

Métodos Computacionais

Departamento de Estatística e Matemática Aplicada

Ronald Targino, Rafael Braz, Juvêncio Nobre e Manoel Santos-Neto

2025-09-06

Índice

Prefácio	4
1 Introdução	5
2 Motivação	6
Da teoria à simulação	6
Um atalho analítico útil	7
O papel da simulação	8
Atividade: Problema do Aniversário (22 jogadores)	9
Exercícios	9
3 Números Uniformes	11
3.1 Geração de sequências $U(0, 1)$	11
3.2 Geradores Congruenciais Lineares	12
3.2.1 Exemplo	12
3.2.2 Implementação em R	12
3.3 Geradores Congruenciais Lineares Mistos	13
3.3.1 Questão de estouro e aritmética modular	14
3.3.2 Implementação em R (com segurança de overflow)	14
3.4 Geradores Congruenciais Lineares Multiplicativos	15
3.4.1 Características e restrições	15
3.4.2 Definição de raiz primitiva	15
3.4.3 Exemplo de implementação em R	16
3.5 Uso de Números Aleatórios na Avaliação de Integrais	16
4 Números Pseudoaleatórios	20
4.1 Introdução	20
4.2 Método da Transformação Inversa	20
Exemplo 1	22
Inversa Generalizada	24
Exemplo 2	25
Exemplo 3	26
Exemplo 4	27
Exercício 5	28
4.3 Método da Aceitação-Rejeição	28
4.4 Método da Composição	31

5	Otimização Numérica	34
5.1	Método de Newton	34
5.2	Método de Newton-Raphson	34
5.3	Método Escore de Fisher	34
5.4	Método BFGS	34
6	Métodos de Reamostragem	35
6.1	Bootstrap	35
6.1.1	Introdução	35
6.1.2	Acurária da média amostral	35
6.1.3	Estimativa bootstrap do erro padrão	36
6.1.4	Bootstrap Paramétrico	40
6.2	Jackknife	41
6.2.1	Introdução	41
6.2.2	Estimador do viés	42
6.2.3	Estimado do erro padrão	42
6.3	Intervalos de Confiança	44
6.4	Intervalos de Confiança Normal Padrão	44
6.5	Intervalo de Confiança t-Student	44
6.6	IC bootstrap- t	45
6.6.1	Intervalos de Confiança bootstrap percentil	46
6.6.2	Intervalos de Confiança bootstrap - versões aprimoradas	49
7	Métodos de Monte Carlo	50
7.1	Introdução	50
7.2	Integração de Monte Carlo	50
7.3	Erro de Monte Carlo	50
7.4	Monte Carlo via Função de Importância	50
7.5	Método de Máxima Verossimilhança	50
8	Algoritmo EM	51
9	Métodos Adicionais	52
	References	53

Prefácio

Este livro resulta de anos de experiência em sala de aula dos professores Ronald Targino, Rafael Braz, Juvêncio Nobre e Manoel Santos-Neto. Destina-se a apoiar os alunos da graduação em Estatística e do Programa de Pós-Graduação em Modelagem e Métodos Quantitativos (PPGMMQ) do Departamento de Estatística e Matemática Aplicada (DEMA) da Universidade Federal do Ceará (UFC).

Ao longo dos capítulos, abordamos a geração de números aleatórios (discretos e contínuos); métodos de suavização; simulação estocástica por inversão, rejeição e composição, bem como métodos de reamostragem; métodos de aproximação e integração; quadratura Gaussiana, integração de Monte Carlo e quadratura adaptativa; métodos de Monte Carlo em sentido amplo; amostradores MCMC, com ênfase em Gibbs e Metropolis–Hastings; otimização numérica via Newton–Raphson, Fisher scoring e quase-Newton, além do algoritmo EM; Bootstrap e Jackknife; diagnóstico de convergência; e aspectos computacionais em problemas práticos, com foco em implementação eficiente, estabilidade numérica e reprodutibilidade dos resultados.

Esperamos que este material sirva não apenas como texto-base para as disciplinas Estatística Computacional (graduação em Estatística) e Métodos Computacionais em Estatística (Mestrado-PPGMMQ), mas também como suporte para aqueles que desejam programar com qualidade na área de Estatística.

1 Introdução

A simulação tem um papel preponderante na estatística moderna, e suas vantagens no ensino de Estatística são conhecidas há muito tempo. Em um de seus primeiros números, o periódico *Teaching Statistics* publicou artigos que aludem precisamente a isso. Thomas e Moore (1980) afirmaram que “a introdução do computador na sala de aula escolar trouxe uma nova técnica para o ensino, a técnica da simulação”. Zieffler e Garfield (2007) e Tintle et al. (2015) discutem o papel e a importância da aprendizagem baseada em simulação no currículo de graduação em Estatística. No entanto, outros autores (por exemplo, Hodgson e Burke 2000) discutem alguns problemas que podem surgir ao ensinar uma disciplina por meio de simulação, a saber, o desenvolvimento de certos equívocos na mente dos estudantes (Martins 2018).

2 Motivação

A Estatística, além de lidar com modelos matemáticos rigorosos, também é permeada por situações em que a intuição humana falha de maneira sistemática. Um exemplo clássico é o **problema do aniversário**, que há décadas desperta curiosidade entre estudantes e pesquisadores.

Enunciado

Em uma sala com r pessoas, qual a probabilidade de que pelo menos duas delas compartilhem o mesmo aniversário?

Um resultado surpreendente é: com apenas **23 pessoas** em uma sala, a probabilidade de que haja pelo menos uma coincidência de aniversários já é **superior a 50%**. Esse resultado é tão interessante que pode ser uma porta de entrada natural para discutir a diferença entre **probabilidade teórica** e **evidência empírica obtida por simulação**.

Da teoria à simulação

Do ponto de vista teórico, a probabilidade de que todos os aniversários sejam distintos entre r pessoas é

$$\Pr(\text{todos distintos}) = \prod_{i=1}^{r-1} \frac{365-i}{365} = \left(1 - \frac{1}{365}\right) \left(1 - \frac{2}{365}\right) \cdots \left(1 - \frac{r-1}{365}\right).$$

Logo, a probabilidade de pelo menos uma coincidência é

$$p_r = 1 - \Pr(\text{todos distintos}).$$

Esse produto é conceitualmente claro, mas fica pouco manejável mentalmente para k moderados. É aqui que a **simulação computacional** pode entrar como aliada didática e científica.

Um atalho analítico útil

O produto acima admite uma **aproximação exponencial simples e acurada**, obtida tomando logaritmo e usando a expansão para argumentos pequenos:

$$\ln(1-x) = -x + o(x), \quad (x \rightarrow 0).$$

Aplicando ao produto,

$$\begin{aligned} \ln(1-p_r) &= \sum_{i=1}^{r-1} \ln\left(1 - \frac{i}{365}\right) \\ &\approx - \sum_{i=1}^{r-1} \frac{i}{365} = - \frac{1+2+\dots+(r-1)}{365} = - \frac{r(r-1)}{2 \cdot 365}. \end{aligned}$$

Exponentiando e isolando p_r , obtemos a aproximação

$$p_r \approx 1 - \exp\left\{-\frac{r(r-1)}{730}\right\}.$$

Essa fórmula tem três virtudes didáticas:

- 1) **Clareza**: exhibe explicitamente o papel do número de pares $\binom{r}{2}$.
- 2) **Rapidez**: permite cálculos aproximados para valores de r de interesse.
- 3) **Boas aproximações** já para r na casa das dezenas.

Exemplo Rápido

- Para **23** pessoas:

$$p_{23}^{(\text{aprox})} = 1 - \exp\left\{-\frac{23 \cdot 22}{730}\right\} = 1 - \exp\{-0.69315\} \approx 0.500,$$

alinhando-se ao resultado clássico de que **23** pessoas já superam 50% de chance de coincidência.

O papel da simulação

A simulação estatística permite reproduzir o experimento de forma empírica: sorteamos aleatoriamente dias de aniversário para os indivíduos e verificamos se há repetições. Repetindo o processo milhares de vezes, obtemos uma estimativa para a probabilidade de coincidência.

Por exemplo, em **R**:

```
k <- 23
birthdays <- sample(1:365, k, replace = TRUE)
any(duplicated(birthdays))
```

```
[1] TRUE
```

Ao repetir esse procedimento muitas vezes (por exemplo, 10.000 simulações), podemos estimar a proporção de conjuntos com coincidência. Pela Lei dos Grandes Números, essa estimativa converge para o valor teórico de aproximadamente 0,507 quando $k = 23$.

```
set.seed(123) #reprodutibilidade

k <- 23
B <- 10000

acertos <- 0L
i <- 0L

repeat {
  i <- i + 1L
  bdays <- sample(1:365, k, replace = TRUE)
  acertos <- acertos + as.integer(any(duplicated(bdays)))
  if (i >= B) break
}

p_hat <- acertos / B
p_hat
```

```
[1] 0.5073
```


Atividade: Problema do Aniversário (22 jogadores)

Nesta motivação consideramos um exemplo discutido em Martins (2018) que é o conhecido e amplamente divulgado problema do aniversário (ver, por exemplo, Falk 2014). Martins (2018) segue o exemplo de Matthews e Stones (1998), considerando duas equipes de futebol e, portanto, coincidências de aniversário entre 22 jogadores. Martins (2018) afirma que um resultado positivo importante dessa atividade é a discussão que surgirá naturalmente entre os estudantes, com o professor atuando como mediador. Além disso, os estudantes adoram jogos e a descoberta prática, e a simulação facilita o engajamento nessas atividades, ao mesmo tempo que ilustra resultados que podem ser não intuitivos, bem como teoria geral, como a **Lei dos Grandes Números**.

Agora iremos considerar o seguinte problema:

O problema: Em uma partida de futebol, qual é a probabilidade de que pelo menos dois dos 22 jogadores façam aniversário no mesmo dia?

Em um país chamado de país do futebol, o contexto é proposital: o futebol é popular e as probabilidades resultantes são contraintuitivas. Antes de qualquer cálculo, considere as hipóteses: (i) todos os 365 dias do ano são igualmente prováveis para qualquer aniversário; (ii) as datas de aniversário dos jogadores são independentes entre si.

Objetivos

- Estimar, via simulação, a probabilidade de coincidência de aniversários.
- Relacionar frequência relativa, Lei dos Grandes Números e variação amostral.
- Comparar o resultado exato e aproximado.

Hipóteses

- 365 dias equiprováveis, datas independentes, ignorar bissexto/gêmeos.

Materiais

- R (ou Posit Cloud), roteiro com comandos `sample()`, `table()`, `mean()`.

Exercícios

- 1) Determinar o menor número de pessoas que deve estar em uma sala para que se possa apostar, com mais de 50% de chance de ganhar, que entre elas existam pelo menos duas com o mesmo aniversário.

- 2) Determinar o menor número de outras pessoas que deve estar em uma sala com você para que se possa apostar, com mais de 50% de chance de ganhar, que pelo menos uma delas tenha o mesmo aniversário que o seu.

3 Números Uniformes

As simulações, de modo geral, requerem uma base inicial formada por números aleatórios. Diz-se que uma sequência R_1, R_2, \dots é composta por números aleatórios quando cada termo segue a distribuição uniforme $U(0, 1)$ e R_i é independente de R_j para todo $i \neq j$. Embora alguns autores utilizem o termo “números aleatórios” para se referir a variáveis amostradas de qualquer distribuição, aqui ele será usado exclusivamente para variáveis com distribuição $U(0, 1)$.

3.1 Geração de sequências $U(0, 1)$

Uma abordagem é utilizar dispositivos físicos aleatorizadores, como máquinas que sorteiam números de loteria, roletas ou circuitos eletrônicos que produzem “ruído aleatório”. Contudo, tais dispositivos apresentam desvantagens:

1. **Baixa velocidade** e dificuldade de integração direta com computadores.
2. **Necessidade de reprodutibilidade** da sequência. Por exemplo, para verificação de código ou comparação de políticas em um modelo de simulação, usando a mesma sequência para reduzir a variância da diferença entre resultados.

Uma forma simples de obter reprodutibilidade é armazenar a sequência em um dispositivo de memória (HD, CD-ROM, livro). De fato, a RAND Corporation publicou *A Million Random Digits with 100 000 Random Normal Deviates* (1955). Entretanto, acessar armazenamento externo milhares ou milhões de vezes torna a simulação lenta.

Assim, a abordagem preferida é **gerar números pseudoaleatórios em tempo de execução**, via recorrências determinísticas sobre inteiros. Isso permite:

- Geração rápida;
- Eliminação do problema de armazenamento;
- Reprodutibilidade controlada.

Entretanto, a escolha inadequada da recorrência pode gerar sequências com baixa qualidade estatística.

3.2 Geradores Congruenciais Lineares

Um **Gerador Congruencial Linear (LGC)** produz uma sequência de inteiros não negativos X_i , $i = 1, 2, \dots$, por meio da relação de recorrência:

$$X_i = (aX_{i-1} + c) \bmod m, \quad i = 1, 2, \dots,$$

em que $a > 0$ é o multiplicador, $X_0 \geq 0$ é a *semente* (*seed*), $c \geq 0$ é o incremento e $m > 0$ é o módulo.

Os valores a, c, X_0 estão no intervalo $[0, m - 1]$. O número pseudoaleatório R_i é obtido por:

$$R_i = \frac{X_i}{m}, \quad R_i \in (0, 1).$$

Se m for suficientemente grande, os valores discretos $0/m, 1/m, \dots, (m-1)/m$ são tão próximos que R_i pode ser tratado como variável contínua.

3.2.1 Exemplo

Seja o gerador:

$$X_i = (9X_{i-1} + 3) \bmod 24, \quad i \geq 1.$$

Escolhendo $X_0 = 3$:

$$X_1 = (9 \times 3 + 3) \bmod 24 = 14$$

$$X_2 = (9 \times 14 + 3) \bmod 24 = 1$$

e assim por diante.

A sequência $R_i = X_i/24$ gerada terá período $\ell = 24$.

3.2.2 Implementação em R

```
# Função LCG genérica
lcg <- function(a, c, m, seed, n) {
  x <- numeric(n)
  x[1] <- seed
  for (i in 2:n) {
    x[i] <- (a * x[i-1] + c) %% m
  }
  r <- x / m
  return(list(X = x, R = r))
}

# Exemplo com a = 9, c = 3, m = 24, seed = 3
resultado <- lcg(a = 9, c = 3, m = 24, seed = 3, n = 20)
resultado$X
```

```
[1] 3 6 9 12 15 18 21 0 3 6 9 12 15 18 21 0 3 6 9 12
```

```
resultado$R
```

```
[1] 0.125 0.250 0.375 0.500 0.625 0.750 0.875 0.000 0.125 0.250 0.375 0.500
[13] 0.625 0.750 0.875 0.000 0.125 0.250 0.375 0.500
```

3.3 Geradores Congruenciais Lineares Mistos

Nos LCGs **mistos** temos $c > 0$. Uma escolha prática é $m = 2^b$, onde b é o número de bits utilizável para inteiros positivos na arquitetura/linguagem. Em muitos ambientes, inteiros usam 32 bits (um para o sinal), implicando $b = 31$ e intervalo $[-2^{31}, 2^{31} - 1]$.

Quando $m = 2^b$, obtemos **período completo** ($\ell = m$) se:

- 1) c é **ímpar** (garante $\gcd(c, m) = 1$);
- 2) $a - 1$ é múltiplo de todos os fatores primos de m e também de 4 (como m é potência de 2).

Essa é a razão de geradores simples com $m = 2^b$, c ímpar e $a \equiv 1 \pmod{4}$ atingirem $\ell = m$.

3.3.1 Questão de estouro e aritmética modular

Em linguagens com inteiros limitados, calcular $aX_{i-1} + c$ pode **transbordar**. Soluções comuns:

- usar precisão estendida (64 bits) ou bibliotecas de inteiros grandes;
- empregar **truques de aritmética modular** (como o método de Schrage) para evitar overflow;
- trabalhar com módulo $m = 2^b$ e aproveitar o “wrap” de bits.

A seguir, implementamos LCG misto com $m = 2^{31}$, $a = 906185749$, $c = 1$. Parâmetros com boas propriedades estatísticas relatadas na literatura.

3.3.2 Implementação em R (com segurança de overflow)

Para garantir a correção do módulo com inteiros grandes, usaremos `bit64` (inteiros de 64 bits) e normalizaremos para $(0, 1)$.

```
#if (!requireNamespace("bit64", quietly = TRUE)) {  
#  install.packages("bit64")  
#}  
  
library(bit64)  
  
lcg_misto <- function(n, seed = 3456L,  
                      a = 906185749L,  
                      c = 1L,  
                      m = bit64::as.integer64(2)^31) {  
  # Trabalha em integer64 para evitar perda de precisão  
  x <- bit64::as.integer64(seed)  
  outX <- bit64::integer64(n)  
  outR <- numeric(n)  
  outX[1] <- x  
  outR[1] <- as.double(x) / as.double(m)  
  for (i in 2:n) {  
    x <- (bit64::as.integer64(a) * x + bit64::as.integer64(c)) %% m  
    outX[i] <- x  
    outR[i] <- as.double(x) / as.double(m)  
  }  
  list(X = outX, R = outR)  
}
```

```
# Exemplo: primeiros 5 números com seed = 3456
set.seed(NULL)
g1 <- lcg_misto(n = 5, seed = 3456L)
g1$X
```

```
integer64
[1] 3456          746789761  460230038  1591485775 1024426876
```

```
g1$R
```

```
[1] 1.609325e-06 3.477511e-01 2.143113e-01 7.410933e-01 4.770359e-01
```

3.4 Geradores Congruenciais Lineares Multiplicativos

No caso **multiplicativo**, temos $c = 0$, e a recorrência fica:

$$X_i = (aX_{i-1}) \bmod m$$

3.4.1 Características e restrições

- Se $X_i = 0$ em algum passo, toda a sequência futura será zero — portanto $X_0 \neq 0$.
- Se $a = 1$, a sequência é constante — também deve ser evitado.
- O **período máximo** possível é $m - 1$, e ele só é atingido quando:
 1. m é primo;
 2. a é uma **raiz primitiva** módulo m .

3.4.2 Definição de raiz primitiva

Um número a é raiz primitiva módulo m se seus poderes geram todos os inteiros não nulos módulo m .

Matematicamente, a satisfaz:

$$m \nmid a^{(m-1)/q} - 1, \quad \forall q \text{ primo que divide } m - 1$$

Esse tipo de gerador é chamado **Gerador de Módulo Primo e Período Máximo**.

3.4.3 Exemplo de implementação em R

A seguir, implementamos um gerador multiplicativo com módulo primo $m = 2^{31} - 1$ (primo de Mersenne) e multiplicador $a = 630360016$, conhecido por apresentar boas propriedades estatísticas.

```
if (!requireNamespace("gmp", quietly = TRUE)) {
  install.packages("gmp")
}
library(gmp)

lcg_mult_primo <- function(n, seed, a = 630360016, m = 2147483647) {
  A <- as.bigz(a); M <- as.bigz(m)
  x <- as.bigz(seed)
  X <- integer(n); R <- numeric(n)
  for (i in seq_len(n)) {
    X[i] <- as.integer(x)
    R[i] <- as.numeric(x) / m
    x <- (A * x) %% M
  }
  list(X = X, R = R)
}

# Exemplo: gerar 10 valores
g2 <- lcg_mult_primo(n = 10, seed = 12345L)
g2$X
```

```
[1]      12345 1461144439 1646755962 423395703 2041926374 720397004
[7] 140279311 597861375 629442282 759842328
```

```
g2$R
```

```
[1] 5.748589e-06 6.803984e-01 7.668305e-01 1.971590e-01 9.508461e-01
[6] 3.354610e-01 6.532264e-02 2.784009e-01 2.931069e-01 3.538292e-01
```

3.5 Uso de Números Aleatórios na Avaliação de Integrais

Uma das primeiras aplicações do uso de números aleatórios foi na resolução de integrais. Considere uma função $g(x)$ e suponha que desejamos calcular uma integral de interesse.

$$\theta = \int_0^1 g(x)dx.$$

Para calcular o valor da integral, observe que, se U é uma variável aleatória com distribuição uniforme no intervalo $(0, 1)$, então podemos reescrever a integral da seguinte forma:

$$\theta = E[g(U)].$$

Se U_1, U_2, \dots, U_n são variáveis aleatórias independentes e uniformes em $(0, 1)$, então as variáveis $g(U_1), g(U_2), \dots, g(U_n)$ são independentes e identicamente distribuídas, todas com esperança igual θ (o valor da integral). Assim, pelo **Teorema da Lei Forte dos Grandes Números**, temos que, com probabilidade 1,

$$\frac{1}{n} \sum_{i=1}^n g(U_i) \rightarrow \theta \quad \text{quando } n \rightarrow \infty.$$

Assim, podemos aproximar o valor da integral gerando um grande número de pontos aleatórios u_i no intervalo $(0, 1)$ e tomando como estimativa a média dos valores $g(u_i)$. Esse procedimento de aproximação de integrais é conhecido como método de **Monte Carlo**.

Se quisermos calcular uma integral mais geral, podemos aplicar a mesma ideia: transformar o problema em uma esperança matemática e, em seguida, aproximá-la por meio de médias amostrais obtidas a partir de números aleatórios. Considere:

$$\theta = \int_a^b g(x)dx.$$

Se quisermos calcular a integral em um intervalo genérico (a, b) , fazemos a substituição

$$u = \frac{x - a}{b - a}, \quad du = \frac{dx}{b - a},$$

o que nos permite reescrevê-la como

$$\theta = \int_a^b g(x)dx = (b - a) \int_0^1 g(a + (b - a)u)du.$$

Definindo

$$h(u) = (b-a)g(a + (b-a)u),$$

temos

$$\theta = \int_0^1 h(u)du.$$

Assim, podemos aproximar θ gerando números aleatórios $u_1, u_2, \dots, u_n \sim U(0, 1)$ e tomando como estimativa a média

$$\theta \approx \frac{1}{n} \sum_{i=1}^n h(u_i).$$

Agora, se nosso objetivo é calcular a integral:

$$\theta = \int_0^\infty g(x)dx.$$

Fazendo a mudança de variável

$$u = \frac{1}{x+1}, \quad du = \frac{-dx}{(x+1)^2} = -u^2 dx.$$

Logo,

$$dx = -\frac{du}{u^2},$$

e a integral resultante é

$$\theta = \int_0^1 h(u)du,$$

com

$$h(u) = \frac{g\left(\frac{1}{u} - 1\right)}{u^2}.$$

A utilidade de empregar números aleatórios para aproximar integrais torna-se ainda mais evidente no caso de integrais multidimensionais. Suponha que g seja uma função com argumento n -dimensional e que estejamos interessados em calcular:

$$\theta = \int_0^1 \int_0^1 \dots \int_0^1 g(x_1, x_2, \dots, x_n) dx_1 dx_2 \dots dx_n.$$

Observe que θ pode ser expresso como o seguinte valor esperado:

$$\theta = E[g(U_1, U_2, \dots, U_n)],$$

em que U_1, U_2, \dots, U_n são variáveis aleatórias independentes uniformemente distribuídas no intervalo $(0, 1)$. Assim, se gerarmos k conjuntos independentes, cada um formado por n variáveis aleatórias independentes com distribuição uniforme em $(0, 1)$, então as variáveis

$$g(U_{i1}, U_{i2}, \dots, U_{in}), \quad i = 1, 2, \dots, k,$$

serão independentes e identicamente distribuídas, todas com esperança igual a θ (o valor da integral). Portanto, podemos aproximar θ por meio da média amostral:

$$\theta \approx \frac{1}{k} \sum_{i=1}^k g(U_{i1}, U_{i2}, \dots, U_{in}).$$

4 Números Pseudoaleatórios

4.1 Introdução

4.2 Método da Transformação Inversa

Suponha que desejamos gerar o valor de uma variável aleatória discreta X com função de massa de probabilidade (f.m.p):

$$\Pr(X = x_j) = p_j, \quad j = 0, 1, \dots, \quad \sum_j p_j = 1.$$

Interesse: Gerar valores da variável aleatória X com distribuição:

X	$P(X = x_j)$
x_0	p_0
x_1	p_1
x_2	p_2
\vdots	\vdots
x_j	p_j
\vdots	\vdots

Para realizar isso, gere um valor u de $U \sim (0, 1)$ e atribua o valor x_j , $j = 0, 1, \dots$, a X conforme as condições abaixo:

$$X = \begin{cases} x_0, & \text{se } u < p_0, \\ x_1, & \text{se } p_0 \leq u < p_0 + p_1, \\ x_2, & \text{se } p_0 + p_1 \leq u < p_0 + p_1 + p_2, \\ x_3, & \text{se } p_0 + p_1 + p_2 \leq u < p_0 + p_1 + p_2 + p_3, \\ \vdots & \\ x_j, & \text{se } \sum_{i=0}^{j-1} p_i \leq u < \sum_{i=0}^j p_i, \\ \vdots & \end{cases}$$

Como, para $0 < a < b < 1$, $\Pr(a \leq U < b) = b - a$, temos que:

$$\Pr(X = x_j) = \Pr\left(\sum_{i=0}^{j-1} p_i \leq U < \sum_{i=0}^j p_i\right) = p_j,$$

e, portanto, X possui a distribuição desejada.

Observações:

1. Algoritmo - O procedimento anterior pode ser escrito como:

Passo 1. Gerar um valor $u \sim U(0, 1)$.

Passo 2. Se $u < p_0$, então $X = x_0$; caso contrário:

- se $u < p_0 + p_1$, então $X = x_1$; caso contrário:
- se $u < p_0 + p_1 + p_2$, então $X = x_2$; caso contrário:
- \vdots

Passo 3. Repetir os passos 1 e 2 n vezes, onde n é o tamanho da amostra.

2. Se os valores, $x_i, i \geq 0$, são ordenados de modo que $x_0 < x_1 < \dots$, e de denotamos

F como função de distribuição de X , então $F(x_j) = \sum_{i=1}^j p_i$. Assim,

X será igual a x_j se $F(x_{j-1}) < U < F(x_j)$.

Em outras palavras, após gerar um número aleatório U , determinamos o valor de X encontrando o intervalo $[F(x_{j-1}), F(x_j))$ no qual U se encontra (ou, de forma equivalente, encontrando o inverso de $F(U)$). É por essa razão que o procedimento acima é denominado *método da transformada inversa discreta* para gerar X .

i Nota

O tempo necessário para gerar uma variável aleatória discreta por esse método é proporcional ao número de intervalos que devem ser pesquisados. Por essa razão, às vezes é vantajoso considerar os possíveis valores x_j de X em ordem decrescente das probabilidades p_j .

Exemplo 1

Simular n valores de X tal que $p_1 = 0.20$, $p_2 = 0.15$, $p_3 = 0.25$, $p_4 = 0.40$, em que $p_j = P(X = j)$.

Algoritmo 1

Passo 1. Gerar n valores $u \sim U(0, 1)$.

Passo 2. Para cada u :

- Se $u < 0.20$, então $X = 1$.
- Caso contrário, se $u < 0.35$, então $X = 2$.
- Caso contrário, se $u < 0.60$, então $X = 3$.
- Caso contrário ($u \geq 0.60$), então $X = 4$.

Passo 3. Repetir n vezes os passos 1 e 2.

A seguir, são apresentados dos códigos escritos na linguagem R:

```
# Proposta 1
n <- 10000
x <- 1:4
p <- c(0.20, 0.15, 0.25, 0.40)
pa <- cumsum(p) # probabilidades acumuladas
a <- c() # vetor para a amostra gerada de X

for(i in 1:n){
  u <- runif(1)
  if (u < pa[1]) {
    a[i] <- x[1]
  } else {
    if (u < pa[2]) {
      a[i] <- x[2]
    } else {
      if (u < pa[3]) {
        a[i] <- x[3]
      } else {
        a[i] <- x[4]
      }
    }
  }
}
```

```

    } else { if (u < pa[3]) {
      a[i] <- x[3]
    } else {
      a[i] <- x[4]
    }
  }
}
}
table(a)/n # tabela de proporções

```

```

a
      1      2      3      4
0.2045 0.1453 0.2551 0.3951

```

```

# Proposta 2
n <- 10000;
x <- 1:4;
p <- c(0.20, 0.15, 0.25, 0.40)
pa <- cumsum(p) # probabilidades acumuladas
a <- c() # vetor para a amostra gerada de X

for(i in 1:n){
  u = runif(1)
  ifelse(u < pa[1], a[i] <- x[1],
    ifelse(u < pa[2], a[i] <- x[2],
      ifelse(u < pa[3], a[i] <- x[3], a[i] <- x[4])))
}
table(a)/n # tabela de proporções

```

```

a
      1      2      3      4
0.1990 0.1527 0.2514 0.3969

```

Algoritmo 2

Passo 1. Gerar um valor $u \sim U(0, 1)$.

Passo 2. Para cada u :

- Se $u < 0.40$, então $X = 4$;
- Caso contrário, se $u < 0.65$, então $X = 3$;
- Caso contrário, se $u < 0.85$, então $X = 1$;

- Caso contrário $u \geq 0.85$, então $X = 2$.

Passo 3. Repetir n vezes os passos 1 e 2.

Observação:

A proposta 2 usa a **ordem decrescente** dos p_j para otimizar a busca.

Algoritmo 3 (mudando as desigualdades)

Passo 1. Gerar $u \sim U(0, 1)$.

Passo 2. Se $u \leq 0.40$, então $X = 4$;

- senão, se $u \leq 0.65$, então $X = 3$;
- senão, se $u \leq 0.85$, então $X = 1$;
- senão ($u > 0.85$), então $X = 2$.

Passo 3. Repetir n vezes os passos 1–2.

Observação:

Com $u \sim U(0, 1)$ é contínua, usar $<$ ou \leq é equivalente em termos de distribuição (a probabilidade de u cair exatamente no ponto de corte é zero). Em implementação numérica, essa escolha apenas define para onde vão raríssimos empates nos pontos cortes.

Inversa Generalizada

Seja F uma função de distribuição qualquer. A inversa generalizada, denotada por F^{-1} , é definida da seguinte forma:

$$F^{-1}(u) = \inf\{x \in \mathbb{R} : F(x) \geq u\}$$

- Tanto a inversa de F como a inversa generalizada estão sendo denotadas por F^{-1} .
- Se a inversa de F existe no sentido usual, ela coincide com a inversa generalizada. Note que, se F for estritamente crescente e contínua, então $\forall x \in \mathbb{R}$ existirá apenas um u tal que $F(x) = u$.
- A proposição seguinte dá base para as simulações de v.a. discretas.

Seja F uma função de distribuição qualquer e F^{-1} sua inversa generalizada. Temos:

$$F^{-1}(u) \leq x \text{ se, e somente se, } u \leq F(x)$$

Se você deseja simular valores de X com distribuição uniforme discreta:

$$P(X = j) = 1/k, \quad j = 1, 2, 3, \dots, k.$$

Basta utilizar a Proposições 1 ou a Proposição 2.

Proposição 1:

Gere $U \sim U(0, 1)$. Defina $X = j$ se

$$\frac{j-1}{k} \leq U < \frac{j}{k},$$

ou, equivalentemente,

$$(j-1) \leq kU < j.$$

Proposição 2:

Gere $U \sim U(0, 1)$. Defina $X = \text{Int}(kU) + 1$, em que $\text{Int}(x)$ é a parte inteira de x .

Exemplo 2

Para simular valores de X com $P(X = j) = 1/10$, $j = 1, 2, 3, \dots, 10$, você pode seguir o procedimento a seguir:

$j-1$	j	U	kU	$X = j$	$X = \text{Int}(kU) + 1$
0	1	0,01	0,1	1	1
1	2	0,31	3,1	4	4
2	3	0,53	5,3	6	6
3	4	0,92	9,2	10	10
4	5	0,45	4,5	5	5
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
9	10	0,74	7,4	8	8

Exercícios

1. Gerar uma permutação dos números $1, 2, 3, \dots, n$, considerando todas as $n!$ possíveis permutações igualmente prováveis (exemplo 4b).
2. Gerar valores de $X \sim \text{Geométrica}(p)$ (exemplo 4d).

3. Implementar o algoritmo para gerar valores de $X \sim \text{Poisson}(\lambda)$ (seção 4.2).
4. Implementar o algoritmo para gerar valores de $X \sim \text{Binomial}(n, p)$ (seção 4.2).

Exemplo 3

Nosso objetivo agora é gerar valores de $X \sim \text{Poisson}(\lambda)$ com:

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

::: {.callout-important} ## **Identidade importante:**

$$p_{i+1} = \frac{\lambda}{i+1} p_i, \quad i \geq 0.$$

:::

Uma forma bastante utilizada para gerar valores de uma variável aleatória $X \sim \text{Poisson}(\lambda)$ é por meio de algoritmos de simulação baseados na identidade acima.

Algoritmo 4

Passo 1. Gerar um número aleatório $u \sim U(0, 1)$.

Passo 2. Inicializar $i = 0$, $p = e^{-\lambda}$, $F = p$.

Passo 3. Se $u < F$, então definir $X = i$ e parar.

Passo 4. Caso contrário:

- atualizar $i \leftarrow i + 1$,
- atualizar $p \leftarrow \frac{\lambda}{i} p$,
- atualizar $F \leftarrow F + p$,
- voltar ao passo 3.

Exercício

Complete o código R abaixo:

```
N = 10^5 # tamanho da amostra
L = 3 # lambda
x = c()
for (j in 1:N){
  u = runif(1)
```

```

i = 0; p = exp(-L); F = p
aceito = "não"
while (aceito != "sim"){
  if(u < F) {
    ...
  } else {
    ...
  }
}
}

```

Exemplo 4

Por fim, iremos gerar valores da variável aleatório X , $X \sim \text{Binomial}(n, p)$, com f.m.p dada por:

$$P(X = i) = \frac{n!}{i!(n-i)!} p^i (1-p)^{n-i}, \quad i = 0, 1, 2, \dots, n.$$

Identidade importante::

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

A partir do resultado acima e do método da transformação inversa podemos escrever o seguinte algoritmo:

Algoritmo 5

Passo 1. Gerar um número aleatório $u \sim U(0, 1)$.

Passo 2. Calcular $k = \frac{p}{1-p}$, inicializar $i = 0$, $p_r = (1-p)^n$, $F = p_r$.

Passo 3. Se $u < F$, definir $X = i$ e parar.

Passo 4. Caso contrário:

- atualizar $p_r \leftarrow \frac{n-i}{i+1} k p_r$,
- atualizar $F \leftarrow F + p_r$,
- atualizar $i \leftarrow i + 1$,
- voltar ao passo 3.

Exercício 5

Escreva um código na linguagem R baseado no Algoritmo 5. Utilize alguma ferramenta gráfica para verificar a coerência dos resultados.

4.3 Método da Aceitação-Rejeição

Suponha que haja interesse em simular valores de uma v.a. X e que não seja possível inverter a sua função de distribuição ou não dispomos de um método para gerar dessa variável aleatória. Entretanto, sabemos como simular de uma outra v.a. Y e que é possível estabelecer uma relação entre as probabilidades associadas às duas variáveis aleatórias (X e Y) de tal modo que valores de Y possam ser admitidos como valores de X .

x	1	2	3	4	5	6	7	8	9	10
p_x	0,11	0,12	0,09	0,08	0,12	0,10	0,09	0,09	0,10	0,10

y	1	2	3	4	5	6	7	8	9	10
q_y	0,10	0,10	0,10	0,10	0,10	0,10	0,10	0,10	0,10	0,10

Contexto:

- **Podemos:** simular valores de Y com f.p. $q_j = P(Y = j), j \geq 0$.
- **Queremos:** simular valores de X com f.p. $p_j = P(X = j), j \geq 0$.
- **Proposta:** simular valor y de Y e aceitar esse valor para X com probabilidade proporcional a $\frac{p_y}{q_y}$.

Algoritmo

Passo 1. Simular y de Y com f.m.p. q_y .

Passo 2. Gerar $u \sim U(0, 1)$.

Passo 3. Se $u < \frac{p_y}{c q_y}$, então aceite $X = y$. Caso contrário, rejeite e não atribua valor a X .

Passo 4. Repetir os passos 1–3 até obter o tamanho de amostra desejado.

O algoritmo aceitação-rejeição gera uma v.a. X tal que $P(X = j) = p_j, j = 0, 1, 2, \dots$. Além disso, o número de iterações que o algoritmo necessita para obter X é uma v.a. geométrica com média c .

Prova do Teorema

1. Em uma iteração, determinar a probabilidade de gerar e ser aceito o valor j :

$$P(Y = j, aceitar) = P(Y = y).P(aceitar/Y = j) = q_j \cdot \frac{p_j}{cq_j} = \frac{p_j}{c}$$

2. Calcular a probabilidade de aceitar um valor j gerado:

$$P(aceitar) = \sum_j P(Y = j, aceitar) = \sum_j \frac{p_j}{c} = \frac{1}{c}$$

3. Como cada iteração independentemente resulta um valor aceitável com probabilidade $\frac{1}{c}$, o número de iterações necessárias segue uma geométrica de média c . Portanto,

$$P(X = j) = \sum_n P(j \text{ aceito na iteração } n) = \sum_n \left(1 - \frac{1}{c}\right)^{n-1} \cdot \frac{p_j}{c} = p_j$$

Observações

- A constante c está relacionada com o número de interações necessárias até a aceitação de um valor de Y para X .
- O valor c deve ser o menor possível.
- O valor de c será o $\max \left\{ \frac{p_y}{q_y} \right\}$.

Nota

$$\begin{aligned} u < \frac{p_y}{c \cdot q_y} \quad \text{e} \quad P(U < \frac{p_y}{c \cdot q_y}) &= \frac{p_y}{c \cdot q_y} \\ \Downarrow \\ \frac{p_y}{c \cdot q_y} &\leq 1 \quad \text{para todo } y \text{ tal que } p_y > 0 \\ \Downarrow \\ c &= \max \left\{ \frac{p_y}{q_y} \right\} \end{aligned}$$

Exemplo 1

Gerar um valor da variável aleatória X com f.m.p.:

j	1	2	3	4	5	6	7	8	9	10
p_j	0,11	0,12	0,09	0,08	0,12	0,10	0,09	0,09	0,10	0,10

Considerando que sabemos gerar de uma v.a. uniforme discreta, assumiremos Y com distribuição

$$P(Y = j) = q_j = \frac{1}{10}; \quad j = 1, 2, \dots, 10.$$

A constante c será determinada por $c = \max \left\{ \frac{p_j}{q_j} \right\} = \frac{0,12}{0,10} = 1,2$.

Algoritmo

1. Simular y de Y : gere $u_1 \sim U(0, 1)$ e faça $y = \text{Int}(10u_1) + 1$.
2. Gerar um segundo número aleatório u_2 .
3. Se $u_2 < \frac{p_y}{0,12}$, faça $X = y$ e pare. Caso contrário, retorne ao passo 1.

i Nota

Suponha $y = 1$. Então, se $u_2 < \frac{p_1}{0,12} = \frac{0,11}{0,12} = 0,9167$, faremos $X = 1$. Isto é, assumiremos que o valor 1 gerado é plausível de ser da distribuição de X . O gráfico a seguir ilustra esse processo.

```
#set.seed(20252)
pseudo_x <- NULL
for(i in seq_len(1000)){

  pj <- c(0.11, 0.12, 0.09, 0.08, 0.12, 0.10, 0.09, 0.09, 0.10, 0.10)

  u1 <- runif(1)
  y <- floor(10*u1) + 1

  repeat{
```

```

u2 <- runif(1)
x <- y

if(u2 < pj[y]/0.12) break
}

pseudo_x[i] <- x
}

round(prop.table(table(pseudo_x)),2)

```

```

pseudo_x
  1    2    3    4    5    6    7    8    9   10
0.09 0.10 0.08 0.10 0.11 0.10 0.11 0.10 0.10 0.10

```

Existe algum problema com os resultados acima? Se sim, faça a correção do código e verifique graficamente a coerência dos resultados.

Observações

- 91,67% de todos os valores 1 gerados de Y serão aceitos
- 100% dos valores 2 gerados de Y serão aceitos
- 75% dos valores 3 gerados de Y serão aceitos
- Qualquer valor da constante c inferior a 1,2 impossibilita gerar a distribuição de X
- Qualquer valor da constante c superior a 1,2 tornaria o processo mais lento para a obtenção da amostra

Exercícios

1. Gere números pseudoaleatórios de X considerando $c = 2, 4$.
2. Compare com os resultados obtidos no Exemplo.

4.4 Método da Composição

Suponha que tenhamos um método eficiente para simular o valor de uma variável aleatória com f.m.p. $p_j, j \geq 0$ ou $q_j, j \geq 0$, e que desejamos simular a variável aleatória X com f.m.p.:

$$\Pr(X = j) = \alpha p_j + (1 - \alpha) q_j, \quad j \geq 0, 0 < \alpha < 1.$$

Se $X_1 \sim p_j$ e $X_2 \sim q_j$, então podemos definir

$$X = \begin{cases} X_1, & \text{com probabilidade } \alpha, \\ X_2, & \text{com probabilidade } 1 - \alpha, \end{cases}.$$

Assim, X terá exatamente a função de probabilidade acima.

Exemplo

Suponha que desejamos gerar o valor de uma variável aleatória X tal que

$$p_j = \Pr(X = j) = \begin{cases} 0.05, & \text{para } j = 1, 2, 3, 4, 5, \\ 0.15, & \text{para } j = 6, 7, 8, 9, 10, \end{cases}$$

Note que $p_j = 0.5 \times p_j^{(1)} + 0.5 \times p_j^{(2)}$, em que

$$p_j^{(1)} = 0.1, \quad j = 1, \dots, 10 \quad \text{e} \quad p_j^{(2)} = \begin{cases} 0, & \text{para } j = 1, 2, 3, 4, 5, \\ 0.2, & \text{para } j = 6, 7, 8, 9, 10, \end{cases}$$

podemos realizar essa simulação gerando primeiro um número aleatório $U \sim U(0, 1)$ e então:

- Se $U < 0.5$, gerar X de uma uniforme discreta sobre $\{1, 2, \dots, 10\}$.
- Caso contrário ($U \geq 0.5$), gerar X de uma uniforme discreta sobre $\{6, 7, 8, 9, 10\}$.

Algoritmo

Passo 1. Gerar $U_1 \sim U(0, 1)$.

Passo 2. Gerar $U_2 \sim U(0, 1)$.

Passo 3. Se $U_1 < 0.5$, definir $X = \text{Int}(10U_1) + 1$. Caso contrário, definir $X = \text{Int}(5U_2) + 6$.

Se F_i , $i = 1, \dots, n$ são funções de distribuição e α_i , $i = 1, \dots, n$ são números não negativos cuja soma é 1, então a função de distribuição:

$$F(x) = \sum_{i=1}^n \alpha_i F_i(x),$$

é uma **mistura**, ou uma **composição**, das funções de distribuição F_i , $i = 1, \dots, n$.

Uma maneira de simular a partir de F é primeiro simular uma variável aleatória I , igual a i com probabilidade α_i , $i = 1, \dots, n$, e então simular a partir da distribuição F_I (Isto é, se o valor simulado de I for $I = j$, então a segunda simulação é feita a partir de F_j). Essa abordagem para simular de F é frequentemente chamada de **método de composição**.

5 Otimização Numérica

5.1 Método de Newton

5.2 Método de Newton-Raphson

5.3 Método Escore de Fisher

5.4 Método BFGS

6 Métodos de Reamostragem

6.1 Bootstrap

6.1.1 Introdução

Bootstrap é um método (computacional) de reamostragem baseado em subamostras de uma amostra observada, sendo introduzido por Efron (1979). Pode ser utilizado com o propósito de estimar erros padrão, viés de estimadores, construir intervalos de confiança, testes de hipóteses, entre outros. Pode ser utilizando sob duas abordagens: **paramétrica** e **não paramétrica**. A abordagem paramétrica exige um modelo estatística, enquanto que na abordagem não paramétrica não há suposição de modelo estatístico; toma-se por base uma distribuição empírica que atribui probabilidade $1/n$ para cada um dos n elementos da amostra. Algumas referências importantes neste tema são: Efron (1979), Wu (1986), Fisher & Hall (1989), Fredman (1986), Efron & Tibshirani (1993), Horowitz (1997), Davison & Hinkley (1997).

6.1.2 Acurária da média amostral

Experimento com 16 ratos, divididos em dois grupos: um grupo recebeu o tratamento e um outro grupo não recebeu o tratamento (controle). O tempo de sobrevivência (dias) é apresentado para cada um dos ratos. O tratamento prolonga a vida?

grupo	tempo 1	tempo 2	tempo 3	n	média	$\widehat{ep}(\text{média})$
tratamento (X)	94	197	16	7	86,86	25,24
	38	99	141			
	23					
controle (Y)	52	104	146	9	56,22	14,14
	10	51	30			
	40	27	46			
diferença					30,63	28,93

Alguns pontos importantes são:

- A resposta à pergunta dependerá de quão acurado(s) é(são) o(s) estimador(es).

- O erro padrão é uma medida (muito usual) de acurácia de estimador.
- erro padrão estimado para a média amostral \bar{X}

$$\widehat{ep}(\bar{X}) = \sqrt{\frac{s^2}{n}},$$

em que $s^2 = \sum_{i=1}^n (x_i - \bar{x})^2 / (n - 1)$.

- erro padrão de qualquer estimador é definido pela raiz quadrada de sua variância

Além disso, temos que:

- erro padrão pequeno \rightarrow acurácia alta
- erro padrão grande \rightarrow acurácia baixa
- acurácia alta (baixa) indica que o estimador apresenta valores próximos (distantes) ao seu valor esperado
- espera-se que 68% dos valores do estimador estejam a menos de um erro padrão do seu valor esperado, e 95%, a menos de dois erros padrões
- erro padrão da diferença ($\bar{X} - \bar{Y}$):

$$28.93 = \sqrt{25.24^2 + 14.14^2}.$$

Resposta à pergunta: a diferença observada 30.63 é somente 30.63/28.93=1.05 erros padrões (estimados) maior que zero, indicando um resultado não significativo, ou seja, o tratamento não aumenta o tempo médio de vida (considerando a teoria dos testes de hipóteses).

O erro padrão para o estimador média amostral apresenta fórmula conhecida, mas há casos em que não dispomos de fórmulas. Suponha que haja interesse em comparar os dois grupos de ratos em relação aos tempos medianos. Temos: md(X)=94 e md(Y)=46. A diferença é 48, maior que a diferença para as médias. Com base na mediana, o tratamento prolonga a vida?

6.1.3 Estimativa bootstrap do erro padrão

Considerações:

- $\tilde{x} = (x_1, x_2, \dots, x_n)$: vetor de dados observado (amostra original de tamanho n)
- $s(\tilde{x})$: estatística de interesse (por exemplo, média amostral)
- Uma amostra bootstrap $\tilde{x}^* = (x_1^*, x_2^*, \dots, x_n^*)$ é obtida pela amostragem aleatória de tamanho n , com reposição, de \tilde{x} . Por exemplo, com $n = 7$, poderíamos obter $\tilde{x}^* = (x_1^*, x_2^*, \dots, x_n^*) = (x_5, x_7, x_5, x_4, x_7, x_3, x_1)$.

O algoritmo bootstrap

- gerar B amostras bootstrap independentes: $\tilde{x}^{*1}, \tilde{x}^{*2}, \dots, \tilde{x}^{*B}$, cada uma de tamanho n e $50 \leq B \leq 200$.
- calcular $s(\tilde{x}^{*b})$, $b = 1, 2, \dots, B$. $s(\tilde{x}^{*b})$ é denominada réplica bootstrap de $s(\tilde{x})$
- calcular $\hat{e}p_{boot} = \hat{e}p_B = \sqrt{\frac{\sum_{b=1}^B [s(\tilde{x}^{*b}) - s(\tilde{x})]^2}{B-1}}$, em que $s(\tilde{x}) = \frac{\sum_{b=1}^B s(\tilde{x}^{*b})}{B}$

Sintaxe do R para calcular a estimativa bootstrap do erro padrão da média do tempo de sobrevida dos ratos do grupo tratamento.

```
set.seed(1234)
# grupo tratamento (amostra original)
x <- c(94, 197, 16, 38, 99, 141, 23)
n <- length(x)
s <- 0 # estatística de interesse
B <- 50 # no. de amostras bootstrap

for(i in 1:B){
  # réplica bootstrap para o estimador média
  s[i] <- mean(sample(x,n,replace=TRUE))
}
# estimativa bootstrap para o erro padrão da média
ep <- sd(s); ep
```

[1] 27.41873

Estimativas bootstrap do erro padrão da média e da mediana do tempo de sobrevivência dos ratos do grupo tratamento. A mediana é menos acurada (erros padrões maiores) que a média para esse conjunto de dados.

B	50	100	250	500	1000	∞
média	19.72	23.63	22.32	23.79	23.02	23.36
mediana	32.21	36.35	34.46	36.72	36.48	37.83

Formalização:

- $\tilde{X} = X_1, X_2, \dots, X_n$: amostra aleatória de uma f.d.a. F
- $\tilde{x} = (x_1, x_2, \dots, x_n)$: amostra aleatória observada de F

- $\Theta = t(F)$: parâmetro (Θ uma função de F)
- $\hat{\Theta} = s(X)$: estimador para Θ
- $\hat{\theta} = s(x)$: estimativa para Θ
- Quão acurado é o estimador $\hat{\Theta}$?
- Seja \hat{F} a distribuição empírica que atribui a probabilidade $1/n$ para cada valor observado x_i , $i = 1, 2, \dots, n$. A amostra bootstrap \tilde{x}^* é definida como a amostra aleatória com reposição de tamanho n extraída de \hat{F} .

$$\tilde{x}^* = (x_1^*, x_2^* \dots, x_n^*)$$

$$\hat{F} \longrightarrow (x_1^*, x_2^* \dots, x_n^*)$$

Nota:

- x^* : o símbolo $*$ indica que a amostra não é a original \tilde{x} , mas uma versão aleatorizada(reamostrada) de \tilde{x}
- a cada amostra bootstrap corresponde uma réplica bootstrap de $\hat{\theta}$, $\hat{\theta}^* = s(\tilde{x}^*)$
- $ep_F(\hat{\Theta})$ é estimado por $ep_{\hat{F}}(\hat{\Theta}^*)$, chamado **estimador bootstrap ideal** para o erro padrão $\hat{\Theta}$
- Raramente faz-se necessário $B \geq 200$ para estimar erro padrão; valores (muito) maiores são necessários, por exemplo, para IC bootstrap.
- $\lim_{B \rightarrow \infty} \hat{ep}_B = ep_{\hat{F}} = ep_{\hat{F}}(\hat{\Theta}^*)$
- O estimador bootstrap ideal $ep_{\hat{F}}(\hat{\Theta}^*)$ e sua aproximação \hat{ep}_B são chamados **estimadores bootstrap não-paramétricos**, pois baseiam-se em \hat{F} , o estimador não-paramétrico de F .
- total de amostras bootstrap distintas (combinação com repetição):

$$\binom{2n-1}{n}.$$

- No R: ver as funções `factorial()`, `choose()`, `combn()`, `combinations()`.

6.1.3.1 Exemplo

Dados de faculdades americanas de direito. População: $N = 82$ faculdades. Amostra aleatória: $n = 15$ faculdades. Variáveis analisadas: LSAT (score médio em um teste), GPA (pontuação média na faculdade).

escola	LSAT	GPA	escola	LSAT	GPA
1	576	3,39	9	651	3,36
2	635	3,30	10	605	3,13
3	558	2,81	11	653	3,12
4	578	3,03	12	575	2,74
5	666	3,44	13	545	2,76
6	580	3,07	14	572	2,88
7	555	3,00	15	594	2,96
8	661	3,43			

- Façamos $Y=LSAT$ e $Z=GPA$. A estatística (estimador) de interesse é o coeficiente de correlação amostral entre as variáveis Y e Z :

$$\hat{\Theta} = corr(Y, Z) = \frac{Cov(Y, Z)}{DP(Y).DP(Z)} = \frac{\sum_{i=1}^n (Y_i - \bar{Y})(Z_i - \bar{Z})/n}{DP(Y).DP(Z)}$$

- Para os dados observados, a estimativa do coeficiente de correlação amostral é 0.776. Quão acurado é o estimador?

Tabela 6.4: **Estimativas bootstrap do erro padrão para $\hat{\Theta} = corr(Y, Z)$**

B	25	50	100	200	400	800	1600	3200
\hat{ep}_B	0,140	0,142	0,151	0,143	0,141	0,137	0,133	0,132

i Nota

No caso de valores extremos inflacionarem fortemente \hat{ep}_B , uma medida mais robusta para o estimador bootstrap do erro padrão é desejável. (ver Efron & Tibshirani (1993)).

i Nota

Inferências baseadas na distribuição normal são “questionáveis” quando o histograma das réplicas bootstrap indica forte assimetria.

6.1.3.2 Exercícios:

- (1) Considere $B = 3200$. Para cada uma das 3200 amostras bootstrap, obtenha a réplica bootstrap $\hat{\theta}^* = \text{corr}(y^*, z^*)$. Faça o histograma das réplicas.
- (2) A Tabela 3.2, p. 21 do livro texto, apresenta os dados populacionais das 82 faculdades. Selecione 3200 amostras aleatórias de tamanho $n = 15$. Para cada uma dessas amostra calcule o coeficiente de correlação e faça o histograma.

6.1.4 Bootstrap Paramétrico

O estimador bootstrap paramétrico do erro padrão é definido por

$$ep_{\hat{F}_{par}}(\hat{\Theta}^*),$$

em que \hat{F}_{par} é um estimador de F derivado do modelo paramétrico para os dados.

Para os dados das faculdades: Vamos supor que a população (LSAT, GPA) possa ser descrita por um modelo paramétrico normal bivariado F . Estimamos F por \hat{F}_{normal} , que denota a f.d.a. de uma normal bivariada com vetor de médias e matriz de covariâncias (\bar{y}, \bar{z}) e $\frac{1}{14} \begin{pmatrix} \sum (y_i - \bar{y})^2 & \sum (y_i - \bar{y})(z_i - \bar{z}) \\ \sum (y_i - \bar{y})(z_i - \bar{z}) & \sum (z_i - \bar{z})^2 \end{pmatrix}$.

O estimador bootstrap paramétrico do erro padrão da correlação $\hat{\Theta}$ será dado por $ep_{\hat{F}_{normal}}(\hat{\Theta}^*)$. Esse estimador bootstrap ideal será aproximado por \hat{ep}_B (conforme algoritmo a seguir).

O algoritmo bootstrap:

- extrair B amostras de tamanho n de \hat{F}_{par} : $\tilde{x}^{*1}, \tilde{x}^{*2}, \dots, \tilde{x}^{*B}$
- calcular $s(\tilde{x}^{*b})$, $b = 1, 2, \dots, B$. $s(\tilde{x}^{*b})$ é a réplica bootstrap de $s(\tilde{x})$
- calcular $\hat{ep}_{boot} = \hat{ep}_B = \sqrt{\frac{\sum_{b=1}^B [s(\tilde{x}^{*b}) - s(\cdot)]^2}{B-1}}$, em que $s(\cdot) = \frac{\sum_{b=1}^B s(\tilde{x}^{*b})}{B}$

No exemplo das faculdades, assumindo o modelo normal bivariado, extraímos B amostras de tamanho $n = 15$ de \hat{F}_{normal} , calculamos o coeficiente de correlação para cada amostra e, por fim, calculamos o desvio padrão desses coeficientes de correlação. Usando $B = 3200$ encontramos $\hat{ep}_B = 0.124$, que é próximo ao valor 0.131 obtido com o bootstrap não-paramétrico.

A fórmula teórica para o erro padrão do coeficiente de correlação é $\frac{1 - \hat{\Theta}^2}{\sqrt{n-3}}$, com $\hat{\Theta} = \text{corr}(Y, Z)$. Vimos que $\hat{\theta} = 0.776$, o que resulta a estimativa 0.115 para o erro padrão do coeficiente de correlação entre as variáveis $Y=\text{LSAT}$ e $Z=\text{GPA}$.

Transformação de Fisher para o coeficiente de correlação $\hat{\Theta}$: $\hat{\zeta} = 0.5 \log \left(\frac{1 + \hat{\Theta}}{1 - \hat{\Theta}} \right)$. Assim, $\hat{\zeta}$ tem distribuição aproximadamente normal com média $0.5 \log \left(\frac{1 + \Theta}{1 - \Theta} \right)$ e variância $\frac{1}{n-3}$. O erro padrão para $\hat{\zeta}$ é $\sqrt{\frac{1}{n-3}}$. Para o exemplo das faculdades, o valor é $\frac{1}{\sqrt{12}} = 0.289$.

A título de comparação com o procedimento bootstrap, a estatística (estimador) $\hat{\zeta}$ foi estimado em cada uma das $B = 3200$ amostras bootstrap. O desvio padrão das réplicas bootstrap resultou 0.290 (muito próximo ao valor teórico 0.289). Histogramas para as correlações $\hat{\theta}^*$ e para os $\hat{\zeta}^*$.

! Importante

Muitas fórmulas para os erros padrões são aproximações baseadas na teoria normal e isso “explica” os resultados próximos obtidos com o uso do bootstrap paramétrico que extrai amostras a partir da distribuição normal.

Vantagens do bootstrap sobre os métodos tradicionais:

- **bootstrap não-paramétrico:** não é necessário fazer suposições de modelos paramétricos para a população;
- **bootstrap paramétrico:** possibilita estimar erros padrões em problemas para os quais não há fórmulas para os erros padrões.

6.2 Jackknife

6.2.1 Introdução

Jackknife é uma técnica para estimar viés e erro padrão de estimadores. É uma técnica que antecede o bootstrap e foi proposta no trabalho pioneiro Quenouille (1949) para reduzir viés do estimador da correlação serial.

Formalização:

- $\tilde{X} = (X_1, X_2, \dots, X_n)$: amostra aleatória de uma f.d.a. F
- $\tilde{x} = (x_1, x_2, \dots, x_n)$: amostra aleatória observada de F
- $\Theta = t(F)$: parâmetro (Θ uma função de F)
- $\hat{\Theta} = s(\tilde{X})$: estimador para Θ
- $\hat{\theta} = s(\tilde{x})$: estimativa para Θ

O jackknife toma como base n amostras de tamanho $n - 1$ selecionadas da amostra aleatória observada. A i -ésima amostra jackknife consiste da amostra observada com a i -ésima observação removida, $i = 1, 2, \dots, n$:

$$x_{\sim(i)} = (x_1, x_2, \dots, x_{i-1}, x_{i+1}, \dots, x_n).$$

6.2.2 Estimador do viés

Para cada amostra i jackknife, é obtida a réplica jackknife $\hat{\theta}_{(i)} = s(x_{\sim(i)})$ e o estimador jackknife do viés de $\hat{\Theta}$ é dado por:

$$\widehat{\text{viés}}_{jack} = (n - 1)(\hat{\Theta}_{(\cdot)} - \hat{\Theta}),$$

em que $\hat{\Theta}_{(\cdot)} = \sum_{i=1}^n \frac{\hat{\Theta}_{(i)}}{n}$.

6.2.3 Estimado do erro padrão

O estimador jackknife do erro padrão de $\hat{\Theta}$ pode ser escrito da seguinte maneira:

$$\widehat{ep}_{jack} = \left[\frac{n-1}{n} \sum_{i=1}^n (\hat{\Theta}_{(i)} - \hat{\Theta}_{(\cdot)})^2 \right]^{\frac{1}{2}},$$

em que $\hat{\Theta}_{(\cdot)} = \sum_{i=1}^n \frac{\hat{\Theta}_{(i)}}{n}$.

Algumas considerações importantes:

- Bootstrap: amostragem aleatória com reposição;
- Jackknife: amostras fixas;
- Jackknife requer o cálculo do estimador apenas para n amostras;
- A acurácia do estimador jackknife do erro padrão depende de quão próximo o estimador é da linearidade. Para funções fortemente não lineares, o jackknife pode ser ineficiente;
- Estimador linear: $\hat{\Theta} = s(X) = \mu + \frac{1}{n} \sum_{i=1}^n \alpha(X_i)$.

6.2.3.1 Exemplo

Considere a amostra observada: $x = (10, 26, 30, 40, 48)$ e o estimador: $\text{mediana}(\hat{\Theta})$. As amostras e réplicas jackknife são dadas, respectivamente, por:

1. $x_{\sim(1)} = (26, 30, 40, 48)$ e $\hat{\theta}_{(1)} = 35$;
2. $x_{\sim(2)} = (10, 30, 40, 48)$ e $\hat{\theta}_{(2)} = 35$;
3. $x_{\sim(3)} = (10, 26, 40, 48)$ e $\hat{\theta}_{(3)} = 33$;
4. $x_{\sim(4)} = (10, 26, 30, 48)$ e $\hat{\theta}_{(4)} = 28$;
5. $x_{\sim(5)} = (10, 26, 30, 40)$ e $\hat{\theta}_{(5)} = 28$.

Desta forma, a estimativa jackknife do erro padrão de $\hat{\Theta}$:

$$\widehat{ep}_{jack} = \left[\frac{4}{5} \sum_{i=1}^5 (\hat{\theta}_{(i)} - \hat{\theta}_{(\cdot)})^2 \right]^{\frac{1}{2}} \approx 6.38,$$

$$\text{com } \hat{\theta}_{(\cdot)} = \sum_{i=1}^5 \frac{\hat{\theta}_{(i)}}{5} = 31.8.$$

A seguir, a sintaxe do R para calcular a estimativa jackknife do erro padrão para a mediana.

```
x <- c(10,26,30,40,48)
n <- length(x)

est.jack <- 0

for(i in 1:n){
  est.jack[i] <- median(x[-i])
}

ep.jack <- sqrt(((n-1)^2/n)*var(est.jack))
ep.jack
```

```
[1] 6.374951
```

6.3 Intervalos de Confiança

⚠ Notação

- $\underset{\sim}{X} = (X_1, x_2, \dots, X_n)$: amostra aleatória de uma f.d.a. F
- $\underset{\sim}{x} = (x_1, x_2, \dots, x_n)$: amostra aleatória observada de F
- $\Theta = t(F)$: parâmetro (Θ uma função de F)
- $\hat{\Theta} = s(\underset{\sim}{X})$: estimador para Θ
- $\hat{\theta} = s(\underset{\sim}{x})$: estimativa para Θ

6.4 Intervalos de Confiança Normal Padrão

Suponha que $\hat{\Theta} \sim N(\Theta, ep(\hat{\Theta})^2)$. Portanto, se $ep(\hat{\Theta})$ é conhecido temos que:

$$Z = \frac{\hat{\Theta} - \Theta}{ep(\hat{\Theta})} \sim N(0, 1),$$

e

$$IC_z(100(1 - \alpha)\%, \Theta) = \hat{\Theta} \pm z_{\frac{\alpha}{2}} ep(\hat{\Theta}).$$

6.5 Intervalo de Confiança t-Student

Suponha que $\hat{\Theta} \sim N(\Theta, ep(\hat{\Theta})^2)$ e se $ep(\hat{\Theta})$ é desconhecido, portanto

$$Z = \frac{\hat{\Theta} - \Theta}{\hat{ep}(\hat{\Theta})} \sim \text{t-Student}(n - 1),$$

e

$$IC_t(100(1 - \alpha)\%, \Theta) = \hat{\Theta} \pm t_{(n-1, \frac{\alpha}{2})} \hat{ep}(\hat{\Theta}).$$

6.6 IC bootstrap- t

Agora vamos definir

$$Z^{*b} = \frac{\hat{\Theta}^{*b} - \hat{\Theta}}{\hat{ep}^{*b}},$$

em que $\hat{\Theta}^{*b} = s(\tilde{X}^{*b})$ e \hat{ep}^{*b} são obtidos para cada amostra bootstrap \tilde{x}^* . O percentil $100 \cdot \alpha$ de Z^{*b} é estimado pelo valor $\hat{t}^{(\alpha)}$, tal que

$$\#\{Z^{*b} \leq \hat{t}^{(\alpha)}\} / B = \alpha.$$

Por exemplo, se $B = 100$ e a confiança para o intervalo é de 90%, a estimativa $\hat{t}^{(0.05)}$ será o quinto maior valor de Z^{*b} e a estimativa $\hat{t}^{(0.95)}$ será o nonagésimo quinto maior valor de Z^{*b} .

Este procedimento estima a distribuição de Z (quantidade pivotal) diretamente dos dados. Não é necessário a suposição teórica de normalidade e sua distribuição é a mesma para qualquer Θ . O intervalo (bilateral) bootstrap- t de confiança $100 \times (1 - 2\alpha)\%$ para o parâmetro Θ , denotado por $IC_t^*(100(1 - 2\alpha)\%, \Theta)$, é definido por:

$$[\hat{\Theta} - \hat{t}^{(1-\alpha)} \hat{ep}(\hat{\Theta}), \hat{\Theta} - \hat{t}^{(\alpha)} \hat{ep}(\hat{\Theta})].$$

Emprega-se estimativas *plug-in* para $\hat{\Theta}$ e $\hat{ep}(\hat{\Theta})$; não sendo possível, o erro padrão é estimado usando bootstrap ou jackknife.

! Importante

Se $B \cdot \alpha$ não resultar um número inteiro?

1. Efron & Tibshirani (1993): supondo $\alpha \leq 0.5$, faça $k = \text{Int}[(B + 1) \cdot \alpha]$ (o maior inteiro $\leq (B + 1) \cdot \alpha$) e defina os percentis empíricos de ordem $100 \cdot \alpha$ e $100 \cdot (1 - \alpha)$, respectivamente, pelo k -ésimo e $(B + 1 - k)$ -ésimo maiores valores de Z^{*b} .
2. Davidson & Hinkley (1997): interpolação dos percentis.

i Considerações

- Este intervalo pode ser fortemente influenciado por poucos valores discrepantes.
- Em geral, é necessário um valor de B muito superior a 200.
- Para grandes amostras, o IC bootstrap- t pode estar mais próximo do nível de cobertura desejado do que os IC Normal ou t-Student.

- Abaixo, percentis da distribuição t -Student com 8 gl, normal padrão e distribuição bootstrap de Z^{*b} (para o grupo controle no experimento com os ratos; $B = 1000$).

Tabela 6.5: Comparação dos percentis de diferentes distribuições.

percentil	5	10	90	95
t_8	-1.86	-1.40	1.40	1.86
Normal	-1.65	-1.28	1.28	1.65
bootstrap-t	-4.53	-2.01	1.19	1.53

O IC bootstrap- t para Θ (média do tempo de sobrevivência de ratos não submetidos ao tratamento (grupo controle)):

$$[56.22 - 1.53 \times 13.33, 56.22 + 4.53 \times 13.33] = [35.82, 116.74],$$

usado a estimativa *plug-in* para o erro padrão da média.



Considerações Finais

- os valores dos percentis bootstrap- t podem não ser simétricos em relação ao zero.
- aplicável para estatísticas de localização (média, mediana etc).
- Cautela no uso envolvendo pequenas amostras (limites do intervalo fora do espaço paramétrico).
- o IC_t^* não é *transformation-respecting* (ver próxima seção)

6.6.1 Intervalos de Confiança bootstrap percentil

O intervalo bootstrap percentil de confiança $100(1-2\alpha)\%$, denotado por $IC_p^*(100(1-2\alpha)\%, \Theta)$, é definido por

$$[\hat{\Theta}^{*(\alpha)}, \hat{\Theta}^{*(1-\alpha)}],$$

em que $\hat{\Theta}^{*(\alpha)}$ é o percentil de ordem 100α da distribuição bootstrap de $\hat{\Theta}^*$. Esta definição refere-se a um número infinito de réplicas.

Em situações práticas, o intervalo é aproximado com base nos resultados de B réplicas bootstrap:

$$\left[\hat{\Theta}_B^{*(\alpha)}, \hat{\Theta}_B^{*(1-\alpha)} \right],$$

em que $\hat{\Theta}_B^{*(\alpha)}$ é dado pelo percentil de ordem 100α dos valores de $\hat{\theta}_B^{*(b)}$, isto é, o $(B \cdot \alpha)$ -ésimo valor ordenado da lista das B réplicas de $\hat{\theta}_B^{*(b)}$.

i Nota

Se a distribuição de $\hat{\Theta}^*$ é aproximadamente normal, os intervalos percentil e normal padrão são próximos.

6.6.1.1 Exemplo

Para uma amostra: X_1, X_2, \dots, X_n , $n = 10$, de uma distribuição (população) normal padrão, considere o parâmetro de interesse Θ . $\Theta = \exp(\mu)$, sendo $\mu = 0$. Sendo a estimativa para Θ : $\hat{\theta} = \exp(\bar{x})$. Além disso, use o bootstrap não paramétrico: $B = 1000$ réplicas $\hat{\theta}^*$ usando a amostra observada: (1.669, -0.411, -0.322, 0.746, -0.868, -0.874, 1.011, -0.173, 0.021, 1.482). Resultando na estimativa para Θ : $\hat{\theta} = \exp(\bar{x}) = 1.256$.

Tabela 6.6: Percentis de $\hat{\theta}^*$ com base em 1000 réplicas bootstrap.

2.5%	5%	10%	16%	50%	84%	90%	95%	97.5%
0.74	0.81	0.89	0.95	1.25	1.65	1.77	2.00	2.13

Desta maneira, temos que

$$IC_p^*(95\%, \Theta) = [0.74, 2.13],$$

e

$$IC_p^*(90\%, \Theta) = [0.81, 2.00].$$

Abaixo um código R para o exemplo.

```
set.seed(1234)
n <- 10
x <- rnorm(n) # amostra observada
B <- 1000     # no. de réplicas bootstrap
theta <- 0
```

```

alfa <- c(0.025,0.05,0.10,0.16,0.50,0.84,0.90,0.95,0.975)

for(i in 1:B){
  a <- sample(x,n,replace=TRUE) # amostra bootstrap
  theta[i] <- exp(mean(a))      # réplica bootstrap
}

exp(mean(x)) # estimativa

```

```
[1] 0.6817056
```

```

sz <- sort(theta)
rbind(100*alfa, round(sz[c(B*alfa)],2))

```

```

      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
[1,]  2.50  5.00 10.00 16.00 50.00 84.00  90 95.0 97.5
[2,]  0.37  0.42  0.48  0.52  0.71  0.93   1  1.1  1.2

```

Agora iremos considerar outra abordagem. Seja $\Phi = \log(\Theta)$ e $\hat{\Phi} = \log(\hat{\Theta}) = \bar{X}$. Usando $IC_z(95\%, \Phi) = \hat{\Phi} \pm z_{\frac{\alpha}{2}} \hat{ep}(\hat{\Phi})$, resulta $[0.228 \pm 1.96 \times 0.28] = [-0.32, 0.78]$, em que $\hat{ep}(\hat{\Phi})$ é o estimador *plug-in* para o erro padrão de $\hat{\Phi}$. Fazendo a transformação inversa, o intervalo para Θ é $[0.73, 2.18]$. Note que o intervalo percentil para Θ assemelha-se ao intervalo normal padrão construído para uma transformação apropriada de Θ e transformado para a escala de Θ .

! Importante

Vantagem do método percentil: Não é necessário conhecer a transformação (normalizadora). O método percentil incorpora automaticamente a transformação adequada.

🔥 Lema

Suponha a transformação $\hat{\Phi} = m(\hat{\Theta})$ e, conseqüentemente, $\hat{\Phi} \sim N(\Phi, c^2)$, para algum erro padrão c . Então, o intervalo percentil para Θ será dado por

$$\left[m^{-1}(\hat{\Phi} - z_{\frac{\alpha}{2}}.c), m^{-1}(\hat{\Phi} + z_{\frac{\alpha}{2}}.c) \right].$$

Nota

- Em situações nas quais o uso de IC padrão é adequado*, o IC percentil produz resultados próximos ao do IC padrão.
- Em situações nas quais o uso do IC padrão é adequado apenas após uma transformação no parâmetro, a aplicação do método percentil automaticamente contempla essa transformação.

Considerações Finais

- Propriedade *transformation-respecting*: o intervalo percentil para qualquer transformação (monotônica) $\Phi = m(\Theta)$ do parâmetro Θ é dado por

$$\left[m(\hat{\Theta}^{*(\alpha)}), m(\hat{\Theta}^{*(1-\alpha)}) \right].$$

- A propriedade também é válida para o intervalo aproximado com base nos resultados de B réplicas bootstrap:

$$\left[m(\hat{\Theta}_B^{*(\alpha)}), m(\hat{\Theta}_B^{*(1-\alpha)}) \right].$$

- Propriedade *range-preserving*: produz intervalos com limites dentro do espaço paramétrico.

6.6.2 Intervalos de Confiança bootstrap - versões aprimoradas

O intervalo bootstrap- t apresenta boa probabilidade de cobertura teórica, mas tende a ser irregular na prática. Já o intervalo bootstrap percentil é menos irregular, mas apresenta probabilidade de cobertura menos satisfatória.

Proposta: intervalo bootstrap BC_a (*bias-corrected and accelerated*) e ABC (*approximate bootstrap confidence*).

substancial melhoria na prática e na teoria. correção de viés do estimador. o ABC exige menos esforço computacional do que o BC_a

7 Métodos de Monte Carlo

7.1 Introdução

7.2 Integração de Monte Carlo

7.3 Erro de Monte Carlo

7.4 Monte Carlo via Função de Importância

7.5 Método de Máxima Verossimilhança

8 Algoritmo EM

9 Métodos Adicionais

References

- Falk, Ruma. 2014. “A Closer Look at the Notorious Birthday Coincidences”. *Teaching Statistics* 36 (2): 41–46. <https://doi.org/10.1111/test.12014>.
- Hodgson, Ted, e Maurice Burke. 2000. “On Simulation and the Teaching of Statistics”. *Teaching Statistics* 22 (3): 91–96. <https://doi.org/10.1111/1467-9639.00033>.
- Martins, Rui Manuel Da Costa. 2018. “Learning the Principles of Simulation Using the Birthday Problem”. *Teaching Statistics* 40 (3): 108–11. <https://doi.org/10.1111/test.12164>.
- Matthews, Robert, e Fiona Stones. 1998. “Coincidences: the truth is out there”. *Teaching Statistics* 20 (1): 17–19. <https://doi.org/https://doi.org/10.1111/j.1467-9639.1998.tb00752.x>.
- Thomas, F. H., e J. L. Moore. 1980. “CUSUM: Computer Simulation for Statistics Teaching”. *Teaching Statistics* 2 (1): 23–28. <https://doi.org/10.1111/j.1467-9639.1980.tb00374.x>.
- Tintle, Nathan, Beth Chance, George Cobb, Soma Roy, Todd Swanson, e Jill VanderStoep. 2015. “Combating Anti-Statistical Thinking Using Simulation-Based Methods Throughout the Undergraduate Curriculum”. *The American Statistician* 69 (4): 362–70. <https://doi.org/10.1080/00031305.2015.1081619>.
- Zieffler, Andrew, e Joan B. Garfield. 2007. “Studying the Role of Simulation in Developing Students’ Statistical Reasoning”. Em *Proceedings of the 56th Session of the International Statistical Institute (ISI)*. International Statistical Institute.