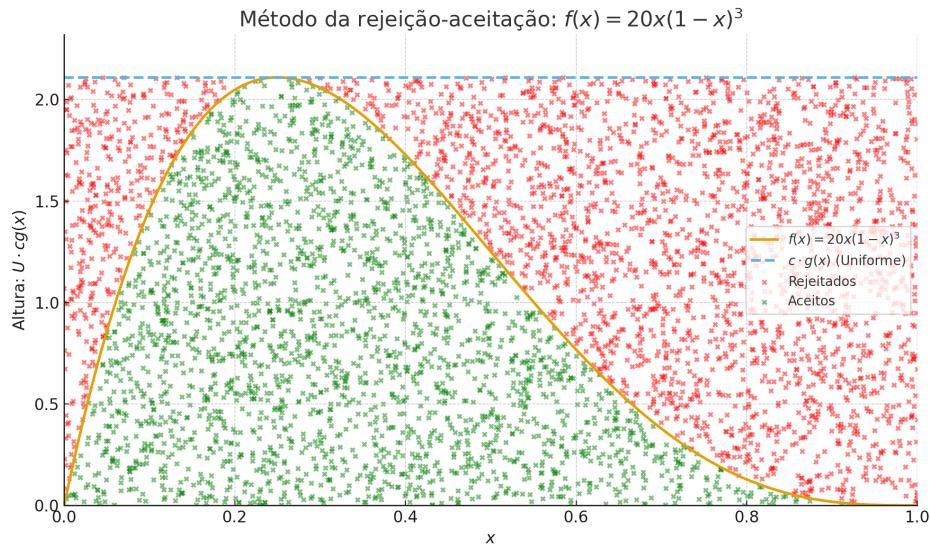
2. Geramos um candidato $Y \sim g$, ou seja, escolhemos um ponto Y aleatório uniformemente em $[0, 1]$.
3. Geramos um valor $U \sim \text{Unif}(0, 1)$, que usaremos para introduzir variabilidade vertical.
4. Calculamos a altura $U \cdot cg(Y)$. Como $g(Y) = 1$, isso equivale a $U \cdot c$, ou seja, sorteamos um ponto dentro do retângulo de altura c sobre o intervalo $[0, 1]$.

5. Comparamos essa altura com o valor da densidade $f(Y)$. Se

$$U \cdot cg(Y) < f(Y),$$

aceitamos o valor Y como amostra de X ; caso contrário, rejeitamos e repetimos o processo.

A interpretação geométrica é simples: estamos sorteando pontos aleatórios dentro do retângulo delimitado por $x \in [0, 1]$ e altura c . Esses pontos têm coordenadas $(Y, U \cdot cg(Y))$. Aceitamos apenas os que caem abaixo da curva $f(x)$. Dessa forma, os pontos aceitos se acumulam na região sob f , replicando a forma da densidade desejada.



Teorema 5. Sejam f e g funções de densidade de probabilidade com suporte em um conjunto $\mathcal{X} \subseteq \mathbb{R}$, e suponha que existe uma constante $c > 0$ tal que

$$\frac{f(x)}{g(x)} \leq c \quad \text{para todo } x \in \mathcal{X}.$$

Considere o algoritmo de geração:

1. Gere $Y \sim g$ e $U \sim \text{Uniform}(0, 1)$, independentes.
2. Retorne $X = Y$ se $U < \frac{f(Y)}{cg(Y)}$. Caso contrário, repita.

Então a variável aleatória X , definida como o primeiro valor Y aceito, possui densidade f . Além disso, o número total de iterações até a aceitação segue uma distribuição geométrica com parâmetro $1/c$.

Demonstração. Seja f a densidade-alvo da qual desejamos amostrar. Usamos uma densidade auxiliar g , com suporte que contém o de f , e uma constante $c \geq \sup_x \frac{f(x)}{g(x)}$.

Queremos mostrar que a variável X aceita tem densidade f . Para isso, analisamos sua função de distribuição acumulada $F_X(x) = \mathbb{P}(X \leq x)$. Pela definição do algoritmo, temos:

$$\mathbb{P}(X \leq x) = \mathbb{P}(Y \leq x \mid \text{aceito}) = \frac{\mathbb{P}\left(Y \leq x, U < \frac{f(Y)}{cg(Y)}\right)}{\mathbb{P}\left(U < \frac{f(Y)}{cg(Y)}\right)}.$$

Usamos a fórmula da probabilidade condicional:

$$\begin{aligned}
 \mathbb{P} \left(Y \leq x, U < \frac{f(Y)}{cg(Y)} \right) &= \int \mathbb{P} \left(Y \leq x, U < \frac{f(Y)}{cg(Y)} \mid Y = y \right) g(y) dy \\
 &= \int \mathbb{P} \left(y \leq x, U < \frac{f(y)}{cg(y)} \mid Y = y \right) g(y) dy \\
 &= \int_{-\infty}^x \mathbb{P} \left(U < \frac{f(y)}{cg(y)} \right) g(y) dy \\
 &= \int_{-\infty}^x \frac{f(y)}{cg(y)} \cdot g(y) dy \\
 &= \frac{1}{c} \int_{-\infty}^x f(y) dy.
 \end{aligned}$$

De forma análoga:

$$\begin{aligned}
 \mathbb{P} \left(U < \frac{f(Y)}{cg(Y)} \right) &= \int \mathbb{P} \left(U < \frac{f(Y)}{cg(Y)} \mid Y = y \right) g(y) dy \\
 &= \int \mathbb{P} \left(U < \frac{f(y)}{cg(y)} \mid Y = y \right) g(y) dy \\
 &= \int \mathbb{P} \left(U < \frac{f(y)}{cg(y)} \right) g(y) dy \\
 &= \int \frac{f(y)}{cg(y)} \cdot g(y) dy \\
 &= \frac{1}{c} \int f(y) dy = \frac{1}{c}.
 \end{aligned}$$

Substituindo numerador e denominador:

$$\mathbb{P}(X \leq x) = \frac{\frac{1}{c} \int_{-\infty}^x f(y) dy}{\frac{1}{c}} = \int_{-\infty}^x f(y) dy = F_X(x).$$

Portanto, $X \sim f$, como queríamos demonstrar.

Além disso, a probabilidade de aceitação em uma única tentativa é dada por:

$$\mathbb{P} \left(U < \frac{f(Y)}{cg(Y)} \right) = \frac{1}{c},$$

ou seja, cada tentativa tem probabilidade $1/c$ de ser aceita. Portanto, o número de repetições até obter um ponto aceito segue uma distribuição geométrica com parâmetro $1/c$.

□

4.2.1 Distribuição normal

A distribuição normal padrão, denotada por $\mathcal{N}(0, 1)$, é uma das distribuições mais importantes da estatística e da probabilidade. Sua densidade é dada por:

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}, \quad x \in \mathbb{R}.$$

Ela possui média $\mathbb{E}[X] = 0$ e variância $\text{Var}(X) = 1$.

Como a função de distribuição acumulada $\Phi(x) = \int_{-\infty}^x f(t) dt$ não possui inversa fechada, o método da inversão não pode ser aplicado diretamente. Em vez disso, uma abordagem alternativa é utilizar uma técnica baseada em outra distribuição mais simples, como a exponencial, combinada com o método de rejeição.

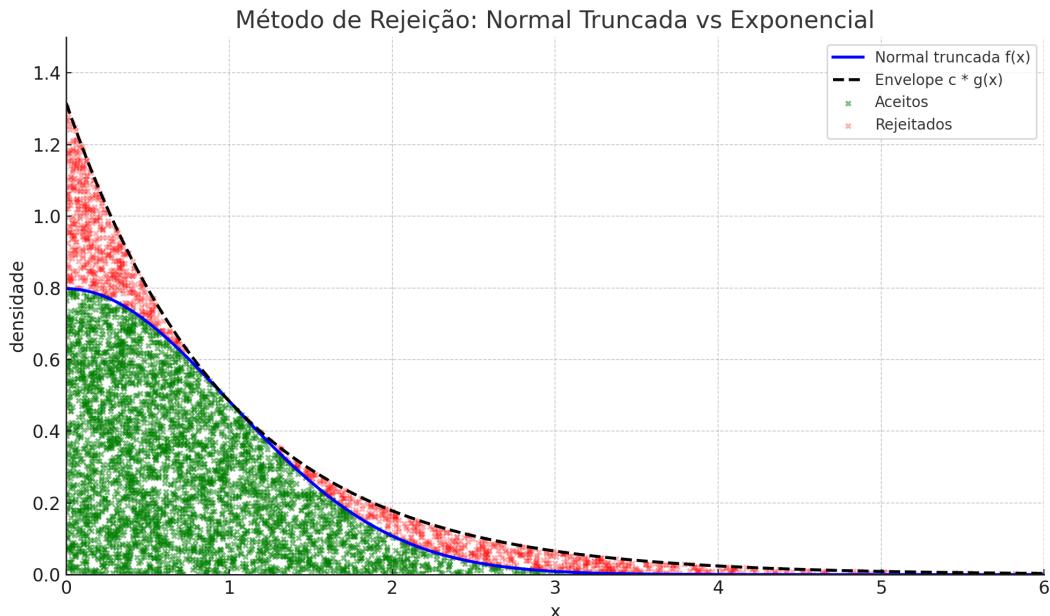
Uma forma eficiente de simular uma variável normal padrão positiva $X \sim \mathcal{N}(0, 1)$ condicionada a $X > 0$ é utilizar o método da rejeição com uma distribuição exponencial como proposta.

Sabemos que a densidade da normal padrão é

$$f(x) = \frac{2}{\sqrt{2\pi}} e^{-x^2/2},$$

e que, para $x > 0$, a função decresce com a cauda $e^{-x^2/2}$. Por outro lado, a densidade da distribuição exponencial com taxa $\lambda = 1$ é

$$g(x) = e^{-x}, \quad x \geq 0.$$



Note que:

$$\frac{f(x)}{g(x)} = \sqrt{\frac{2}{\pi}} e^{-x^2/2+x}.$$

Para aplicar o método da rejeição, precisamos encontrar o ponto de máximo da razão $f(x)/g(x)$, ou seja, maximizar a função $x - x^2/2$. Derivando:

$$\frac{d}{dx} \left(x - \frac{x^2}{2} \right) = 1 - x = 0 \Rightarrow x = 1.$$

Portanto, o máximo ocorre em $x = 1$, e o valor da constante c será:

$$c = \sqrt{\frac{2e}{\pi}}.$$

A razão $f(x)/cg(x)$ pode então ser escrita como:

$$\frac{f(x)}{cg(x)} = \exp\left(-\frac{1}{2}(x-1)^2\right),$$

o que nos leva ao seguinte algoritmo:

1. Gere $Y \sim \text{Exp}(1)$.
2. Gere $U \sim \text{Uniforme}(0, 1)$.
3. Se $U < \exp\left(-\frac{(Y-1)^2}{2}\right)$, aceite Y como amostra da normal positiva.
4. Caso contrário, volte ao passo 1.

Finalmente, se $X \sim |\mathcal{N}(0, 1)|$, isto é, uma normal padrão truncada para valores positivos (obtida via o algoritmo anterior).

Para gerar uma normal padrão simétrica $Z \sim \mathcal{N}(0, 1)$, basta sortear um sinal $S \sim \text{Bernoulli}(1/2)$, e definir:

$$Z = \begin{cases} X, & \text{se } S = 1, \\ -X, & \text{se } S = 0. \end{cases}$$

Dessa forma, Z tem distribuição simétrica em torno de zero, com densidade normal padrão, como desejado.

Intuição geométrica

O método da rejeição pode ser visualizado como um processo de *amostragem de pontos aleatórios em uma região do plano*, com o objetivo de “pintar” a curva da densidade alvo $f(x)$.



Imagine que temos uma função auxiliar $g(x)$, da qual sabemos simular facilmente, e uma constante de majoração $c > 0$ tal que $f(x) \leq cg(x)$ para todo x . Isso nos permite usar $cg(x)$ como um *envelope* que cobre toda a curva de $f(x)$.

A cada tentativa, sorteamos:

- Um valor $Y \sim g(x)$: isso escolhe uma posição no eixo x , com densidade g ;

- Um valor $U \sim \text{Unif}(0, 1)$: isso define uma altura relativa no intervalo $[0, cg(Y)]$, formando o ponto $(Y, U \cdot cg(Y))$ dentro do retângulo sob o envelope.

O ponto é *aceito* se estiver abaixo da curva de f , ou seja, se

$$U < \frac{f(Y)}{cg(Y)}.$$

Caso contrário, o ponto é rejeitado.

Assim, ao longo do tempo, os pontos aceitos se acumulam nas regiões onde $f(x)$ é maior, formando uma amostra com exatamente a distribuição desejada.

Perceba que a constante c controla a *eficiência* do método: quanto maior c , maior a área total do envelope $cg(x)$ em relação à curva alvo $f(x)$, e mais pontos são desperdiçados. O valor ideal de c é o menor possível que ainda garanta $f(x) \leq cg(x)$ para todo x ; nesse caso, a taxa de aceitação é maximizada e igual a $1/c$.

4.3 Distribuição Gamma

A distribuição Gamma com parâmetros $\alpha > 0$ (forma) e $\theta > 0$ (escala) é definida pela densidade

$$f(x) = \frac{1}{\Gamma(\alpha) \theta^\alpha} x^{\alpha-1} e^{-x/\theta}, \quad x > 0,$$

onde $\Gamma(\alpha)$ é a função Gama de Euler.

Uma maneira natural de interpretar essa distribuição é por analogia com modelos discretos. No mundo discreto, a distribuição Geométrica mede o número de ensaios necessários até observar o primeiro sucesso em uma sequência de Bernoullis. Se quisermos o número de ensaios até o r -ésimo sucesso, obtemos a Binomial Negativa, que pode ser vista como a soma de variáveis Geométricas independentes.

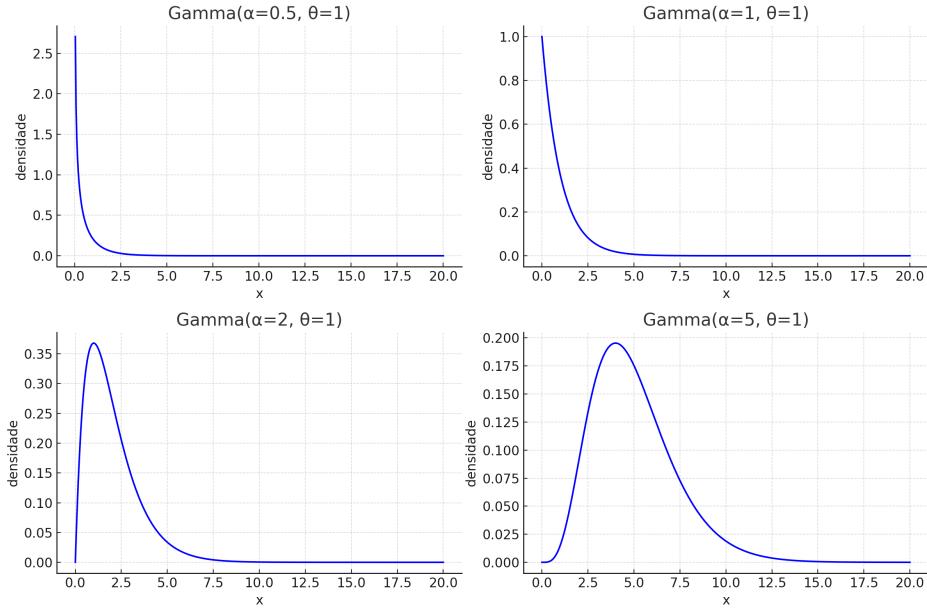
No mundo contínuo, a Exponencial exerce um papel análogo ao da Geométrica: ela mede o tempo de espera até o primeiro sucesso. Seguindo a mesma lógica, a distribuição Gamma surge como soma de várias variáveis Exponenciais independentes, representando o tempo de espera até o α -ésimo sucesso.

Alguns casos particulares:

- Quando $\alpha = 1$, a Gamma coincide exatamente com a Exponencial.
- Quando α é um inteiro maior que 1, a Gamma pode ser entendida como a soma de α Exponenciais independentes.
- Mesmo para α não inteiro, a distribuição Gamma mantém a interpretação de “tempo de espera até sucessos acumulados”, generalizando de forma contínua a Binomial Negativa.

O parâmetro de forma α regula quantos sucessos estão sendo acumulados e, portanto, afeta diretamente a forma da distribuição:

- Para $\alpha < 1$, a densidade concentra-se fortemente perto de zero.



- Para $\alpha = 1$, a densidade é simplesmente a Exponencial decrescente.
- Para $\alpha > 1$, a densidade é unimodal, com máximo em $(\alpha - 1)\theta$.

Já o parâmetro de escala θ atua como fator multiplicativo, alongando ou comprimindo a distribuição. A média e a variância crescem proporcionalmente a ele, conforme:

$$\mathbb{E}[X] = \alpha\theta, \quad \text{Var}(X) = \alpha\theta^2.$$

4.3.1 Simulando quando α é inteiro

Quando o parâmetro de forma é inteiro, $\alpha = k \in \mathbb{N}$, a distribuição $\text{Gamma}(k, \theta)$ (escala $\theta > 0$) recebe o nome de *Erlang*. Ela pode ser obtida como a soma de k variáveis exponenciais independentes. Na parametrização por taxa $\lambda = 1/\theta$:

$$X \sim \text{Gamma}(k, \lambda) \iff X \stackrel{d}{=} \sum_{i=1}^k E_i, \quad E_i \stackrel{\text{i.i.d.}}{\sim} \text{Exp}(\lambda).$$

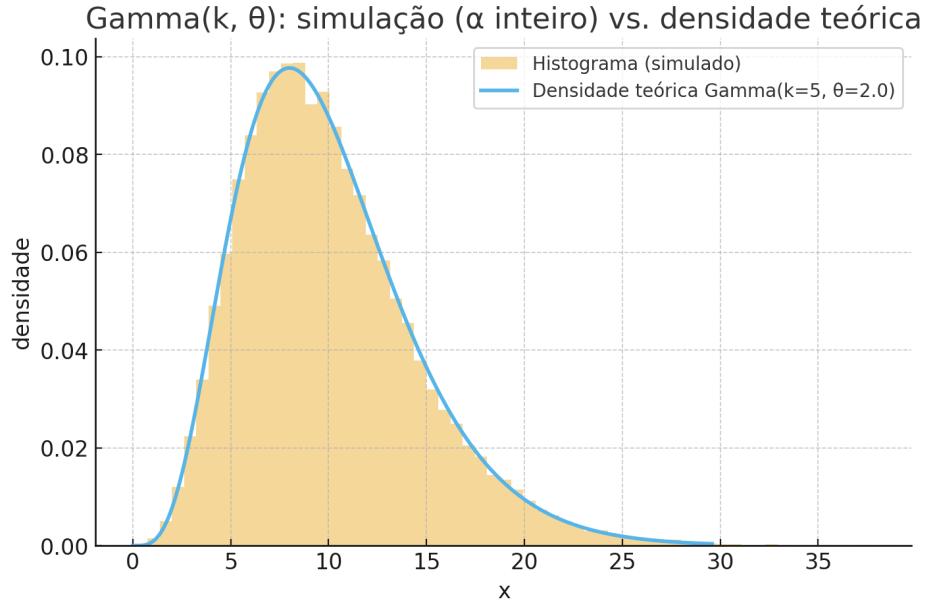
O algoritmo de simulação é o seguinte:

1. Fixe $k \in \mathbb{N}$ e $\lambda > 0$ (ou $\theta = 1/\lambda$).
2. Gere $E_i \sim \text{Exp}(\lambda)$ de forma independente, para $i = 1, \dots, k$.
3. Calcule $X = \sum_{i=1}^k E_i$. O resultado segue $X \sim \text{Gamma}(k, \lambda)$.

4.3.2 Simulando quando $\alpha > 1$ via aceitação–rejeição com Exponencial

Considere $X \sim \text{Gamma}(\alpha, \lambda)$ com $\alpha > 1$ (parametrização por taxa λ). A densidade alvo é

$$f(x) = \frac{\lambda^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\lambda x}, \quad x > 0.$$



Usaremos como proposta $Y \sim \text{Exp}(\mu)$, com densidade

$$g(x) = \mu e^{-\mu x}, \quad x > 0.$$

Para que o método de aceitação-rejeição seja válido, precisamos de uma constante c tal que $f(x) \leq c g(x)$ para todo $x > 0$. O quociente

$$\frac{f(x)}{g(x)} = \frac{\lambda^\alpha}{\Gamma(\alpha) \mu} x^{\alpha-1} e^{-(\lambda-\mu)x}$$

mostra que é necessário ter $\mu < \lambda$, pois caso contrário o termo exponencial não decai e o quociente não tem máximo finito. Quando $\mu < \lambda$, o máximo ocorre em

$$x^* = \frac{\alpha - 1}{\lambda - \mu},$$

com valor

$$c(\mu) = \frac{\lambda^\alpha}{\Gamma(\alpha) \mu} \left(\frac{\alpha - 1}{\lambda - \mu} \right)^{\alpha-1} e^{-(\alpha-1)}.$$

A probabilidade de aceitação por tentativa é $1/c(\mu)$ e, portanto, o número médio de tentativas até aceitar uma amostra é $c(\mu)$. Interpretando em tempo contínuo como um processo de Poisson de taxa 1, o thinning com probabilidade $1/c(\mu)$ gera um processo aceito com taxa $1/c(\mu)$, de modo que o tempo médio entre aceitações é $c(\mu)$.

O algoritmo é o seguinte:

1. Escolha $\mu \in (0, \lambda)$, por exemplo $\mu = \mu^* = \lambda/\alpha$ que minimiza $c(\mu)$.
2. Gere $Y \sim \text{Exp}(\mu)$ e $U \sim \text{Unif}(0, 1)$.
3. Aceite $X = Y$ se $U \leq f(Y)/(c(\mu) g(Y))$; caso contrário, volte ao passo 2.

O valor aceito X tem distribuição $\text{Gamma}(\alpha, \lambda)$.

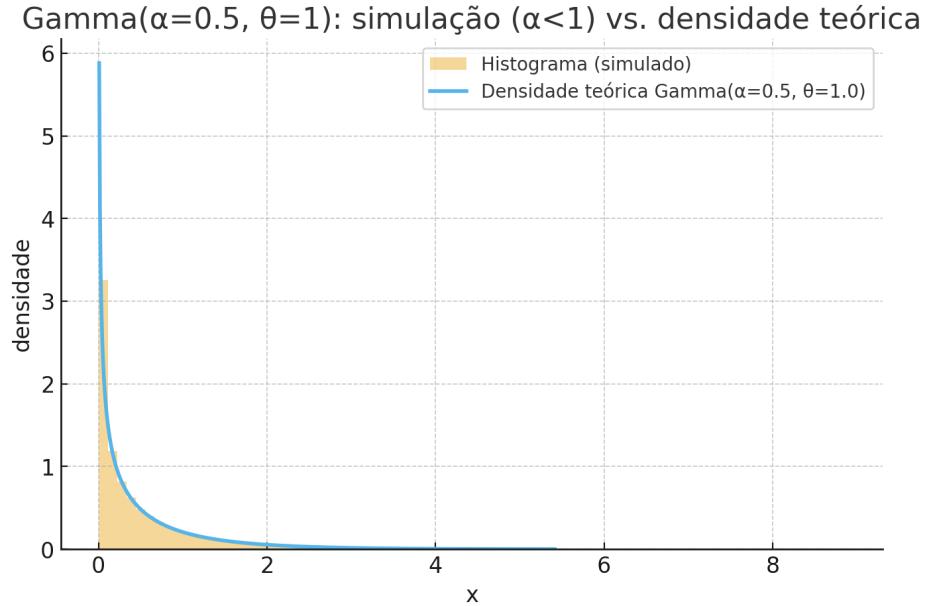
A tabela a seguir mostra a constante c^* e a taxa de aceitação $1/c^*$ para alguns valores de α na escolha ótima $\mu = \lambda/\alpha$ (os valores independem de λ):

α	$\mu^* = \lambda/\alpha$	$c^* = \frac{\alpha^\alpha}{\Gamma(\alpha)} e^{-(\alpha-1)}$	aceitação ($1/c^*$)
1.5	$\frac{2}{3}\lambda$	1.2573	0.7953
2	$\frac{1}{2}\lambda$	1.4715	0.6796
3	$\frac{1}{3}\lambda$	1.8270	0.5473
5	$\frac{1}{5}\lambda$	2.3848	0.4193
8	$\frac{1}{8}\lambda$	3.0355	0.3294
12	$\frac{1}{12}\lambda$	3.7306	0.2681

4.3.3 Simulando quando $\alpha < 1$

Para $0 < \alpha < 1$, uma forma simples de simular $\Gamma(\alpha, \theta)$ é usar a identidade

$$\text{se } G \sim \Gamma(\alpha + 1, \theta) \text{ e } U \sim \text{Unif}(0, 1) \text{ (indep.)}, \quad X = G U^{1/\alpha} \sim \Gamma(\alpha, \theta).$$



Para entender essa relação, considere as variáveis independentes (U, G) com $U \sim \text{Unif}(0, 1)$ e $G \sim \Gamma(\alpha + 1, \theta)$, $0 < \alpha < 1$. Defina a transformação

$$(x, g) = T(u, g) = (g u^{1/\alpha}, g),$$

cuja inversa é

$$(u, g) = T^{-1}(x, g) = ((x/g)^\alpha, g).$$

O suporte transformado é $x > 0$ e $g > x$ (pois $u \in (0, 1)$ implica $x/g \in (0, 1)$).

A densidade conjunta de (U, G) é

$$f_{U,G}(u, g) = f_U(u) f_G(g) = \mathbf{1}_{(0,1)}(u) \frac{g^\alpha e^{-g/\theta}}{\Gamma(\alpha + 1) \theta^{\alpha+1}}, \quad g > 0.$$

Pela fórmula de mudança de variável,

$$f_{X,G}(x,g) = f_{U,G}((x/g)^\alpha, g) \left| \det \frac{\partial(u,g)}{\partial(x,g)} \right|.$$

Calculamos o jacobiano usando a inversa $u = (x/g)^\alpha$:

$$\frac{\partial u}{\partial x} = \alpha x^{\alpha-1} g^{-\alpha}, \quad \frac{\partial u}{\partial g} = -\alpha x^\alpha g^{-(\alpha+1)}, \quad \frac{\partial g}{\partial x} = 0, \quad \frac{\partial g}{\partial g} = 1,$$

logo

$$\left| \det \frac{\partial(u,g)}{\partial(x,g)} \right| = \left| \frac{\partial u}{\partial x} \cdot 1 - 0 \right| = \alpha x^{\alpha-1} g^{-\alpha}.$$

Portanto,

$$f_{X,G}(x,g) = \mathbf{1}_{(x,\infty)}(g) \frac{g^\alpha e^{-g/\theta}}{\Gamma(\alpha+1) \theta^{\alpha+1}} \alpha x^{\alpha-1} g^{-\alpha} = \mathbf{1}_{(x,\infty)}(g) \frac{\alpha x^{\alpha-1}}{\Gamma(\alpha+1) \theta^{\alpha+1}} e^{-g/\theta}.$$

Integrando em g para obter a marginal de X :

$$f_X(x) = \int_x^\infty f_{X,G}(x,g) dg = \frac{\alpha x^{\alpha-1}}{\Gamma(\alpha+1) \theta^{\alpha+1}} \int_x^\infty e^{-g/\theta} dg = \frac{\alpha x^{\alpha-1}}{\Gamma(\alpha+1) \theta^{\alpha+1}} \theta e^{-x/\theta}.$$

Usando $\Gamma(\alpha+1) = \alpha \Gamma(\alpha)$, obtemos

$$f_X(x) = \frac{x^{\alpha-1} e^{-x/\theta}}{\Gamma(\alpha) \theta^\alpha}, \quad x > 0,$$

que é exatamente a densidade $\Gamma(\alpha, \theta)$. Logo, $X = G U^{1/\alpha} \sim \Gamma(\alpha, \theta)$.

O algoritmo de simulação é:

1. Dado $0 < \alpha < 1$ e $\theta > 0$, defina $\alpha' = \alpha + 1$.
2. Gere $G \sim \Gamma(\alpha', \theta)$ (por exemplo, via Marsaglia-Tsang, pois $\alpha' > 1$).
3. Gere $U \sim \text{Unif}(0, 1)$, independente de G .
4. Retorne $X = G U^{1/\alpha}$. Então $X \sim \Gamma(\alpha, \theta)$.

4.4 Distribuição Beta

A distribuição Beta é uma das mais importantes distribuições contínuas em estatística, definida no intervalo unitário $[0, 1]$ e parametrizada por dois parâmetros de forma $\alpha > 0$ e $\beta > 0$. Sua densidade é dada por

$$f(x) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1} (1-x)^{\beta-1}, \quad 0 < x < 1,$$

onde

$$B(\alpha, \beta) = \int_0^1 u^{\alpha-1} (1-u)^{\beta-1} du = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

é a função Beta de Euler.

A interpretação intuitiva da distribuição Beta é como um modelo de incerteza sobre probabilidades. Se pensamos em x como a probabilidade de sucesso em uma sequência de ensaios de Bernoulli, a Beta aparece naturalmente como distribuição a posteriori em modelos Bayesianos conjugados: começando com uma priori Beta(α, β), após observar s sucessos e f fracassos, a posteriori é Beta($\alpha + s, \beta + f$).

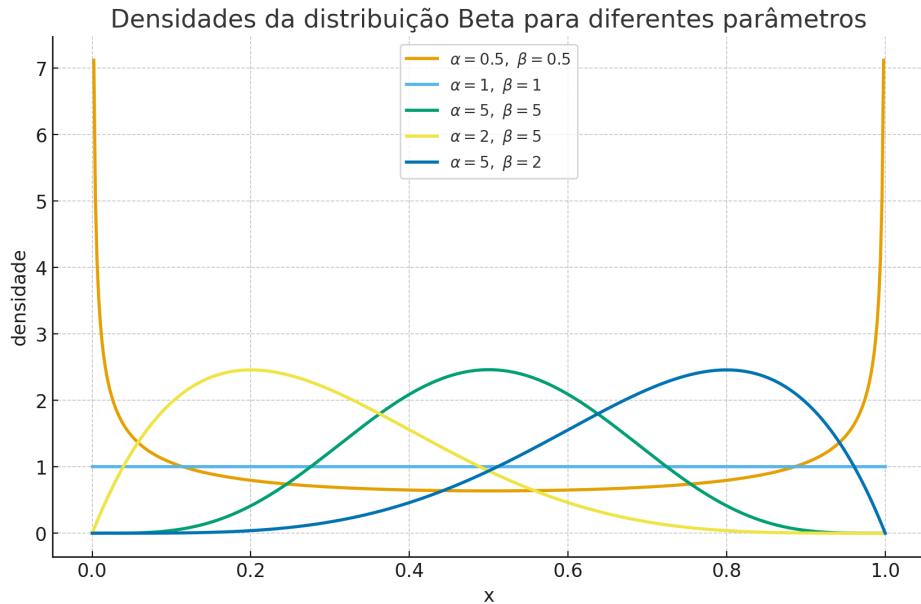
A forma da densidade é bastante flexível:

- Para $\alpha, \beta < 1$, a densidade concentra-se nos extremos 0 e 1.
- Para $\alpha = \beta = 1$, temos a uniforme no intervalo (0, 1).
- Para $\alpha > 1$ e $\beta > 1$, a densidade é unimodal, com máximo em $(\alpha - 1)/(\alpha + \beta - 2)$.

Os momentos principais são:

$$\mathbb{E}[X] = \frac{\alpha}{\alpha + \beta}, \quad \text{Var}(X) = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)}.$$

Essas fórmulas mostram como α e β podem ser interpretados como “pseudocontagens” de sucessos e fracassos, de forma que $\alpha + \beta$ controla a concentração da distribuição em torno da média.



4.4.1 Simulando a Beta via aceitação–rejeição com proposta uniforme

A distribuição Beta(α, β) tem suporte em (0, 1), de modo que uma escolha natural de proposta é $Y \sim \text{Unif}(0, 1)$. A densidade da uniforme é $g(y) = 1$ para $0 < y < 1$, e precisamos de uma constante c tal que

$$f(y) \leq c g(y) = c, \quad 0 < y < 1.$$

O algoritmo de aceitação–rejeição é:

1. Gere $Y \sim \text{Unif}(0, 1)$ e $U \sim \text{Unif}(0, 1)$ independentes.

2. Aceite $X = Y$ se $U \leq f(Y)/c$, caso contrário repita o passo 1.

O valor aceito X terá distribuição Beta(α, β).

A escolha ótima de c pode ser caracterizado em três casos:

- Se $\alpha > 1$ e $\beta > 1$, a densidade é unimodal, com modo em

$$y^* = \frac{\alpha - 1}{\alpha + \beta - 2},$$

e portanto

$$c = f(y^*) = \frac{1}{B(\alpha, \beta)} (y^*)^{\alpha-1} (1 - y^*)^{\beta-1}.$$

- Se $\alpha = \beta = 1$, a distribuição é uniforme, logo $c = 1$.
- Se $\alpha < 1$ ou $\beta < 1$, a densidade diverge em uma das extremidades (0 ou 1), e assim $\sup f(y) = \infty$. Nesse caso não existe constante finita c e o método de aceitação–rejeição com uniforme como proposta não pode ser aplicado.

4.4.2 Simulando a Beta via Gammas independentes

O método mais utilizado e geral para simular variáveis Beta(α, β) explora a relação entre as distribuições Beta e Gama. Seja

$$G_1 \sim \Gamma(\alpha, 1), \quad G_2 \sim \Gamma(\beta, 1),$$

independentes. Então vale a identidade

$$X = \frac{G_1}{G_1 + G_2} \sim \text{Beta}(\alpha, \beta).$$

A prova segue do fato de que o vetor normalizado $(G_1, G_2)/(G_1 + G_2)$ tem distribuição Dirichlet(α, β), e portanto sua primeira coordenada é Beta(α, β). Outra forma é calcular a densidade conjunta de (X, T) , com $X = \frac{G_1}{G_1 + G_2}$ e $T = G_1 + G_2$, e verificar que a marginal de X coincide com a densidade da Beta.

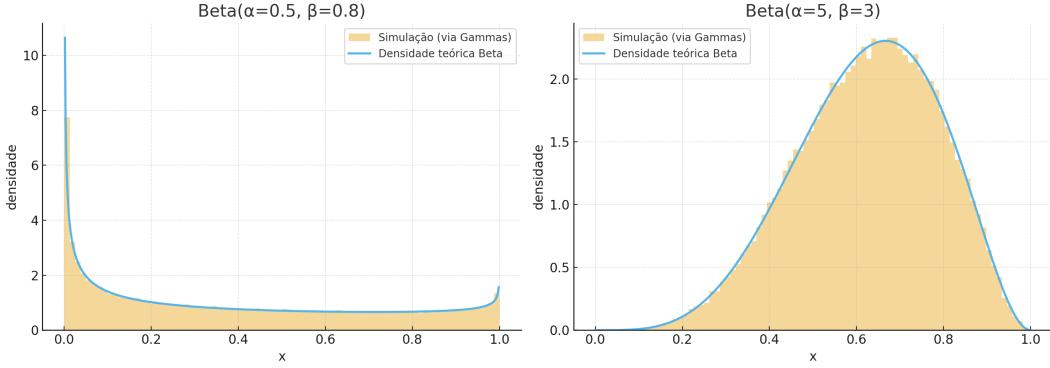
O algoritmo é simples e eficiente:

1. Gere $G_1 \sim \Gamma(\alpha, 1)$ e $G_2 \sim \Gamma(\beta, 1)$ de forma independente.
2. Retorne $X = \frac{G_1}{G_1 + G_2}$.

Esse procedimento funciona para qualquer $\alpha, \beta > 0$, inclusive quando são menores que 1, ao contrário do método de aceitação–rejeição com proposta uniforme.

4.5 Transformações de Variáveis Aleatórias

Neste capítulo estudaremos transformações de variáveis e vetores aleatórios. A ideia central é a seguinte: dado um modelo probabilístico inicial, frequentemente precisamos aplicar funções a variáveis ou vetores aleatórios para obter novas quantidades de interesse. O objetivo, então, é caracterizar a distribuição resultante após a transformação, seja ela univariada ou multivariada.



4.5.1 Geração de Normais via Método de Box–Muller

Seja $U \sim \text{Unif}(0, 2\pi)$ e $T \sim \text{Expo}(1)$ independentes, com densidade conjunta

$$f_{U,T}(u, t) = \frac{1}{2\pi} e^{-t}, \quad u \in (0, 2\pi), t > 0.$$

Definimos a transformação

$$X = \sqrt{2T} \cos U, \quad Y = \sqrt{2T} \sin U.$$

O objetivo é determinar a densidade conjunta de (X, Y) . Para isso usamos a fórmula de mudança de variáveis

$$f_{X,Y}(x, y) = f_{U,T}(u, t) \left| \det \frac{\partial(u, t)}{\partial(x, y)} \right|,$$

onde (u, t) é obtido a partir de (x, y) .

Primeiro observamos que

$$x^2 + y^2 = 2t(\cos^2 u + \sin^2 u) = 2t,$$

de modo que

$$t = \frac{1}{2}(x^2 + y^2), \quad u = \arctan\left(\frac{y}{x}\right) \quad (\text{ajustado para o quadrante correto}).$$

Assim, a transformação é invertível.

O próximo passo é calcular o jacobiano da transformação direta $(u, t) \mapsto (x, y)$. Temos

$$\begin{aligned} \frac{\partial x}{\partial u} &= -\sqrt{2t} \sin u, & \frac{\partial x}{\partial t} &= \frac{1}{\sqrt{2t}} \cos u, \\ \frac{\partial y}{\partial u} &= \sqrt{2t} \cos u, & \frac{\partial y}{\partial t} &= \frac{1}{\sqrt{2t}} \sin u. \end{aligned}$$

Logo,

$$J = \begin{pmatrix} -\sqrt{2t} \sin u & \frac{1}{\sqrt{2t}} \cos u \\ \sqrt{2t} \cos u & \frac{1}{\sqrt{2t}} \sin u \end{pmatrix}.$$

O determinante é

$$\det(J) = \left(-\sqrt{2t} \sin u \right) \left(\frac{1}{\sqrt{2t}} \sin u \right) - \left(\frac{1}{\sqrt{2t}} \cos u \right) \left(\sqrt{2t} \cos u \right).$$

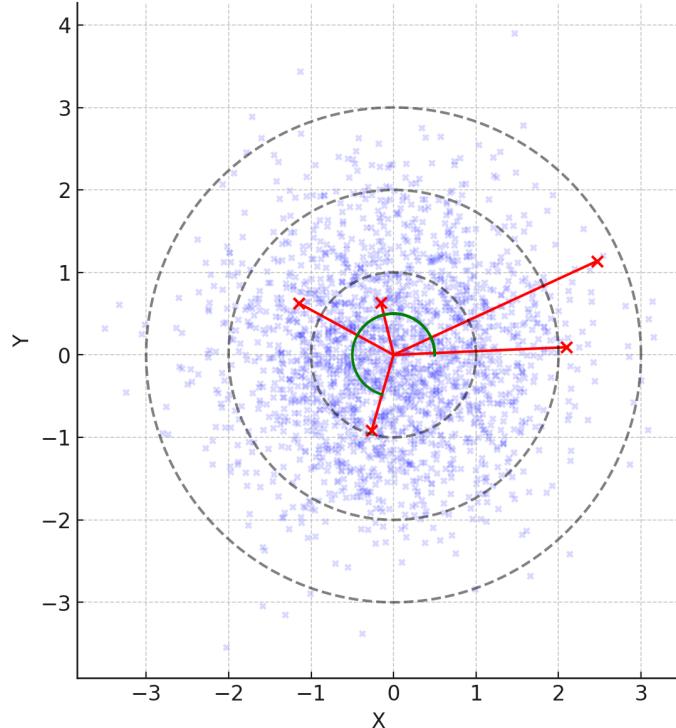
Simplificando,

$$\det(J) = -\sin^2 u - \cos^2 u = -1.$$

Portanto,

$$|\det(J)| = 1.$$

Box-Muller: representação em coordenadas polares



Aplicando a fórmula de mudança de variáveis,

$$f_{X,Y}(x,y) = f_{U,T}(u,t) \left| \det \frac{\partial(u,t)}{\partial(x,y)} \right| = \frac{1}{2\pi} e^{-t} \cdot 1.$$

Substituindo $t = \frac{1}{2}(x^2 + y^2)$,

$$f_{X,Y}(x,y) = \frac{1}{2\pi} \exp(-\frac{1}{2}(x^2 + y^2)).$$

Finalmente, notamos que

$$f_{X,Y}(x,y) = \left(\frac{1}{\sqrt{2\pi}} e^{-x^2/2} \right) \left(\frac{1}{\sqrt{2\pi}} e^{-y^2/2} \right),$$

o que mostra que X e Y são independentes e ambos têm distribuição Normal padrão.

Com essa dedução, concluímos que (X, Y) definidos acima são variáveis independentes com distribuição Normal padrão. Assim, o método de Box-Muller pode ser usado diretamente para gerar Normais a partir de variáveis Uniformes e Exponenciais. Na prática, o algoritmo segue os seguintes passos:

1. Gere $U \sim \text{Unif}(0, 2\pi)$.

2. Gere $T \sim \text{Expo}(1)$.
3. Calcule $X = \sqrt{2T} \cos U$ e $Y = \sqrt{2T} \sin U$.
4. Então X e Y são independentes e possuem distribuição $\mathcal{N}(0, 1)$.

Intuitivamente, o que estamos fazendo é gerar um par de variáveis Normais independentes (X, Y) e representá-las em coordenadas polares: o ângulo é sorteado uniformemente e o raio vem de uma distribuição que garante a forma circular da densidade Normal.

4.5.2 Geração da normal bivariada

Nosso objetivo agora é mostrar como simular uma Normal bivariada (Z, W) com marginais $\mathcal{N}(0, 1)$ e correlação ρ , onde $-1 < \rho < 1$. A ideia é construir (Z, W) a partir de variáveis independentes mais simples.

Sejam $X, Y \sim \mathcal{N}(0, 1)$ independentes. Definimos

$$Z = X, \quad W = \rho X + \tau Y, \quad \tau = \sqrt{1 - \rho^2}.$$

Então Z e W têm marginais $\mathcal{N}(0, 1)$ e $\text{Corr}(Z, W) = \rho$. Este é um procedimento construtivo que permite gerar diretamente a Normal bivariada a partir de duas Normais independentes.

Para verificar a validade da construção, vamos obter a densidade conjunta de (Z, W) . A densidade de (X, Y) é

$$f_{X,Y}(x, y) = \frac{1}{2\pi} \exp\left(-\frac{1}{2}(x^2 + y^2)\right).$$

Como a transformação é

$$z = x, \quad w = \rho x + \tau y,$$

a inversa é

$$x = z, \quad y = \frac{w - \rho z}{\tau}.$$

Aplicando a fórmula de mudança de variáveis,

$$f_{Z,W}(z, w) = f_{X,Y}(x, y) \left| \det \frac{\partial(x, y)}{\partial(z, w)} \right|,$$

com (x, y) dados pela inversa acima.

O jacobiano da transformação inversa é

$$\frac{\partial(x, y)}{\partial(z, w)} = \begin{pmatrix} \frac{\partial x}{\partial z} & \frac{\partial x}{\partial w} \\ \frac{\partial y}{\partial z} & \frac{\partial y}{\partial w} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -\frac{\rho}{\tau} & \frac{1}{\tau} \end{pmatrix},$$

portanto

$$\left| \det \frac{\partial(x, y)}{\partial(z, w)} \right| = \frac{1}{\tau}.$$

Substituindo em $f_{Z,W}$,

$$f_{Z,W}(z, w) = \frac{1}{2\pi\tau} \exp\left(-\frac{1}{2}\left(z^2 + \left(\frac{w - \rho z}{\tau}\right)^2\right)\right).$$

Fazendo as contas e lembrando que $\rho^2 + \tau^2 = 1$, obtemos

$$f_{Z,W}(z,w) = \frac{1}{2\pi\tau} \exp\left(-\frac{1}{2\tau^2}(z^2 - 2\rho zw + w^2)\right).$$

Essa é exatamente a forma conhecida da densidade Normal bivariada com matriz de covariância

$$\Sigma = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}, \quad |\Sigma| = 1 - \rho^2 = \tau^2.$$

De fato, podemos escrever

$$f_{Z,W}(z,w) = \frac{1}{2\pi|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(z,w) \Sigma^{-1} (z,w)^\top\right).$$

Do ponto de vista de simulação, esse resultado mostra que basta gerar $X, Y \sim \mathcal{N}(0,1)$ independentes (e.g. via Box–Muller) e aplicar a transformação acima. O algoritmo é:

1. Gere $X \sim \mathcal{N}(0,1)$.
2. Gere $Y \sim \mathcal{N}(0,1)$ independentemente.
3. Calcule

$$Z = X, \quad W = \rho X + \sqrt{1 - \rho^2} Y.$$

4. O par (Z, W) tem distribuição Normal bivariada com matriz de covariância Σ .

Generalizando para normal multivariada

No caso bivariado, construímos

$$\begin{pmatrix} Z \\ W \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \rho & \sqrt{1 - \rho^2} \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix}, \quad X, Y \sim \mathcal{N}(0,1) \text{ independentes.}$$

Chamando a matriz de transformação de A , temos

$$\begin{pmatrix} Z \\ W \end{pmatrix} = A \begin{pmatrix} X \\ Y \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 0 \\ \rho & \sqrt{1 - \rho^2} \end{pmatrix}.$$

Como X, Y são independentes com variância 1, temos

$$\text{Cov}\left(\begin{pmatrix} Z \\ W \end{pmatrix}\right) = A \text{Cov}\left(\begin{pmatrix} X \\ Y \end{pmatrix}\right) A^\top = A I_2 A^\top = A A^\top.$$

Fazendo as contas,

$$A A^\top = \begin{pmatrix} 1 & 0 \\ \rho & \sqrt{1 - \rho^2} \end{pmatrix} \begin{pmatrix} 1 & \rho \\ 0 & \sqrt{1 - \rho^2} \end{pmatrix} = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix},$$

que é exatamente a matriz de covariância da Normal bivariada desejada.

No caso geral em d dimensões, seguimos a mesma ideia. Seja

$$X = (X_1, \dots, X_d)^\top \sim \mathcal{N}_d(0, I_d),$$

um vetor de Normais independentes. Para qualquer matriz $A \in \mathbb{R}^{d \times d}$, definimos

$$Z = AX.$$

Então

$$\text{Cov}(Z) = A \text{Cov}(X) A^\top = A I_d A^\top = AA^\top.$$

Portanto, dado Σ simétrica definida positiva, basta encontrar A tal que $\Sigma = AA^\top$. Assim, podemos gerar

$$Z \sim \mathcal{N}_d(0, \Sigma).$$

Se quisermos uma média não nula $\mu \in \mathbb{R}^d$, basta considerar

$$Z = \mu + AX.$$

Para encontrar A a partir de uma matriz de covariância Σ simétrica definida positiva, existem diferentes decomposições possíveis:

- **Decomposição de Cholesky:** escreve-se

$$\Sigma = LL^\top,$$

onde L é triangular inferior. Neste caso, podemos tomar $A = L$.

- **Decomposição espectral:** escreve-se

$$\Sigma = Q\Lambda Q^\top,$$

onde Q é ortogonal e $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_d)$ contém os autovalores de Σ . Como Σ é definida positiva, $\lambda_i > 0$. Assim, podemos tomar

$$A = Q\Lambda^{1/2},$$

com

$$\Lambda^{1/2} = \text{diag}(\sqrt{\lambda_1}, \dots, \sqrt{\lambda_d}).$$

- **Outros métodos:** em aplicações numéricas, também se pode usar decomposições de tipo QR ou fatorações aproximadas, dependendo da estabilidade computacional desejada.

Exemplo 8 (Simulação de Normal trivariada). *Considere o vetor aleatório $(Z_1, Z_2, Z_3)^\top \sim \mathcal{N}_3(0, \Sigma)$ com média nula e matriz de covariância*

$$\Sigma = \begin{pmatrix} 1 & 0.8 & 0.3 \\ 0.8 & 1 & 0.5 \\ 0.3 & 0.5 & 1 \end{pmatrix}.$$

Para gerar amostras desse vetor, seguimos a ideia de representar uma Normal multivariada como transformação linear de Normais independentes. Seja

$$\mathbf{X} = (X_1, X_2, X_3)^\top \sim \mathcal{N}_3(0, I_3),$$

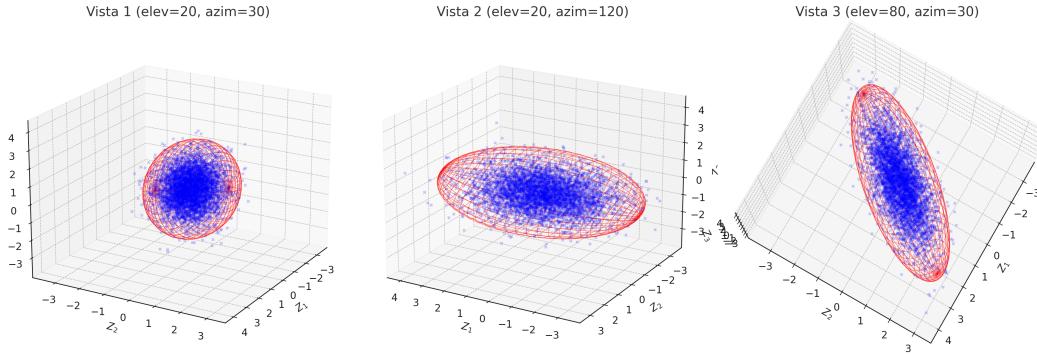
isto é, X_1, X_2, X_3 independentes $\mathcal{N}(0, 1)$. Se encontrarmos uma matriz L tal que

$$\Sigma = LL^\top,$$

então

$$\mathbf{Z} = L\mathbf{X}$$

terá distribuição $\mathcal{N}_3(0, \Sigma)$.



No caso desta matriz, a decomposição de Cholesky fornece

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 0.8 & 0.6 & 0 \\ 0.3 & 0.566 & 0.768 \end{pmatrix},$$

de modo que, se gerarmos $X_1, X_2, X_3 \sim \mathcal{N}(0, 1)$ independentes e calcularmos

$$\begin{pmatrix} Z_1 \\ Z_2 \\ Z_3 \end{pmatrix} = L \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix},$$

obtemos um vetor Normal trivariado com a covariância desejada.

4.5.3 Distribuição Qui-quadrado

A distribuição qui-quadrado surge naturalmente como uma transformação de variáveis normais independentes.

Sejam $Z_1, \dots, Z_k \stackrel{\text{i.i.d.}}{\sim} N(0, 1)$. Definimos

$$Q = \sum_{i=1}^k Z_i^2.$$

Dizemos que Q segue a distribuição qui-quadrado com k graus de liberdade, denotada por

$$Q \sim \chi_k^2.$$

A densidade dessa distribuição pode ser derivada observando que $Z_i^2 \sim \Gamma(\frac{1}{2}, 2)$, e que a soma de variáveis Gama independentes com mesma escala também é Gama. Assim,

$$Q \sim \Gamma\left(\frac{k}{2}, 2\right).$$

Logo, a densidade é

$$f(q) = \frac{1}{2^{k/2} \Gamma(k/2)} q^{\frac{k}{2}-1} e^{-q/2}, \quad q > 0.$$

Os principais momentos são

$$\mathbb{E}[Q] = k, \quad \text{Var}(Q) = 2k.$$

Essa distribuição aparece com frequência em estatística, por exemplo em testes de hipóteses e intervalos de confiança, pois estatísticas do tipo “soma de quadrados de erros padronizados” têm exatamente essa forma.

Simulando a qui-quadrado via soma de Normais

A definição da distribuição qui-quadrado já fornece um método direto de simulação. Seja $k \in \mathbb{N}$ o número de graus de liberdade. Gere variáveis independentes

$$Z_1, Z_2, \dots, Z_k \stackrel{\text{i.i.d.}}{\sim} N(0, 1).$$

Então

$$Q = \sum_{i=1}^k Z_i^2 \sim \chi_k^2.$$

O algoritmo é:

1. Fixe $k \in \mathbb{N}$.
2. Gere $Z_1, \dots, Z_k \stackrel{\text{i.i.d.}}{\sim} N(0, 1)$.
3. Retorne $Q = \sum_{i=1}^k Z_i^2$.

Esse procedimento é simples e mostra claramente a origem da distribuição qui-quadrado como soma de quadrados de Normais padrão. Além disso, destaca a ligação entre a χ_k^2 e a distribuição Normal, que será essencial para a construção de outras distribuições clássicas, como a t de Student e a F de Fisher.

Simulando a qui-quadrado via Gamma

Outra forma de simular uma variável χ_k^2 é usar sua equivalência com a distribuição Gama. Sabemos que

$$Q \sim \chi_k^2 \iff Q \sim \Gamma\left(\frac{k}{2}, 2\right),$$

isto é, uma Gama com parâmetro de forma $k/2$ e escala 2.

Portanto, para simular $Q \sim \chi_k^2$ podemos simplesmente gerar

$$Q \sim \Gamma\left(\frac{k}{2}, 2\right).$$

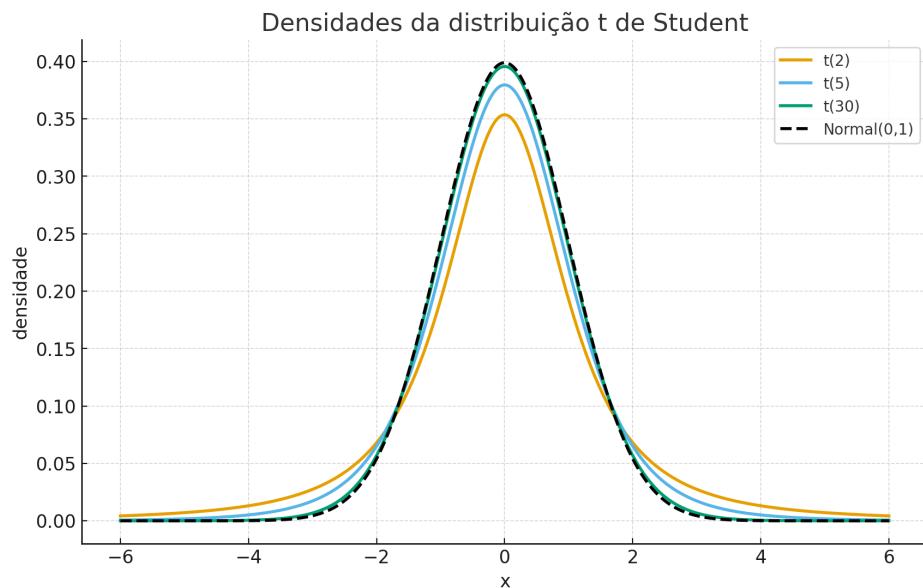
O algoritmo é:

1. Fixe $k \in \mathbb{N}$.
2. Gere $G \sim \Gamma\left(\frac{k}{2}, 2\right)$.
3. Retorne $Q = G$.

Esse método é geralmente mais eficiente em termos computacionais, já que muitas bibliotecas numéricas possuem rotinas otimizadas para a geração de variáveis Gama com parâmetros arbitrários. Assim, para valores grandes de k , pode ser preferível usar diretamente a simulação via Gama em vez da soma de muitos Normais.

4.5.4 Simulando a distribuição t de Student

A distribuição t de Student foi introduzida em 1908 por William Gosset, que trabalhava como mestre cervejeiro na Guinness. Por restrições da empresa, ele publicou seus resultados sob o pseudônimo *Student*, dando origem ao nome da distribuição. Essa distribuição aparece naturalmente em problemas de inferência estatística, especialmente em testes de hipóteses, mas aqui nos interessa sua definição e como simulá-la.



Seja $Z \sim N(0,1)$ e $Q \sim \chi^2_\nu$ independentes, com ν graus de liberdade. Definimos

$$T = \frac{Z}{\sqrt{Q/\nu}}.$$

Dizemos que T segue a distribuição t de Student com ν graus de liberdade, denotada por

$$T \sim t_\nu.$$

Algumas propriedades ajudam a caracterizar essa distribuição: ela é simétrica em torno de zero, de modo que se $T \sim t_\nu$ então também $-T \sim t_\nu$; no caso particular $\nu = 1$, a distribuição t_1 coincide com a distribuição de Cauchy; e quando $\nu \rightarrow \infty$, a distribuição t_ν converge para a Normal padrão $N(0,1)$. Essas propriedades mostram que, com poucos graus de liberdade,

a t de Student tem caudas mais pesadas do que a Normal, refletindo a incerteza adicional ao estimar a variância. À medida que ν cresce, a distribuição se aproxima da Normal, tornando-se praticamente indistinguível dela.

A intuição da fórmula pode ser entendida a partir da estatística de teste para a média de uma Normal. Se a variância σ^2 fosse conhecida, teríamos

$$Z = \frac{\bar{X} - \mu}{\sigma / \sqrt{n}} \sim N(0, 1).$$

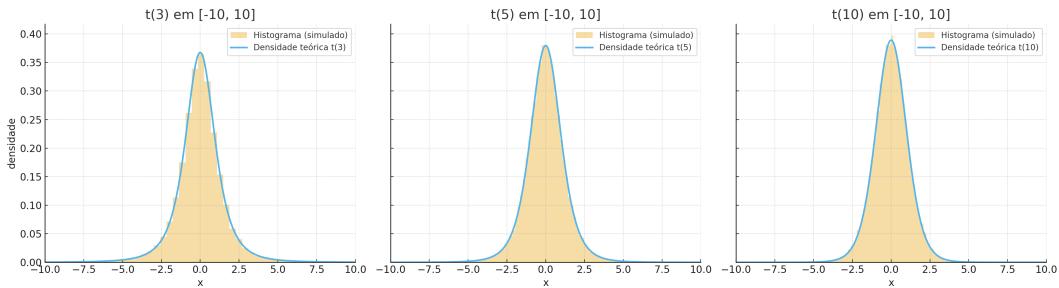
No entanto, como σ^2 é geralmente desconhecida, substituímos σ pela estimativa amostral S . Um resultado clássico mostra que

$$Q = \frac{(n-1)S^2}{\sigma^2} \sim \chi_{n-1}^2.$$

Assim, a estatística de interesse torna-se

$$T = \frac{Z}{\sqrt{Q/\nu}}, \quad \nu = n-1,$$

o que justifica a definição da distribuição t . O numerador Z mede a variabilidade da média padronizada, enquanto o denominador $\sqrt{Q/\nu}$ introduz a incerteza extra pela estimação da variância, resultando em caudas mais pesadas.



A própria definição sugere um algoritmo de simulação simples:

1. Fixe o número de graus de liberdade ν .
2. Gere $Z \sim N(0, 1)$.
3. Gere $Q \sim \chi_{\nu}^2$, independentemente de Z .
4. Retorne $T = Z / \sqrt{Q/\nu}$, que segue a lei t_{ν} .

Capítulo 5

Simulação via Monte Carlo

A ideia fundamental da simulação de Monte Carlo é usar amostras aleatórias para aproximar quantidades numéricas que, de outra forma, seriam difíceis ou impossíveis de calcular analiticamente. Em sua forma mais simples, o método baseia-se na *Lei dos Grandes Números*.

5.1 Estimando médias

Seja X uma variável aleatória com distribuição f , e desejamos estimar

$$\mu = \mathbb{E}[h(X)].$$

Gerando X_1, \dots, X_n independentes de f , definimos o estimador de Monte Carlo:

$$\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^n h(X_i).$$

Pela Lei dos Grandes Números, $\hat{\mu}_n \rightarrow \mu$ quase certamente quando $n \rightarrow \infty$. Uma maneira simples de justificar essa convergência é pela *desigualdade de Chebyshev*. Se $\text{Var}[h(X)] = \sigma^2 < \infty$, então

$$\text{Var}[\hat{\mu}_n] = \frac{\sigma^2}{n}.$$

Logo, para qualquer $\varepsilon > 0$,

$$\mathbb{P}(|\hat{\mu}_n - \mu| > \varepsilon) \leq \frac{\text{Var}[\hat{\mu}_n]}{\varepsilon^2} = \frac{\sigma^2}{n\varepsilon^2}.$$

Portanto, $\mathbb{P}(|\hat{\mu}_n - \mu| > \varepsilon) \rightarrow 0$ quando $n \rightarrow \infty$, mostrando que $\hat{\mu}_n$ converge para μ em probabilidade, o que é precisamente a versão fraca da Lei dos Grandes Números.

5.1.1 Exemplos

Exemplo 9. Considere a integral

$$I = \int_0^1 e^{-x^2} dx.$$

Embora não exista uma expressão analítica simples para essa integral, podemos aproximá-la por simulação de Monte Carlo.

Geramos $X_1, \dots, X_n \sim \text{Uniforme}(0, 1)$ e usamos a identidade

$$I = \mathbb{E} [e^{-X^2}] .$$

Assim, o estimador de Monte Carlo é

$$\hat{I}_n = \frac{1}{n} \sum_{i=1}^n e^{-X_i^2} .$$

O valor aproximado da integral é $I \approx 0.7468$.

Exemplo 10. Considere $X \sim \mathcal{N}(0, 1)$ e o evento $A = \{X > 1\}$. Queremos estimar a probabilidade

$$p = \mathbb{P}(X > 1) .$$

Geramos $X_1, \dots, X_n \sim \mathcal{N}(0, 1)$ e usamos o estimador de Monte Carlo

$$\hat{p}_n = \frac{1}{n} \sum_{i=1}^n \mathbb{1}\{X_i > 1\} .$$

Pela Lei dos Grandes Números, $\hat{p}_n \rightarrow p$ quando $n \rightarrow \infty$.

O valor verdadeiro é

$$p = 1 - \Phi(1) \approx 0.1587 ,$$

onde Φ denota a CDF da normal padrão.

Exemplo 11. Podemos estimar o valor de π por simulação de Monte Carlo usando uma interpretação geométrica.

Considere o quadrado $[0, 1] \times [0, 1]$ e o quarto de círculo de raio 1 centrado na origem, definido por

$$x^2 + y^2 \leq 1 .$$

A área do quarto de círculo é $\pi/4$. Assim, se gerarmos pontos (X_i, Y_i) uniformemente distribuídos no quadrado, a fração que cai dentro do círculo aproxima a razão entre as áreas, isto é,

$$\frac{\text{número de pontos no círculo}}{\text{número total de pontos}} \approx \frac{\pi}{4} .$$

Logo, o estimador de Monte Carlo é

$$\hat{\pi}_n = 4 \times \frac{1}{n} \sum_{i=1}^n \mathbb{1}\{X_i^2 + Y_i^2 \leq 1\} .$$

Pela Lei dos Grandes Números, $\hat{\pi}_n \rightarrow \pi$ quando $n \rightarrow \infty$.

Exemplo 12. Considere $X \sim \text{Uniforme}(0, 1)$. Queremos estimar simultaneamente $\mathbb{E}[X]$ e $\text{Var}[X]$ por simulação.

Geramos X_1, \dots, X_n independentes e usamos

$$\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^n X_i , \quad \hat{\sigma}_n^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \hat{\mu}_n)^2 .$$

Exemplo 13. Considere a integral em duas dimensões

$$I = \int_0^1 \int_0^1 e^{-(x^2+y^2)} dx dy.$$

Geramos $(X_i, Y_i) \sim \text{Uniforme}([0, 1]^2)$ e usamos o estimador

$$\hat{I}_n = \frac{1}{n} \sum_{i=1}^n e^{-(X_i^2+Y_i^2)}.$$

Exemplo 14. Considere $X \sim \text{Exponencial}(1)$ e queremos estimar $\mathbb{E}[e^{-X}]$. Geramos X_1, \dots, X_n independentes e calculamos

$$\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^n e^{-X_i}.$$

O valor verdadeiro é $\mathbb{E}[e^{-X}] = \frac{1}{2}$.

Exemplo 15. Considere a integral

$$I = \int_0^\infty \frac{e^{-x}}{1+x} dx.$$

Essa integral não tem forma fechada simples, mas pode ser expressa como uma esperança sob uma distribuição conveniente.

Observe que $f(x) = e^{-x}\mathbf{1}_{\{x>0\}}$ é a densidade de uma distribuição Exponencial(1). Assim, podemos escrever

$$I = \int_0^\infty \frac{e^{-x}}{1+x} dx = \int_0^\infty \frac{1}{1+x} f(x) dx = \mathbb{E}\left[\frac{1}{1+X}\right], \quad X \sim \text{Exponencial}(1).$$

Logo, podemos estimar I por Monte Carlo gerando $X_1, \dots, X_n \sim \text{Exponencial}(1)$ e calculando

$$\hat{I}_n = \frac{1}{n} \sum_{i=1}^n \frac{1}{1+X_i}.$$

O valor verdadeiro da integral é aproximadamente $I \approx 0.5963$.

Exemplo 16. Considere a integral

$$I = \int_0^\infty \frac{\sin(x)}{x} dx.$$

Não há uma densidade de probabilidade aparecendo explicitamente, mas podemos introduzir uma para reescrever a integral como uma esperança.

Escolha, por conveniência, a densidade exponencial $f(x) = e^{-x}\mathbf{1}_{\{x>0\}}$. Então,

$$I = \int_0^\infty \frac{\sin(x)}{x} dx = \int_0^\infty \frac{\sin(x)}{xe^{-x}} f(x) dx = \mathbb{E}\left[\frac{\sin(X)}{Xe^{-X}}\right], \quad X \sim \text{Exponencial}(1).$$

Assim, a integral pode ser estimada por Monte Carlo sem precisar integrar diretamente uma função oscilatória:

$$\hat{I}_n = \frac{1}{n} \sum_{i=1}^n \frac{\sin(X_i)}{X_i e^{-X_i}}, \quad X_i \sim \text{Exponencial}(1).$$

O valor exato da integral é $I = \frac{\pi}{2} \approx 1.5708$.

5.2 Intervalos de Confiança

Uma estimativa obtida por simulação de Monte Carlo é aleatória. Mesmo quando o estimador é não tendencioso, seu valor varia a cada execução devido à variabilidade amostral. Por isso, é importante quantificar essa incerteza por meio de um *intervalo de confiança*.

Seja $\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^n h(X_i)$ um estimador de Monte Carlo para $\mu = \mathbb{E}[h(X)]$. Pelo Teorema Central do Limite,

$$\sqrt{n} \frac{\hat{\mu}_n - \mu}{\sigma} \xrightarrow{d} \mathcal{N}(0, 1),$$

onde $\sigma^2 = \text{Var}[h(X)]$. Assim, para n grande,

$$\mathbb{P}\left(|\hat{\mu}_n - \mu| \leq z_{1-\alpha/2} \frac{\sigma}{\sqrt{n}}\right) \approx 1 - \alpha,$$

onde $z_{1-\alpha/2}$ é o quantil da normal padrão.

Na prática, a variância σ^2 é desconhecida. Usamos a estimativa amostral

$$s_n^2 = \frac{1}{n-1} \sum_{i=1}^n (h(X_i) - \hat{\mu}_n)^2.$$

Substituindo σ por s_n , obtemos o intervalo de confiança assintótico

$$\hat{\mu}_n \pm z_{1-\alpha/2} \frac{s_n}{\sqrt{n}}.$$

O intervalo representa a faixa de valores plausíveis para μ , dada a variabilidade da amostra. Para um nível de confiança de 95%, usamos $z_{0.975} \approx 1.96$, obtendo

$$\hat{\mu}_n \pm 1.96 \frac{s_n}{\sqrt{n}}.$$

Exemplo 17. Considere novamente a estimativa

$$I = \mathbb{E}[e^{-X^2}], \quad X \sim \text{Uniforme}(0, 1).$$

Queremos construir um intervalo de confiança para I com base em $n = 10^5$ simulações.

(1) O estimador de Monte Carlo é

$$\hat{I}_n = \frac{1}{n} \sum_{i=1}^n e^{-X_i^2}.$$

A média amostral obtida foi

$$\hat{I}_n = 0.7472.$$

(2) O desvio padrão amostral das observações $e^{-X_i^2}$ é

$$s_n = 0.289.$$

(3) O erro padrão do estimador é

$$\text{EP} = \frac{s_n}{\sqrt{n}} = \frac{0.289}{\sqrt{100000}} = 0.000914.$$

(4) Pela regra empírica 68–95–99.7, sabemos que:

- cerca de 68% das observações estão dentro de 1 desvio padrão da média;
- cerca de 95% estão dentro de 2 desvios padrão;
- e cerca de 99.7% estão dentro de 3 desvios padrão.

Assim, podemos construir um intervalo aproximado de 95% de confiança usando dois desvios padrão em vez de 1.96.

(5) O termo de margem é então

$$2 \times EP = 2 \times 0.000914 = 0.00183.$$

(6) O intervalo de confiança é

$$[0.7472 - 0.00183, 0.7472 + 0.00183] = [0.7454, 0.7490].$$

Em outras palavras, esperamos que cerca de 95% das repetições do experimento de Monte Carlo produzam valores de \hat{I}_n dentro de dois erros padrão da média verdadeira. O valor teórico $I = 0.7468$ está de fato dentro desse intervalo.

Exercício 25. Ache intervalos de confianças para todos os exemplos anteriores.

Capítulo 6

Redução de variância

Em um estudo de simulação, é comum que se deseje estimar um parâmetro θ associado a um modelo estocástico. Para isso, o modelo é executado a fim de gerar uma variável de saída X , cuja esperança é $\theta = \mathbb{E}[X]$.

Realizam-se então n repetições independentes da simulação, sendo que a i -ésima repetição fornece o valor X_i . A partir dessas observações, a estimativa natural de θ é a média amostral

$$\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i.$$

Note que \bar{X} é um estimador não viesado de θ , de modo que

$$\mathbb{E}[\bar{X}] = \theta.$$

Assim, o erro quadrático médio do estimador coincide com sua variância:

$$\text{MSE}(\bar{X}) = \mathbb{E}[(\bar{X} - \theta)^2] = \text{Var}(\bar{X}) = \frac{\text{Var}(X)}{n}.$$

Portanto, se for possível construir um outro estimador não viesado de θ com variância menor do que a de \bar{X} , obteremos uma estimativa mais eficiente. Este é o ponto de partida para as técnicas de redução de variância que discutiremos a seguir.

6.1 Uso de variáveis antitéticas

Considere o problema de estimar $\theta = \mathbb{E}[X]$ por simulação. Se gerarmos duas observações X_1 e X_2 , identicamente distribuídas com esperança θ , uma estimativa natural é a média

$$\hat{\theta} = \frac{X_1 + X_2}{2}.$$

A variância desse estimador pode ser escrita como

$$\text{Var}(\hat{\theta}) = \frac{1}{4} \text{Var}(X_1 + X_2) = \frac{1}{4} (\text{Var}(X_1) + \text{Var}(X_2) + 2 \text{Cov}(X_1, X_2)).$$

Como X_1 e X_2 têm a mesma distribuição, $\text{Var}(X_1) = \text{Var}(X_2)$, segue que

$$\text{Var}(\hat{\theta}) = \frac{1}{2} \text{Var}(X_1) + \frac{1}{2} \text{Cov}(X_1, X_2).$$

Portanto:

- se X_1 e X_2 forem independentes, $\text{Cov}(X_1, X_2) = 0$ e $\text{Var}(\hat{\theta}) = \frac{1}{2} \text{Var}(X_1)$;
- se conseguirmos construir X_1 e X_2 de modo que a covariância seja **negativa**, então a variância de $\hat{\theta}$ será ainda menor.

A questão, então, é: como gerar dois valores X_1 e X_2 com a mesma distribuição, mas negativamente correlacionados?

Suponha que X_1 seja função de m números aleatórios independentes, isto é,

$$X_1 = h(U_1, \dots, U_m),$$

onde U_1, \dots, U_m são independentes e uniformemente distribuídos em $(0, 1)$.

Observe que, se $U \sim U(0, 1)$, então também $1 - U \sim U(0, 1)$. Assim, se definirmos

$$X_2 = h(1 - U_1, \dots, 1 - U_m),$$

teremos que X_2 possui a mesma distribuição que X_1 .

Além disso, como $1 - U$ é negativamente correlacionado com U , é razoável esperar que X_2 seja negativamente correlacionado com X_1 .

Para tornar a ideia mais clara, considere o caso em que X_1 depende apenas de uma variável uniforme. Seja $U \sim U(0, 1)$ e uma função monótona crescente $h : [0, 1] \rightarrow \mathbb{R}$. Definimos

$$X_1 = h(U), \quad X_2 = h(1 - U).$$

Note que X_1 e X_2 têm a mesma distribuição, pois U e $1 - U$ são identicamente distribuídos. Além disso, se U assume um valor grande, então $X_1 = h(U)$ também será grande, mas nesse caso $1 - U$ será pequeno, de modo que $X_2 = h(1 - U)$ será pequeno. Assim, valores altos de X_1 tendem a estar associados a valores baixos de X_2 , e vice-versa, o que implica correlação negativa.

No caso particular em que $h(u) = u$, temos

$$X_1 = U, \quad X_2 = 1 - U.$$

Claramente, $\mathbb{E}[X_1] = \mathbb{E}[X_2] = \frac{1}{2}$, de modo que

$$\mathbb{E}[X_1]\mathbb{E}[X_2] = \frac{1}{4}.$$

Por outro lado,

$$\mathbb{E}[X_1 X_2] = \int_0^1 u(1 - u) du = \int_0^1 (u - u^2) du = \frac{1}{2} - \frac{1}{3} = \frac{1}{6}.$$

Assim,

$$\text{Cov}(X_1, X_2) = \frac{1}{6} - \frac{1}{4} = -\frac{1}{12} < 0,$$

mostrando explicitamente a correlação negativa entre X_1 e X_2 . Esse raciocínio se estende naturalmente para funções h monótonas em várias variáveis. Sejam U_1, \dots, U_m variáveis independentes uniformes em $(0, 1)$ e definamos

$$X_1 = h(U_1, \dots, U_m), \quad X_2 = h(1 - U_1, \dots, 1 - U_m),$$

com h crescente em cada coordenada.

Nesse caso, X_1 é uma função crescente do vetor (U_1, \dots, U_m) , enquanto X_2 é decrescente. Considerando

$$g(U_1, \dots, U_m) = -X_2,$$

vemos que g também é crescente em cada coordenada.

Ora, quando duas funções de um mesmo conjunto de variáveis independentes são monótonas no mesmo sentido (ambas crescentes ou ambas decrescentes), seus valores tendem a variar em conjunto, de modo que a covariância é não-negativa. Aplicando esse raciocínio a X_1 e g , concluímos que

$$\text{Cov}(X_1, -X_2) \geq 0,$$

o que implica

$$\text{Cov}(X_1, X_2) \leq 0.$$

Portanto, para qualquer função h crescente em cada coordenada, o par (X_1, X_2) é negativamente correlacionado, e o uso de variáveis antitéticas reduz (ou, no pior caso, não aumenta) a variância do estimador.

Em resumo, o *método das variáveis antitéticas* consiste em explorar a correlação negativa entre pares de simulações para reduzir a variância do estimador. Em vez de gerar duas réplicas independentes X_1 e X_2 , construímos o par de forma que ambas tenham a mesma distribuição, mas sejam negativamente correlacionadas. A variável antitética é dado por

$$X' = \frac{X_1 + X_2}{2},$$

o qual satisfaz $\mathbb{E}[X'] = \theta$, mas possui variância menor ou igual à do estimador baseado em amostras independentes.

Um algoritmo simples para aplicar o método pode ser descrito da seguinte forma:

1. Gere $U_1, \dots, U_m \sim \text{Uniforme}(0, 1)$ independentes.
2. Calcule $X_1 = h(U_1, \dots, U_m)$.
3. Calcule $X_2 = h(1 - U_1, \dots, 1 - U_m)$.
4. Defina a variável antitética como

$$X' = \frac{X_1 + X_2}{2}.$$

Sempre que utilizamos o método da inversão para gerar variáveis aleatórias, podemos aplicar diretamente a técnica das variáveis antitéticas. De fato, se $U \sim U(0, 1)$ gera a variável desejada via a transformação $X = F^{-1}(U)$, então $1 - U$ também é uniforme em $(0, 1)$, e portanto $X' = F^{-1}(1 - U)$ tem a mesma distribuição de X .

A grande vantagem é que, em vez de gerar duas variáveis independentes U_1 e U_2 para obter duas amostras de X , basta gerar uma única variável uniforme U . Com ela, obtemos simultaneamente o par antitético (X, X') , o que não apenas economiza custo computacional como também pode reduzir a variância do resultado final.

Exemplo 18. Considere a geração de uma variável aleatória exponencial com parâmetro $\lambda > 0$. Pelo método da inversão, se $U \sim U(0, 1)$, então

$$X = -\frac{1}{\lambda} \log(U)$$

segue a distribuição $Exp(\lambda)$.

Para aplicar o método das variáveis antitéticas, em vez de gerar duas variáveis independentes $U_1, U_2 \sim U(0, 1)$, usamos o par $(U, 1 - U)$. Assim, obtemos

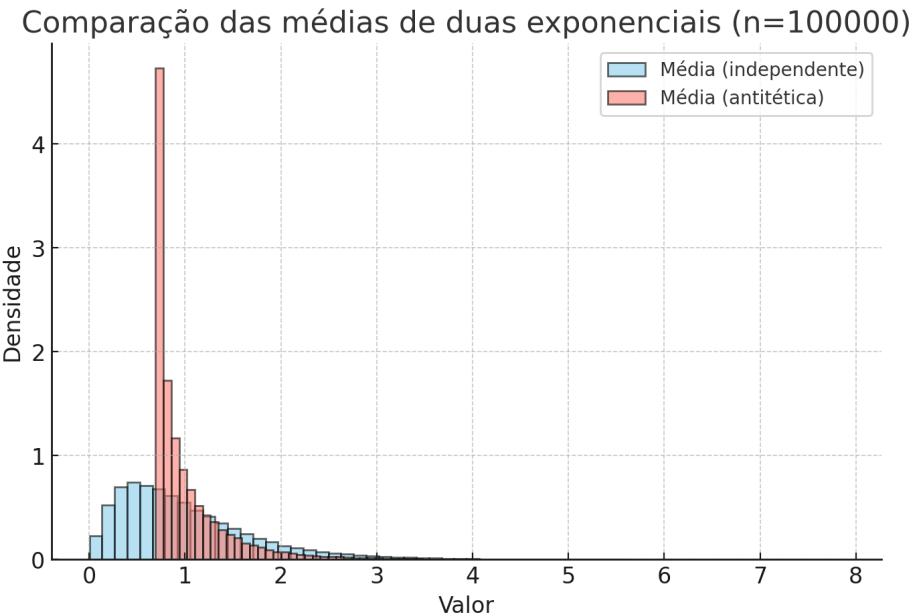
$$X_1 = -\frac{1}{\lambda} \log(U), \quad X_2 = -\frac{1}{\lambda} \log(1 - U).$$

Definimos, então, a variável final como a média

$$Z = \frac{X_1 + X_2}{2}.$$

O algoritmo é:

1. Gere $U \sim Uniforme(0, 1)$.
2. Calcule $X_1 = -\frac{1}{\lambda} \log(U)$.
3. Calcule $X_2 = -\frac{1}{\lambda} \log(1 - U)$.
4. Defina $Z = (X_1 + X_2)/2$.



No método independente, como cada variável exponencial pode assumir valores próximos de zero (quando $U \rightarrow 1$), a média também pode se aproximar de zero. Já no método antitético temos, supondo $\lambda = 1$,

$$Z = -\frac{1}{2} \log(U(1 - U)).$$

Como $U(1 - U) \leq 1/4$, segue que

$$Z \geq \frac{1}{2} \log 4 = \log 2 \approx 0.693.$$

Ou seja, a variável construída por antitéticos nunca assume valores menores que $\log 2$.

Esse resultado explica por que, ao comparar os histogramas, a média independente pode assumir valores próximos de zero, enquanto a antitética tem suporte a partir de $\log 2$. Além disso, no experimento com $n = 10^5$, o erro quadrático médio foi aproximadamente 0.505 no caso independente e apenas 0.174 no caso antitético, mostrando a expressiva redução de variância obtida pelo método.

Exemplo 19. Considere a integral

$$I = \int_0^\infty \log(1 + x^2) e^{-x} dx.$$

Observe que o termo e^{-x} corresponde à densidade de uma variável $X \sim \text{Exp}(1)$. Assim, podemos reescrever a integral como

$$I = \mathbb{E}[\log(1 + X^2)], \quad X \sim \text{Exp}(1).$$

Portanto, a solução via Monte Carlo é imediata: basta gerar amostras $X_i \sim \text{Exp}(1)$, calcular $\log(1 + X_i^2)$ e tirar a média. O algoritmo segue os passos:

1. Gerar $U_i \sim U(0, 1)$.
2. Transformar em $X_i = -\log(U_i)$.
3. Calcular $\log(1 + X_i^2)$ e tirar a média.

Para reduzir a variância, podemos usar variáveis antitéticas. Nesse caso, ao invés de gerar apenas U_i , usamos também $1 - U_i$. Isso produz

$$X_i = -\log(U_i), \quad X'_i = -\log(1 - U_i),$$

e então o estimador final é

$$\hat{I}_{ant} = \frac{1}{n} \sum_{i=1}^n \frac{1}{2} \left(\log(1 + X_i^2) + \log(1 + (X'_i)^2) \right).$$

Note que nem sempre variáveis antitéticas reduzem a variância: essa técnica é mais eficaz quando a função aplicada às amostras (aqui, $\log(1 + x^2)$) é monotônica, pois nesse caso os pares $(U, 1 - U)$ tendem a gerar correlação negativa entre os valores simulados.

Exemplo 20. Considere a integral

$$J = \int_{-\infty}^{\infty} e^x \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx.$$

O integrando envolve a densidade da normal padrão $N(0, 1)$, logo podemos escrever

$$J = \mathbb{E}[e^Z], \quad Z \sim N(0, 1).$$

O valor exato é conhecido:

$$J = e^{1/2}.$$

Para estimar J via Monte Carlo, seguimos os passos:

1. Gerar $Z_i \sim N(0, 1)$.
2. Calcular e^{Z_i} .
3. Tomar a média sobre as n amostras.

Note que a distribuição normal é simétrica em torno de zero, isto é, $Z \sim N(0, 1)$ implica que também $-Z \sim N(0, 1)$. Assim, para cada Z_i gerado, podemos considerar o par $(Z_i, -Z_i)$ e formar o estimador

$$\hat{J}_{ant} = \frac{1}{n} \sum_{i=1}^n \frac{1}{2} (e^{Z_i} + e^{-Z_i}).$$

Neste caso, como e^x é uma função monotônica crescente, os valores e^{Z_i} e e^{-Z_i} tendem a se compensar, gerando correlação negativa e uma variância muito menor na estimativa. O ponto essencial é que a esperança se mantém inalterada, mas o uso da antitética torna o estimador mais eficiente.

6.2 O uso de variáveis de controle

Suponha que desejamos estimar

$$\theta = \mathbb{E}[X],$$

onde X é o resultado de uma simulação. Agora suponha que exista outra variável Y cuja esperança é conhecida, digamos

$$\mathbb{E}[Y] = \mu_Y.$$

Então, para qualquer constante c , o estimador

$$Z = X + c(Y - \mu_Y)$$

é não-viesado para θ , pois $\mathbb{E}[Z] = \theta$.

A variância deste estimador é

$$\text{Var}(Z) = \text{Var}(X + c(Y - \mu_Y)) = \text{Var}(X) + c^2 \text{Var}(Y) + 2c \text{Cov}(X, Y).$$

Minimizando em relação a c , obtemos

$$c^* = -\frac{\text{Cov}(X, Y)}{\text{Var}(Y)}.$$

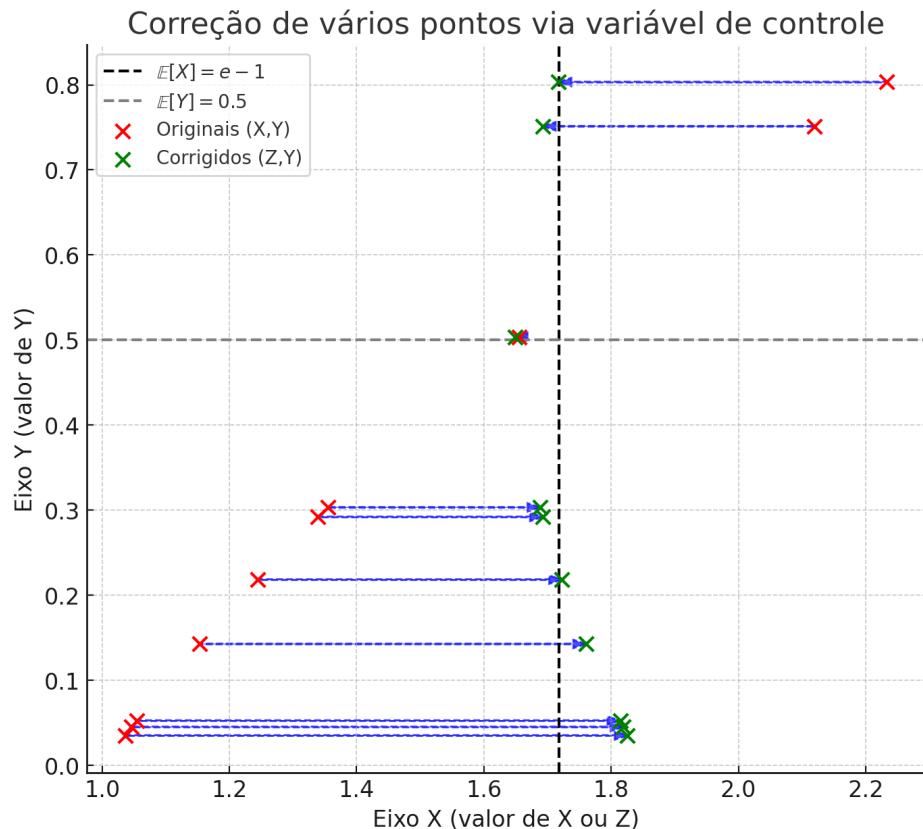
Substituindo este valor na expressão da variância, resulta

$$\text{Var}(Z) = \text{Var}(X) - \frac{\text{Cov}(X, Y)^2}{\text{Var}(Y)}.$$

A variável Y é chamada de *variável de controle*. A intuição é a seguinte: se X e Y são positivamente correlacionados, então $c^* < 0$. Nesse caso, quando Y assume um valor acima da sua média conhecida μ_Y , é provável que X também esteja acima de sua média θ . Para compensar esse excesso, reduzimos o valor de X ao somar $c(Y - \mu_Y)$ com $c < 0$. De forma análoga, se Y estiver abaixo de sua média, provavelmente X também estará, e nesse caso o termo $c(Y - \mu_Y)$ corrige o valor de X para cima. Quando X e Y são negativamente correlacionados, o raciocínio

é simétrico: nesse caso $c^* > 0$, e se Y está acima de sua média, é provável que X esteja abaixo; o termo $c(Y - \mu_Y)$ corrige então X para cima. Do mesmo modo, quando Y está abaixo de μ_Y , X tende a estar acima, e a correção ajusta X para baixo. Assim, tanto em correlação positiva quanto em correlação negativa o método funciona: o que importa não é o sinal, mas sim a intensidade da correlação.

Esse ajuste reduz a variância porque parte da flutuação de X pode ser explicada pela sua correlação com Y . O desvio de Y em relação à sua média atua como um indicador do desvio de X , e ao subtrair essa componente previsível obtemos um estimador mais estável. Do ponto de vista matemático, a variância de X é decomposta em uma parte explicável pela covariância com Y e uma parte residual; ao introduzir a variável de controle, eliminamos a parte explicável e restamos apenas com a parte residual, que é menor. Assim, a variância do estimador nunca aumenta e, se $\rho(X, Y) \neq 0$, ela é estritamente reduzida.



Além disso, ao dividir pela variância de X , obtemos

$$\frac{\text{Var}(Z)}{\text{Var}(X)} = 1 - \rho(X, Y)^2,$$

onde

$$\rho(X, Y) = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X) \text{Var}(Y)}}$$

é a correlação entre X e Y . Isso mostra que a redução relativa de variância obtida é de $100 \cdot \rho(X, Y)^2$ por cento, independentemente de a correlação ser positiva ou negativa.

Na prática, $\text{Cov}(X, Y)$ e $\text{Var}(Y)$ não são conhecidos de antemão e precisam ser estimados a partir dos dados simulados. Se n simulações são realizadas, gerando pares (X_i, Y_i) , podemos calcular

$$\widehat{\text{Cov}}(X, Y) = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y}), \quad \widehat{\text{Var}}(Y) = \frac{1}{n-1} \sum_{i=1}^n (Y_i - \bar{Y})^2,$$

e então definir

$$\hat{c}^* = -\frac{\widehat{\text{Cov}}(X, Y)}{\widehat{\text{Var}}(Y)}.$$

O estimador final com variável de controle é dado por

$$\hat{\theta}_{\text{ctrl}} = \frac{1}{n} \sum_{i=1}^n \left(X_i + \hat{c}^*(Y_i - \mu_Y) \right).$$

O procedimento pode ser resumido no seguinte algoritmo:

1. Gerar amostras (X_i, Y_i) da simulação, $i = 1, \dots, n$.
2. Calcular \bar{X} e \bar{Y} .
3. Estimar $\widehat{\text{Cov}}(X, Y)$ e $\widehat{\text{Var}}(Y)$.
4. Determinar $\hat{c}^* = -\widehat{\text{Cov}}(X, Y)/\widehat{\text{Var}}(Y)$.
5. Formar o estimador $\hat{\theta}_{\text{ctrl}} = \frac{1}{n} \sum_{i=1}^n (X_i + \hat{c}^*(Y_i - \mu_Y))$.

Exemplo 21. Considere a integral

$$\theta = \int_0^1 e^x dx = e - 1.$$

Podemos reescrevê-la como uma esperança, notando que se $U \sim U(0, 1)$ então

$$\theta = \mathbb{E}[e^U].$$

Assim, definindo $X = e^U$, podemos estimar θ por Monte Carlo a partir da média amostral de X .

Agora considere a variável $Y = U$, cuja esperança é conhecida, $\mu_Y = \mathbb{E}[U] = 0.5$. Como $X = e^U$ e $Y = U$ são fortemente correlacionados positivamente, podemos utilizar Y como variável de controle. O estimador controlado é dado por

$$Z = X + c^*(Y - \mu_Y),$$

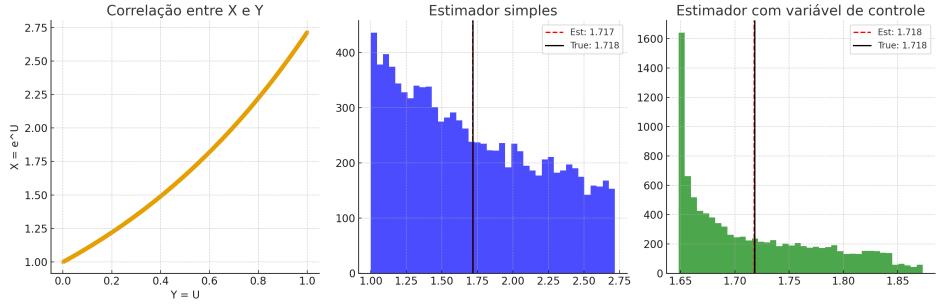
onde

$$c^* = -\frac{\text{Cov}(X, Y)}{\text{Var}(Y)}.$$

Na prática, basta gerar pares (X_i, Y_i) a partir de $U_i \sim U(0, 1)$, calcular as estimativas amostrais de $\text{Cov}(X, Y)$ e $\text{Var}(Y)$ para obter \hat{c}^* , e então construir o estimador

$$\hat{\theta}_{\text{ctrl}} = \frac{1}{n} \sum_{i=1}^n \left(X_i + \hat{c}^*(Y_i - \mu_Y) \right).$$

A correlação positiva entre X e Y faz com que o desvio de Y em relação à sua média indique a direção do desvio de X em relação a θ . O termo de ajuste $c^*(Y - \mu_Y)$ corrige esse efeito, reduzindo drasticamente a variância do estimador. Na simulação, observamos que a variância caiu de aproximadamente 0.24 (sem



controle) para 0.004 (com controle), com estimativas muito mais concentradas em torno do valor verdadeiro $e - 1 \approx 1.718$.

O procedimento pode ser resumido no seguinte algoritmo:

1. Gerar n amostras $U_i \sim U(0, 1)$, $i = 1, \dots, n$.
2. Calcular $X_i = e^{U_i}$ e $Y_i = U_i$.
3. Estimar $\widehat{\text{Cov}}(X, Y)$ e $\widehat{\text{Var}}(Y)$ a partir dos dados.
4. Determinar $\hat{c}^* = -\widehat{\text{Cov}}(X, Y)/\widehat{\text{Var}}(Y)$.
5. Construir o estimador controlado

$$\hat{\theta}_{ctrl} = \frac{1}{n} \sum_{i=1}^n \left(X_i + \hat{c}^*(Y_i - \mu_Y) \right).$$

6.3 Redução de Variância por Condicionamento

Uma técnica bastante útil para reduzir a variância em simulação é o uso de condicionamento. A ideia se baseia diretamente na fórmula da variância condicional:

$$\text{Var}(X) = \mathbb{E}[\text{Var}(X | Y)] + \text{Var}(\mathbb{E}[X | Y]).$$

A demonstração é simples. Por definição,

$$\text{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2.$$

Pela propriedade da esperança condicional,

$$\text{Var}(X) = \mathbb{E}[\mathbb{E}[X^2 | Y]] - (\mathbb{E}[\mathbb{E}[X | Y]])^2.$$

Agora, observe que

$$\mathbb{E}[X^2 | Y] = \text{Var}(X | Y) + (\mathbb{E}[X | Y])^2.$$

Portanto,

$$\mathbb{E}[\mathbb{E}[X^2 | Y]] = \mathbb{E}[\text{Var}(X | Y)] + \mathbb{E}[(\mathbb{E}[X | Y])^2].$$

Substituindo de volta,

$$\text{Var}(X) = \mathbb{E}[\text{Var}(X | Y)] + \left(\mathbb{E}[(\mathbb{E}[X | Y])^2] - (\mathbb{E}[X])^2 \right),$$

e o termo entre parênteses é precisamente $\text{Var}(\mathbb{E}[X | Y])$.

Assim, chegamos à decomposição

$$\text{Var}(X) = \mathbb{E}[\text{Var}(X | Y)] + \text{Var}(\mathbb{E}[X | Y]).$$

Como $\mathbb{E}[\text{Var}(X | Y)] \geq 0$, segue que

$$\text{Var}(X) \geq \text{Var}(\mathbb{E}[X | Y]),$$

portanto, podemos utilizar $Z = \mathbb{E}[X | Y]$ como variável para simularmos, já que $\mathbb{E}[Z] = \mathbb{E}[X]$.

Uma forma de entender por que o condicionamento reduz a variância é pensar no problema de estimar a altura média de uma população. Se representarmos por X a altura de uma pessoa escolhida ao acaso, cada sorteio pode resultar em valores muito diferentes, como um homem de 1,90 m ou uma mulher de 1,55 m, e essa variabilidade individual é refletida em $\text{Var}(X)$. Agora, suponha que introduzimos uma variável Y que indica o grupo ao qual a pessoa pertence, por exemplo, sexo masculino ou feminino. Em vez de registrar a altura individual X , passamos a registrar a média do grupo correspondente, isto é,

$$Z = \mathbb{E}[X | Y].$$

Se a pessoa sorteada for um homem, usamos a média de alturas dos homens (digamos, 175 cm); se for uma mulher, usamos a média das mulheres (digamos, 162 cm). Note que a média global continua correta: metade das vezes registramos 175, metade das vezes 162, o que resulta em 168,5 cm, exatamente a média real da população.

A identidade

$$\text{Var}(X) = \mathbb{E}[\text{Var}(X | Y)] + \text{Var}(\mathbb{E}[X | Y])$$

mostra como essa substituição reduz a variância. O primeiro termo corresponde à variabilidade dentro de cada grupo (diferenças entre indivíduos do mesmo sexo), enquanto o segundo termo corresponde à variabilidade entre as médias dos grupos (diferença entre a média dos homens e a das mulheres). Quando usamos diretamente X , ambos os termos estão presentes; quando usamos $\mathbb{E}[X | Y]$, eliminamos o primeiro termo e ficamos apenas com a variabilidade entre grupos. Assim, o estimador permanece não-viesado, mas com menor variância. Em outras palavras, condicionar equivale a substituir um indivíduo ruidoso pela média de seu grupo, preservando a esperança e reduzindo a dispersão.

O procedimento para estimar $\theta = \mathbb{E}[X]$ via condicionamento pode ser descrito da seguinte forma:

1. Identificar uma variável auxiliar Y em relação à qual seja possível calcular $\mathbb{E}[X | Y]$ de forma analítica ou computacionalmente simples.
2. Gerar amostras Y_1, Y_2, \dots, Y_n da distribuição de Y .
3. Para cada Y_i , calcular $Z_i = \mathbb{E}[X | Y_i]$.

4. Usar a média amostral

$$\hat{\theta} = \frac{1}{n} \sum_{i=1}^n Z_i$$

como estimador de θ .

Esse algoritmo gera um estimador não-viesado de θ , mas com variância reduzida em comparação ao estimador usual baseado diretamente em X .

Exemplo 22. Queremos estimar π via simulação. Podemos gerar dois números aleatórios $U_1, U_2 \sim U(0, 1)$ e definir

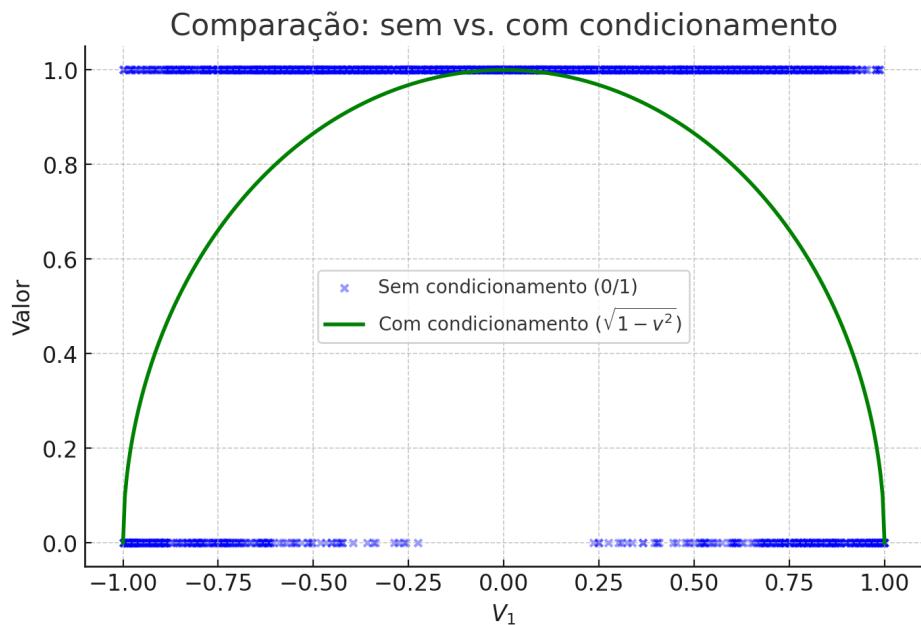
$$V_i = 2U_i - 1, \quad i = 1, 2,$$

de modo que $V_1, V_2 \sim U(-1, 1)$. Se considerarmos

$$I = \mathbf{1}\{V_1^2 + V_2^2 \leq 1\},$$

então $\mathbb{E}[I] = \pi/4$, pois a probabilidade de um ponto uniforme em $[-1, 1]^2$ cair dentro do círculo unitário é exatamente a razão entre a área do círculo e a área do quadrado. Assim, uma estimativa usual seria

$$\hat{\pi} = \frac{4}{n} \sum_{i=1}^n I_i.$$



Podemos, porém, melhorar esse estimador aplicando a técnica de condicionamento. Em vez de usar I diretamente, consideremos $\mathbb{E}[I | V_1]$. Temos

$$\mathbb{E}[I | V_1 = v] = P(V_1^2 + V_2^2 \leq 1 | V_1 = v).$$

Isso equivale a

$$\mathbb{E}[I | V_1 = v] = P(V_2^2 \leq 1 - v^2).$$

Como $V_2 \sim U(-1, 1)$ e é independente de V_1 , temos

$$P(V_2^2 \leq 1 - v^2) = P(-\sqrt{1-v^2} \leq V_2 \leq \sqrt{1-v^2}).$$

Portanto,

$$\mathbb{E}[I \mid V_1 = v] = \int_{-\sqrt{1-v^2}}^{\sqrt{1-v^2}} \frac{1}{2} dx = \sqrt{1-v^2}.$$

Assim, obtemos o estimador condicionado

$$Z = \sqrt{1 - V_1^2},$$

que satisfaz $\mathbb{E}[Z] = \pi/4$, mas tem variância menor do que I .

Finalmente, se $U \sim U(0, 1)$, temos $V_1 = 2U - 1$, e portanto

$$Z = \sqrt{1 - (2U - 1)^2}.$$

Logo, podemos simular π a partir do estimador

$$\hat{\pi} = \frac{4}{n} \sum_{i=1}^n \sqrt{1 - (2U_i - 1)^2},$$

que é mais eficiente do que usar diretamente o indicador I .

Exemplo 23. Considere a seguinte modelagem para a altura em uma população. Seja $Y \sim Bernoulli(p)$ a variável que indica o sexo do indivíduo, em que $Y = 0$ representa mulher e $Y = 1$ representa homem. Condicionalmente a Y , a altura X tem distribuição normal

$$X \mid Y = 0 \sim N(\mu_f, \sigma_f^2), \quad X \mid Y = 1 \sim N(\mu_m, \sigma_m^2).$$

Nosso objetivo é estimar a média da população,

$$\theta = \mathbb{E}[X] = (1-p)\mu_f + p\mu_m.$$

Se gerarmos indivíduos completos, isto é, sorteando Y e depois $X \mid Y$, o estimador de Monte Carlo é

$$\hat{\theta}_{simples} = \frac{1}{n} \sum_{i=1}^n X_i,$$

que é não viésado para θ , mas apresenta variância

$$\text{Var}(X) = (1-p)\sigma_f^2 + p\sigma_m^2 + p(1-p)(\mu_m - \mu_f)^2.$$

Em vez de usar diretamente X , podemos aplicar condicionamento. Nesse caso, registramos $Z = \mathbb{E}[X \mid Y]$, ou seja, μ_f se $Y = 0$ e μ_m se $Y = 1$. O estimador correspondente é

$$\hat{\theta}_{cond} = \frac{1}{n} \sum_{i=1}^n Z_i,$$

que também é não viésado para θ , mas com variância

$$\text{Var}(Z) = p(1-p)(\mu_m - \mu_f)^2,$$

estritamente menor do que $\text{Var}(X)$, pois elimina a variabilidade interna de cada grupo (σ_f^2 e σ_m^2). Assim, ao invés de considerar a altura ruidosa de cada indivíduo, utilizamos a média condicional do grupo, que é mais estável e resulta em um estimador mais eficiente.

6.4 Amostragem por importância

Considere uma variável aleatória X com densidade $f(x)$. Nosso objetivo é calcular

$$\theta = \mathbb{E}[h(X)] = \int h(x)f(x) dx.$$

Em alguns casos, uma simulação direta de $X \sim f$ pode ser ineficiente:

- pode ser difícil gerar amostras segundo f ;
- a variância de $h(X)$ sob f pode ser grande;
- ou ainda uma combinação desses fatores.

Uma alternativa é escolher uma outra densidade $g(x)$ tal que $f(x) = 0$ sempre que $g(x) = 0$.

Nesse caso, podemos reescrever

$$\theta = \int h(x) \frac{f(x)}{g(x)} g(x) dx = \mathbb{E}_g \left[h(X) \frac{f(X)}{g(X)} \right],$$

onde \mathbb{E}_g denota esperança em relação à densidade g .

Assim, se gerarmos $X_1, \dots, X_n \sim g$, um estimador natural é

$$\hat{\theta} = \frac{1}{n} \sum_{i=1}^n h(X_i) \frac{f(X_i)}{g(X_i)}.$$

Se a densidade instrumental g for bem escolhida, a variância do peso $h(X)f(X)/g(X)$ pode ser bem menor do que a variância de $h(X)$ sob f , resultando em uma estimação mais eficiente.

Note que $f(x)$ e $g(x)$ representam as probabilidades relativas de se observar x quando $X \sim f$ ou $X \sim g$. Quando $X \sim g$, em geral a razão $f(x)/g(x)$ é menor que 1, mas como

$$\mathbb{E}_g \left[\frac{f(X)}{g(X)} \right] = 1,$$

ela ocasionalmente assume valores grandes, podendo gerar alta variância.

A ideia central da amostragem por importância é escolher g de modo que nesses pontos onde $f(x)/g(x)$ é grande, a função $h(x)$ seja pequena (ou mesmo nula). Dessa forma, o produto

$$W(X) = h(X) \frac{f(X)}{g(X)}$$

permanece controlado, evitando explosões na variância.

Esse raciocínio mostra por que a técnica é especialmente eficaz na estimação de probabilidades raras. Nesse caso, $h(x)$ é uma função indicadora de um conjunto A pouco provável sob f . Se escolhermos g de modo que A seja mais frequente, então:

- para $x \in A$, temos $h(x) = 1$ e a razão $f(x)/g(x)$ é moderada;
- para $x \notin A$, temos $h(x) = 0$, logo não importa se $f(x)/g(x)$ é grande.

Assim, o estimador se torna muito mais estável e com variância reduzida, o que torna a amostragem por importância uma ferramenta poderosa para lidar com eventos raros.

6.4.1 Densidades Inclinadas (Tilted Densities)

Uma questão central em amostragem por importância é a escolha da densidade instrumental $g(x)$. Uma família bastante útil é a das *densidades inclinadas*, definidas a partir da função geradora de momentos.

Seja $X \sim f$ uma variável aleatória com função geradora de momentos

$$M(t) = \mathbb{E}_f[e^{tX}] = \int e^{tx} f(x) dx.$$

Definição 1. A densidade inclinada de f , associada ao parâmetro $t \in \mathbb{R}$, é definida por

$$f_t(x) = \frac{e^{tx} f(x)}{M(t)}.$$

Intuitivamente, a densidade f_t dá mais peso a valores grandes de X quando $t > 0$ e mais peso a valores pequenos quando $t < 0$. Em muitos casos, f_t pertence à mesma família paramétrica de f , mas com parâmetros modificados.

Alguns exemplos:

- **Normal.** Se $X \sim N(\mu, \sigma^2)$, então f_t é $N(\mu + t\sigma^2, \sigma^2)$. Nesse caso, o tilt desloca a média, concentrando a massa de probabilidade à direita quando $t > 0$ e à esquerda quando $t < 0$.
- **Exponencial.** Se $X \sim \text{Exp}(\lambda)$, então f_t é $\text{Exp}(\lambda - t)$, válido para $t < \lambda$. Aqui, o tilt altera o comportamento da cauda: para $t > 0$, a distribuição decai mais rápido (cauda mais leve), enquanto para $t < 0$ a cauda se torna mais pesada.
- **Gama.** Se $X \sim \text{Gamma}(\alpha, \beta)$, então f_t é $\text{Gamma}(\alpha, \beta - t)$, válido para $t < \beta$. Assim como na exponencial (caso particular da gama), o tilt controla a espessura da cauda, deixando-a mais leve quando $t > 0$ e mais pesada quando $t < 0$.
- **Poisson.** Se $X \sim \text{Poisson}(\lambda)$, então f_t é $\text{Poisson}(\lambda e^t)$. Nesse caso, o tilt modifica a média exponencialmente: para $t > 0$, a distribuição se desloca para valores grandes, enquanto para $t < 0$ se concentra em valores pequenos.
- **Binomial.** Se $X \sim \text{Binomial}(n, p)$, então f_t é $\text{Binomial}(n, p_t)$ com

$$p_t = \frac{pe^t}{1 - p + pe^t}.$$

Aqui, o tilt altera diretamente a probabilidade de sucesso: quando $t > 0$ temos $p_t > p$, o que força mais sucessos, e quando $t < 0$ temos $p_t < p$, forçando mais fracassos.

Exercício 26. Prove as afirmações acima.

Exemplo 24 (Estimando probabilidades raras). Sejam X_1, \dots, X_n variáveis aleatórias independentes com densidades (funções de massa ou de probabilidade) f_i , para $i = 1, \dots, n$. Defina

$$S = \sum_{i=1}^n X_i, \quad \mu = \sum_{i=1}^n \mathbb{E}[X_i].$$

Nosso objetivo é estimar a probabilidade de que S seja maior do que um limiar a , onde $a \gg \mu$, isto é,

$$\theta = \mathbb{P}(S > a) = \mathbb{E}[\mathbf{1}_{\{S>a\}}].$$

Quando a é muito maior que a média μ , esse evento é raro, e portanto uma simulação direta via Monte Carlo ingênuo é ineficiente, pois apenas uma fração ínfima das amostras contribui para o cálculo do estimador. Uma alternativa é utilizar a técnica de amostragem por importância com densidades inclinadas.

Seja

$$f_{i,t}(x) = \frac{e^{tx} f_i(x)}{M_i(t)}, \quad M_i(t) = \mathbb{E}[e^{tX_i}],$$

a densidade inclinada de X_i , onde $t > 0$ é um parâmetro comum a todas as variáveis. Ao simular cada X_i segundo $f_{i,t}$, obtemos que

$$\theta = \mathbb{E}_t \left[\mathbf{1}_{\{S>a\}} \exp(-tS) \prod_{i=1}^n M_i(t) \right],$$

onde \mathbb{E}_t denota esperança sob as densidades inclinadas.

A partir dessa representação, segue naturalmente um estimador de Monte Carlo:

$$\hat{\theta} = \frac{1}{N} \sum_{j=1}^N \mathbf{1}_{\{S^{(j)}>a\}} \exp(-tS^{(j)}) \prod_{i=1}^n M_i(t),$$

onde $S^{(j)} = \sum_{i=1}^n X_i^{(j)}$ e cada $X_i^{(j)}$ é simulado de $f_{i,t}$.

A escolha de t é crucial: se for muito pequeno, a distribuição inclinada pouco difere da original e o evento $\{S > a\}$ continua raro. Se for muito grande, os pesos podem se tornar instáveis, aumentando a variância. O critério usual é escolher t de forma que

$$\mathbb{E}_t[S] \approx a,$$

ou seja, deslocar a média da soma sob a medida inclinada para próximo do limiar a . Dessa forma, amostras são concentradas justamente nas regiões que mais contribuem para o evento raro, aumentando a eficiência do método.

O algoritmo pode ser resumido da seguinte forma:

1. Escolha $t > 0$ de modo que $\mathbb{E}_t[S] \approx a$.

2. Para $j = 1, \dots, N$:

(a) Gere $X_1^{(j)}, \dots, X_n^{(j)}$ independentemente segundo as densidades inclinadas $f_{i,t}$.

(b) Calcule $S^{(j)} = \sum_{i=1}^n X_i^{(j)}$.

(c) Associe o peso

$$W^{(j)} = \mathbf{1}_{\{S^{(j)}>a\}} \exp(-tS^{(j)}) \prod_{i=1}^n M_i(t).$$

3. Estime θ por

$$\hat{\theta} = \frac{1}{N} \sum_{j=1}^N W^{(j)}.$$

No caso particular em que cada $X_i \sim N(0, 1)$, temos que a soma $S = \sum_{i=1}^n X_i$ é normal $N(0, n)$. Quando $n = 1$, por exemplo, estimar $\mathbb{P}(S > 5)$ é um evento extremamente raro, pois o valor exato dessa probabilidade é

$$\theta = \mathbb{P}(S > 5) \approx 2.87 \times 10^{-7}.$$

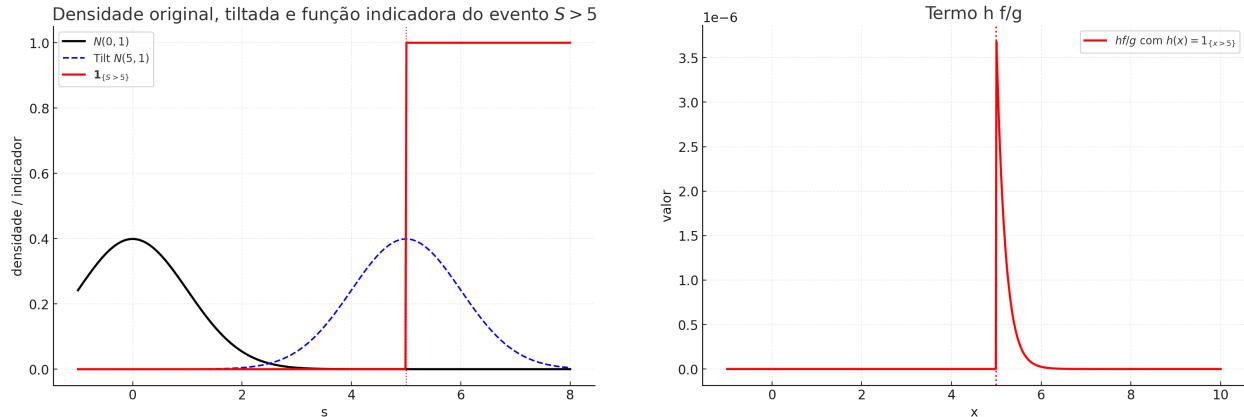
Um procedimento de Monte Carlo ingênuo, baseado apenas em amostrar de $N(0, 1)$, é ineficiente: em uma simulação com $N = 200,000$ repetições, a estimativa obtida foi de aproximadamente 5.0×10^{-6} , um valor que não coincide com o verdadeiro devido à raridade do evento.

Aplicando a técnica de densidades inclinadas, obtemos que a tilted density é

$$f_t(x) = \frac{e^{tx} f(x)}{M(t)}, \quad M(t) = \exp\left(\frac{1}{2}t^2\right),$$

o que implica que f_t é a densidade de uma normal $N(t, 1)$. Ao escolher $t = 5$, a distribuição inclinada desloca a média exatamente para o limiar de interesse. O peso associado a cada amostra é dado por

$$W = \mathbf{1}_{\{S>5\}} \exp(-tS + \frac{1}{2}t^2).$$



Nesse caso, a estimativa via amostragem por importância com $N = 200,000$ simulações foi

$$\hat{\theta}_{tilt} \approx 2.87 \times 10^{-7},$$

em perfeito acordo com o valor teórico.

Capítulo 7

Cadeias de Markov e MCMC

A simulação é uma técnica extremamente poderosa em probabilidade e estatística. Quando não conseguimos calcular analiticamente quantidades como a média ou a variância de uma variável aleatória X , podemos gerar amostras independentes X_1, X_2, \dots, X_n dessa distribuição e aproximar os valores verdadeiros por meio de estimadores amostrais:

$$\mathbb{E}(X) \approx \frac{1}{n}(X_1 + \dots + X_n) = \bar{X}_n, \quad \text{Var}(X) \approx \frac{1}{n-1} \sum_{j=1}^n (X_j - \bar{X}_n)^2.$$

A lei dos grandes números garante que essas aproximações serão boas se n for grande. Podemos melhorar cada vez mais a aproximação aumentando n , bastando rodar o computador por mais tempo (em vez de lidar com uma soma ou integral possivelmente intratável). Como discutido no Capítulo 10, essa abordagem, em que geramos valores aleatórios para aproximar uma quantidade, é chamada de método de *Monte Carlo*.

Uma limitação importante da ideia de Monte Carlo é que precisamos saber como gerar as amostras X_1, \dots, X_n (de preferência de forma eficiente, pois queremos n grande). Por exemplo, suponha que desejamos simular de uma distribuição contínua com função densidade

$$f(x) \propto x^{3.1}(1-x)^{4.2}, \quad 0 < x < 1.$$

Olhando apenas para a função de densidade, não é óbvio como obter uma variável aleatória com essa distribuição. Reconhecemos, no entanto, que se trata de uma Beta(4.1, 5.2). Mesmo assim, inverter a função de distribuição acumulada é difícil, e em distribuições mais complicadas nem sequer conhecemos a constante de normalização da densidade.

Essas dificuldades motivam o uso de um conjunto poderoso de algoritmos que revolucionaram a estatística e a computação científica: os métodos de *Monte Carlo via Cadeias de Markov* (MCMC). A ideia central é construir uma cadeia de Markov cuja distribuição estacionária seja justamente a distribuição de interesse.

Antes de estudarmos o MCMC em si, precisamos entender melhor o objeto central que sustenta esses métodos: as *cadeias de Markov*.

7.1 Cadeias de Markov (Resumo)

Cadeias de Markov “vivem” tanto no espaço quanto no tempo: o conjunto de valores possíveis de X_n é chamado de *espaço de estados*, enquanto o índice n representa a evolução do processo ao longo do tempo.

O espaço de estados de uma cadeia de Markov pode ser discreto ou contínuo, e o tempo também pode ser discreto ou contínuo. Neste capítulo vamos focar exclusivamente em cadeias de Markov com *tempo discreto* e *espaço de estados finito*. Em particular, assumiremos que X_n assume valores em um conjunto finito, que usualmente denotamos por $\{1, 2, \dots, M\}$ ou $\{0, 1, \dots, M\}$.

Definição 2 (Cadeia de Markov). *Uma sequência de variáveis aleatórias X_0, X_1, X_2, \dots com valores no espaço de estados $\{1, 2, \dots, M\}$ é chamada de cadeia de Markov se, para todo $n \geq 0$,*

$$\mathbb{P}(X_{n+1} = j \mid X_n = i, X_{n-1} = i_{n-1}, \dots, X_0 = i_0) = \mathbb{P}(X_{n+1} = j \mid X_n = i).$$

A condição acima é chamada de *propriedade de Markov*. Ela afirma que, dado o presente, o passado e o futuro são condicionalmente independentes. Ou seja, para prever o próximo estado, basta conhecer o estado atual.

Para descrever a dinâmica de uma cadeia de Markov, precisamos conhecer as probabilidades de transição de qualquer estado para qualquer outro.

Definição 3 (Matriz de transição). *Seja X_0, X_1, X_2, \dots uma cadeia de Markov com espaço de estados $\{1, 2, \dots, M\}$. Definimos*

$$q_{ij} = \mathbb{P}(X_{n+1} = j \mid X_n = i),$$

como a probabilidade de transição do estado i para o estado j . A matriz $Q = (q_{ij})_{i,j=1}^M$ é chamada de matriz de transição da cadeia.

A matriz de transição Q é não-negativa e cada linha soma 1, pois, dado um estado inicial i , a cadeia deve transitar para algum estado do espaço.

Exemplo 25 (Cadeia chuva-sol). *Suponha que em cada dia o clima possa ser chuvoso (R) ou ensolarado (S). Se hoje está chuvoso, amanhã estará chuvoso com probabilidade $1/3$ e ensolarado com probabilidade $2/3$. Se hoje está ensolarado, amanhã estará chuvoso com probabilidade $1/2$ e ensolarado com probabilidade $1/2$.*

Definindo X_n como o clima no dia n , temos uma cadeia de Markov com espaço de estados $\{R, S\}$ e matriz de transição

$$Q = \begin{pmatrix} 1/3 & 2/3 \\ 1/2 & 1/2 \end{pmatrix},$$

onde a primeira linha corresponde ao estado R e a segunda ao estado S .

Uma vez conhecida a matriz de transição Q de uma cadeia de Markov, podemos calcular as probabilidades de transição em horizontes de tempo maiores que um passo.

Definição 4 (Probabilidade de transição em n passos). *A probabilidade de transição em n passos do estado i para o estado j é a probabilidade de estarmos em j exatamente n passos após termos começado em i . Denotamos por*

$$q_{ij}^{(n)} = \mathbb{P}(X_n = j \mid X_0 = i).$$

Por exemplo, para $n = 2$, temos

$$q_{ij}^{(2)} = \sum_k q_{ik} q_{kj},$$

pois para ir de i a j em dois passos, a cadeia precisa ir primeiro de i até algum estado intermediário k , e depois de k até j . A propriedade de Markov garante a independência condicional dessas transições.

Note que o lado direito corresponde exatamente ao elemento (i, j) da matriz Q^2 , pela definição de multiplicação de matrizes. Portanto, a matriz Q^2 fornece as probabilidades de transição em dois passos.

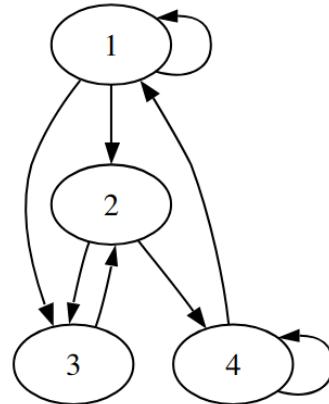
De maneira geral, por indução obtemos que

$$q_{ij}^{(n)} \text{ é a entrada } (i, j) \text{ da matriz } Q^n.$$

Exemplo 26 (Matriz de transição de uma cadeia de Markov com 4 estados). Considere a cadeia de Markov com 4 estados representada na Figura abaixo. Quando não há probabilidades escritas sobre as setas, isso significa que todas as transições saindo de um mesmo estado são igualmente prováveis.

Por exemplo, existem 3 setas saindo do estado 1, de modo que as transições $1 \rightarrow 3$, $1 \rightarrow 2$ e $1 \rightarrow 1$ ocorrem cada uma com probabilidade $1/3$. Portanto, a matriz de transição da cadeia é

$$Q = \begin{pmatrix} 1/3 & 1/3 & 1/3 & 0 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 1 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 \end{pmatrix}.$$



Para calcular a probabilidade de que a cadeia esteja no estado 3 após 5 passos, partindo do estado 1, basta olhar para o elemento $(1, 3)$ da matriz Q^5 .

Usando um computador, obtemos

$$Q^5 = \begin{pmatrix} 853/3888 & 509/1944 & 52/243 & 395/1296 \\ 173/864 & 85/432 & 31/108 & 91/288 \\ 37/144 & 29/72 & 1/9 & 11/48 \\ 499/2592 & 395/1296 & 71/324 & 245/864 \end{pmatrix}.$$

Assim,

$$q_{13}^{(5)} = \frac{52}{243}.$$

A matriz de transição Q codifica a distribuição condicional de X_1 dado o estado inicial da cadeia. Especificamente, a i -ésima linha de Q é a PMF condicional de X_1 dado $X_0 = i$, escrita como um vetor linha. De forma análoga, a i -ésima linha de Q^n corresponde à PMF condicional de X_n dado $X_0 = i$.

Para obter as distribuições marginais de X_0, X_1, \dots , precisamos especificar não apenas a matriz de transição, mas também as condições iniciais da cadeia. O estado inicial X_0 pode ser especificado de forma determinística, ou de forma aleatória segundo alguma distribuição. Seja $t = (t_1, t_2, \dots, t_M)$ a PMF de X_0 , vista como vetor linha, em que $t_i = \mathbb{P}(X_0 = i)$.

Proposição 1 (Distribuição marginal de X_n). *Seja $t = (t_1, t_2, \dots, t_M)$ o vetor de probabilidades iniciais, com $t_i = \mathbb{P}(X_0 = i)$. Então a distribuição marginal de X_n é dada por*

$$tQ^n.$$

Em particular, a j -ésima componente do vetor tQ^n é

$$\mathbb{P}(X_n = j).$$

Demonstração. Pela lei da probabilidade total, condicionando em X_0 , a probabilidade de a cadeia estar no estado j após n passos é

$$\mathbb{P}(X_n = j) = \sum_{i=1}^M \mathbb{P}(X_0 = i) \mathbb{P}(X_n = j | X_0 = i) = \sum_{i=1}^M t_i q_{ij}^{(n)}.$$

Mas essa soma corresponde exatamente à j -ésima componente do vetor tQ^n , pela definição de multiplicação de matrizes. \square

Exemplo 27 (Distribuições marginais de uma cadeia de Markov com 4 estados). *Considere novamente a cadeia de Markov com 4 estados representada na Figura anterior. Suponha que as condições iniciais sejam dadas por*

$$t = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right),$$

isto é, a cadeia começa com igual probabilidade em cada um dos quatro estados.

Seja X_n a posição da cadeia no tempo n . A distribuição marginal de X_1 é

$$tQ = \left(\frac{1}{4} \quad \frac{1}{4} \quad \frac{1}{4} \quad \frac{1}{4}\right) \begin{pmatrix} 1/3 & 1/3 & 1/3 & 0 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 1 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \end{pmatrix} = \left(\frac{5}{24}, \frac{1}{3}, \frac{5}{24}, \frac{1}{4}\right).$$

A distribuição marginal de X_5 é

$$tQ^5 = \left(\frac{1}{4} \quad \frac{1}{4} \quad \frac{1}{4} \quad \frac{1}{4}\right) \begin{pmatrix} 853/3888 & 509/1944 & 52/243 & 395/1296 \\ 173/864 & 85/432 & 31/108 & 91/288 \\ 37/144 & 29/72 & 1/9 & 11/48 \\ 499/2592 & 395/1296 & 71/324 & 245/864 \end{pmatrix} = \left(\frac{3379}{15552}, \frac{2267}{7776}, \frac{101}{486}, \frac{1469}{5184}\right).$$

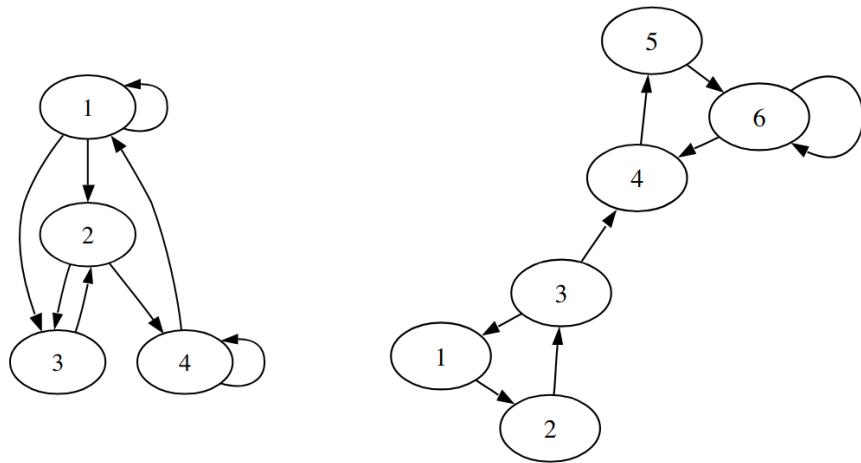
Neste caso utilizamos o computador para realizar as multiplicações de matrizes.

7.1.1 Classificação dos estados

Nesta parte introduziremos a terminologia usada para descrever as várias características de uma cadeia de Markov. Os estados de uma cadeia podem ser classificados como *recorrentes* ou *transientes*, dependendo de o processo retornar ou não a eles ao longo do tempo. Além disso, cada estado possui um *período*, que é um número inteiro positivo que resume a quantidade de passos que pode decorrer entre visitas sucessivas a esse estado.

Essas características são importantes porque determinam o comportamento de longo prazo da cadeia de Markov, que será estudado mais adiante.

Os conceitos de recorrência e transiência são melhor ilustrados com um exemplo.



Na cadeia de Markov mostrada à esquerda da Figura, uma partícula se movendo entre os estados continuará visitando todos os quatro estados indefinidamente, pois é possível transitar de qualquer estado para qualquer outro.

Em contraste, considere a cadeia à direita da Figura, e suponha que a partícula comece no estado 1. Durante algum tempo, a cadeia pode permanecer no triângulo formado pelos estados 1, 2, 3, mas eventualmente atingirá o estado 4. A partir do momento em que entra no estado 4, a cadeia nunca mais retorna a 1, 2, 3, e passa a se mover apenas entre os estados 4, 5, 6 para sempre.

Assim, os estados 1, 2, 3 são *transientes*, enquanto os estados 4, 5, 6 são *recorrentes*.

Definição 5 (Estados recorrentes e transientes). *Um estado i de uma cadeia de Markov é dito recorrente se, partindo de i , a probabilidade de que a cadeia eventualmente retorne a i é igual a 1.*

Caso contrário, o estado é dito transitente, o que significa que, se a cadeia começar em i , existe probabilidade positiva de nunca mais retornar a i .

Embora a definição de estado transitente apenas exija que haja probabilidade positiva de nunca retornar ao estado, podemos dizer algo mais forte: sempre que existir probabilidade positiva de abandonar i para sempre, a cadeia inevitavelmente deixará o estado i em algum momento.

Além disso, é possível caracterizar a distribuição do número de retornos ao estado.

Proposição 2 (Número de retornos a um estado transitente é Geométrico). *Seja i um estado transitente de uma cadeia de Markov. Suponha que a probabilidade de nunca retornar a i , partindo de i , seja*

$p > 0$. Então, partindo de i , o número de vezes que a cadeia retorna a i antes de sair para sempre é uma variável aleatória com distribuição

$$\text{Geom}(p).$$

Demonstração. A demonstração segue pela interpretação da distribuição Geométrica. Cada vez que a cadeia está em i , temos um ensaio de Bernoulli: ocorre “sucesso” se a cadeia sair de i para sempre, e ocorre “falha” se a cadeia eventualmente retornar a i . Esses ensaios são independentes pela propriedade de Markov.

O número de retornos ao estado i corresponde ao número de falhas antes do primeiro sucesso, exatamente a história da distribuição Geométrica. E como uma variável Geométrica assume valores finitos com probabilidade 1, concluímos que, após um número finito de visitas, a cadeia deixará o estado i para sempre. \square

Se o número de estados não for muito grande, uma maneira de classificar estados como recorrentes ou transientes é desenhar o diagrama da cadeia de Markov e aplicar o mesmo tipo de raciocínio feito na análise dos exemplos anteriores.

Um caso especial em que podemos concluir imediatamente que todos os estados são recorrentes ocorre quando a cadeia é *irredutível*, isto é, quando é possível ir de qualquer estado a qualquer outro.

Definição 6 (Cadeia irredutível e redutível). *Uma cadeia de Markov com matriz de transição Q é dita irredutível se, para quaisquer dois estados i e j , for possível ir de i até j em um número finito de passos, com probabilidade positiva.*

Isto é, para quaisquer estados i, j , existe um número inteiro $n > 0$ tal que a entrada (i, j) de Q^n é positiva.

Uma cadeia que não é irredutível é chamada de redutível.

Proposição 3 (Irredutibilidade implica recorrência de todos os estados). *Em uma cadeia de Markov irredutível com espaço de estados finito, todos os estados são recorrentes.*

Demonstração. É claro que pelo menos um estado deve ser recorrente; se todos fossem transientes, a cadeia eventualmente abandonaria todos os estados para sempre, o que é impossível.

Sem perda de generalidade, suponha que o estado 1 seja recorrente. Considere outro estado i . Pela definição de irredutibilidade, existe algum n tal que $q_{1i}^{(n)} > 0$.

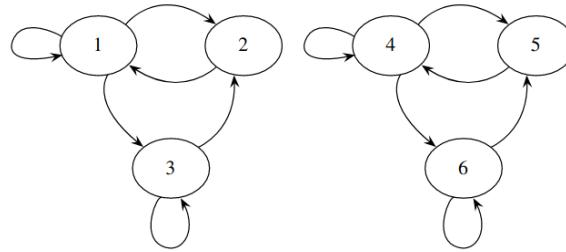
Assim, toda vez que a cadeia visita o estado 1, há uma probabilidade positiva de que, após n passos, ela esteja no estado i . Como a cadeia visita o estado 1 infinitas vezes (por recorrência), concluirá inevitavelmente no estado i .

Além disso, partindo de i , a cadeia retorna ao estado 1, já que este é recorrente. Aplicando o mesmo argumento recursivamente, a cadeia visitará o estado i infinitas vezes.

Como i foi arbitrário, segue que todos os estados são recorrentes. \square

A recíproca da proposição anterior é falsa: é possível ter uma cadeia de Markov *redutível* em que todos os estados sejam recorrentes.

Um exemplo é a cadeia ilustrada na Figura abaixo, que consiste em duas “ilhas” de estados.



Outra forma de classificar estados é de acordo com seus *períodos*. O período de um estado resume quanto tempo pode se passar entre visitas sucessivas a esse estado.

Definição 7 (Período de um estado, cadeia periódica e aperiódica). *O período de um estado i em uma cadeia de Markov é o máximo divisor comum (mdc) dos números de passos em que é possível retornar a i , partindo de i . Mais precisamente, o período de i é*

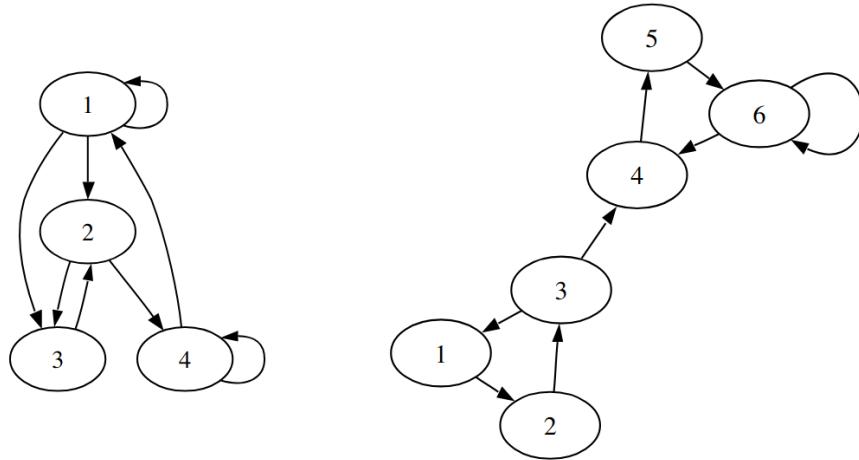
$$d(i) = \gcd\{ n \geq 1 : (Q^n)_{ii} > 0 \}.$$

Se nunca for possível retornar a i , definimos $d(i) = \infty$.

Um estado é dito aperiódico se $d(i) = 1$, e periódico caso contrário.

A cadeia como um todo é chamada aperiódica se todos os seus estados forem aperiódicos, e periódica caso contrário.

Exemplo 28 (Periodicidade em cadeias de Markov). *Considere novamente as duas cadeias de Markov da Figura abaixo.*



Na cadeia com 6 estados (à direita), partindo do estado 1, é possível retornar a ele após 3 passos, 6 passos, 9 passos e assim por diante. No entanto, não é possível retornar ao estado 1 em um número de passos que não seja múltiplo de 3. Portanto, o estado 1 tem período 3. De forma análoga, os estados 2 e 3 também têm período 3.

Por outro lado, os estados 4, 5, 6 possuem período 1. Como nem todos os estados têm período 1, a cadeia é considerada periódica.

Em contraste, na cadeia da Figura 2 (à esquerda), todos os estados são aperiódicos, de modo que a cadeia como um todo é aperiódica.

IMPORTANTE!!! Vale destacar uma diferença importante: algumas propriedades pertencem a *estados individuais* da cadeia de Markov, enquanto outras pertencem à *cadeia como um todo*. A tabela abaixo resume essa distinção, destacando os conceitos principais lado a lado para facilitar a comparação.

Propriedades de estados	Propriedades da cadeia
Estado recorrente: Partindo de i , a probabilidade de eventualmente retornar a i é 1.	Cadeia irredutível: É possível ir de qualquer estado i para qualquer estado j em um número finito de passos, com probabilidade positiva.
Estado transitente: Partindo de i , existe probabilidade positiva de nunca mais retornar a i .	Cadeia redutível: Não é possível ir de alguns estados i para outros j em um número finito de passos, com probabilidade positiva.
Estado aperiódico: O período do estado i é 1, ou seja, é possível retornar a i em tempos arbitrários suficientemente grandes.	Cadeia aperiódica: Todos os estados da cadeia são aperiódicos.
Estado periódico: O período do estado i é maior que 1, ou seja, o retorno a i só pode ocorrer em múltiplos de algum inteiro $d > 1$.	Cadeia periódica: Pelo menos um estado da cadeia é periódico.

7.1.2 Distribuição estacionária

Os conceitos de recorrência e transiência são fundamentais para compreender o comportamento de longo prazo de uma cadeia de Markov. Inicialmente, a cadeia pode passar algum tempo em estados transitórios; porém, com o tempo, ela tende a permanecer apenas nos estados recorrentes. Surge então uma pergunta natural: qual fração do tempo a cadeia passará em cada um desses estados recorrentes?

A resposta é dada pela *distribuição estacionária* da cadeia, também chamada de *distribuição em regime permanente*. Veremos nesta seção que, para cadeias de Markov irredutíveis e aperiódicas, a distribuição estacionária descreve o comportamento assintótico da cadeia, independentemente das condições iniciais. Ela fornece tanto a probabilidade de longo prazo de estar em um determinado estado quanto a proporção de tempo que a cadeia passará nesse estado.

Definição 8 (Distribuição estacionária). Um vetor linha $s = (s_1, \dots, s_M)$, com $s_i \geq 0$ e $\sum_i s_i = 1$, é dito uma *distribuição estacionária* para uma cadeia de Markov com matriz de transição Q se

$$\sum_i s_i q_{ij} = s_j, \quad \text{para todo } j.$$

Esse sistema de equações lineares pode ser escrito de forma compacta como

$$sQ = s.$$

Recorde que, se s é a distribuição de X_0 , então sQ é a distribuição marginal de X_1 . Assim, a igualdade $sQ = s$ significa que, se X_0 tem distribuição s , então X_1 também terá distribuição s . Pelo mesmo raciocínio, X_2, X_3, \dots também seguirão a mesma distribuição. Em outras palavras, uma cadeia de Markov cuja distribuição inicial é a distribuição estacionária s permanecerá nessa distribuição para sempre.

Observação 2. Podemos ter uma interpretação intuitiva da distribuição estacionária a partir de uma simulação mental. Imagine que temos um número muito grande de partículas (por exemplo, um bilhão), e que a distribuição inicial dessas partículas entre os estados é proporcional à distribuição inicial da cadeia. Em seguida, fazemos todas as partículas evoluírem segundo a matriz de transição Q .

Após um certo número de passos n , contamos quantas partículas estão em cada estado. Quando a cadeia atinge o regime estacionário, essas proporções se estabilizam: se contarmos novamente após aplicar Q mais uma vez, as frações relativas de partículas em cada estado permanecerão essencialmente as mesmas.

Assim, a distribuição estacionária s representa justamente essa configuração de equilíbrio em que a aplicação de Q não altera mais as proporções — isto é, $sQ = s$.

Exemplo 29 (Distribuição estacionária para uma cadeia com dois estados). Considere a matriz de transição

$$Q = \begin{pmatrix} \frac{1}{3} & \frac{2}{3} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

A distribuição estacionária é da forma $s = (s, 1 - s)$. Devemos então resolver

$$(s, 1 - s) \begin{pmatrix} \frac{1}{3} & \frac{2}{3} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} = (s, 1 - s),$$

o que é equivalente ao sistema

$$\begin{cases} \frac{1}{3}s + \frac{1}{2}(1 - s) = s, \\ \frac{2}{3}s + \frac{1}{2}(1 - s) = 1 - s. \end{cases}$$

A única solução é $s = \frac{3}{7}$. Portanto, a distribuição estacionária única dessa cadeia de Markov é

$$s = \left(\frac{3}{7}, \frac{4}{7} \right).$$

Mais geralmente, suponha que $q_{12} = a$ e $q_{21} = b$, com $0 < a < 1$ e $0 < b < 1$. A matriz de transição é então

$$Q = \begin{pmatrix} 1 - a & a \\ b & 1 - b \end{pmatrix}.$$

Escrevendo $s = (s_1, s_2)$, a equação $sQ = s$ fornece o sistema linear

$$\begin{cases} (1 - a)s_1 + bs_2 = s_1, \\ as_1 + (1 - b)s_2 = s_2. \end{cases}$$

Ambas as equações se reduzem a

$$as_1 = bs_2.$$

Como $s_2 = 1 - s_1$, obtemos a solução única

$$s = \left(\frac{b}{a + b}, \frac{a}{a + b} \right).$$

Existência, unicidade e convergência

Surge naturalmente a questão: uma distribuição estacionária sempre existe? E, caso exista, ela é única? Para cadeias de Markov com espaço de estados finito, a resposta é afirmativa: sempre existe uma distribuição estacionária. Além disso, quando a cadeia é irredutível, essa distribuição é única.

Teorema 6 (Existência e unicidade da distribuição estacionária). *Para qualquer cadeia de Markov irredutível, existe uma única distribuição estacionária. Nessa distribuição, todos os estados possuem probabilidade positiva.*

Esse resultado decorre de um teorema clássico da álgebra linear conhecido como *teorema de Perron–Frobenius*.

Além da existência e unicidade, também é importante entender a *convergência* para a distribuição estacionária. Já afirmamos de forma informal que a distribuição estacionária descreve o comportamento de longo prazo da cadeia: se a cadeia for executada por tempo suficiente, a distribuição marginal de X_n tende à distribuição estacionária s . O resultado a seguir formaliza essa ideia.

Teorema 7 (Convergência para a distribuição estacionária). *Se (X_0, X_1, \dots) é uma cadeia de Markov irredutível e aperiódica com distribuição estacionária s e matriz de transição Q , então*

$$\mathbb{P}(X_n = i) \longrightarrow s_i \quad \text{quando } n \rightarrow \infty.$$

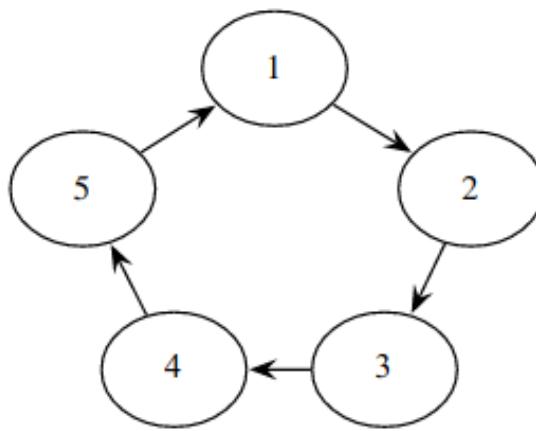
Em termos matriciais, Q^n converge para uma matriz cujas linhas são todas iguais a s .

Portanto, após um número suficientemente grande de passos, a probabilidade de que a cadeia esteja em um estado i se aproxima de s_i , independentemente das condições iniciais. Isso mostra que cadeias irredutíveis e aperiódicas são particularmente agradáveis de trabalhar, pois o seu comportamento assintótico é estável e previsível.

Observação 3. *De modo intuitivo, a condição adicional de aperiodicidade serve para evitar cadeias que apenas “giram em ciclos”, alternando de forma determinística entre grupos de estados. Por exemplo, em cadeias onde certos estados só são acessíveis após um número par de passos, enquanto outros apenas após um número ímpar, a convergência não ocorre sem essa hipótese. A combinação de irredutibilidade e aperiodicidade garante que a cadeia possa se misturar completamente no espaço de estados e, portanto, converge para sua distribuição estacionária.*

Exemplo 30 (Cadeia periódica). *Considere a cadeia de Markov ilustrada abaixo, em que cada estado possui período 5.*

$$Q = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$



Pode-se verificar facilmente que

$$s = \left(\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}\right)$$

é uma distribuição estacionária dessa cadeia, e que ela é única.

No entanto, suponha que a cadeia comece em $X_0 = 1$. Nesse caso, a distribuição de X_n atribui probabilidade 1 ao estado $(n \bmod 5) + 1$ e probabilidade 0 a todos os outros estados. Em particular, a distribuição de X_n não converge para s quando $n \rightarrow \infty$.

De forma equivalente, a matriz Q^n não converge para uma matriz em que todas as linhas são iguais a s . As transições dessa cadeia são determinísticas, e portanto cada Q^n continua sendo uma matriz composta apenas por zeros e uns. Esse exemplo mostra que, embora a distribuição estacionária exista e seja única, a convergência não ocorre quando a cadeia é periódica.

Observação 4. A condição de irredutibilidade é essencial para que a distribuição estacionária seja única e represente o comportamento de longo prazo da cadeia.

Se a cadeia não for irredutível, o espaço de estados pode se decompor em vários subconjuntos fechados — ou seja, conjuntos de estados dos quais não é possível sair. Cada um desses subconjuntos pode ter a sua própria distribuição estacionária, o que implica que não há uma única distribuição que descreva o comportamento de toda a cadeia.

Por exemplo, suponha que existam dois conjuntos de estados A e B tais que, uma vez que a cadeia entra em A , nunca mais pode ir para B , e vice-versa. Então, a probabilidade de longo prazo de estar em A ou em B dependerá da condição inicial. Isso impede a convergência para uma distribuição estacionária única.

A irredutibilidade elimina esse problema: ela garante que todos os estados se comunicam entre si, isto é, para quaisquer i e j , existe algum número de passos n tal que $(Q^n)_{ij} > 0$. Com isso, a cadeia pode eventualmente alcançar qualquer estado a partir de qualquer outro, o que assegura tanto a existência quanto a unicidade da distribuição estacionária e a convergência para ela.

Tempo médio de retorno e comportamento de longo prazo

Além de descrever o comportamento assintótico da cadeia, a distribuição estacionária também está relacionada ao tempo médio entre visitas a um estado específico.

Teorema 8 (Tempo esperado de retorno). *Considere uma cadeia de Markov irredutível com distribuição estacionária s . Seja r_i o tempo esperado para a cadeia retornar ao estado i , dado que ela comece em i . Então,*

$$s_i = \frac{1}{r_i}.$$

Esse resultado mostra que estados com maior probabilidade estacionária são visitados com mais frequência — em média, o tempo até retornar a eles é menor.

Exemplo 31 (Comportamento de longo prazo de uma cadeia com dois estados). *Considere novamente a cadeia de dois estados discutida anteriormente, cuja matriz de transição é*

$$Q = \begin{pmatrix} \frac{1}{3} & \frac{2}{3} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

A distribuição estacionária é

$$s = \left(\frac{3}{7}, \frac{4}{7} \right).$$

No longo prazo, a cadeia passará aproximadamente $3/7$ do tempo no estado 1 e $4/7$ do tempo no estado 2. Começando no estado 1, o tempo médio para retornar a esse estado é $r_1 = 7/3$, em conformidade com o teorema acima, pois $s_1 = 1/r_1$.

Além disso, as potências da matriz de transição convergem para uma matriz em que cada linha coincide com a distribuição estacionária:

$$Q^n = \begin{pmatrix} \frac{1}{3} & \frac{2}{3} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}^n \rightarrow \begin{pmatrix} \frac{3}{7} & \frac{4}{7} \\ \frac{3}{7} & \frac{4}{7} \end{pmatrix}, \quad \text{quando } n \rightarrow \infty.$$

7.1.3 Reversibilidade

Vimos que a distribuição estacionária de uma cadeia de Markov é extremamente útil para compreender seu comportamento de longo prazo. Entretanto, em muitos casos pode ser computacionalmente difícil determinar essa distribuição, especialmente quando o espaço de estados é grande. Nesta seção, estudamos um caso especial importante em que é possível evitar o cálculo direto das equações de autovalor associadas à matriz de transição.

Definição 9 (Reversibilidade). *Seja $Q = (q_{ij})$ a matriz de transição de uma cadeia de Markov. Dizemos que a cadeia é reversível em relação a um vetor $s = (s_1, \dots, s_M)$, com $s_i \geq 0$ e $\sum_i s_i = 1$, se*

$$s_i q_{ij} = s_j q_{ji}, \quad \text{para todos os estados } i, j.$$

Essa equação é chamada de condição de equilíbrio detalhado (ou detailed balance condition).

A intuição por trás da reversibilidade é a seguinte: uma cadeia reversível, iniciada segundo sua distribuição estacionária, se comporta da mesma forma independentemente de o tempo estar sendo observado para frente ou para trás. Mais precisamente, quando a cadeia está em equilíbrio, a probabilidade de sair do estado i e ir para o estado j em um passo é $s_i q_{ij}$, e essa probabilidade é exatamente igual à de sair de j e voltar para i , que é $s_j q_{ji}$. Em outras palavras, o fluxo de probabilidade de i para j é o mesmo que o de j para i :

$$s_i q_{ij} = s_j q_{ji}.$$

Isso significa que, no regime estacionário, as transições “para frente” e “para trás” ocorrem com a mesma frequência média, de modo que, se observarmos a cadeia no tempo inverso, ela parecerá estatisticamente idêntica à original.

Outra maneira de entender a reversibilidade é pensar na cadeia como um sistema com um grande número de partículas que se movem de forma independente de acordo com as probabilidades de transição. No longo prazo, a proporção de partículas em cada estado j é dada pela probabilidade estacionária s_j . O equilíbrio estacionário garante que, em média, o fluxo de partículas que sai de cada estado é igual ao fluxo de partículas que entra nele.

Mais precisamente, seja n o número total de partículas e s o vetor de proporções atuais de partículas em cada estado. Temos que s é estacionário se, e somente se,

$$s_j = \sum_i s_i q_{ij} = s_j q_{jj} + \sum_{i \neq j} s_i q_{ij}, \quad \text{para todo } j.$$

Multiplicando por n , obtemos

$$ns_j(1 - q_{jj}) = \sum_{i \neq j} ns_i q_{ij}.$$

O lado esquerdo representa o número médio de partículas que sairão do estado j no próximo passo, enquanto o lado direito representa o número médio de partículas que entrarão em j . Portanto, há um equilíbrio entre entrada e saída de partículas em cada estado.

A condição de reversibilidade impõe uma forma ainda mais forte de equilíbrio: para cada par de estados distintos i e j ,

$$ns_i q_{ij} = ns_j q_{ji}.$$

O lado esquerdo corresponde ao número médio de partículas que vão de i para j , e o lado direito ao número médio que vai de j para i . Assim, a reversibilidade garante que, par a par, o fluxo entre dois estados é perfeitamente equilibrado.

Proposição 4 (Reversibilidade implica estacionariedade). *Se $Q = (q_{ij})$ é a matriz de transição de uma cadeia de Markov reversível em relação a um vetor $s = (s_1, \dots, s_M)$ não negativo, com soma dos componentes igual a 1, então s é uma distribuição estacionária da cadeia.*

Demonstração. Temos

$$\sum_i s_i q_{ij} = \sum_i s_j q_{ji} = s_j \sum_i q_{ji} = s_j,$$

onde a última igualdade decorre do fato de que a soma das probabilidades em cada linha de Q é igual a 1. Logo, $sQ = s$, e portanto s é estacionária. \square

Esse é um resultado poderoso, pois frequentemente é mais simples verificar a condição de reversibilidade do que resolver o sistema completo de equações $sQ = s$.

Um caso importante e simples de cadeia reversível ocorre quando a matriz de transição Q é simétrica. Se Q é simétrica, isto é, $q_{ij} = q_{ji}$ para todos os i, j , então a distribuição estacionária é uniforme sobre o espaço de estados:

$$s = \left(\frac{1}{M}, \frac{1}{M}, \dots, \frac{1}{M}\right).$$

De fato, se $q_{ij} = q_{ji}$, a condição de reversibilidade $s_i q_{ij} = s_j q_{ji}$ é satisfeita sempre que $s_i = s_j$ para todos os pares (i, j) .

Esse é um caso particular de um fato mais geral: quando cada coluna de Q também soma 1, a distribuição uniforme continua sendo estacionária.

Proposição 5 (Distribuição estacionária uniforme). *Se cada coluna da matriz de transição Q soma 1, então a distribuição uniforme sobre todos os estados,*

$$s = \left(\frac{1}{M}, \frac{1}{M}, \dots, \frac{1}{M} \right),$$

é uma distribuição estacionária da cadeia de Markov.

Demonstração. Se cada coluna de Q soma 1, então o vetor linha $v = (1, 1, \dots, 1)$ satisfaz $vQ = v$. Dividindo por M , obtemos

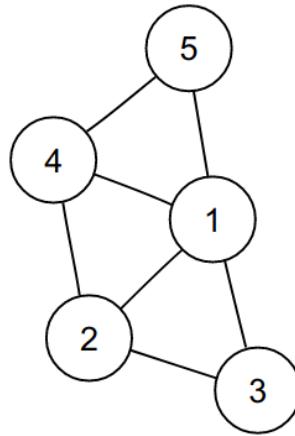
$$\left(\frac{1}{M}, \frac{1}{M}, \dots, \frac{1}{M} \right) Q = \left(\frac{1}{M}, \frac{1}{M}, \dots, \frac{1}{M} \right),$$

logo a distribuição uniforme é estacionária. \square

Uma matriz cujas linhas e colunas somam 1 é chamada de *matriz duplamente estocástica*. Toda cadeia de Markov cuja matriz de transição é duplamente estocástica possui distribuição estacionária uniforme.

Exemplo 32 (Caminhada aleatória em uma rede não direcionada). *Uma rede é um conjunto de nós conectados por arestas. A rede é dita não direcionada se as arestas puderem ser percorridas em ambos os sentidos, isto é, se não houver “ruas de mão única”.*

Considere um caminhante que percorre aleatoriamente as arestas de uma rede não direcionada. A partir de um nó i , o caminhante escolhe uma das arestas conectadas a i com probabilidades iguais e então atravessa a aresta escolhida.



O grau de um nó é o número de arestas conectadas a ele. A sequência de graus de uma rede com nós $1, 2, \dots, n$ é o vetor

$$d = (d_1, d_2, \dots, d_n),$$

onde d_j é o grau do nó j . Areias que ligam um nó a ele mesmo (self-loops) são permitidas e contam como 1 no grau desse nó.

Por exemplo, para a rede acima, a sequência de graus é

$$d = (4, 3, 2, 3, 2).$$

Note que, para todos os pares de nós i, j ,

$$d_i q_{ij} = d_j q_{ji},$$

pois $q_{ij} = 1/d_i$ se $\{i, j\}$ é uma aresta e $q_{ij} = 0$ caso contrário (para $i \neq j$). Logo, pela proposição anterior, a distribuição estacionária é proporcional à sequência de graus:

$$s_i \propto d_i.$$

De forma intuitiva, os nós com maior grau são mais “bem conectados”, e portanto o caminhante passa mais tempo neles no longo prazo. No exemplo acima, isso resulta em

$$s = \left(\frac{4}{14}, \frac{3}{14}, \frac{2}{14}, \frac{3}{14}, \frac{2}{14} \right),$$

que é a distribuição estacionária da caminhada aleatória nessa rede.

Mais geralmente, é possível considerar uma caminhada aleatória em uma rede não direcionada ponderada, onde cada aresta possui um peso. Nesse caso, o caminhante escolhe seu próximo nó a partir de i com probabilidades proporcionais aos pesos das arestas que partem de i. Esse processo também define uma cadeia de Markov reversível. De fato, todo processo de Markov reversível pode ser interpretado como uma caminhada aleatória sobre uma rede não direcionada com pesos apropriados nas arestas.

7.2 Markov Chain Monte Carlo

Ao longo destas notas vimos que a simulação é uma ferramenta extremamente poderosa em probabilidade. Quando o raciocínio analítico é complicado ou pouco intuitivo, a simulação permite verificar resultados de forma empírica.

De maneira semelhante, se não soubermos calcular explicitamente a média e a variância de uma variável aleatória X , mas soubermos gerar amostras independentes X_1, X_2, \dots, X_n , podemos aproximar os valores verdadeiros por

$$\mathbb{E}[X] \approx \frac{1}{n} \sum_{j=1}^n X_j = \bar{X}_n, \quad \text{Var}[X] \approx \frac{1}{n-1} \sum_{j=1}^n (X_j - \bar{X}_n)^2.$$

Pelo teorema da lei dos grandes números, essas aproximações se tornam mais precisas à medida que n cresce. Podemos obter resultados cada vez melhores apenas aumentando o tempo de simulação, sem precisar lidar com integrais ou somas intratáveis. Esse tipo de abordagem é conhecido como *método de Monte Carlo*.

Um dos principais desafios dos métodos de Monte Carlo é a necessidade de saber gerar amostras X_1, X_2, \dots, X_n da distribuição desejada. Em muitos casos, isso não é nada trivial. Por exemplo, suponha que queremos gerar valores de uma variável contínua com densidade

$$f(x) \propto x^{3.1}(1-x)^{4.2}, \quad 0 < x < 1.$$

Reconhecemos aqui a forma de uma distribuição Beta(4.1, 5.2). Mesmo assim, a função de distribuição acumulada (CDF) da Beta é complicada, e inverter essa função para gerar amostras é praticamente inviável na prática.

Em aplicações reais, as distribuições costumam ser muito mais complexas do que a Beta. Muitas vezes, o termo de normalização da densidade (ou massa de probabilidade) é desconhecido e impossível de calcular com precisão, mesmo com computadores modernos e técnicas avançadas de integração numérica.

O objetivo desta seção é introduzir o método *Markov Chain Monte Carlo* (MCMC), uma família de algoritmos que permite gerar amostras de distribuições complexas a partir de cadeias de Markov. O desenvolvimento do MCMC revolucionou a estatística e a computação científica, pois tornou possível simular distribuições de alta dimensão ou com normalizadores desconhecidos.

A ideia central é inverter o problema estudado anteriormente. Antes, conhecíamos a matriz de transição Q e procurávamos sua distribuição estacionária s . Agora, o processo é o oposto: partimos de uma distribuição s que desejamos simular e construímos uma cadeia de Markov cuja distribuição estacionária é exatamente s .

Se essa cadeia for executada por tempo suficiente, a distribuição de seus estados se aproximará da distribuição alvo s . O aspecto surpreendente é que isso pode ser feito sem conhecer o termo de normalização de s , o que torna o método aplicável em uma enorme variedade de contextos.

7.3 Algoritmo de Metropolis–Hastings

O algoritmo de *Metropolis–Hastings* é um método geral para construir cadeias de Markov cuja distribuição estacionária seja uma distribuição alvo arbitrária. A ideia central é inverter o problema usual: em vez de começar de uma cadeia e buscar sua distribuição estacionária, queremos agora *construir* uma cadeia cuja distribuição estacionária seja uma distribuição prescrita $s = (s_1, \dots, s_M)$.

Suponha que o espaço de estados seja finito, $\{1, \dots, M\}$, e que $s_i > 0$ para todo i . Considere uma cadeia de Markov conhecida, com matriz de transição $P = (p_{ij})$. Essa cadeia é escolhida apenas porque sabemos simulá-la facilmente — isto é, conseguimos gerar um próximo estado j a partir de um estado atual i segundo as probabilidades p_{ij} . Entretanto, P em geral *não* tem s como distribuição estacionária. Nossa objetivo é modificar o mecanismo de transição de P para obter uma nova cadeia $Q = (q_{ij})$ que preserve s como estacionária.

A modificação consiste em introduzir um mecanismo de *aceitação e rejeição*. A cadeia original P propõe um possível novo estado, e o algoritmo decide se essa proposta será aceita ou rejeitada. Essa etapa de aceitação é cuidadosamente escolhida para garantir que a cadeia resultante satisfaça a condição de reversibilidade com respeito a s , o que implica que s é estacionária.

O algoritmo procede da seguinte forma:

1. Escolha um estado inicial X_0 (de forma aleatória ou determinística).
2. No passo n , suponha que a cadeia está em $X_n = i$.

3. Proponha um novo estado j de acordo com as probabilidades da linha i da matriz P ; isto é, escolha j com probabilidade p_{ij} .
4. Calcule a *probabilidade de aceitação*

$$a_{ij} = \min\left(\frac{s_j p_{ji}}{s_i p_{ij}}, 1\right).$$

5. Com probabilidade a_{ij} , aceite a proposta e defina $X_{n+1} = j$; caso contrário, rejeite a proposta e mantenha $X_{n+1} = i$.

A matriz P é chamada de *matriz de proposta*, pois serve apenas para gerar possíveis movimentos da cadeia. As probabilidades de aceitação a_{ij} ajustam essas propostas para que o equilíbrio da nova cadeia Q obedeça à distribuição s . Em notação compacta, a nova matriz de transição é

$$q_{ij} = \begin{cases} p_{ij}a_{ij}, & i \neq j, \\ 1 - \sum_{k \neq i} p_{ik}a_{ik}, & i = j. \end{cases}$$

Um aspecto importante é que o algoritmo *não requer o conhecimento da constante de normalização* de s . Se $s \propto f$, com f conhecida apenas até uma constante multiplicativa, o quociente $s_j/s_i = f(j)/f(i)$ elimina o fator comum. Isso torna o método aplicável mesmo quando s é conhecida apenas de forma não normalizada, o que ocorre frequentemente em problemas de inferência bayesiana.

Além disso:

- se $p_{ij} = 0$, a transição $i \rightarrow j$ nunca é proposta, logo não precisa ser avaliada;
- se $p_{ii} > 0$, pode ocorrer de a proposta coincidir com o estado atual, e a cadeia naturalmente permanece em i .

Em resumo, o algoritmo de Metropolis–Hastings constrói uma nova cadeia de Markov a partir de uma cadeia proposta P , aceitando ou rejeitando cada movimento de forma a garantir que a distribuição estacionária da cadeia resultante seja exatamente a distribuição desejada s .

Proposição 6 (Reversibilidade da cadeia de Metropolis–Hastings). *Seja $Q = (q_{ij})$ a matriz de transição da cadeia gerada pelo algoritmo de Metropolis–Hastings. Então a cadeia é reversível em relação à distribuição s , e portanto s é sua distribuição estacionária.*

Demonstração. Precisamos verificar a condição de reversibilidade

$$s_i q_{ij} = s_j q_{ji}, \quad \text{para todos os } i, j.$$

O caso $i = j$ é imediato, pois ambos os lados são iguais a $s_i q_{ii}$. Consideremos agora $i \neq j$.

Se $q_{ij} > 0$, então $p_{ij} > 0$ — a cadeia só pode propor uma transição possível — e também $p_{ji} > 0$, pois, do contrário, a probabilidade de aceitação seria nula. De forma análoga, se $p_{ij}, p_{ji} > 0$, então $q_{ji} > 0$. Portanto, q_{ij} e q_{ji} são simultaneamente nulos ou não nulos.

Para $i \neq j$ com $q_{ij} > 0$, temos

$$q_{ij} = p_{ij}a_{ij},$$

pois, partindo de i , a única forma de alcançar j é propor essa transição (com probabilidade p_{ij}) e aceitá-la (com probabilidade a_{ij}).

Caso 1: se $s_j p_{ji} \leq s_i p_{ij}$, então

$$a_{ij} = \frac{s_j p_{ji}}{s_i p_{ij}} \quad \text{e} \quad a_{ji} = 1.$$

Logo,

$$s_i q_{ij} = s_i p_{ij} a_{ij} = s_i p_{ij} \frac{s_j p_{ji}}{s_i p_{ij}} = s_j p_{ji} = s_j p_{ji} a_{ji} = s_j q_{ji}.$$

Caso 2: se $s_j p_{ji} > s_i p_{ij}$, o mesmo raciocínio vale trocando i e j , de modo que novamente

$$s_i q_{ij} = s_j q_{ji}.$$

Portanto, a cadeia de Metropolis–Hastings satisfaz a condição de reversibilidade em relação a s . Como cadeias reversíveis são estacionárias com respeito à mesma distribuição, segue que s é a distribuição estacionária da cadeia com matriz de transição Q . \square

Observação 5 (Intuição via fluxo de partículas). *Podemos interpretar a cadeia de Metropolis–Hastings em termos de um sistema com n partículas que se movem entre os estados de acordo com as probabilidades de transição q_{ij} . No equilíbrio, a fração de partículas no estado i é s_i , de modo que existem aproximadamente ns_i partículas em i .*

Durante um passo da cadeia, cada partícula em i propõe se mover para j com probabilidade p_{ij} , e essa proposta é aceita com probabilidade a_{ij} . O fluxo esperado de partículas que saem de i e entram em j é, portanto,

$$ns_i p_{ij} a_{ij}.$$

Analogamente, o fluxo esperado de partículas indo de j para i é

$$ns_j p_{ji} a_{ji}.$$

A ideia central do algoritmo é ajustar as probabilidades de aceitação a_{ij} de modo que, no regime estacionário, esses dois fluxos se equilibrem:

$$ns_i p_{ij} a_{ij} = ns_j p_{ji} a_{ji},$$

ou, equivalentemente,

$$s_i q_{ij} = s_j q_{ji}.$$

Essa é exatamente a condição de reversibilidade (ou balanço detalhado).

A definição

$$a_{ij} = \min\left(\frac{s_j p_{ji}}{s_i p_{ij}}, 1\right)$$

garante esse equilíbrio:

- Se o fluxo proposto $s_i p_{ij}$ é maior que o fluxo de volta $s_j p_{ji}$, isso significa que, no estado atual, há “muitas partículas” tentando sair de i em direção a j , em comparação com o número que retorna de j para i . Para evitar que o sistema se desequilibre (isto é, que o estado j acumule partículas e o estado i esvazie), o algoritmo reduz a probabilidade de aceitação a_{ij} , permitindo que apenas uma fração dessas tentativas de saída seja efetivamente realizada. Esse “freio probabilístico” impede que o fluxo líquido entre i e j seja diferente de zero.
- Se o fluxo proposto é menor, isto é, há poucas partículas saindo de i em relação às que retornam de j , o sistema já tem menos movimento em direção a j . Nesse caso, todas as propostas são aceitas ($a_{ij} = 1$), pois não há risco de desequilíbrio: permitir todas as transições ajuda a compensar o déficit de fluxo, mantendo o equilíbrio entre os dois estados.

Com esse ajuste, o sistema tende a um equilíbrio dinâmico, no qual a taxa média de partículas indo de i para j é igual à taxa de partículas voltando de j para i . Não há acúmulo nem escoamento líquido de probabilidade entre os estados, e a igualdade

$$s_i q_{ij} = s_j q_{ji}$$

expressa precisamente essa situação de equilíbrio. Assim, a cadeia de Metropolis–Hastings é reversível em relação a s , e s é sua distribuição estacionária.

Exemplo 33 (Amostrador independente para a distribuição Beta). Vamos aplicar o algoritmo de Metropolis–Hastings para gerar amostras de uma distribuição Beta(a, b). Até aqui, introduzimos o método apenas para espaços de estado finitos, mas as mesmas ideias se aplicam a espaços contínuos.

Uma escolha simples para a cadeia de proposta é utilizar variáveis aleatórias independentes Unif(0, 1). Ou seja, o próximo estado proposto é sempre um valor $u \in (0, 1)$ gerado de forma independente do estado atual. Esse caso particular é chamado de amostrador independente (independence sampler), pois as propostas não dependem do ponto atual da cadeia.

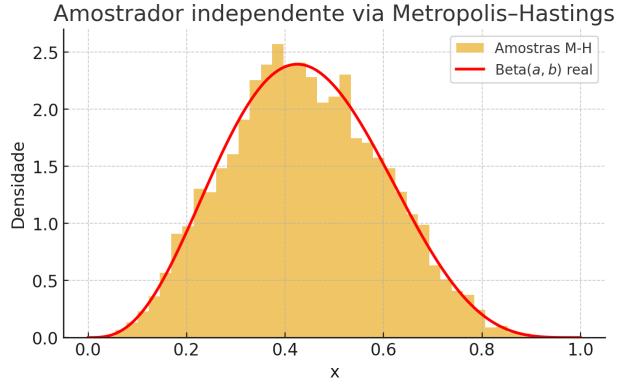
Seja W_0 um estado inicial arbitrário. A cadeia W_0, W_1, \dots é construída da seguinte forma:

1. No passo atual, suponha que a cadeia esteja em $W_n = w$.
2. Gere uma proposta $u \sim \text{Unif}(0, 1)$.
3. Calcule a probabilidade de aceitação

$$a(w, u) = \min\left(\frac{u^{a-1}(1-u)^{b-1}}{w^{a-1}(1-w)^{b-1}}, 1\right).$$

4. Com probabilidade $a(w, u)$, aceite a proposta e defina $W_{n+1} = u$; caso contrário, rejeite a proposta e mantenha $W_{n+1} = w$.

Observe que o algoritmo não requer o conhecimento da constante de normalização da densidade Beta(a, b), pois ela se cancela na razão entre as densidades no numerador e no denominador. Aqui, a densidade Beta(a, b) desempenha o papel de s (a distribuição estacionária desejada), enquanto a densidade uniforme Unif(0, 1) desempenha o papel de p_{ij} e p_{ji} , já que as propostas são sempre independentes do estado atual.



Após um número suficientemente grande de iterações, as variáveis W_n, W_{n+1}, \dots seguem aproximadamente a distribuição Beta(a, b). Essas amostras, entretanto, não são independentes: a cadeia gera uma sequência de variáveis correlacionadas, que oscilam em torno da distribuição alvo conforme o processo evolui.

Perceba que no algoritmo geral de Metropolis–Hastings, a probabilidade de aceitação é dada por

$$a(x, u) = \min\left(1, \frac{s(u) p(x | u)}{s(x) p(u | x)}\right),$$

onde $p(u | x)$ é a densidade da proposta, isto é, a probabilidade de propor o ponto u dado o estado atual x .

O termo $\frac{p(x|u)}{p(u|x)}$ está presente para corrigir possíveis assimetrias da distribuição de proposta. Por exemplo, se é mais fácil propor de x para u do que o contrário, o fator $\frac{p(x|u)}{p(u|x)}$ compensa essa diferença, garantindo que o fluxo médio de partículas entre x e u permaneça equilibrado:

$$s(x) p(u | x) a(x, u) = s(u) p(x | u) a(u, x).$$

No caso da simulação da distribuição Beta, utilizamos um independence sampler, em que as propostas são independentes do estado atual:

$$p(u | x) = p(u) = \text{Unif}(0, 1).$$

Consequentemente,

$$p(x | u) = p(x) = \text{Unif}(0, 1), \quad \text{e portanto} \quad \frac{p(x | u)}{p(u | x)} = 1.$$

O fator de correção se cancela, e a fórmula de aceitação se reduz para

$$a(x, u) = \min\left(1, \frac{s(u)}{s(x)}\right),$$

que é exatamente a expressão usada no caso da Beta.

Em resumo, o termo da proposta desaparece porque a distribuição de proposta é simétrica e independente do estado atual, de modo que não há assimetria a corrigir.

Observação 6 (Período de aquecimento (*burn-in*)). Mesmo quando uma cadeia de Markov possui s como distribuição estacionária, isso significa apenas que s é um estado de equilíbrio: se a cadeia for iniciada com distribuição s , ela permanecerá em s para sempre.

Na prática, porém, a cadeia é iniciada em um ponto fixo $X_0 = i$ ou segundo alguma distribuição inicial diferente de s . As primeiras iterações servem para que a cadeia se desloque gradualmente em direção ao equilíbrio, aproximando-se da distribuição estacionária. Durante esse período inicial, as distribuições de X_n ainda refletem o estado inicial e não representam bem o comportamento estacionário.

Esse intervalo é chamado de período de aquecimento, ou burn-in. Se denotarmos por $p^{(n)}$ o vetor de probabilidades no tempo n , temos

$$p^{(n+1)} = p^{(n)}Q.$$

Conforme n cresce, ocorre a convergência

$$p^{(n)} \longrightarrow s,$$

isto é, a distribuição da cadeia tende à distribuição estacionária s . Somente após essa fase de convergência é que as amostras podem ser consideradas representativas do regime estacionário.

Intuitivamente, podemos imaginar muitas cópias da cadeia evoluindo em paralelo. Inicialmente, há um acúmulo de partículas em certos estados e um déficit em outros. À medida que o tempo passa, os fluxos de transição entre estados se equilibram, até que a proporção de partículas em cada estado se estabilize segundo s . Descartar as primeiras amostras equivale a ignorar essa fase de ajuste até o equilíbrio.

7.4 Amostragem de Gibbs

A amostragem de Gibbs é um algoritmo de Monte Carlo empregado para gerar amostras aproximadas de uma distribuição conjunta. A ideia central é simples: atualizar sucessivamente uma variável de cada vez, amostrando-a de sua distribuição condicional dado o valor atual das demais. Esse método é particularmente útil quando as distribuições condicionais são fáceis de manipular e de amostrar diretamente.

Considere o caso de duas variáveis aleatórias discretas X e Y , com função de probabilidade conjunta

$$p_{X,Y}(x,y) = P(X = x, Y = y).$$

Desejamos construir uma cadeia de Markov (X_n, Y_n) cuja distribuição estacionária seja $p_{X,Y}$.

Existem duas versões principais do algoritmo de Gibbs, dependendo de como as variáveis são atualizadas: (1) o *Gibbs sistemático*, no qual as variáveis são atualizadas em ordem fixa e alternada; e (2) o *Gibbs aleatório*, no qual a variável a ser atualizada é escolhida aleatoriamente a cada iteração.

O procedimento pode ser descrito da seguinte forma:

1. No passo atual, suponha que a cadeia esteja em $(X_n, Y_n) = (x_n, y_n)$.
2. Gere um novo valor x_{n+1} a partir da distribuição condicional de X dado $Y = y_n$, isto é,

$$x_{n+1} \sim p(x \mid Y = y_n).$$

3. Em seguida, gere um novo valor y_{n+1} a partir da distribuição condicional de Y dado $X = x_{n+1}$:

$$y_{n+1} \sim p(y \mid X = x_{n+1}).$$

4. Atualize o estado da cadeia para $(X_{n+1}, Y_{n+1}) = (x_{n+1}, y_{n+1})$.

Repetindo esses passos indefinidamente, a cadeia (X_n, Y_n) converge para a distribuição estacionária $p_{X,Y}$.

Na versão aleatória, escolhe-se a cada iteração qual variável será atualizada, com probabilidades iguais. O procedimento segue:

1. No passo atual, suponha que a cadeia esteja em $(X_n, Y_n) = (x_n, y_n)$.

2. Escolha aleatoriamente qual componente será atualizado:

- com probabilidade $1/2$, atualize X ;
- com probabilidade $1/2$, atualize Y .

3. Se X for escolhido:

- (a) Gere $x_{n+1} \sim p(x | Y = y_n)$;
- (b) Defina $(X_{n+1}, Y_{n+1}) = (x_{n+1}, y_n)$.

4. Caso Y seja escolhido:

- (a) Gere $y_{n+1} \sim p(y | X = x_n)$;
- (b) Defina $(X_{n+1}, Y_{n+1}) = (x_n, y_{n+1})$.

Repetindo o processo, obtemos novamente uma cadeia cuja distribuição estacionária é $p_{X,Y}$.

O algoritmo de Gibbs se estende naturalmente para o caso de d variáveis aleatórias. Nesse caso, o estado da cadeia é um vetor $W_n = (W_n^{(1)}, \dots, W_n^{(d)})$. Em cada iteração:

1. Escolhe-se (de forma determinística ou aleatória) um índice $j \in \{1, \dots, d\}$;

2. Amostra-se o componente $W_n^{(j)}$ da distribuição condicional

$$W_{n+1}^{(j)} \sim p(w^{(j)} | w^{(-j)}),$$

onde $w^{(-j)}$ denota todos os outros componentes fixos;

3. Mantêm-se os demais componentes inalterados.

Observação 7. O amostrador de Gibbs pode ser interpretado como um caso especial do algoritmo de Metropolis–Hastings. Enquanto o Metropolis–Hastings requer uma distribuição proposta e uma etapa de aceitação ou rejeição, o Gibbs utiliza propostas que sempre são aceitas, pois cada amostra é retirada exatamente da distribuição condicional correta.

Teorema 9 (Gibbs aleatório como caso particular de Metropolis–Hastings). *O amostrador de Gibbs aleatório é um caso particular do algoritmo de Metropolis–Hastings, no qual toda proposta é sempre aceita. Em particular, isso implica que a distribuição estacionária do amostrador de Gibbs aleatório é exatamente a distribuição conjunta desejada.*

Demonstração. Apresentaremos a demonstração no caso bidimensional, embora o argumento se estenda naturalmente a qualquer número de dimensões.

Sejam X e Y variáveis aleatórias discretas cuja distribuição conjunta $p(x, y) = P(X = x, Y = y)$ é a distribuição estacionária que queremos obter.

O algoritmo de Metropolis–Hastings, no estado atual (x, y) , procede da seguinte forma: propõe um novo estado (x', y') segundo uma distribuição proposta $q((x', y') | (x, y))$ e aceita essa proposta com probabilidade

$$a((x, y), (x', y')) = \min \left\{ 1, \frac{p(x', y') q((x, y) | (x', y'))}{p(x, y) q((x', y') | (x, y))} \right\}.$$

No caso do Gibbs aleatório, a proposta consiste em escolher aleatoriamente uma das coordenadas e atualizá-la de acordo com sua distribuição condicional verdadeira. Mais precisamente:

1. Com probabilidade $1/2$, atualiza-se X a partir de $p(x' | Y = y)$, mantendo $Y' = y$.
2. Com probabilidade $1/2$, atualiza-se Y a partir de $p(y' | X = x)$, mantendo $X' = x$.

Vamos considerar o segundo caso, em que apenas Y é atualizado (o caso simétrico em que X é atualizado é análogo). Assim,

$$q((x, y') | (x, y)) = \frac{1}{2} p(y' | x) \quad \text{e} \quad q((x, y) | (x, y')) = \frac{1}{2} p(y | x).$$

Substituindo esses termos na fórmula de aceitação, obtemos:

$$a((x, y), (x, y')) = \min \left\{ 1, \frac{p(x, y') p(y | x)}{p(x, y) p(y' | x)} \right\}.$$

Como $p(x, y) = p(x) p(y | x)$, temos:

$$\frac{p(x, y') p(y | x)}{p(x, y) p(y' | x)} = \frac{p(x) p(y' | x) p(y | x)}{p(x) p(y | x) p(y' | x)} = 1.$$

Logo,

$$a((x, y), (x, y')) = 1.$$

Isto é, toda proposta é sempre aceita. Consequentemente, o algoritmo de Metropolis–Hastings, com essa escolha específica de distribuição proposta, coincide exatamente com o amostrador de Gibbs aleatório, e ambos têm a mesma distribuição estacionária $p(x, y)$. \square

Exemplo 34 (O problema da galinha e dos ovos). *Uma galinha põe um número N de ovos, onde $N \sim \text{Poisson}(\lambda)$. Cada ovo choca com probabilidade p , onde p é desconhecido e tem distribuição*

$$p \sim \text{Beta}(a, b).$$

Os parâmetros λ, a, b são conhecidos. O problema é que não observamos o número total de ovos N , apenas o número de ovos chocados, denotado por X . Nossa objetivo é estimar a esperança posterior

$$\mathbb{E}[p | X = x],$$

isto é, a média de p após observar x ovos chocados.

1. A distribuição de X dado p é (pense no porquê)

$$X \mid p \sim \text{Poisson}(\lambda p).$$

Logo, a densidade posterior de p é proporcional a

$$f(p \mid X = x) \propto P(X = x \mid p) f(p) \propto e^{-\lambda p} (\lambda p)^x p^{a-1} (1-p)^{b-1}.$$

Essa distribuição não tem forma fechada conhecida, o que dificulta a amostragem direta. Para contornar isso, introduzimos a variável latente N , correspondente ao número total de ovos postos.

2. Condisionalmente a $N = n$ e p , o número de ovos chocados segue

$$X \mid N = n, p \sim \text{Binomial}(n, p).$$

Assim, ao condicionar em N , recuperamos a conjugação Beta–Binomial:

$$p \mid X = x, N = n \sim \text{Beta}(x + a, n - x + b).$$

O fato de que essa forma condicional é simples motiva o uso da amostragem de Gibbs: alternaremos entre amostrar p dado N e N dado p .

3. Desejamos gerar amostras da distribuição conjunta de (p, N) condicionada a $X = x$. O algoritmo segue os seguintes passos:

(a) Faça suposições iniciais para p e N .

(b) Repita até a convergência:

i. **Atualização de p :**

$$p \mid X = x, N = n \sim \text{Beta}(x + a, n - x + b).$$

ii. **Atualização de N :** Seja $Y = N - X$ o número de ovos que não chocaram. Condisionalmente a p , o número de ovos não chocados segue

$$Y \mid p \sim \text{Poisson}(\lambda(1 - p)).$$

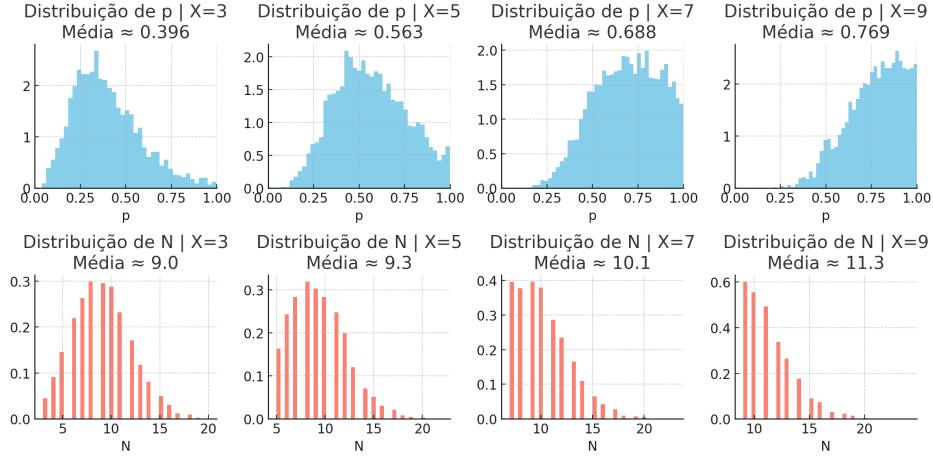
Assim, sorteamos $Y \sim \text{Poisson}(\lambda(1 - p))$ e definimos

$$N = X + Y.$$

4. Após muitas iterações, obtemos amostras $(p^{(1)}, N^{(1)}), (p^{(2)}, N^{(2)}), \dots$ extraídas aproximadamente de $f(p, N \mid X = x)$. A esperança posterior é então estimada pela média amostral:

$$\mathbb{E}(p \mid X = x) \approx \frac{1}{T} \sum_{t=1}^T p^{(t)}.$$

Os resultados obtidos por simulação do amostrador de Gibbs são resumidos na Tabela 4. Nas simulações, utilizou-se $\lambda = 10$ como valor esperado do número total de ovos postos, e um prior Beta(a, b) com $a = b = 1$, correspondendo a uma distribuição uniforme sobre $[0, 1]$ para o parâmetro p . Foram



considerados diferentes valores observados de ovos chocados $X \in \{3, 5, 7, 9\}$, enquanto os demais parâmetros permaneceram fixos.

X	$\mathbb{E}[p X = x]$	$\mathbb{E}[N X = x]$
3	0.40	9.0
5	0.56	9.3
7	0.69	10.1
9	0.77	11.3

Observa-se que, à medida que o número de ovos chocados X aumenta, a média posterior de p cresce de forma aproximadamente monotônica, variando de cerca de 0.4 para 0.77. Esse comportamento é exatamente o esperado, pois $X | p \sim \text{Poisson}(\lambda p)$: quanto maior o número de ovos chocados, maior deve ser a probabilidade de sucesso p .

Além disso, a média posterior de N também aumenta levemente com X , o que é coerente com o modelo $N = X + Y$, em que $Y | p \sim \text{Poisson}(\lambda(1 - p))$. Ou seja, observar mais ovos chocados implica, em média, um número total ligeiramente maior de ovos postos. Esses resultados confirmam que o amostrador de Gibbs captura corretamente a relação entre X , p e N , produzindo inferências consistentes com o modelo teórico.

Capítulo 8

Processos de Difusão

8.1 Movimento Browniano e SDE

A solução de uma *equação diferencial estocástica* (SDE, do inglês *stochastic differential equation*) é um **processo estocástico**. Um processo estocástico é uma coleção de variáveis aleatórias $X_t \in \mathbb{R}^d$ que evoluem ao longo do tempo $t \geq 0$. Embora cada X_t seja aleatória para um instante fixo, o interesse está em compreender como valores em tempos diferentes se relacionam — isto é, como X_{t+s} depende de X_t .

Um exemplo fundamental de processo estocástico é o **movimento browniano**, denotado por W_t . Ele é caracterizado por duas propriedades essenciais:

- **Incrementos normais:** os incrementos têm distribuição normal com variância proporcional ao intervalo de tempo:

$$W_{t+s} - W_t \sim \mathcal{N}(0, sI_d), \quad \text{para todo } t, s \geq 0.$$

- **Incrementos independentes:** para quaisquer $t_1 > t_2 > t_3$, os incrementos $W_{t_1} - W_{t_2}$ e $W_{t_2} - W_{t_3}$ são independentes.

Essas propriedades fazem do movimento browniano o modelo canônico de ruído contínuo, servindo como base para a definição das equações diferenciais estocásticas.

Podemos aproximar numericamente uma trajetória de movimento browniano discretizando o tempo. Se tomamos passos igualmente espaçados de tamanho $s > 0$, o incremento em cada passo é dado por

$$W_{t+s} = W_t + \sqrt{s} \varepsilon,$$

onde $\varepsilon \sim \mathcal{N}(0, I_d)$ é uma variável aleatória independente a cada passo. O parâmetro s representa o **tamanho do passo temporal** — isto é, o intervalo entre duas amostragens consecutivas do processo. O fator \sqrt{s} garante que a variância dos incrementos cresça linearmente com o tempo, como exige a definição de movimento browniano:

$$\text{Var}(W_{t+s} - W_t) = sI_d.$$

Essa relação mostra que quanto maior o intervalo s , maior a variabilidade esperada do incremento, refletindo a natureza difusiva do processo.

Para ilustrar, vejamos como podemos **simular numericamente uma trajetória de movimento browniano**. A ideia é construir uma sequência de valores $(W_0, W_s, W_{2s}, \dots, W_T)$ que satisfaça as propriedades do processo.

1. **Escolha dos parâmetros:** Defina o tempo total de simulação $T > 0$ e o número de passos n . O tamanho do passo será $s = T/n$.
2. **Inicialização:** Comece com $W_0 = 0$, que é a condição inicial típica do movimento browniano.
3. **Geração dos incrementos:** Para cada passo $k = 1, 2, \dots, n$, gere um ruído gaussiano independente

$$\varepsilon_k \sim \mathcal{N}(0, 1),$$

e compute o incremento

$$\Delta W_k = \sqrt{s} \varepsilon_k.$$

4. **Construção da trajetória:** Atualize o valor do processo de forma recursiva:

$$W_k = W_{k-1} + \Delta W_k.$$

O vetor (W_0, W_1, \dots, W_n) representa uma amostra discreta da trajetória de W_t no intervalo $[0, T]$.

O resultado é mostrado na Figura 8.1, que exibe duas trajetórias com diferentes tamanhos de passo s . Quanto menor o passo, mais suave e precisa é a aproximação da trajetória contínua de um movimento browniano.

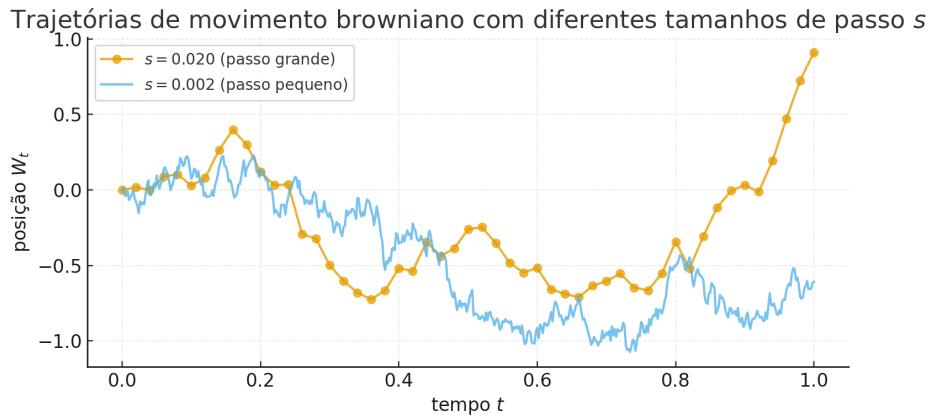


Figura 8.1: Simulação de duas trajetórias de movimento browniano com diferentes tamanhos de passo s .

Exercício 27. *Simule movimentos brownianos com diferente passos.*

Com base nesse ruído contínuo W_t , uma **equação diferencial estocástica** é uma equação que descreve a evolução de um processo X_t segundo

$$dX_t = f(X_t, t) dt + g(X_t, t) dW_t.$$

O primeiro termo, $f(X_t, t) dt$, representa a tendência média do movimento e é chamado de **drift**; o segundo termo, $g(X_t, t) dW_t$, modela a **difusão**, responsável pelas flutuações aleatórias.

Para obter uma intuição mais concreta sobre a dinâmica dessa equação, é útil pensar em sua forma *discretizada no tempo*. Consideremos pequenos intervalos de tempo $\Delta > 0$. O movimento browniano B_t possui incrementos

$$B_{t+\Delta} - B_t \sim \mathcal{N}(0, \Delta I_d),$$

e, além disso, esses incrementos são independentes para intervalos disjuntos. Assim, podemos representar o incremento como

$$B_{t+\Delta} - B_t = \sqrt{\Delta} \varepsilon_t,$$

onde $\varepsilon_t \sim \mathcal{N}(0, I_d)$ é uma variável aleatória independente a cada passo.

Substituindo esse termo na SDE

$$dX_t = f(X_t, t) dt + g(X_t, t) dB_t,$$

obtemos a versão discreta aproximada:

$$X_{t+\Delta} \approx X_t + f(X_t, t) \Delta + g(X_t, t) (B_{t+\Delta} - B_t).$$

Usando a expressão acima para o incremento browniano, isso se torna

$$X_{t+\Delta} \approx X_t + f(X_t, t) \Delta + g(X_t, t) \sqrt{\Delta} \varepsilon_t.$$

Essa equação descreve como o processo X_t evolui passo a passo: a cada intervalo Δ , há um deslocamento determinístico dado por $f(X_t, t) \Delta$ (o *drift*) e um deslocamento aleatório $g(X_t, t) \sqrt{\Delta} \varepsilon_t$ (a *difusão*).

Essa formulação é conhecida como o **esquema de Euler–Maruyama**, uma generalização estocástica do método de Euler para equações diferenciais ordinárias. À medida que $\Delta \rightarrow 0$, a sequência $X_{t+\Delta}$ converge, sob condições adequadas, para a solução contínua da SDE.

Exercício 28. Considere o esquema de Euler–Maruyama para simular uma equação diferencial estocástica (SDE) da forma

$$dX_t = f(X_t) dt + \sigma dB_t,$$

onde B_t é um movimento browniano padrão e $\sigma > 0$ controla a intensidade do ruído.

(a) Implemente a simulação de trajetórias para dois campos de drift diferentes:

$$f_1(x) = -x \quad e \quad f_2(x) = x - x^3.$$

(b) Gere várias trajetórias para cada caso, mantendo o mesmo valor de σ e do passo temporal Δ .

(c) Compare os comportamentos obtidos:

- Como o termo de drift influencia a dispersão das trajetórias?
- Em que sentido o caso $f_2(x) = x - x^3$ pode ser interpretado como um sistema com dois estados de equilíbrio?

(d) Plote em um mesmo gráfico uma trajetória com drift e outra sem drift (isto é, $f(x) = 0$) para visualizar a diferença qualitativa entre difusão pura e dinâmica com força restauradora.

8.2 Equação de Fokker–Planck

Considere um processo de difusão governado pela SDE geral

$$dX_t = f(X_t, t) dt + g(X_t, t) dB_t,$$

onde f é o campo de *drift* e g o coeficiente de *difusão*. A função g controla como o ruído atua sobre cada componente do sistema — por exemplo, se g é uma matriz, o ruído pode ter intensidades e correlações diferentes em cada direção.

A densidade de probabilidade $p(x, t)$ associada a X_t satisfaz a **equação de Fokker–Planck** (também chamada de equação de Kolmogorov para frente):

$$\frac{\partial p}{\partial t} = -\nabla \cdot (f(x, t) p(x, t)) + \frac{1}{2} \nabla \cdot ((\nabla \cdot (g g^\top))(x, t) p(x, t)).$$

O primeiro termo representa o transporte determinístico da densidade pelo campo de drift, enquanto o segundo termo descreve a difusão espacial causada pelo ruído multiplicativo $g(x, t) dB_t$.

No caso em que o ruído é *isotrópico e constante*, isto é, $g(x, t) = \sqrt{2D} I_d$, a equação se simplifica para

$$\frac{\partial p}{\partial t} = -\nabla \cdot (f(x) p(x, t)) + D \Delta p(x, t),$$

onde $D > 0$ é o coeficiente de difusão e Δ é o operador Laplaciano.

Essa forma mostra claramente a dualidade entre drift e difusão:

- o termo $-\nabla \cdot (f p)$ concentra ou transporta massa segundo o fluxo determinístico;
- o termo $D \Delta p$ espalha a densidade, suavizando descontinuidades e representando o efeito do ruído.

A equação de Fokker–Planck pode ainda ser escrita como uma **equação de continuidade**:

$$\frac{\partial p}{\partial t} = -\nabla \cdot J, \quad J = f(x) p(x, t) - D \nabla p(x, t),$$

onde J é o *fluxo de probabilidade*. O primeiro termo de J corresponde ao transporte devido ao drift, e o segundo termo é o fluxo difusivo induzido pela variação espacial da densidade.

Aqui, o operador $\nabla \cdot J$ representa o **divergente** do campo de fluxo J , isto é, a taxa líquida de probabilidade que sai (ou entra) de uma pequena região do espaço. De forma intuitiva:

- se $\nabla \cdot J > 0$, há mais probabilidade saindo do que entrando — a densidade $p(x, t)$ diminui localmente;
- se $\nabla \cdot J < 0$, há mais probabilidade entrando do que saindo — a densidade aumenta naquela região.

Portanto, a equação de continuidade expressa o *princípio de conservação de probabilidade*: a densidade muda no tempo apenas devido ao fluxo de probabilidade atravessando as fronteiras das regiões do espaço.

8.3 Amostragem de Langevin

A ideia central da *amostragem de Langevin* é usar uma dinâmica estocástica contínua no tempo cuja distribuição estacionária coincide com uma densidade-alvo $p(x)$. Queremos, portanto, construir uma SDE cuja solução X_t satisfaça

$$\lim_{t \rightarrow \infty} \mathcal{L}(X_t) = p(x),$$

onde $\mathcal{L}(X_t)$ denota a lei (ou distribuição) do processo no tempo t .

Uma escolha natural é definir o drift como o gradiente do logaritmo da densidade:

$$dX_t = \nabla \log p(X_t) dt + \sqrt{2} dB_t.$$

Essa SDE é conhecida como **dinâmica de Langevin** (ou *overdamped Langevin equation*). O termo determinístico $\nabla \log p(X_t)$ empurra as partículas em direção às regiões de maior probabilidade da distribuição, enquanto o ruído gaussiano $\sqrt{2} dB_t$ garante que o processo explore todo o espaço de estados.

Se tomarmos $p_\infty(x) = p(x)$, queremos verificar que essa escolha satisfaz a condição estacionária da equação de Fokker-Planck:

$$\nabla \cdot (\nabla \log p(x) p(x)) = \Delta p(x).$$

Recordemos que o operador ∇ representa o *gradiente*, e que o operador $\nabla \cdot$ representa o *divergente*. Aplicar o divergente a um gradiente produz o **Laplaciano**, denotado por Δ :

$$\Delta p(x) = \nabla \cdot (\nabla p(x)).$$

Em uma dimensão, o Laplaciano reduz-se simplesmente à segunda derivada $\frac{d^2 p}{dx^2}$.

Pela regra da cadeia, sabemos que

$$\nabla \log p(x) = \frac{\nabla p(x)}{p(x)}.$$

Multiplicando ambos os lados por $p(x)$, obtemos

$$\nabla p(x) = p(x) \nabla \log p(x).$$

Substituímos essa identidade no lado direito da equação de equilíbrio:

$$\Delta p(x) = \nabla \cdot (\nabla p(x)) = \nabla \cdot (p(x) \nabla \log p(x)).$$

Portanto, o lado direito e o lado esquerdo da equação estacionária são idênticos:

$$\nabla \cdot (\nabla \log p(x) p(x)) = \nabla \cdot (p(x) \nabla \log p(x)) = \Delta p(x).$$

Essa relação entre a SDE de Langevin e a Fokker-Planck mostra que o processo preserva a densidade $p(x)$ no equilíbrio. Na prática, a amostragem de Langevin consiste em discretizar essa SDE (via o método de Euler–Maruyama) e usar as iterações resultantes para gerar amostras aproximadamente distribuídas segundo $p(x)$.

Exercício 29. Considere a densidade alvo $p(x) = \mathcal{N}(x; 0, 1)$, isto é,

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}.$$

Queremos usar a dinâmica de Langevin para gerar amostras dessa distribuição:

$$dX_t = \nabla \log p(X_t) dt + \sqrt{2} dB_t.$$

- Derive o gradiente de $\log p(x)$. Mostre que

$$\nabla \log p(x) = -x.$$

Observe que o termo constante $-\frac{1}{2} \log(2\pi)$ desaparece na derivada. Discuta o motivo pelo qual o termo normalizador da densidade não afeta o gradiente.

- Implemente a dinâmica de Langevin:

$$X_{t+\Delta} = X_t - X_t \Delta + \sqrt{2\Delta} \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, 1).$$

Use $\Delta = 0.01$ e inicialize $X_0 \sim \mathcal{N}(0, 1)$. Gere trajetórias e verifique que o histograma das amostras converge para a densidade $p(x)$.

- Versão sem difusão: Repita a simulação removendo o termo aleatório,

$$X_{t+\Delta} = X_t - X_t \Delta.$$

O processo converge para qual valor? Explique o papel do ruído na manutenção da variabilidade da amostra.

- Interpretação: Analise a dinâmica obtida. O termo $-X_t \Delta$ atua como uma força de retorno à origem (um poço de potencial), enquanto o termo aleatório $\sqrt{2\Delta} \varepsilon_t$ impede que o processo colapse nesse ponto, mantendo a distribuição estacionária $\mathcal{N}(0, 1)$.

Note que o termo normalizador $\frac{1}{\sqrt{2\pi}}$ não influencia o gradiente $\nabla \log p(x)$, pois é constante. Assim, o comportamento da dinâmica de Langevin depende apenas da forma relativa de $p(x)$, e não da sua escala global. Esse fato é fundamental: em métodos baseados em gradientes (como Langevin e Hamiltonian Monte Carlo), apenas a derivada do logaritmo da densidade importa.

Exercício 30. Considere o problema de gerar amostras de uma **mistura bimodal de normais**:

$$p(x) = \frac{1}{2} \mathcal{N}(x; -3, 1^2) + \frac{1}{2} \mathcal{N}(x; 3, 1^2).$$

O objetivo é usar a dinâmica de Langevin para aproximar amostras dessa distribuição:

$$dX_t = \nabla \log p(X_t) dt + \sqrt{2} dB_t.$$

- Derive o gradiente de $\log p(x)$. Expresse $\nabla \log p(x)$ em termos das densidades de cada componente da mistura e de suas médias.

2. Implemente a dinâmica de Langevin:

$$X_{t+\Delta} = X_t + \nabla \log p(X_t) \Delta + \sqrt{2\Delta} \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, 1).$$

Use $\Delta = 0.02$ e initialize $X_0 \sim \mathcal{N}(0, 5)$. Gere trajetórias e compare o histograma das amostras obtidas com a densidade-alvo $p(x)$.

3. Versão sem difusão: Repita a simulação removendo o termo aleatório,

$$X_{t+\Delta} = X_t + \nabla \log p(X_t) \Delta,$$

e observe o comportamento do processo. Ele converge para algum ponto específico? O que muda em relação à versão estocástica?

4. Influência da condição inicial: Inicialize o processo com $X_0 = 1$ e repita o experimento. O processo é capaz de explorar os dois modos da mistura? Discuta por que a escolha de X_0 pode influenciar a região da distribuição onde o processo permanece por mais tempo.

Exercício 31. Considere novamente a mistura bimodal

$$p(x) = \frac{1}{2} \mathcal{N}(x; -3, 1^2) + \frac{1}{2} \mathcal{N}(x; 3, 1^2),$$

e a dinâmica de Langevin

$$dX_t = \nabla \log p(X_t) dt + \sqrt{2} dB_t.$$

1. Simule uma **trajetória longa** da dinâmica, iniciando em $X_0 \sim \mathcal{N}(0, 5)$. Plote o histograma das amostras e o gráfico de posição $t \mapsto X_t$.

Discuta o que está acontecendo no processo e como isso se liga com a matéria de MCMC.

2. Repita o experimento com **várias trajetórias independentes** cada uma com condição inicial diferente, e compare os histogramas dos dois métodos.

8.4 Denoising Score Matching

Como vimos, a dinâmica de Langevin

$$dX_t = \frac{1}{2} \nabla_x \log p(X_t) dt + dW_t$$

possui como distribuição estacionária a própria densidade alvo $p(x)$. De fato, pela equação de Fokker–Planck associada,

$$\frac{\partial p_t(x)}{\partial t} = -\nabla_x \cdot \left(\frac{1}{2} \nabla_x \log p(x) p_t(x) \right) + \frac{1}{2} \Delta p_t(x),$$

vemos que $p_t(x) = p(x)$ é uma solução estacionária.

O problema, entretanto, é que na prática não conhecemos $p(x)$ de forma explícita e portanto não temos acesso ao seu gradiente $\nabla_x \log p(x)$, o *score*, que aparece diretamente no termo de drift da dinâmica de Langevin. O que dispomos, em geral, são apenas amostras independentes

$$x_1, \dots, x_n \sim p(x),$$

e queremos, a partir delas, construir um estimador para o score ou, de forma equivalente, um campo vetorial $s_\theta(x) \approx \nabla_x \log p(x)$ que possa ser usado em dinâmicas como a de Langevin para simular a distribuição alvo.

O objetivo do *Denoising Score Matching* (DSM) é aprender o *score* de uma distribuição de probabilidade desconhecida $p(x)$, definido por

$$s^*(x) = \nabla_x \log p(x).$$

A ideia central do DSM é transformar o problema de estimar o score em uma tarefa de regressão supervisionada. Para isso, partimos de amostras $x \sim p(x)$ e adicionamos ruído gaussiano $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, obtendo dados corrompidos

$$\tilde{x} = x + \varepsilon.$$

O método define como alvo de regressão

$$t = \frac{x - \tilde{x}}{\sigma^2},$$

e treina um modelo $s_\theta(\tilde{x}, \sigma)$ para aproximar esse alvo a partir do dado corrompido. A função de perda considerada é

$$\mathcal{L}(\theta) = \mathbb{E}_{x \sim p} \mathbb{E}_{\varepsilon \sim \mathcal{N}(0, \sigma^2)} [\|s_\theta(\tilde{x}, \sigma) - t\|^2].$$

Para entender por que esse procedimento funciona, começamos observando que a variável corrompida \tilde{x} possui densidade

$$q_\sigma(\tilde{x}) = \int p(x) \varphi_\sigma(\tilde{x} - x) dx,$$

onde φ_σ é a densidade gaussiana $\mathcal{N}(0, \sigma^2)$. Portanto, q_σ é a convolução de p com uma gaussiana. O score dessa densidade suavizada é

$$\nabla_{\tilde{x}} \log q_\sigma(\tilde{x}) = \frac{\nabla_{\tilde{x}} q_\sigma(\tilde{x})}{q_\sigma(\tilde{x})}.$$

Derivando sob o sinal de integral,

$$\nabla_{\tilde{x}} q_\sigma(\tilde{x}) = \int p(x) \nabla_{\tilde{x}} \varphi_\sigma(\tilde{x} - x) dx,$$

e como

$$\nabla_{\tilde{x}} \varphi_\sigma(\tilde{x} - x) = -\frac{\tilde{x} - x}{\sigma^2} \varphi_\sigma(\tilde{x} - x),$$

temos

$$\nabla_{\tilde{x}} q_\sigma(\tilde{x}) = \int \frac{x - \tilde{x}}{\sigma^2} p(x) \varphi_\sigma(\tilde{x} - x) dx.$$

Dividindo por $q_\sigma(\tilde{x})$ e lembrando que a densidade conjunta de (x, \tilde{x}) é dada por $p(x, \tilde{x}) = p(x) \varphi_\sigma(\tilde{x} - x)$, podemos reescrever o integrando em termos da distribuição condicional de x dado \tilde{x} :

$$p(x | \tilde{x}) = \frac{p(x) \varphi_\sigma(\tilde{x} - x)}{q_\sigma(\tilde{x})}.$$

Substituindo essa expressão na fórmula do gradiente, obtemos

$$\nabla_{\tilde{x}} \log q_\sigma(\tilde{x}) = \int \frac{x - \tilde{x}}{\sigma^2} p(x | \tilde{x}) dx,$$

o que mostra que o score de q_σ é precisamente a esperança condicional

$$\nabla_{\tilde{x}} \log q_\sigma(\tilde{x}) = \mathbb{E} \left[\frac{x - \tilde{x}}{\sigma^2} \mid \tilde{x} \right].$$

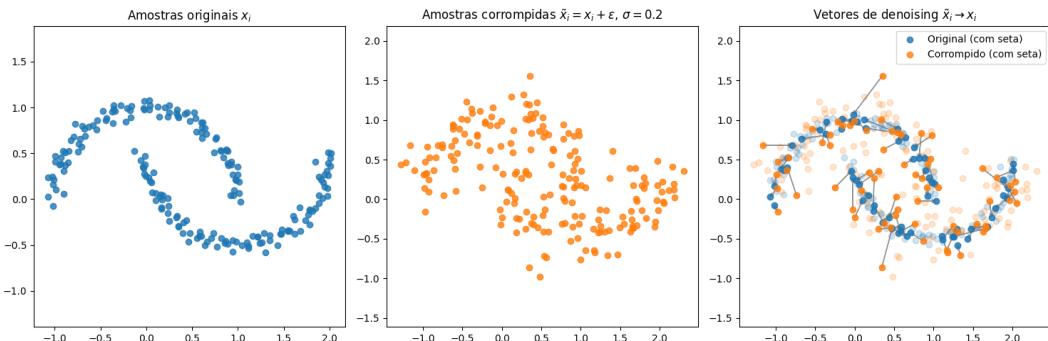
Assim, mostramos que a esperança condicional do alvo t dado \tilde{x} é exatamente o score da densidade suavizada q_σ . Como a perda quadrática é minimizada quando $s_\theta(\tilde{x}, \sigma) = \mathbb{E}[t | \tilde{x}]$, segue que o estimador ótimo do DSM é

$$s^*(\tilde{x}, \sigma) = \nabla_{\tilde{x}} \log q_\sigma(\tilde{x}).$$

Note que, fixado o valor de σ , o DSM está aprendendo o *score* não da densidade original $p(x)$, mas da versão *suavizada* $q_\sigma(x) = (p * \varphi_\sigma)(x)$, obtida ao convoluir p com uma gaussiana de variância σ^2 . Ou seja, estamos removendo detalhes finos da distribuição original e focando nas variações mais suaves de p .

O campo vetorial $\nabla_x \log q_\sigma(x)$ aponta na direção em que a densidade suavizada mais cresce: ele indica como deveríamos mover uma partícula que foi corrompida pelo ruído para restaurar a estrutura de p .

Assim, quando treinamos $s_\theta(x, \sigma)$ para aproximar $\mathbb{E}[(x - \tilde{x})/\sigma^2 | \tilde{x} = x]$, estamos aprendendo como “desfazer” o ruído gaussiano de nível σ . Por isso o nome *denoising score matching*.



8.4.1 Estimando o score na prática

Na dedução anterior vimos que, para um valor fixo de σ , o modelo $s_\theta(\tilde{x}, \sigma)$ treinado com o *Denoising Score Matching* aprende o score da densidade suavizada $q_\sigma(x)$. Na prática, entretanto, não queremos apenas um único valor de σ , mas uma coleção de níveis de ruído que permitam capturar a estrutura de $p(x)$ em diferentes escalas.

Usualmente escolhemos uma *escala geométrica* de valores

$$\sigma_k = \sigma_{\min} \left(\frac{\sigma_{\max}}{\sigma_{\min}} \right)^{k/(K-1)}, \quad k = 0, \dots, K-1,$$

de modo que os níveis de ruído cubram uniformemente várias ordens de magnitude, indo de ruído forte (σ_{\max}) a ruído fraco (σ_{\min}). Para cada amostra x_i e para cada σ_k , podemos gerar várias versões corrompidas

$$\tilde{x}_{i,j,k} = x_i + \varepsilon_{i,j,k}, \quad \varepsilon_{i,j,k} \sim \mathcal{N}(0, \sigma_k^2 I),$$

e calcular os respectivos alvos de regressão

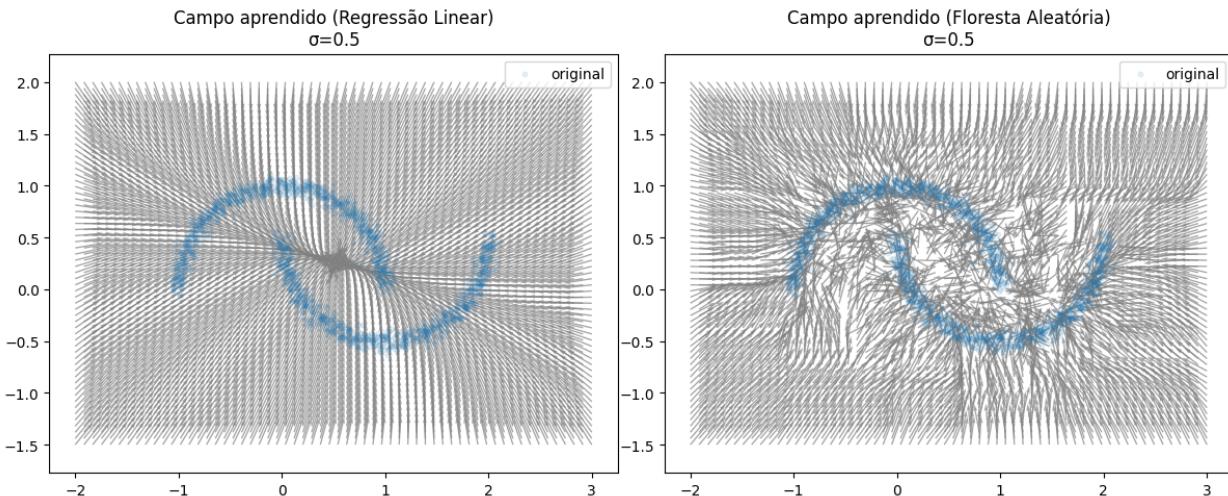
$$t_{i,j,k} = -\frac{\varepsilon_{i,j,k}}{\sigma_k^2}.$$

Dessa forma, podemos *expandir o conjunto de dados* criando várias amostras supervisionadas $(\tilde{x}_{i,j,k}, \sigma_k, t_{i,j,k})$, que descrevem, para diferentes níveis de ruído, a direção de denoising a ser aprendida.

O próximo passo é ajustar um modelo de regressão s_θ que receba como entrada o ponto corrompido \tilde{x} e o valor de σ , e aprenda a prever o vetor t . Esse modelo pode ser uma rede neural, mas também algo mais simples, como uma árvore de decisão ou um modelo de regressão não linear. O objetivo é que, após o treinamento, o campo aprendido satisfaça aproximadamente

$$s_\theta(\tilde{x}, \sigma) \approx \nabla_{\tilde{x}} \log q_\sigma(\tilde{x}),$$

fornecendo uma boa estimativa do score da densidade suavizada. Com isso, temos um modelo capaz de indicar, para cada ponto corrompido, em que direção ele deve se mover para recuperar regiões de alta densidade de $p(x)$.



Em resumo, o treinamento do DSM pode ser realizado seguindo os seguintes passos:

1. Definir uma escala geométrica de valores de ruído $\sigma_1, \dots, \sigma_K$, que vai do ruído mais forte (σ_{\max}) ao mais fraco (σ_{\min}). Essa escala define o quanto cada amostra será corrompida.
2. Para cada amostra x_i do conjunto de dados e para cada nível de ruído σ_k , gerar algumas versões corrompidas $\tilde{x}_{i,j,k} = x_i + \varepsilon_{i,j,k}$, com $\varepsilon_{i,j,k} \sim \mathcal{N}(0, \sigma_k^2 I)$. Isso aumenta o tamanho do conjunto de treinamento e ajuda o modelo a aprender a remover diferentes intensidades de ruído.

3. Calcular o alvo de regressão para cada par $(\tilde{x}_{i,j,k}, \sigma_k)$ como $t_{i,j,k} = -\varepsilon_{i,j,k}/\sigma_k^2$, que representa a direção na qual a amostra corrompida deve ser movida para retornar à distribuição original.
4. Montar um conjunto de dados supervisionado formado por pares de entrada e saída $([\tilde{x}_{i,j,k}, \sigma_k], t_{i,j,k})$. O valor de σ_k é incluído como uma *feature* adicional para indicar o nível de ruído daquela amostra.
5. Ajustar um modelo de regressão — que pode ser simples, como uma Floresta Aleatória — usando essas amostras expandidas. O modelo deve aprender a prever t a partir de $[\tilde{x}, \sigma]$.
6. Após o treinamento, o modelo resultante $s_\theta(\tilde{x}, \sigma)$ fornece uma aproximação do campo de score $\nabla_{\tilde{x}} \log q_\sigma(\tilde{x})$, indicando para cada ponto corrompido em qual direção ele deve se mover para se aproximar de regiões de alta densidade de $p(x)$.

Exercício 32. Neste exercício, vamos implementar o treinamento de um modelo de Denoising Score Matching (DSM) em um conjunto de dados sintético. O objetivo é aprender o campo de score $s_\theta(\tilde{x}, \sigma)$ a partir de amostras corrompidas, conforme discutido em aula.

1. Gere um conjunto de dados bidimensional usando a função `make_moons` abaixo.
2. Construa uma escala geométrica de valores de ruído usando a função `geometric_sigmas`.
3. Para cada valor de σ , corrompa as amostras adicionando ruído gaussiano $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$, e calcule o alvo $t = -\varepsilon/\sigma^2$.
4. Monte um conjunto de dados supervisionado contendo como entrada o par $[\tilde{x}, \sigma]$ e como saída o vetor t .
5. Treine um modelo de regressão à sua escolha (por exemplo, regressão linear, rede neural, ou floresta aleatória) para aprender a mapear $[\tilde{x}, \sigma] \mapsto t$.
6. Fixe um valor de σ e visualize o campo aprendido sobre uma grade bidimensional de pontos, comparando visualmente os resultados de diferentes modelos.

Capítulo 9

Estratégias para acelerar códigos em Python

9.1 Profiling com cProfile

Antes de otimizar, é essencial medir onde o tempo realmente está sendo gasto. O módulo `cProfile`, da biblioteca padrão do Python, permite gerar um perfil de execução mostrando quantas vezes cada função foi chamada e quanto tempo ela consumiu.

O uso mais simples é direto pelo terminal, aplicando o profiler a um script:

```
1 # Executa o script inteiro e mostra estatísticas
2 python -m cProfile meu_script.py
3
4 # Salva os resultados em arquivo para análise posterior
5 python -m cProfile -o saida.prof meu_script.py
```

Rodando pelo terminal

O arquivo gerado pode ser inspecionado com o módulo `pstats`, que permite ordenar e filtrar resultados:

```
1 python -m pstats saida.prof
2 # Comandos úteis no prompt do pstats:
3 #   sort time      (ordena pelo tempo interno da função)
4 #   sort cumtime   (ordena pelo tempo acumulado)
5 #   stats 20       (mostra as 20 funções mais custosas)
6 #   callers func   (quem chama 'func')
7 #   callees func   (quem 'func' chama)
```

Explorando com `pstats` (terminal)

Também é possível usar `cProfile` dentro do código, o que facilita em notebooks ou quando queremos medir apenas um trecho específico:

```
1 import cProfile, pstats, io
2
3 pr = cProfile.Profile()
```

```

4 pr.enable()
5
6 # --- código a ser medido ---
7 resultado = algoritmo_pesado()
8 # -----
9
10 pr.disable()
11 s = io.StringIO()
12 ps = pstats.Stats(pr, stream=s).sort_stats("cumtime")
13 ps.print_stats(10)    # mostra as 10 funções mais custosas
14 print(s.getvalue())

```

Usando cProfile dentro do código

As duas métricas principais são:

- **time**: tempo gasto apenas dentro da função, sem contar chamadas internas.
- **cumtime**: tempo acumulado, incluindo todas as funções chamadas.

Em geral, começa-se ordenando por `cumtime` para encontrar o caminho mais caro da execução. Depois, olhar o `time` ajuda a identificar funções individuais que valem otimização.

A seguir montamos um experimento simples para evidenciar como o `cProfile` ajuda a localizar gargalos: comparamos uma multiplicação de matrizes feita de forma ingênuia em Python (três laços) com a versão vetorizada do NumPy (delegada à BLAS).

O código abaixo implementa as duas versões e usa uma função auxiliar para rodar o profiler em cada uma delas, exibindo as funções mais custosas. O script pode ser salvo como `profile_matmul.py`.

```

1 import numpy as np
2 import math
3 import cProfile, pstats, io
4 import time
5
6 # Versão ingenua: 3 loops em Python
7 def matmul_naive(A, B):
8     n, m = A.shape
9     m2, p = B.shape
10    assert m == m2
11    C = np.zeros((n, p))
12    for i in range(n):
13        for j in range(p):
14            s = 0.0
15            for k in range(m):
16                s += A[i, k] * B[k, j]
17            C[i, j] = s
18    return C
19

```

```

20 # Versao NumPy (vetorizada/BLAS)
21 def matmul_numpy(A, B):
22     return A @ B
23
24 def profile_func(func, *args, top=15):
25     pr = cProfile.Profile()
26     pr.enable()
27     t0 = time.perf_counter()
28     result = func(*args)
29     t1 = time.perf_counter()
30     pr.disable()
31     s = io.StringIO()
32     ps = pstats.Stats(pr, stream=s).sort_stats("cumtime")
33     ps.print_stats(top)
34     print(f"\n>>> Tempo total (parede): {t1 - t0:.3f} s\n")
35     return s.getvalue()
36
37 A = np.random.rand(n, n)
38 B = np.random.rand(n, n)
39
40 print("==== Profiling matmul_naive ====")
41 out_naive = profile_func(matmul_naive, A, B)
42 print(out_naive)
43
44 print("==== Profiling matmul_numpy ====")
45 out_np = profile_func(matmul_numpy, A, B)
46 print(out_np)

```

Esse script pode ser executado normalmente com `python profile_matmul.py`. Outra forma é rodar o profiler diretamente no terminal, usando `python -m cProfile -o saída.prof profile_matmul.py`. Nesse caso o resultado fica salvo em `saída.prof`, e podemos explorá-lo depois com o módulo `pstats` de forma interativa, usando comandos como `sort cumtime`, `stats 20` ou `callers matmul_naive`.

Rodando a versão ingênuo, a saída típica mostra que praticamente todo o tempo foi consumido dentro de `matmul_naive`:

```

1      7 function calls in 8.532 seconds
2
3 Ordered by: cumulative time
4      ncalls  tottime  percall  cumtime  percall filename:lineno(function)
5          1    8.523    8.523    8.523    8.523 profile_matmul.py:11(
6              matmul_naive)
7          1    0.009    0.009    0.009    0.009 {built-in method builtins.
8                  print}
9          ...

```

Ao comparar com a versão vetorizada, vemos que a execução termina em milésimos de segundo, com o tempo todo acumulado em `matmul_numpy`:

```

1      7 function calls in 0.020 seconds
2
3
4 Ordered by: cumulative time
5 ncalls  tottime  percall  cumtime  percall filename:lineno(function)
6       1    0.019    0.019    0.019    0.019 profile_matmul.py:28(
7         matmul_numpy)
8
9 ...

```

Os números exatos variam conforme o tamanho das matrizes e a biblioteca BLAS instalada, mas o padrão é claro: a implementação ingênuo em Python puro consome segundos de CPU, enquanto a versão NumPy é milhares de vezes mais rápida.

As colunas do profiler têm significados diferentes. O campo `ncalls` mostra o número de chamadas à função. O `tottime` corresponde ao tempo gasto apenas dentro da função, sem contar chamadas internas. Já o `cumtime` indica o tempo acumulado incluindo funções chamadas dentro dela. Em geral, ordenar por `cumtime` ajuda a encontrar o caminho mais custoso da execução, enquanto olhar para `tottime` revela funções “folha” particularmente lentas.

Quando esse mesmo código é rodado em um notebook Jupyter, o output tende a ficar mais “poluído”, aparecendo referências a `asyncio`, `zmq` e outros componentes do kernel. Isso acontece porque o profiler mede tudo o que roda no processo, não apenas a nossa função. Para uma visão limpa e didática, vale a pena executar o script direto no terminal.

9.2 Paralelização com `joblib.Parallel`

A biblioteca `joblib` fornece uma forma simples de paralelizar *loops* *embaraçosamente paralelos* em Python, isto é, situações em que várias tarefas independentes podem ser executadas ao mesmo tempo. A ideia básica é escrever um laço `for` como uma compreensão preguiçosa de chamadas a uma função via `delayed`, e despachar essas tarefas para `Parallel`, que se encarrega de distribuí-las entre diferentes trabalhadores.

```

1 from joblib import Parallel, delayed
2 from math import sqrt
3
4 # aplicar sqrt a 0^2, 1^2, ..., 9^2 em paralelo
5 res = Parallel(n_jobs=4)(
6     delayed(sqrt)(i**2)
7     for i in range(10)
8 )

```

Receita de bolo

No exemplo acima, o parâmetro `n_jobs` define quantos trabalhadores serão usados (tipicamente o número de CPUs lógicas da máquina). A função `delayed` apenas empacota a chamada para que ela possa ser enviada a um worker, enquanto `Parallel` recolhe todas as tarefas e coordena sua execução.

Uma forma intuitiva de entender esse mecanismo é pensar em uma cozinha: se temos apenas um cozinheiro (um `for` sequencial), cada prato é preparado do início ao fim antes do próximo

começar. Já com vários cozinheiros (workers), cada um recebe um prato e trabalha nele independentemente, de modo que vários ficam prontos ao mesmo tempo. Essa estratégia funciona muito bem, mas há alguns cuidados: se uma tarefa demora muito enquanto outras são rápidas, pode haver desequilíbrio entre os workers; por outro lado, se existem milhares de tarefas minúsculas, o custo de despachá-las pode ser maior que o ganho da paralelização. Para reduzir esse problema, o joblib agrupa chamadas em lotes (*batching*), enviando várias de uma vez só.

Outro detalhe importante está no backend usado. Em Python, o *Global Interpreter Lock* (GIL) impede que várias threads executem código Python puro ao mesmo tempo. Por isso, o backend padrão (loky) cria processos separados, que contornam o GIL e escalam bem em cálculos pesados. Já o backend threading mantém as tarefas no mesmo processo, sendo útil em funções que passam a maior parte do tempo esperando I/O ou que já liberam o GIL (como operações NumPy). Existe ainda o multiprocessing, mas o locy tende a ser mais robusto.

```

1 # Uso de threads porque a função processa_io
2 # passa a maior parte do tempo esperando rede.
3 res = Parallel(n_jobs=8, backend="threading")(
4     delayed(processa_io)(u) for u in urls
5 )

```

Exemplo com I/O

Em resumo: use locy (padrão) para tarefas CPU-bound, threading para tarefas I/O-bound, e sempre ajuste o número de jobs de acordo com o hardware disponível. Paralelizar acelera muito, mas nem sempre compensa: quando as tarefas são pequenas demais, o overhead pode superar o benefício.

Um exemplo clássico de tarefa CPU-bound é calcular números primos ou executar operações pesadas de álgebra linear. Nesses casos, vale usar o backend padrão:

```

1 from joblib import Parallel, delayed
2 import math
3
4 def eh_primo(n):
5     for i in range(2, int(math.sqrt(n))+1):
6         if n % i == 0:
7             return False
8     return True
9
10 nums = range(10**6, 10**6+1000)
11 res = Parallel(n_jobs=4)(delayed(eh_primo))(n) for n in nums)

```

Exemplo CPU-bound

Aqui, cada worker testa um conjunto de números independentemente. Quanto mais núcleos disponíveis, mais rápido o processamento.

Já um exemplo I/O-bound seria baixar várias páginas da web. Cada tarefa fica a maior parte do tempo esperando a rede, e usar processos separados não traz vantagem; nesse caso o backend threading é mais leve:

```

1 import requests
2 urls = ["https://httpbin.org/delay/1"] * 20
3
4 def baixa(url):
5     return requests.get(url).status_code
6
7 res = Parallel(n_jobs=8, backend="threading")(
8     delayed(baixa)(u) for u in urls
9 )

```

Exemplo I/O-bound

Se cada requisição demora cerca de 1 segundo, com 8 threads as 20 requisições terminam em poucos segundos, em vez de mais de 20.

Por fim, um caso em que a paralelização atrapalha é quando as tarefas são rápidas demais, por exemplo calcular o quadrado de números pequenos:

```

1 def quadrado(n):
2     return n*n
3
4 nums = range(1000)
5 res = Parallel(n_jobs=4)(delayed(quadrado)(n) for n in nums)

```

Exemplo de overhead

Aqui o custo de organizar as tarefas, mandar para os workers e reunir os resultados é maior do que simplesmente rodar um `for` sequencial. Nesse cenário, a paralelização pode ser mais lenta.

9.3 Compilação Just-In-Time com Numba

Numba é um compilador JIT (Just-In-Time) para Python focado em acelerar código numérico. Ele “traduz” funções Python (que operam sobre tipos e arrays compatíveis) para código nativo via LLVM, reduzindo drasticamente o overhead dos laços em Python puro. A ideia prática é simples: decorar funções críticas com `@njit` (ou `@jit(nopython=True)`), evitar objetos Python dentro dessas funções e, quando fizer sentido, ativar paralelização com `parallel=True` e `prange`.

O primeiro cuidado ao medir é lembrar do custo de compilação: na *primeira* chamada de cada assinatura de tipos, Numba compila a função (demora mais). Depois disso, as chamadas seguintes usam o código nativo já gerado.

O exemplo abaixo acelera uma multiplicação de matrizes ingênua (três laços) sem recorrer ao NumPy `@`. Primeiro mostramos a versão `njit` sequencial; em seguida, a variação paralela (`parallel=True + prange`). Usamos `perf_counter` para mostrar o tempo da primeira chamada (com compilação) e das chamadas seguintes (sem compilação).

```

1 import numpy as np
2 import time
3 from numba import njit, prange

```

```

4
5 # Versao Python pura (referencia)
6 def matmul_naive(A, B):
7     n, m = A.shape
8     m2, p = B.shape
9     assert m == m2
10    C = np.zeros((n, p))
11    for i in range(n):
12        for j in range(p):
13            s = 0.0
14            for k in range(m):
15                s += A[i, k] * B[k, j]
16            C[i, j] = s
17    return C
18
19 # Versao Numba: nopython mode (sem objetos Python dentro)
20 @njit
21 def matmul_numba(A, B):
22     n, m = A.shape
23     m2, p = B.shape
24     C = np.zeros((n, p))
25     for i in range(n):
26         for j in range(p):
27             s = 0.0
28             for k in range(m):
29                 s += A[i, k] * B[k, j]
30             C[i, j] = s
31     return C
32
33 # Versao Numba paralela: requer parallel=True e uso de prange
34 @njit(parallel=True)
35 def matmul_numba_parallel(A, B):
36     n, m = A.shape
37     m2, p = B.shape
38     C = np.zeros((n, p))
39     for i in prange(n): # <-- prange permite paralelizar esse loop
40         externo
41         for j in range(p):
42             s = 0.0
43             for k in range(m):
44                 s += A[i, k] * B[k, j]
45             C[i, j] = s
46     return C
47
48 # Benchmark simples: separa "primeira chamada" e "repetidas"
49 def bench(func, *args, repeat=3, label=""):
50     # primeira chamada (inclui compilacao JIT quando aplicavel)
51     t0 = time.perf_counter()

```

```

51     out = func(*args)
52     t1 = time.perf_counter()
53     print(f"{label} [1a chamada]: {t1 - t0:.3f} s")
54
55     # chamadas seguintes (ja compilado)
56     best = float("inf")
57     for _ in range(repeat):
58         t0 = time.perf_counter()
59         func(*args)
60         t1 = time.perf_counter()
61         best = min(best, t1 - t0)
62     print(f"{label} [melhor chamada subsequente]: {best:.3f} s")
63     return out
64
65 if __name__ == "__main__":
66     n = 600
67     A = np.random.rand(n, n)
68     B = np.random.rand(n, n)
69
70     # Referencia Python puro (lento)
71     bench(matmul_naive, A, B, label="naive (Python)")
72
73     # Numba sequencial
74     bench(matmul_numba, A, B, label="Numba (njit)")
75
76     # Numba paralelo
77     bench(matmul_numba_parallel, A, B, label="Numba (parallel)")
```

Acelerando loops com Numba (@njit)

Na prática, você deverá observar algo assim: a versão Python pura leva segundos; a versão @njit cai para frações (ou poucos segundos em matrizes grandes) após a compilação; a versão paralela tende a ganhar mais em máquinas com vários núcleos, desde que o tamanho do problema justifique o overhead de criar e sincronizar threads. Nem todo laço se beneficia de parallel=True; se o problema é pequeno, o custo extra pode superar o ganho.

Outro modo útil de Numba é compilar funções *elementwise* com @vectorize, criando uma ufunc ao estilo NumPy; isso permite aplicar a função diretamente sobre arrays, com broadcast, sem escrever laços em Python. O exemplo a seguir define uma ufunc para uma transformação escalar simples e a aplica a um array grande.

```

1 import numpy as np
2 from numba import vectorize, float64
3
4 @vectorize([float64(float64)])
5 def transform(x):
6     # alguma transformacao escalar (exemplo)
7     return (x * x + 0.5) / (x + 1.0)
8
```

```

9 x = np.random.rand(1_000_000)
10 y = transform(x) # aplica como ufunc, sem laços explícitos em Python

```

UFunc com @vectorize (estilo NumPy)

Algumas recomendações práticas ao usar Numba: (i) mantenha dentro das funções JIT apenas operações suportadas (aritmética, indexação NumPy, algumas funções math/numpy); (ii) evite objetos Python (listas que crescem, dicionários, set) e chamadas que exijam o interpretador; (iii) prefira arrays com *dtype* numéricos (float64, int64, etc.) e formatos contíguos; (iv) tome cuidado com alocação excessiva dentro do laço; (v) ative parallel=True apenas após confirmar que o gargalo é CPU-bound e que o tamanho do problema compensa a paralelização; (vi) lembre-se do “aquecimento”: meça separando a primeira chamada (com compilação) das seguintes; (vii) quando a função estabilizar, @njit(cache=True) pode salvar o binário no disco e reduzir o tempo de compilação em execuções futuras (útil em scripts).

Por fim, se você já tem uma versão vetorizada eficiente em NumPy (que usa BLAS), muitas vezes ela será tão rápida quanto (ou mais rápida que) reimplementar em Numba, a menos que o seu padrão de acesso/cálculo seja muito específico. O ponto forte do Numba é acelerar *laços* e lógicas numéricas que seriam lentas em Python puro, mantendo o código próximo ao original, sem partir direto para C/C++.

9.4 Paralelismo simples em Bash

O Bash permite escrever pequenos scripts para automatizar tarefas repetitivas. Um dos recursos mais uteis é a possibilidade de rodar varios comandos em paralelo, sem esperar um terminar para comeclar o proximo. Para isso usamos o operador &.

No exemplo abaixo, usamos o comando sleep (que apenas dorme por alguns segundos) para simular tarefas demoradas. Cada chamada ao sleep é enviada ao plano de fundo com &, de modo que o laço continua imediatamente para a proxima iteracao.

```

1 #!/bin/bash
2
3 for i in $(seq 1 5)
4 do
5     echo "Iniciando tarefa $i"
6     sleep 3 &
7 done
8
9 echo "Todas as tarefas foram lancadas!"

```

Rodando sleeps em paralelo

Nesse script, as cinco tarefas começam quase ao mesmo tempo e, apos cerca de tres segundos, todas terminam juntas. Se tirassemos o &, o script levaria cerca de 15 segundos, pois cada sleep 3 seria executado em sequencia.

Para visualizar essa diferença, vejamos primeiro a execucao sequencial:

```

1 #!/bin/bash

```

```

1
2
3 for i in $(seq 1 5)
4 do
5     echo "Rodando tarefa $i"
6     sleep 3
7 done
8
9 echo "Todas as tarefas terminaram (sequencial)"

```

Execucao sequencial

E agora a versao em paralelo, onde o tempo total cai para cerca de 3 segundos:

```

1 #!/bin/bash
2
3 for i in $(seq 1 5)
4 do
5     echo "Rodando tarefa $i"
6     sleep 3 &
7 done
8
9 wait
10 echo "Todas as tarefas terminaram (paralelo)"

```

Execucao em paralelo

Para garantir que o script so finalize depois que todas as tarefas concluirem, podemos usar explicitamente o comando `wait`:

```

1 #!/bin/bash
2
3 for i in $(seq 1 5)
4 do
5     sleep 3 &
6 done
7
8 wait
9 echo "Todas as tarefas terminaram!"

```

Sincronizando com `wait`

Tambem é possivel limitar quantas tarefas rodam em paralelo. Uma tecnica simples é controlar com um contador e usar `wait -n` para esperar pelo menos um job terminar antes de lancar o proximo:

```

1 #!/bin/bash
2
3 N=2    # no maximo 2 processos ao mesmo tempo
4
5 for i in $(seq 1 5)
6 do

```

```
7 sleep 3 &
8 if (( $(jobs -r | wc -l) >= N )); then
9     wait -n
10    fi
11 done
12
13 wait
14 echo "Fim das tarefas"
```

Limitando jobs simultaneos

Esses exemplos usam apenas comandos nativos (`sleep`, `echo`), mas a ideia é exatamente a mesma se quisermos chamar um script Python ou outro programa no lugar.

Referências Bibliográficas