

Estatística Computacional

Pedro Rafael Diniz Marinho

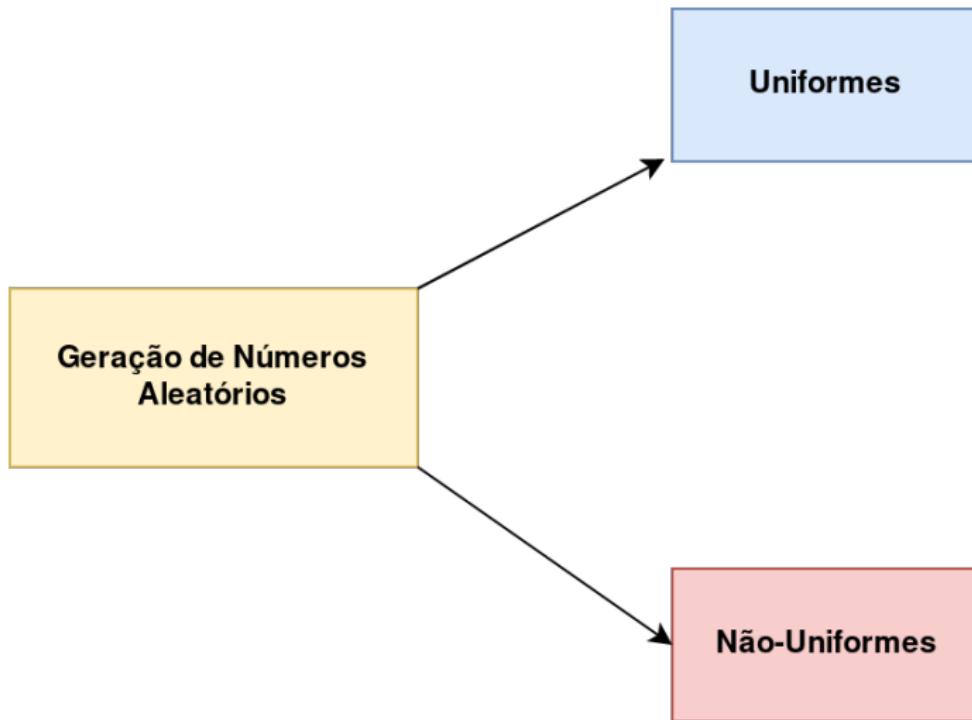
Universidade Federal da Paraíba
Departamento de Estatística

2019.1

Geração de Números Pseudo-Aleatórios



Geração de Números Pseudo-Aleatórios



Geração de Números Pseudo-Aleatórios

Nota: Diversos materiais usam o título **Geração de Números Aleatórios**. Trata-se de um “péssimo” título uma vez que resultados puramente aleatórios **não são** reproduzíveis.

Um título mais adequado seria **Sequência Pseudo-Aleatória de Números**. De toda forma, sempre que for mencionado o título **Geração de Números Aleatórios** entenda que se estar a falar sobre **Sequência Pseudo-Aleatória de Números**.

Importante: A sequência de números pseudo-aleatórios são reproduzíveis uma vez que tal sequência é obtida por meio de um algoritmo.

Geração de Números Pseudo-Aleatórios



Geração de Números Pseudo-Aleatórios



Nota: A entrada normalmente é chamada de **semente** do gerador de números pseudo-aleatórios.

Geração de Números Pseudo-Aleatórios



Nota: A entrada normalmente é chamada de **semente** do gerador de números pseudo-aleatórios.

Fato: A partir de sequências uniformes, podemos gerar sequências não-uniformes.

Geração de Números Pseudo-Aleatórios

Definição (Transformação Integral de Probabilidade): Seja X uma variável aleatória com função de distribuição F_X . A transformação de X tal que $U = F_X(X)$ é denominada **transformação integral de probabilidade**.

Geração de Números Pseudo-Aleatórios

Definição (Transformação Integral de Probabilidade): Seja X uma variável aleatória com função de distribuição F_X . A transformação de X tal que $U = F_X(X)$ é denominada **transformação integral de probabilidade**.

Observe que o uso da transformação acima depende da possibilidade de invertermos a função F . A função inversa tem domínio em $[0, 1]$. Porém, se F tiver saltos ou for escada, F não admitirá inversa. Dessa forma, utilizaremos a **função inversa generalizada** e que por abuso de notação será representada por F^{-1} .

Geração de Números Pseudo-Aleatórios

Definição (**Inversa Generalizada de F**): Seja F uma função de distribuição qualquer. A inversa generalizada denotada por F^{-1} é definida como:

Geração de Números Pseudo-Aleatórios

Definição (**Inversa Generalizada de F**): Seja F uma função de distribuição qualquer. A inversa generalizada denotada por F^{-1} é definida como:

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

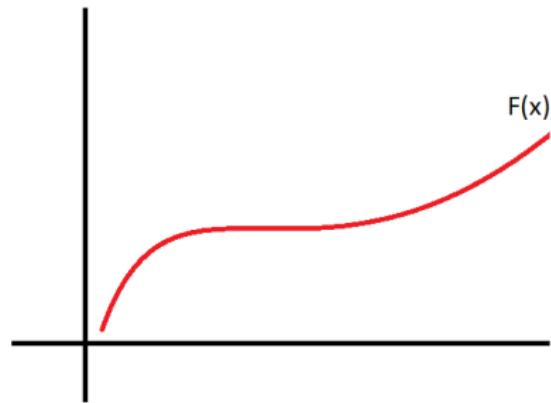
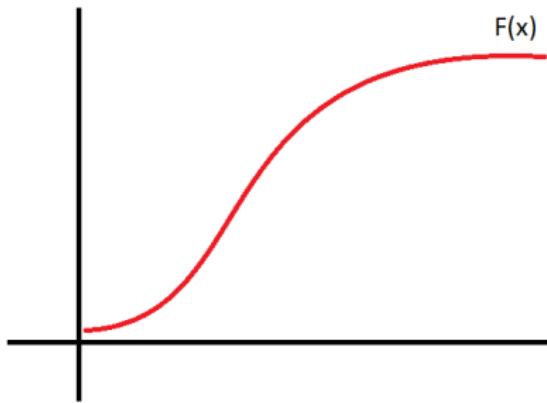
Geração de Números Pseudo-Aleatórios

Definição (**Inversa Generalizada de F**): Seja F uma função de distribuição qualquer. A inversa generalizada denotada por F^{-1} é definida como:

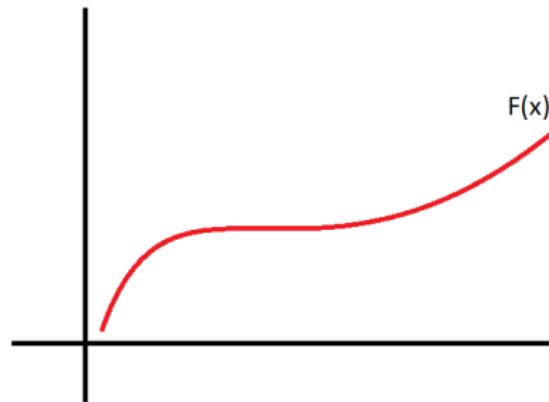
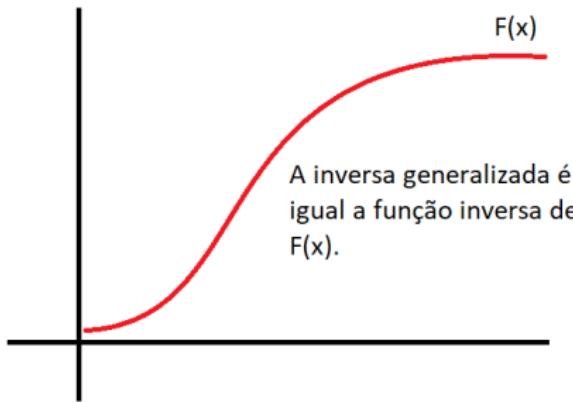
$$F^{-1}(u) = \inf\{x \in \mathbb{R} : F(x) \geq u\}.$$

Nota: Caso a função inversa de F exista no sentido usual, esta coincidirá com a função inversa generalizada de F .

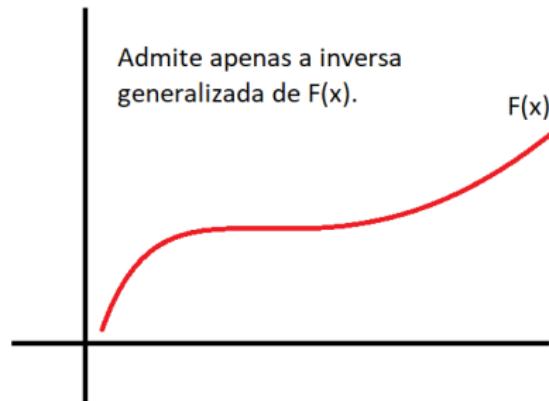
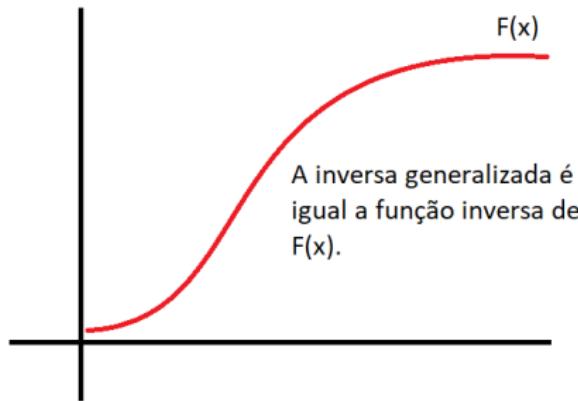
Geração de Números Pseudo-Aleatórios



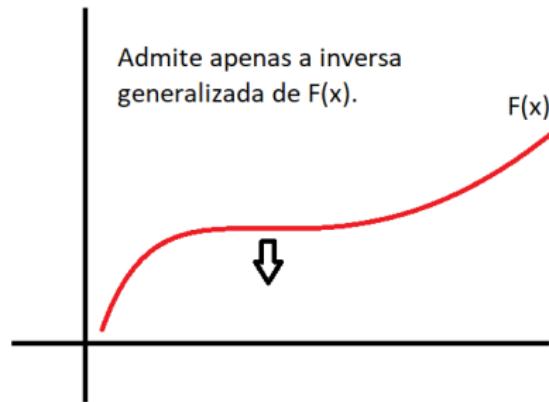
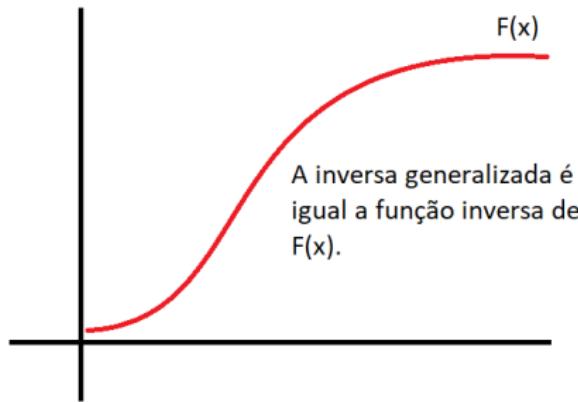
Geração de Números Pseudo-Aleatórios



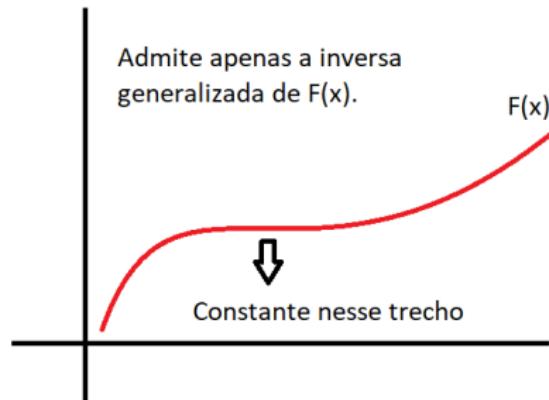
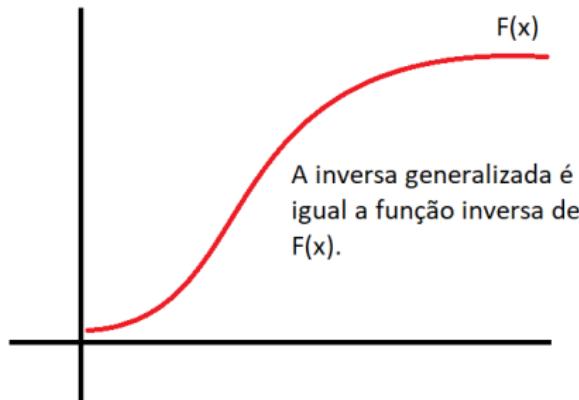
Geração de Números Pseudo-Aleatórios



Geração de Números Pseudo-Aleatórios



Geração de Números Pseudo-Aleatórios



Geração de Números Pseudo-Aleatórios

Observação Importante: A inversa generalizada cuida de situações em que a função F não é invertível.

Proposição: Seja X uma variável aleatória contínua com função de distribuição F . Sendo $U \sim U_c[0, 1]$, então $X = F_X^{-1}(U)$ terá função de distribuição F .

Geração de Números Pseudo-Aleatórios

Observação Importante: A inversa generalizada cuida de situações em que a função F não é invertível.

Proposição: Seja X uma variável aleatória contínua com função de distribuição F . Sendo $U \sim U_c[0, 1]$, então $X = F_X^{-1}(U)$ terá função de distribuição F .

Prova:

$$P(X \leq x) = P(F_X^{-1}(U) \leq x) = P(U \leq F_X(x)) = F(x).$$

Geração de Números Pseudo-Aleatórios

O método poderá ser aplicado para geração de observações de variáveis aleatórias contínuas ou discrete de uma determinada distribuição de probabilidade.

Algoritmo (Método da Inversão)

Geração de Números Pseudo-Aleatórios

O método poderá ser aplicado para geração de observações de variáveis aleatórias contínuas ou discrete de uma determinada distribuição de probabilidade.

Algoritmo (Método da Inversão)

- 1 Obtenha a inversa $F_X^{-1}(u)$.

Geração de Números Pseudo-Aleatórios

O método poderá ser aplicado para geração de observações de variáveis aleatórias contínuas ou discrete de uma determinada distribuição de probabilidade.

Algoritmo (Método da Inversão)

- 1 Obtenha a inversa $F_X^{-1}(u)$.
- 2 Escreva um comando ou função que calcule $F_X^{-1}(u)$.

Geração de Números Pseudo-Aleatórios

O método poderá ser aplicado para geração de observações de variáveis aleatórias contínuas ou discrete de uma determinada distribuição de probabilidade.

Algoritmo (Método da Inversão)

- 1 Obtenha a inversa $F_X^{-1}(u)$.
- 2 Escreva um comando ou função que calcule $F_X^{-1}(u)$.
- 3 Para cada observação de uma variável aleatória, faça:

Geração de Números Pseudo-Aleatórios

O método poderá ser aplicado para geração de observações de variáveis aleatórias contínuas ou discrete de uma determinada distribuição de probabilidade.

Algoritmo (Método da Inversão)

- 1 Obtenha a inversa $F_X^{-1}(u)$.
- 2 Escreva um comando ou função que calcule $F_X^{-1}(u)$.
- 3 Para cada observação de uma variável aleatória, faça:
 - a) Gere uma observação u de uma variável aleatória $U \sim U_c[0, 1]$.

Geração de Números Pseudo-Aleatórios

O método poderá ser aplicado para geração de observações de variáveis aleatórias contínuas ou discrete de uma determinada distribuição de probabilidade.

Algoritmo (Método da Inversão)

- 1 Obtenha a inversa $F_X^{-1}(u)$.
- 2 Escreva um comando ou função que calcule $F_X^{-1}(u)$.
- 3 Para cada observação de uma variável aleatória, faça:
 - a) Gere uma observação u de uma variável aleatória $U \sim U_c[0, 1]$.
 - b) Calcule $x = F_X^{-1}(u)$.

Geração de Números Pseudo-Aleatórios

Exercício: Considere a função densidade de probabilidade $f_X(x) = 3x^2$, $0 \leq x \leq 1$. Obtenha utilizando o algoritmo acima uma sequência de 1000 números pseudo-aleatórios com distribuição $F_X(x)$.

Geração de Números Pseudo-Aleatórios

Solução:

Temos que $F_X(x) = \int_0^x f_X(x)dx = x^3$, com $0 \leq x \leq 1$. Fazendo $u = x^3$, temos que

$$F_X^{-1}(u) = u^{1/3}, \quad 0 \leq u \leq 1.$$

Utilizaremos o algoritmo apresentado e $F_X^{-1}(u)$ obtido logo acima para obter uma sequência de observações x_1, \dots, x_n .

Geração de Números Pseudo-Aleatórios

Código R:

```
1: # Função densidade de X.  
2: pdf_f <- function(x){  
3:   3 * x^2  
4: }  
5:  
6: # Gerando mil números pseudo-aleatórios  
7: # com distribuição U[0,1].  
8: set.seed(0); u <- runif(n = 1000, min = 0, max = 1)  
9:  
10: # Aplicando u à inversa de F_X(x).  
11: x <- u^(1/3)
```

Geração de Números Pseudo-Aleatórios

```
12: # Histograma da densidade da amostra.  
13: hist(x, prob = TRUE, xlab = "Dominio",  
14:         ylab = "Densidade",  
15:         main = expression(f(x)==3*x^2))  
16: dominio <- seq(from=0, to = 1, by = .01)  
17: lines(dominio, pdf_f(dominio), lwd = 2, col = "red")
```

Será que deu certo?

Geração de Números Pseudo-Aleatórios

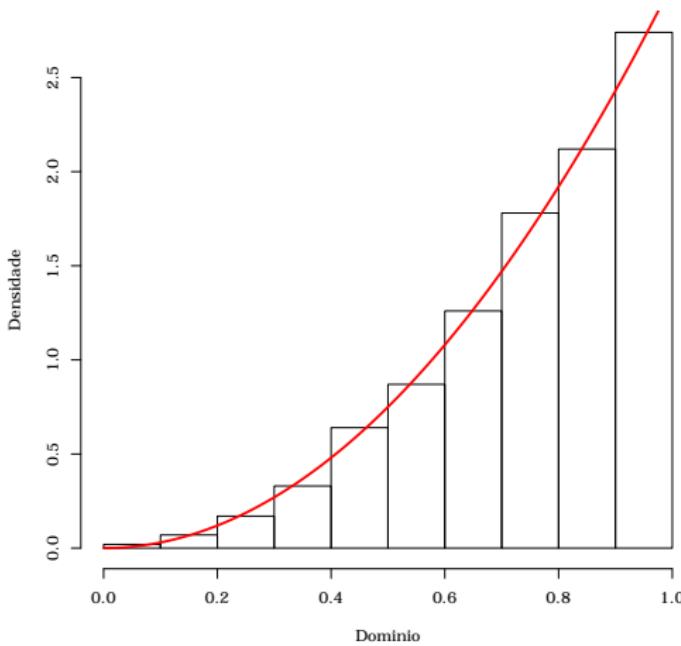
```
12: # Histograma da densidade da amostra.  
13: hist(x, prob = TRUE, xlab = "Dominio",  
14:         ylab = "Densidade",  
15:         main = expression(f(x)==3*x^2))  
16: dominio <- seq(from=0, to = 1, by = .01)  
17: lines(dominio, pdf_f(dominio), lwd = 2, col = "red")
```

Será que deu certo?

**Observemos o ajustamento da distribuição
 $f_X(x) = 3x^2$, $0 \leq x \leq 1$ aos dados gerados.**

Geração de Números Pseudo-Aleatórios

$$f(x) = 3x^2$$



Geração de Números Pseudo-Aleatórios

Nota: Na Figura acima, o título possui uma expressão matemática. Este título é obtido especificando no argumento `main` a função `expression()` com a expressão matemática desejada. Consulte a documentação da linguagem para maiores detalhes.

Observação: O método da inversão é o algoritmo mais geral e utilizado para geração de números pseudo-aleatórios de uma distribuição qualquer.

Exercício: Seja $X \sim \text{Exp}(\lambda)$, tal que $f_X(x) = \lambda e^{-\lambda x}$, com $x > 0$ e $\lambda > 0$. Sem utilizar a função `rexp()`, construa uma função para geração de números pseudo-aleatórios de uma distribuição $\text{Exp}(\lambda)$.

Geração de Números Pseudo-Aleatórios

Exercício: É possível fazer uso do método da inversão para implementarmos uma função que faz uso do método da inversão para geração de números pseudo-aleatórios de uma variável aleatória $X \sim \mathcal{N}(\mu, \sigma^2)$? Justifique sua resposta.

Geração de Números Pseudo-Aleatórios

Para gerarmos números pseudo-aleatórios de uma distribuição qualquer, foi preciso gerar números pseudo-aleatórios proveniente de uma distribuição uniforme.

Definição: Um gerador de números pseudo-aleatórios é um **algoritmo** que iniciando em um valor (**semente**) e usando uma transformação determinística D , produz uma sequência $u_i = D^i(u_0)$ de valores em $(0, 1)$.

Observação: O conhecimento de u_1, \dots, u_n não deverá conduzir ao conhecimento de u_{n+1}, u_{n+2}, \dots . Dessa forma, desejamos que a sequência não poderá ser previsível.

Geração de Números Pseudo-Aleatórios

Por exemplo, se gerarmos uma sequência $\{0, 1, \dots, n\}$ (com n muito grande) e ao final dividirmos cada número por n , teremos uma sequência em $[0, 1]$, mas nesse caso há previsibilidade.

Muito Importante

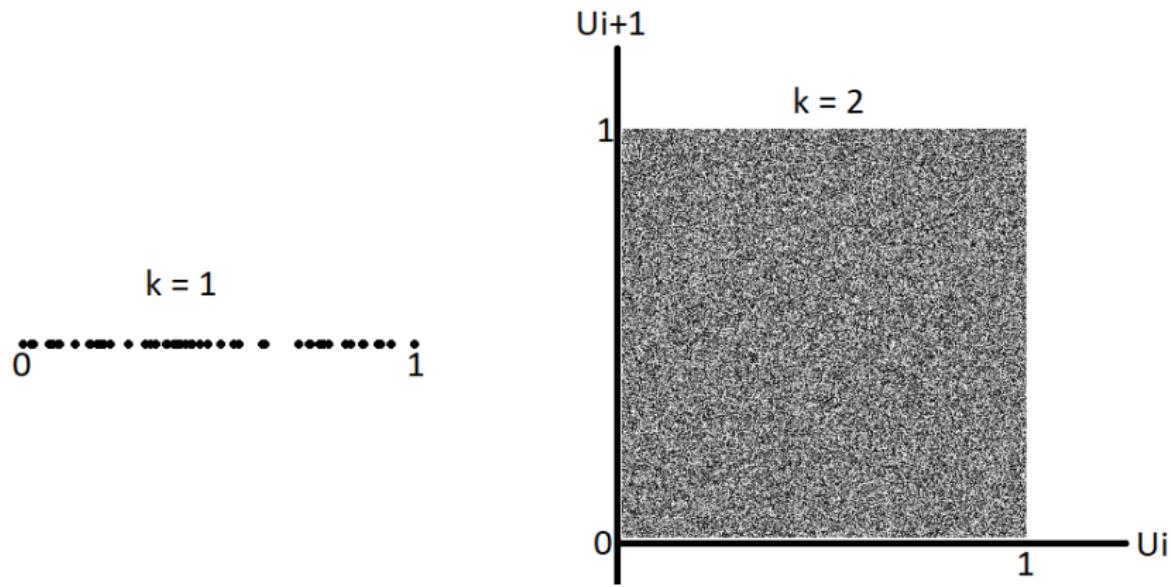
Um bom gerador de números pseudo-aleatórios com distribuição uniforme não permite a detecção de qualquer padrão em $[0, 1]^k$, com $k = 1, \dots, 6$, isto é, não é possível detectar padrões até a sexta dimensão. Em outras palavras, é preciso que a uniformidade esteja presente até a sexta dimensão.

Geração de Números Pseudo-Aleatórios

Por exemplo, para $k = 1$ e $k = 2$, desejamos:

Geração de Números Pseudo-Aleatórios

Por exemplo, para $k = 1$ e $k = 2$, desejamos:



Geração de Números Pseudo-Aleatórios

**Porém, lembre-se, precisamos de uniformidade
até a sexta dimensão.**

Geração de Números Pseudo-Aleatórios



Figura: John von Neumann.

Um dos primeiros algoritmos de número pseudo-aleatório (**Midsquare**) foi idealizado pelo matemático húngaro (naturalizado americano) **John von Neumann** (1903 - 1957).

von Neumann deixou contribuições em diversas áreas como ciência da computação, economia, teoria dos jogos, análise funcional, teoria dos conjuntos, entre outras. von Neumann foi um dos construtores do primeiro computador digital eletrônico (ENIAC).

Geração de Números Pseudo-Aleatórios

Algoritmo Midsquare de von Neumann (~ 1949):

Geração de Números Pseudo-Aleatórios

Algoritmo Midsquare de von Neumann (~ 1949):

- 1 Tome uma semente: um número inteiro de 4 dígitos.

Geração de Números Pseudo-Aleatórios

Algoritmo Midsquare de von Neumann (~ 1949):

- 1 Tome uma semente: um número inteiro de 4 dígitos.
- 2 Calcule o quadrado desse número e preencha, se necessário, com zero à esquerda, para que o número gerado tenha 8 dígitos.

Geração de Números Pseudo-Aleatórios

Algoritmo Midsquare de von Neumann (~ 1949):

- 1 Tome uma semente: um número inteiro de 4 dígitos.
- 2 Calcule o quadrado desse número e preencha, se necessário, com zero à esquerda, para que o número gerado tenha 8 dígitos.
- 3 Repita os passos 1 e 2 a quantidade de vezes desejadas e ao final divida cada número obtido, incluindo a semente por 10 mil.

Geração de Números Pseudo-Aleatórios

Exercício: Construa, em R, uma função que implemente o gerador Midsquare. **Dica:** Talvez seja necessário manipular strings para efetuar o passo 2 do algoritmo. As funções `strsplit()` e `paste()` podem ser úteis. Procurem maiores detalhes na documentação da linguagem.

Geração de Números Pseudo-Aleatórios

Solução:

```
1: midsquare <- function(n = 1, semente = 1987){  
2:   valores <- NULL; i = 1  
3:   repeat{  
4:     semente <- semente^2  
5:     quadrado <- unlist(strsplit(as.character(semente),  
6:                               split=""))  
7:     quadrado <- c(rep(0,8 - length(quadrado)),  
8:                   quadrado)  
9:     valores[i] <- as.numeric(paste(quadrado[3:6],  
10:                                collapse = ""))
```

Geração de Números Pseudo-Aleatórios

```
12:     semente <- valores[i]
13:     i <- i + 1
14:     if (i > n) break
15: }
16: valores
17:}
18: midsquare(n=4, semente = 1348)
#> [1] 8171 7652 5531 5919
```

Geração de Números Pseudo-Aleatórios

Exercício: O que acontece se um dado valor gerado $x_i = 0$? Discutam.

Geração de Números Pseudo-Aleatórios

Exercício: O que acontece se um dado valor gerado $x_i = 0$? Discutam.

Resposta: A sequência fica “presa” no zero, isto é:

$$x_{i+1} = x_{i+2} = \dots = 0.$$

Geração de Números Pseudo-Aleatórios

Exercício: O que acontece se um dado valor gerado $x_i = 0$? Discutam.

Resposta: A sequência fica “presa” no zero, isto é:

$$x_{i+1} = x_{i+2} = \dots = 0.$$

Exercício: Qual o problema do algoritmo ao considerar a semente ($x_0 = 3792$)? Discutam.

Geração de Números Pseudo-Aleatórios

Exercício: O que acontece se um dado valor gerado $x_i = 0$? Discutam.

Resposta: A sequência fica “presa” no zero, isto é:

$$x_{i+1} = x_{i+2} = \dots = 0.$$

Exercício: Qual o problema do algoritmo ao considerar a semente ($x_0 = 3792$)? Discutam.

Resposta: A sequência não é aparentemente aleatória.

Geração de Números Pseudo-Aleatórios

Exercício: Qual o problema do algoritmo ao considerar $x_0 = 2100$?

Geração de Números Pseudo-Aleatórios

Exercício: Qual o problema do algoritmo ao considerar $x_0 = 2100$?

Resposta: O ciclo do gerador é muito curto.

Geração de Números Pseudo-Aleatórios

Exercício: Qual o problema do algoritmo ao considerar $x_0 = 2100$?

Resposta: O ciclo do gerador é muito curto.

Queremos geradores que possuam ciclos longos.

Geração de Números Pseudo-Aleatórios

Exercício: Qual o problema do algoritmo ao considerar $x_0 = 2100$?

Resposta: O ciclo do gerador é muito curto.

Queremos geradores que possuam ciclos longos.

Sendo assim, precisaremos de um algoritmo melhor...

Geração de Números Pseudo-Aleatórios

Gerador Congruencial:

$$x_i = (a * x_{i-1} + c) \bmod M, i = 1, 2, \dots,$$

em que x_0 é a semente do gerador, a é o multiplicador, c é o deslocamento e M é o módulo.

Observação: Usualmente, tem-se que:

$$a, c, x_i \in \{0, 1, 2, \dots, M - 1\}.$$

Geração de Números Pseudo-Aleatórios

Gerador Congruencial:

$$x_i = (a * x_{i-1} + c) \bmod M, i = 1, 2, \dots,$$

em que x_0 é a semente do gerador, a é o multiplicador, c é o deslocamento e M é o módulo.

Observação: Usualmente, tem-se que:

$$a, c, x_i \in \{0, 1, 2, \dots, M - 1\}.$$

Definição: Chamaremos de **período** de um gerador ao número de termos gerador a partir de uma semente x_0 sem a sequência até então gerada se repetir.

Geração de Números Pseudo-Aleatórios

Nota: $a \bmod b$ representa o **resto da divisão** entre a e b .

Terminologias:

Geração de Números Pseudo-Aleatórios

Nota: $a \bmod b$ representa o **resto da divisão** entre a e b .

Terminologias:

- Se $c \neq 0$, diremos que o gerador é **misto** (período máximo igual à M).

Geração de Números Pseudo-Aleatórios

Nota: $a \bmod b$ representa o **resto da divisão** entre a e b .

Terminologias:

- Se $c \neq 0$, diremos que o gerador é **misto** (período máximo igual à M).
- Se $c = 0$, diremos que o gerador é **multiplicativo** (período máximo igual à $M - 1$). Nesse caso, excluímos o valor 0 da sequência.

Geração de Números Pseudo-Aleatórios

Exercício: Utilizando $M = 64$, $x_0 = 1$, $a = 19$ e $c = 15$, gere uma sequência de 10 números pseudo-aleatórios obtido pelo gerador congruencial.

Geração de Números Pseudo-Aleatórios

Exercício: Utilizando $M = 64$, $x_0 = 1$, $a = 19$ e $c = 15$, gere uma sequência de 10 números pseudo-aleatórios obtido pelo gerador congruencial.

Exercício: Utilizando a linguagem R, implemente uma função para o algoritmo do gerador congruencial.

Geração de Números Pseudo-Aleatórios

```
1: rcongruencial <- function(n = 10, semente = 1987,
2:                               parametros, unif = TRUE){
3:
4:   a <- parametros[1]; c <- parametros[2]
5:   M <- parametros[3]
6:
7:   i <- 2; vetor <- NULL;
8:   vetor[1] <- semente
9:
10:  repeat{
11:    vetor[i] <- (a * vetor[i-1] + c) %% M
12:    i <- i + 1
13:
14:    if(i > n + 1) break
15:  }
```

Geração de Números Pseudo-Aleatórios

```
16: ifelse(unif == TRUE, vetor <- vetor[-1]/M,
17:         vetor <- vetor[-1])
18:         vetor
19:}
20:
21:rcongruencial(n = 10, semente = 1,
22:                  parametros = c(19,15,64), unif = F)
#> [1] 34 21 30 9 58 29 54 17 18 37
```

Geração de Números Pseudo-Aleatórios

Na linguagem C, a função `rand()` da biblioteca padrão **stdlib.h** utiliza-se do gerador. Mais especificamente, alguns compiladores de C, consideram:

Geração de Números Pseudo-Aleatórios

Na linguagem C, a função `rand()` da biblioteca padrão **stdlib.h** utiliza-se do gerador. Mais especificamente, alguns compiladores de C, consideram:

- 1 **gcc**: M^{32} , $a = 69069$, $c = 5$ (misto);

Geração de Números Pseudo-Aleatórios

Na linguagem C, a função `rand()` da biblioteca padrão **stdlib.h** utiliza-se do gerador. Mais especificamente, alguns compiladores de C, consideram:

- 1 **gcc**: M^{32} , $a = 69069$, $c = 5$ (misto);
- 2 **Microsoft**: M^{32} , $a = 214013$, $c = 2531011$ (misto);

Geração de Números Pseudo-Aleatórios

Na linguagem C, a função `rand()` da biblioteca padrão **stdlib.h** utiliza-se do gerador. Mais especificamente, alguns compiladores de C, consideram:

- 1 **gcc**: M^{32} , $a = 69069$, $c = 5$ (misto);
- 2 **Microsoft**: M^{32} , $a = 214013$, $c = 2531011$ (misto);
- 3 **Borland**: M^{32} , $a = 22695477$, $c = 1$ (misto).

Observação: Em C, a função `rand()` gera números entre 0 e `RAND_MAX` (constante simbólica que representa o maior inteiro representável pelo computador).

Geração de Números Pseudo-Aleatórios

Na linguagem de programação Ox, desenvolvida por Jurgen Doornik há o gerador de **Park-Miller** que é um gerador de números pseudo-aleatório congruencial multiplicativo. Nesse gerador, considera-se:

$$M = 2^{32} - 1, a = 16807, c = 0.$$

Geração de Números Pseudo-Aleatórios

Na linguagem de programação Ox, desenvolvida por Jurgen Doornik há o gerador de **Park-Miller** que é um gerador de números pseudo-aleatório congruencial multiplicativo. Nesse gerador, considera-se:

$$M = 2^{32} - 1, a = 16807, c = 0.$$

Comentário particular do professor: Não aconselharia o uso da linguagem Ox por se tratar de uma linguagem fechada mantida especialmente por uma única pessoa. Maiores detalhes em <http://www.doornik.com/>. Muito embora Ox é vendida como uma linguagem eficiente, temos a disposição diversas outras linguagem ainda mais eficientes e livres como é o caso de C e C++. Além disso, linguagem como, por exemplo, C e C++ conversam facilmente com outras linguagem de programação.

Geração de Números Pseudo-Aleatórios

Gerador Randu:

$$x_{i+1} = 65539 * x_i \bmod 31.$$

Observação: Há diversos materiais falando sobre este gerador em que podemos citar Donald E. Knuth, *The Art of Computer Programming*, Volume 2: *Seminumerical Algorithms*, 3rd edition (Addison-Wesley, Boston, 1998).

Nota: O gerador randu é um gerador congruencial que foi utilizado por muito tempo em manframes entre as décadas de 60 e 70. Porém, este gerador possui erros consideráveis e com o tempo ele foi deixado de lado. **Ele é considerado um dos piores algoritmos geradores de números pseudo-aleatórios já criado.**

Geração de Números Pseudo-Aleatórios

Exercício: Gere uma sequência de números pseudo-aleatórios utilizando o gerador randu.

Exercício: Implemente, utilizando a linguagem R, uma função para geração de números pseudo-aleatórios utilizando o gerador randu.

Geração de Números Pseudo-Aleatórios

Solução:

```
1: rrandu <- function(n, semente){  
2:   vetor <- NULL  
3:   vetor[1] <- semente  
4:   i <- 2  
5:   repeat{  
6:     vetor[i] <- (65539 * vetor[i-1]) %% 2^(31)  
7:     i <- i + 1  
8:     if (i > n+1) break  
9:   }  
10:  vetor[-1]  
11:}
```

Geração de Números Pseudo-Aleatórios

Exemplo: Corra o código abaixo e descreva o motivo pelo qual o gerador não é razoável. Explique!

```
1: a <- NULL
2: for(i in 1:10)
3:   a[i] <- rrandu(n=1,semente=i)
4: plot(a, xlab = "x_i", ylab = "x_{i+1}",
5:       main = "Gerador Randu",
6:       pch = 16)
```

Geração de Números Pseudo-Aleatórios

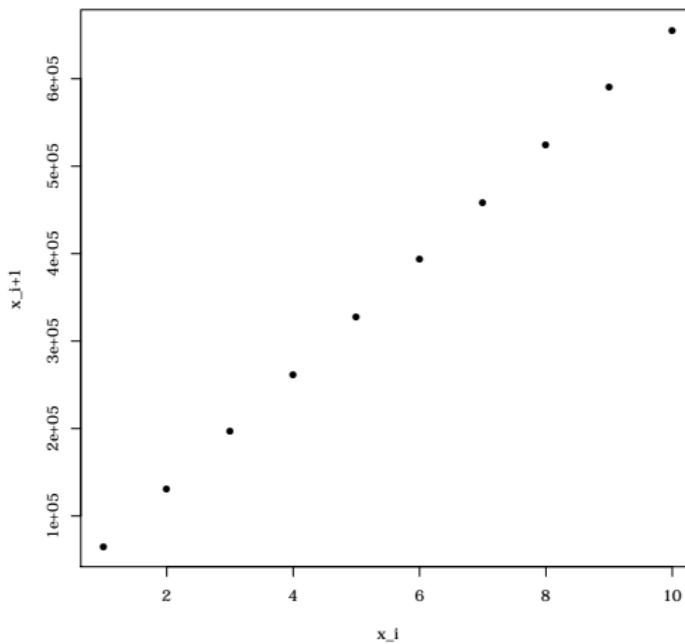
Exemplo: Corra o código abaixo e descreva o motivo pelo qual o gerador não é razoável. Explique!

```
1: a <- NULL  
2: for(i in 1:10)  
3:   a[i] <- rrandu(n=1,semente=i)  
4: plot(a, xlab = "x_i", ylab = "x_{i+1}",  
5:       main = "Gerador Randu",  
6:       pch = 16)
```

Observando o gráfico de x_i por x_{i+1} , é fácil perceber o comportamento previsível do gerador randu. Trata-se de uma característica muito indesejável em um gerador.

Geração de Números Pseudo-Aleatórios

Comportamento previsível do Randu



Propriedades de Geradores Congruenciais

Notação: $a \equiv b \pmod{M}$.

Lê-se: “ a é congruente para b módulo M ”.

Tal notação apresentada acima significa: $a - b$ é divisível por M .

Exemplo:

Propriedades de Geradores Congruenciais

Notação: $a \equiv b \pmod{M}$.

Lê-se: “ a é congruente para b módulo M ”.

Tal notação apresentada acima significa: $a - b$ é divisível por M .

Exemplo:

a) $10 \equiv 4 \pmod{2}$;

Propriedades de Geradores Congruenciais

Notação: $a \equiv b \pmod{M}$.

Lê-se: “ a é congruente para b módulo M ”.

Tal notação apresentada acima significa: $a - b$ é divisível por M .

Exemplo:

- a) $10 \equiv 4 \pmod{2}$;
- b) $7 \equiv 1 \pmod{3}$;

Propriedades de Geradores Congruenciais

Notação: $a \equiv b \pmod{M}$.

Lê-se: “ a é congruente para b módulo M ”.

Tal notação apresentada acima significa: $a - b$ é divisível por M .

Exemplo:

- a) $10 \equiv 4 \pmod{2}$;
- b) $7 \equiv 1 \pmod{3}$;
- c) $-10 \equiv 2 \pmod{2}$;

Propriedades de Geradores Congruenciais

Notação: $a \equiv b \pmod{M}$.

Lê-se: “ a é congruente para b módulo M ”.

Tal notação apresentada acima significa: $a - b$ é divisível por M .

Exemplo:

- a) $10 \equiv 4 \pmod{2}$;
- b) $7 \equiv 1 \pmod{3}$;
- c) $-10 \equiv 2 \pmod{2}$;
- d) $10 \equiv 0 \pmod{5}$.

Propriedades de Geradores Congruenciais

Definição: a é raiz primitiva de M , se e só se:

Propriedades de Geradores Congruenciais

Definição: a é raiz primitiva de M , se e só se:

- c1) $a^{M-1} - 1 \equiv 0 \pmod{M}$, isto é, $a^{M-1} - 1$ é divisível por M ;

Propriedades de Geradores Congruenciais

Definição: a é raiz primitiva de M , se e só se:

- c1) $a^{M-1} - 1 \equiv 0 \pmod{M}$, isto é, $a^{M-1} - 1$ é divisível por M ;
- c2) Para todo inteiro positivo I tal que $I < M - 1$, temos que $(a^I - 1)/M$ **não** é inteiro.

Propriedades de Geradores Congruenciais

Teorema (Gerador linear congruencial multiplicativo): Seja M um número primo. O gerador congruencial multiplicativo tem **período completo**, isto é, $M - 1$, se e somente se, a (multiplicador) é raiz primitiva de M .

Propriedades de Geradores Congruenciais

Teorema (Gerador linear congruencial multiplicativo): Seja M um número primo. O gerador congruencial multiplicativo tem **período completo**, isto é, $M - 1$, se e somente se, a (multiplicador) é raiz primitiva de M .

Exercício: Postule um gerador congruencial linear considerando $M = 13$ e aplique o teorema acima para determinar se seu gerador possui período completo igual à 12.

Propriedades de Geradores Congruenciais

Exercício: Considere o gerador:

$$x_i = a * x_{i-1} \bmod 11.$$

Considere também $a = 2, 3, \dots, 10$. Para quais valores de a somos conduzidos a período completo igual à 10?

Propriedades de Geradores Congruenciais

Exercício: Considere o gerador:

$$x_i = a * x_{i-1} \bmod 11.$$

Considere também $a = 2, 3, \dots, 10$. Para quais valores de a somos conduzidos a período completo igual à 10?

Resposta: $a \in \{2, 6, 7, 8\}$ conduzem o gerador $x_i = a * x_{i-1} \bmod 11$ a ter período completo.

Propriedades de Geradores Congruenciais

Conhecendo uma raiz primitiva de M , é possível facilmente obter outros valores de a que conduzem a período completo considerando a regra abaixo:

Regra: Se a é raiz primitiva de M , então $b = a^k \text{ mod } M$ também é raiz primitiva de M se, e somente se, k e $M - 1$ forem primos relativos (**coprimos**), isto é, se o único divisor comum entre eles é o número 1.

Propriedades de Geradores Congruenciais

Conhecendo uma raiz primitiva de M , é possível facilmente obter outros valores de a que conduzem a período completo considerando a regra abaixo:

Regra: Se a é raiz primitiva de M , então $b = a^k \text{ mod } M$ também é raiz primitiva de M se, e somente se, k e $M - 1$ forem primos relativos (**coprimos**), isto é, se o único divisor comum entre eles é o número 1.

Exercício: Voltando ao exercício anterior, supomos que conhecemos que $a = 2$ é raiz primitiva de $M = 11$. Utilize a regra a cima para obter os valores de a que conduzem à período completo.

Propriedades de Geradores Congruenciais

Resposta:

Propriedades de Geradores Congruenciais

Resposta:

- $2^1 \bmod 11 = 2;$

Propriedades de Geradores Congruenciais

Resposta:

- $2^1 \text{ mod } 11 = \mathbf{2};$
- $2^3 \text{ mod } 11 = \mathbf{8};$

Propriedades de Geradores Congruenciais

Resposta:

- $2^1 \bmod 11 = 2;$
- $2^3 \bmod 11 = 8;$
- $2^7 \bmod 11 = 7;$

Propriedades de Geradores Congruenciais

Resposta:

- $2^1 \bmod 11 = \mathbf{2};$
- $2^3 \bmod 11 = \mathbf{8};$
- $2^7 \bmod 11 = \mathbf{7};$
- $2^9 \bmod 11 = \mathbf{6}.$

Propriedades de Geradores Congruenciais

Resposta:

- $2^1 \bmod 11 = 2;$
- $2^3 \bmod 11 = 8;$
- $2^7 \bmod 11 = 7;$
- $2^9 \bmod 11 = 6.$

Assim, $a \in \{2, 6, 7, 8\}$ conduz a período completo.

Propriedades de Geradores Congruenciais

Teorema (Gerador congruencial linear misto, $c \neq 0$): Considere o gerador congruencial linear misto $x_i = (a * x_{i-1} + c) \bmod M$, com $c > 0$ (inteiro). Para $M = 2^\beta$ (β inteiro positivo), o gerador possui período completo (isto é, 2^β), se e somente se:

Propriedades de Geradores Congruenciais

Teorema (Gerador congruencial linear misto, $c \neq 0$): Considere o gerador congruencial linear misto $x_i = (a * x_{i-1} + c) \bmod M$, com $c > 0$ (inteiro). Para $M = 2^\beta$ (β inteiro positivo), o gerador possui período completo (isto é, 2^β), se e somente se:

- c1) $a \equiv 1 \pmod{4}$ ($a - 1$ é divisível por 4);

Propriedades de Geradores Congruenciais

Teorema (Gerador congruencial linear misto, $c \neq 0$): Considere o gerador congruencial linear misto $x_i = (a * x_{i-1} + c) \bmod M$, com $c > 0$ (inteiro). Para $M = 2^\beta$ (β inteiro positivo), o gerador possui período completo (isto é, 2^β), se e somente se:

- c1) $a \equiv 1 \pmod{4}$ ($a - 1$ é divisível por 4);
- c2) $\text{mod}(c, M) = 1$ (c e M são primos relativos, isto é, coprimos).

Propriedades de Geradores Congruenciais

Teorema (Gerador congruencial linear misto, $c \neq 0$): Considere o gerador congruencial linear misto $x_i = (a * x_{i-1} + c) \bmod M$, com $c > 0$ (inteiro). Para $M = 2^\beta$ (β inteiro positivo), o gerador possui período completo (isto é, 2^β), se e somente se:

- c1) $a \equiv 1 \pmod{4}$ ($a - 1$ é divisível por 4);
- c2) $\text{mod}(c, M) = 1$ (c e M são primos relativos, isto é, coprimos).

Nota: Qualquer valor ímpar de c satisfaz a propriedade c2. Além disso, não é necessário que M seja um número primo.

Propriedades de Geradores Congruenciais

Exemplo: O gerador utilizado pelo compilador **gcc** da linguagem C é um gerador congruencial linear, tal que:

- $a = 69069$;
- $c = 5$ (é um gerador misto e satisfaz a propriedade c2);
- $M = 2^{32}$ (M não é primo).

Gerador Mersenne Twister

O gerador **Mersenne Twister** foi proposto por Makoto Matsumoto e Takuji Nishimura, 1998. O gerador é de longe o mais utilizado na computação e em estudos de simulações. Sendo assim, o Mersenne Twister é, normalmente, o gerador de números pseudo-aleatórios padrão de diversas linguagens, softwares e bibliotecas como é o caso das linguagens **R**, **Julia**, **Python**, **Ruby**, das bibliotecas **GNU Scientific Library (GSL)**, **GNU Multiple Precision Arithmetic Library**, **Cuda** e de softwares como **Mathematica** e **Maple**.

Gerador Mersenne Twister

Algumas características do gerador Mersenne Twister são:

Gerador Mersenne Twister

Algumas características do gerador Mersenne Twister são:

- Não é um gerador congruencial linear;

Gerador Mersenne Twister

Algumas características do gerador Mersenne Twister são:

- Não é um gerador congruencial linear;
- Produz observações geradas de forma pseudo-aleatória com alta qualidade;

Gerador Mersenne Twister

Algumas características do gerador Mersenne Twister são:

- Não é um gerador congruencial linear;
- Produz observações geradas de forma pseudo-aleatória com alta qualidade;
- É um gerador rápido;

Gerador Mersenne Twister

Algumas características do gerador Mersenne Twister são:

- Não é um gerador congruencial linear;
- Produz observações geradas de forma pseudo-aleatória com alta qualidade;
- É um gerador rápido;
- Passou por diversos testes estatísticos para mensurar sua qualidade;

Gerador Mersenne Twister

Algumas características do gerador Mersenne Twister são:

- Não é um gerador congruencial linear;
- Produz observações geradas de forma pseudo-aleatória com alta qualidade;
- É um gerador rápido;
- Passou por diversos testes estatísticos para mensurar sua qualidade;
- Tem período de ocorrência $2^{19937} - 1$;

Gerador Mersenne Twister

Algumas características do gerador Mersenne Twister são:

- Não é um gerador congruencial linear;
- Produz observações geradas de forma pseudo-aleatória com alta qualidade;
- É um gerador rápido;
- Passou por diversos testes estatísticos para mensurar sua qualidade;
- Tem período de ocorrência $2^{19937} - 1$;
- É baseado em números primos de Mersenne, isto é, se a é um número primo e $M_a = 2^a - 1$ também é, diremos que M_a é número primo de Mersenne.

Gerando Normal

Pelo método da inversão, necessitamos conhecer F_X^{-1} para gerarmos observações de X . Computacionalmente não há problemas em utilizarmos o método da inversão para gerarmos observações gaussianas, uma vez que há implementado em R a a inversa da acumulada da normal que é chamada de **função erro**, em que

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt.$$

Gerando Normal

Algoritmo do método de Box-Muller

Gerando Normal

Algoritmo do método de Box-Muller

- 1 Gere $U_1, U_2 \sim \mathcal{U}(0, 1)$;

Gerando Normal

Algoritmo do método de Box-Muller

- 1 Gere $U_1, U_2 \sim \mathcal{U}(0, 1)$;
- 2 Faça $R^2 = -2 \log U_1$ (distribuição exponencial) e $S^2 = 2\pi U_2$ (distribuição uniforme);

Gerando Normal

Algoritmo do método de Box-Muller

- 1 Gere $U_1, U_2 \sim \mathcal{U}(0, 1)$;
- 2 Faça $R^2 = -2 \log U_1$ (distribuição exponencial) e $S^2 = 2\pi U_2$ (distribuição uniforme);
- 3 Retorne $X = R \cos(S^2)$ e $Y = R \sin(S^2)$.

As ocorrências X e Y são ocorrências da distribuição normal padrão.

Gerando Normal

Algoritmo do método de Box-Muller

- 1 Gere $U_1, U_2 \sim \mathcal{U}(0, 1)$;
- 2 Faça $R^2 = -2 \log U_1$ (distribuição exponencial) e $S^2 = 2\pi U_2$ (distribuição uniforme);
- 3 Retorne $X = R \cos(S^2)$ e $Y = R \sin(S^2)$.

As ocorrências X e Y são ocorrências da distribuição normal padrão.

Nota: Perceba que em cada execução completa do algoritmo geramos duas novas observações de $X \sim \mathcal{N}(0, 1)$.

Gerando Normal

Algoritmo pelo método polar:

Gerando Normal

Algoritmo pelo método polar:

- 1 Gere $U_1, U_2 \sim \mathcal{U}(0, 1)$;

Gerando Normal

Algoritmo pelo método polar:

- 1 Gere $U_1, U_2 \sim \mathcal{U}(0, 1)$;
- 2 Transforme $U_1 = 2U_1 - 1$, $U_2 = 2U_2 - 1$ e faça $W = U_1^2 + U_2^2$;

Gerando Normal

Algoritmo pelo método polar:

- 1 Gere $U_1, U_2 \sim \mathcal{U}(0, 1)$;
- 2 Transforme $U_1 = 2U_1 - 1$, $U_2 = 2U_2 - 1$ e faça $W = U_1^2 + U_2^2$;
- 3 Se $W > 1$, volte ao primeiro passo;

Gerando Normal

Algoritmo pelo método polar:

- 1 Gere $U_1, U_2 \sim \mathcal{U}(0, 1)$;
- 2 Transforme $U_1 = 2U_1 - 1$, $U_2 = 2U_2 - 1$ e faça $W = U_1^2 + U_2^2$;
- 3 Se $W > 1$, volte ao primeiro passo;
- 4 Retorne

$$X = \sqrt{\frac{-\log W}{W}} \times U_1 \text{ e } Y = \sqrt{\frac{-\log W}{W}} \times U_2.$$

Gerando Normal

Algoritmo pelo método polar:

- 1 Gere $U_1, U_2 \sim \mathcal{U}(0, 1)$;
- 2 Transforme $U_1 = 2U_1 - 1$, $U_2 = 2U_2 - 1$ e faça $W = U_1^2 + U_2^2$;
- 3 Se $W > 1$, volte ao primeiro passo;
- 4 Retorne

$$X = \sqrt{\frac{-\log W}{W}} \times U_1 \text{ e } Y = \sqrt{\frac{-\log W}{W}} \times U_2.$$

A vantagem do algoritmo é que não requer avaliação de funções trigonométricas. Porém, a desvantagem é que normalmente precisamos gerar mais de duas uniformes para obter duas ocorrências gaussianas.

Gerando Normal

Exercício: Implemente uma função para geração de números pseudo-aleatórios com distribuição normal padrão. A função deverá implementar o método de Box-Muller e o método polar. Ao final obtenha um histograma com os números gerados (mil valores) e realize um teste de normalidade.

Gerando de distribuições discretas

Suponhamos que desejamos gerar valores de uma variável aleatória X discreta com função massa de probabilidade

$$P(X = x_j) = p_j, \quad j = 0, 1, \dots, \text{ e } \sum_j p_j = 1.$$

Para realizar isto, geramos uma observação u de uma variável aleatória $U \sim \mathcal{U}(0, 1)$, tal que

Gerando de distribuições discretas

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

Gerando de distribuições discretas

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

Note que para $0 < a < b < 1$, $P(a \leq U < b) = b - a$.

Gerando de distribuições discretas

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

Note que para $0 < a < b < 1$, $P(a \leq U < b) = b - a$. Assim, temos que

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

Dessa forma, X tem a distribuição discreta que desejamos.

Gerando de distribuições discretas

Gerando de distribuições discretas

Algoritmo:

Gere uma observação de U e compare na sequência:

Gerando de distribuições discretas

Algoritmo:

Gere uma observação de U e compare na sequência:

Gerando de distribuições discretas

Algoritmo:

Gere uma observação de U e compare na sequência:

- 1 Se $u < p_0$, faça $X = x_0$ e pare.

Gerando de distribuições discretas

Algoritmo:

Gere uma observação de U e compare na sequência:

- 1 Se $u < p_0$, faça $X = x_0$ e pare.
- 2 Se $u < p_0 + p_1$, faça $X = x_1$ e pare.

Gerando de distribuições discretas

Algoritmo:

Gere uma observação de U e compare na sequência:

- 1 Se $u < p_0$, faça $X = x_0$ e pare.
- 2 Se $u < p_0 + p_1$, faça $X = x_1$ e pare.
- 3 Se $u < p_0 + p_1 + p_2$, faça $X = x_2$ e pare.

Gerando de distribuições discretas

Algoritmo:

Gere uma observação de U e compare na sequência:

- 1 Se $u < p_0$, faça $X = x_0$ e pare.
- 2 Se $u < p_0 + p_1$, faça $X = x_1$ e pare.
- 3 Se $u < p_0 + p_1 + p_2$, faça $X = x_2$ e pare.
- 4 :

Gerando de distribuições discretas

Exercício: Construa uma função em R que simule uma variável aleatória X tal que

$$p_1 = 0.20, \ p_2 = 0.15, \ p_3 = 0.25, \ p_4 = 0.40 \text{ quando } p_j = P(X = j).$$

Gerando de distribuições discretas

Observação

O único caso em que não há a necessidade de checar caso a caso do algoritmo anterior é quando desejamos simular observações de uma variável aleatória com distribuição uniforme discreta, isto é

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

Gerando de distribuições discretas

Observação

Temos nesse caso que

$$X = j \text{ se } \frac{j-1}{n} \leq U < \frac{j}{n}.$$

Gerando de distribuições discretas

Observação

Temos nesse caso que

$$X = j \text{ se } \frac{j-1}{n} \leq U < \frac{j}{n}.$$

Assim, X será igual à j se $j-1 \leq nU < j$.

Gerando de distribuições discretas

Observação

Temos nesse caso que

$$X = j \text{ se } \frac{j-1}{n} \leq U < \frac{j}{n}.$$

Assim, X será igual à j se $j-1 \leq nU < j$. Em outras palavras, temos que

$$X = \text{Int}(nU) + 1.$$

Nota: A notação $\text{Int}(x)$ significa parte inteira de x . Alguns livros denotam por $[x]$.

Gerando de distribuições discretas

Exercício: Construa uma função em R que gera números de $1, \dots, n$ provenientes de uma variável aleatória com distribuição uniforme discreta.

Gerando de distribuições discretas

Exercício: Lembre-se que X é dito ser uma variável aleatória discreta com parâmetro p se

$$P(X = i) = pq^{i-1}, \text{ com } i \geq 1, \text{ e } q = 1 - p,$$

isto é $X \sim \text{Geo}(p)$. X representa o número de ensaios até a ocorrência do primeiro **sucesso**. Construa uma função em R que gere observações de X .

Gerando de distribuições discretas

Estudando o problema acima

Note que

$$\begin{aligned}\sum_{i=1}^{j-1} P(X = i) &= 1 - P(X > j-1) = 1 - P(X \geq j) \\ &= 1 - \sum_{k=j}^{\infty} P(X = k) = 1 - \sum_{k=j}^{\infty} pq^{k-1},\end{aligned}$$

com $j \geq 1$.

Gerando de distribuições discretas

Estudando o problema acima

Nota: Lembre-se que em uma progressão geométrica, temos que

Gerando de distribuições discretas

Estudando o problema acima

Nota: Lembre-se que em uma progressão geométrica, temos que

$$\sum_{k=m}^{\infty} ar^k = \frac{ar^m}{1-r}.$$

Então,

$$\sum_{i=1}^{j-1} P(X = i) = 1 - \sum_{k=j}^{\infty} pq^{k-1} = 1 - q^{j-1}.$$

Gerando de distribuições discretas

Estudando o problema acima

Assim, temos que $1 - q^{j-1} \leq U < 1 - q^j$.

Gerando de distribuições discretas

Estudando o problema acima

Assim, temos que $1 - q^{j-1} \leq U < 1 - q^j$. De forma equivalente, temos

$$q^j < 1 - U \leq q^{j-1}.$$

Daí, podemos definir $X = \min\{j : q^j < 1 - U\}$.

Gerando de distribuições discretas

Estudando o problema acima

Assim, temos que $1 - q^{j-1} \leq U < 1 - q^j$. De forma equivalente, temos

$$q^j < 1 - U \leq q^{j-1}.$$

Daí, podemos definir $X = \min\{j : q^j < 1 - U\}$. Então,

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

Gerando de distribuições discretas

Estudando o problema acima

Então, teremos que

Gerando de distribuições discretas

Estudando o problema acima

Então, teremos que

$$X = \text{Int} \left(\frac{\log(1 - U)}{\log(q)} \right) + 1.$$

Gerando de distribuições discretas

Estudando o problema acima

Então, teremos que

$$X = \text{Int} \left(\frac{\log(1 - U)}{\log(q)} \right) + 1.$$

Por fim, note que $1 - U$ também é uma variável aleatória com distribuição $\mathcal{U}(0, 1)$.

Gerando de distribuições discretas

Estudando o problema acima

Então, teremos que

$$X = \text{Int} \left(\frac{\log(1 - U)}{\log(q)} \right) + 1.$$

Por fim, note que $1 - U$ também é uma variável aleatória com distribuição $\mathcal{U}(0, 1)$. Então,

$$X \equiv \text{Int} \left(\frac{\log(U)}{\log(q)} \right) + 1$$

é uma observação de $X \sim \text{Geo}(p)$.

Gerando por transformação

Resultado de importante de probabilidade

Seja X uma variável aleatória em um espaço de probabilidade $(\Omega, \mathcal{A}, \mathcal{P})$. Uma função $h : X \rightarrow \mathbb{R}$ conhecida também é uma variável aleatória no mesmo espaço de probabilidade. Assim, dado o conhecimento de X através de sua distribuição de probabilidade, desejamos obter o comportamento da transformação $h(X)$.

Gerando por transformação

Resultado de importante de probabilidade

Seja X uma variável aleatória em um espaço de probabilidade $(\Omega, \mathcal{A}, \mathcal{P})$. Uma função $h : X \rightarrow \mathbb{R}$ conhecida também é uma variável aleatória no mesmo espaço de probabilidade. Assim, dado o conhecimento de X através de sua distribuição de probabilidade, desejamos obter o comportamento da transformação $h(X)$.

Normalmente utilizamos dois métodos para obter a distribuição de $h(X)$. São ele:

Gerando por transformação

Resultado de importante de probabilidade

Seja X uma variável aleatória em um espaço de probabilidade $(\Omega, \mathcal{A}, \mathcal{P})$. Uma função $h : X \rightarrow \mathbb{R}$ conhecida também é uma variável aleatória no mesmo espaço de probabilidade. Assim, dado o conhecimento de X através de sua distribuição de probabilidade, desejamos obter o comportamento da transformação $h(X)$.

Normalmente utilizamos dois métodos para obter a distribuição de $h(X)$. São ele:

- 1 Método direto.

Gerando por transformação

Resultado de importante de probabilidade

Seja X uma variável aleatória em um espaço de probabilidade $(\Omega, \mathcal{A}, \mathcal{P})$. Uma função $h : X \rightarrow \mathbb{R}$ conhecida também é uma variável aleatória no mesmo espaço de probabilidade. Assim, dado o conhecimento de X através de sua distribuição de probabilidade, desejamos obter o comportamento da transformação $h(X)$.

Normalmente utilizamos dois métodos para obter a distribuição de $h(X)$. São ele:

- 1 Método direto.
- 2 Método jacobiano.

Gerando por transformação

Exercício: Seja X uma variável aleatória em $(\Omega, \mathcal{A}, \mathcal{P})$ e suponha que $X \sim \mathcal{U}(0, 1)$. Obtenha a distribuição de $Y = -\log(X)$.

Gerando por transformação

Exercício: Seja X uma variável aleatória em $(\Omega, \mathcal{A}, \mathcal{P})$ e suponha que $X \sim \mathcal{U}(0, 1)$. Obtenha a distribuição de $Y = -\log(X)$.

Exercício: Após a obtenção da distribuição de Y no exercício acima, implemente uma função em R que gere observações da variável aleatória Y .

Gerando por transformação

Observação

Gerando por transformação

Observação

Note que computacionalmente, não necessitamos obter analiticamente seja por método direto ou método jacobiano a distribuição da transformação. As únicas coisas que precisaremos é conhecer a transformação $h(\cdot)$ e sabermos gerar observações da variável aleatória X .

Gerando por transformação

Observação

Note que computacionalmente, não necessitamos obter analiticamente seja por método direto ou método jacobiano a distribuição da transformação. As únicas coisas que precisaremos é conhecer a transformação $h(\cdot)$ e sabermos gerar observações da variável aleatória X .

Prova: De fato, seja $Y = h(X)$, então, temos que

Gerando por transformação

Observação

Note que computacionalmente, não necessitamos obter analiticamente seja por método direto ou método jacobiano a distribuição da transformação. As únicas coisas que precisaremos é conhecer a transformação $h(\cdot)$ e sabermos gerar observações da variável aleatória X .

Prova: De fato, seja $Y = h(X)$, então, temos que

$$u = F_Y(y) = P(Y \leq y) = P(Y \leq g(x)) = F_Y(g(x)).$$

Gerando por transformação

Observação

Note que computacionalmente, não necessitamos obter analiticamente seja por método direto ou método jacobiano a distribuição da transformação. As únicas coisas que precisaremos é conhecer a transformação $h(\cdot)$ e sabermos gerar observações da variável aleatória X .

Prova: De fato, seja $Y = h(X)$, então, temos que

$$u = F_Y(y) = P(Y \leq y) = P(Y \leq g(x)) = F_Y(g(x)).$$

Daí, se F_Y é invertível, temos que

Gerando por transformação

Observação

Note que computacionalmente, não necessitamos obter analiticamente seja por método direto ou método jacobiano a distribuição da transformação. As únicas coisas que precisaremos é conhecer a transformação $h(\cdot)$ e sabermos gerar observações da variável aleatória X .

Prova: De fato, seja $Y = h(X)$, então, temos que

$$u = F_Y(y) = P(Y \leq y) = P(Y \leq g(x)) = F_Y(g(x)).$$

Daí, se F_Y é invertível, temos que

$$g(x) = F_Y^{-1}(u).$$

Gerando por transformação

Exercício: Considerando a observação acima, gere a observações da variável aleatória Y sem considerar a distribuição de Y . Observe por meio de um histograma a distribuição amostral de Y . **Lembre-se:** aqui, $Y = -\log(X)$, com $X \sim \mathcal{U}(0, 1)$.

Gerando por transformação

Exercício: Conhecendo a distribuição de probabilidade da variável aleatória X , implemente um gerador para as transformações abaixo:

- a) Se $X \sim \text{Exp}(\lambda)$, com $x \geq 0$ e $\lambda > 0$ e
 $Y = \sum_{i=1}^n X_i \sim \Gamma(n, \lambda)$, implemente um gerador para Y .

Gerando por transformação

Exercício: Conhecendo a distribuição de probabilidade da variável aleatória X , implemente um gerador para as transformações abaixo:

- a) Se $X \sim \text{Exp}(\lambda)$, com $x \geq 0$ e $\lambda > 0$ e
 $Y = \sum_{i=1}^n X_i \sim \Gamma(n, \lambda)$, implemente um gerador para Y .
- b) Se $X \sim \text{Exp}(\lambda)$, com $x \geq 0$ e $\lambda > 0$ e
 $Y = \mu - \beta \log(\lambda X) \sim \text{Gumbel}(\mu, \beta)$, com $\mu \in \mathbb{R}$ e $\beta > 0$,
implemente um gerador para Y .

Gerando por transformação

Exercício: Conhecendo a distribuição de probabilidade da variável aleatória X , implemente um gerador para as transformações abaixo:

- a) Se $X \sim \text{Exp}(\lambda)$, com $x \geq 0$ e $\lambda > 0$ e
 $Y = \sum_{i=1}^n X_i \sim \Gamma(n, \lambda)$, implemente um gerador para Y .
- b) Se $X \sim \text{Exp}(\lambda)$, com $x \geq 0$ e $\lambda > 0$ e
 $Y = \mu - \beta \log(\lambda X) \sim \text{Gumbel}(\mu, \beta)$, com $\mu \in \mathbb{R}$ e $\beta > 0$,
implemente um gerador para Y .
- c) Se $X \sim U(0, 1)$ (contínua) e $Y = m + s(-\log(X))^{-1/\alpha} \sim \text{Fréchet}(\alpha, s, m)$, com $x, \alpha, s > 0$ e $m \in \mathbb{R}$, implemente um gerador para Y .

Método da Aceitação-Rejeição

O método da aceitação-rejeição também chamado de algoritmo da aceitação-rejeição trata-se de um método de Monte-Carlo que estudaremos mais a frente. Tal método foi proposto inicialmente pelo matemático John von Neumann e poderá ser utilizado para gerar observações de uma variável aleatória discreta ou contínua.

Método da Aceitação-Rejeição

O método da aceitação-rejeição também chamado de algoritmo da aceitação-rejeição trata-se de um método de Monte-Carlo que estudaremos mais a frente. Tal método foi proposto inicialmente pelo matemático John von Neumann e poderá ser utilizado para gerar observações de uma variável aleatória discreta ou contínua.

Observação

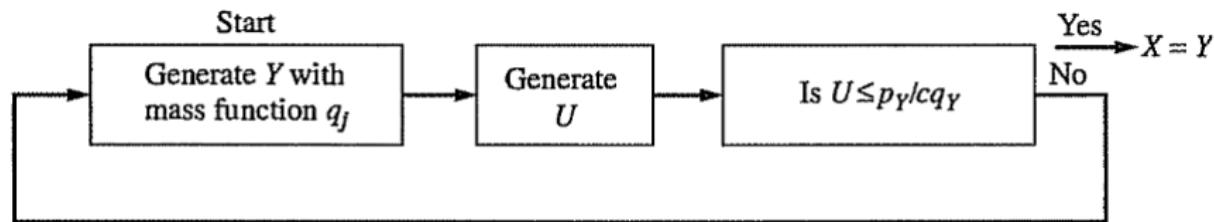
Para que possamos gerar observações de uma variável aleatória X precisaremos ter um meio de gerar observações de uma variável aleatória Y e **não** necessariamente é preciso que as variáveis aleatórias estejam relacionadas matematicamente ou que suas distribuições de probabilidade tenham alguma relação matemática.

Método da Aceitação-Rejeição

Teorema: Seja X uma variável aleatória ao qual queremos gerar observações, tal que $P(X = j) = p_j$, com $j = 0, \dots$. O procedimento abaixo permite gerar observações de X .

Método da Aceitação-Rejeição

Teorema: Seja X uma variável aleatória ao qual queremos gerar observações, tal que $P(X = j) = p_j$, com $j = 0, \dots$. O procedimento abaixo permite gerar observações de X .



em que $P(Y = j) = q_j$, com $j = 0, \dots$, c é uma constante tal que $p_j/q_j \leq c$ e U é uma variável aleatória uniforme no $(0,1)$.

Método da Aceitação-Rejeição

Prova:

Método da Aceitação-Rejeição

Prova: Precisamos provar que seguir o procedimento acima fará com que venhamos aceitar $X = Y$ com probabilidade p_Y/cq_y e fazer isso garante que $P(X = y) = p_y$. Daí,

Método da Aceitação-Rejeição

Prova: Precisamos provar que seguir o procedimento acima fará com que venhamos aceitar $X = Y$ com probabilidade p_Y/cq_y e fazer isso garante que $P(X = y) = p_y$. Daí,

$$\begin{aligned} P(Y = j, U \leq p_y/cq_y) &= P(Y = j)P(U \leq p_y/cq_y | Y_j) \\ &= q_j \frac{p_j}{cq_j} = \frac{p_j}{c}. \end{aligned}$$

Além disso, a probabilidade de aceitar $X = Y$ é dada por

Método da Aceitação-Rejeição

Prova: Precisamos provar que seguir o procedimento acima fará com que venhamos aceitar $X = Y$ com probabilidade p_Y/cq_y e fazer isso garante que $P(X = y) = p_y$. Daí,

$$\begin{aligned} P(Y = j, U \leq p_y/cq_y) &= P(Y = j)P(U \leq p_y/cq_y | Y_j) \\ &= q_j \frac{p_j}{cq_j} = \frac{p_j}{c}. \end{aligned}$$

Além disso, a probabilidade de aceitar $X = Y$ é dada por

$$P\left(U \leq \frac{p_j}{cq_j}\right) = \sum_j \frac{p_j}{c} = \frac{1}{c}.$$

Método da Aceitação-Rejeição

Note também que as iterações do algoritmo são independentes e que o processo aleatório de se obter um sucesso (aceitar $X = Y$) segue uma distribuição geométrica de parâmetro $(1/c)$, isto é, o número médio de repetições sé c . Daí, temos que

Método da Aceitação-Rejeição

Note também que as iterações do algoritmo são independentes e que o processo aleatório de se obter um sucesso (aceitar $X = Y$) segue uma distribuição geométrica de parâmetro $(1/c)$, isto é, o número médio de repetições sé c . Daí, temos que

$$P(X = j) = \sum_{n \in \mathbb{N}} \left(1 - \frac{1}{c}\right)^{n-1} \frac{1}{c} p_j = p_j.$$

Método da Aceitação-Rejeição

Além disso, note que precisamos que

$$\frac{p_y}{cq_y} \leq 1,$$

uma vez que aceitamos $X = Y$ quando $U \leq \frac{p_y}{cq_y}$, e U tem distribuição uniforme em $(0,1)$.

Método da Aceitação-Rejeição

Além disso, note que precisamos que

$$\frac{p_y}{cq_y} \leq 1,$$

uma vez que aceitamos $X = Y$ quando $U \leq \frac{p_y}{cq_y}$, e U tem distribuição uniforme em $(0,1)$. Assim, queremos que

$$\frac{p_y}{q_y} \leq c.$$

Método da Aceitação-Rejeição

Importante

Note que para que a última desigualdade seja sempre satisfeita, basta tomarmos os o valor de c tal que

$$\max \frac{p_y}{q_y} \leq c.$$

Método da Aceitação-Rejeição

Importante

Note que para que a última desigualdade seja sempre satisfeita, basta tomarmos o valor de c tal que

$$\max \frac{p_y}{q_y} \leq c.$$

Nota: É sempre uma ótima estratégia tomar $c = \max p_y/q_y$.

Método da Aceitação-Rejeição

Importante

Note que para que a última desigualdade seja sempre satisfeita, basta tomarmos o valor de c tal que

$$\max \frac{p_y}{q_y} \leq c.$$

Nota: É sempre uma ótima estratégia tomar $c = \max p_y/q_y$. Isso se deve ao fato de que como o método da aceitação-rejeição tem distribuição geométrica com probabilidade de aceitação $1/c$ ("sucesso"), necessitamos o menor valor de c que satisfaça a desigualdade acima para maximizar essa probabilidade.

Método da Aceitação-Rejeição

Exercício: Suponha que desejamos simular valores de uma variável aleatória X que assume os valores $1, 2, \dots, 10$ com as respectivas probabilidades $0.11, 0.12, 0.09, 0.08, 0.12, 0.10, 0.09, 0.09, 0.10, 0.10$. Suponha também que sabemos gerar observações de uma variável aleatória Y que assume as mesmas observações de X com probabilidade $q_j = 1/10, j = 1, \dots, 10$.

Método da Aceitação-Rejeição

Exercício: Suponha que desejamos simular valores de uma variável aleatória X que assume os valores $1, 2, \dots, 10$ com as respectivas probabilidades $0.11, 0.12, 0.09, 0.08, 0.12, 0.10, 0.09, 0.09, 0.10, 0.10$. Suponha também que sabemos gerar observações de uma variável aleatória Y que assume as mesmas observações de X com probabilidade $q_j = 1/10, j = 1, \dots, 10$. Qual o valor de c ideal?

Método da Aceitação-Rejeição

Exercício: Suponha que desejamos simular valores de uma variável aleatória X que assume os valores $1, 2, \dots, 10$ com as respectivas probabilidades $0.11, 0.12, 0.09, 0.08, 0.12, 0.10, 0.09, 0.09, 0.10, 0.10$. Suponha também que sabemos gerar observações de uma variável aleatória Y que assume as mesmas observações de X com probabilidade $q_j = 1/10, j = 1, \dots, 10$. Qual o valor de c ideal? Construa uma função em R que implemente o método da aceitação e rejeição.

Método da Aceitação-Rejeição

Exercício: Escreva um algoritmo eficiente para simular as observações da variável aleatória X tal que

$$P(X = 1) = 0.3, P(X = 2) = 0.2, P(X = 3) = 0.35, P(X = 4) = 0.15.$$

Método da Aceitação-Rejeição

Exercício: Escreva um algoritmo eficiente para simular as observações da variável aleatória X tal que

$$P(X = 1) = 0.3, P(X = 2) = 0.2, P(X = 3) = 0.35, P(X = 4) = 0.15.$$

Exercício: Escreva um algoritmo eficiente para simular as observações da variável aleatória X tal que

$$P(X = 0) = 0.01, P(X = 1) = 0.04, P(X = 2) = 0.12,$$

$$P(X = 3) = 0.27, P(X = 4) = 0.44, P(X = 5) = 0.62,$$

$$P(X = 6) = 0.76, P(X = 7) = 0.87, P(X = 8) = 0.93,$$

$$P(X = 9) = 0.97, P(X = 10) = 0.99, P(X = 11) = 0.99,$$

$$P(X = 12) = 1.$$

Método da Aceitação-Rejeição

Nota

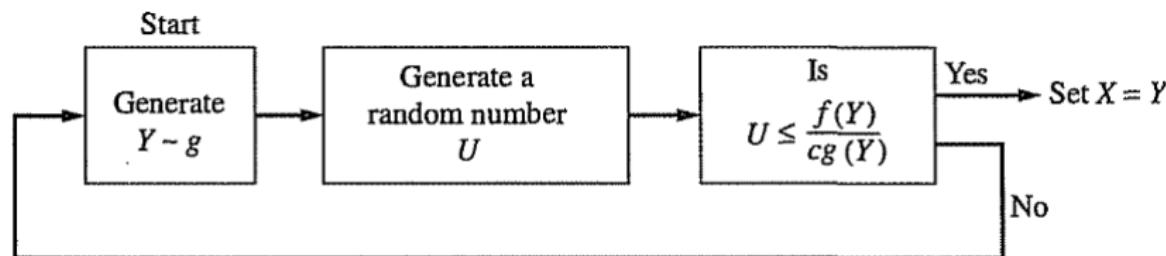
No caso em que desejamos gerar observações de uma variável aleatória discreta pelo método da aceitação-rejeição, uma escolha conveniente para a função de probabilidade de Y é considerar a distribuição uniforme discreta com probabilidade de ocorrência de cada observação igual à $\frac{1}{n}$.

Método da Aceitação-Rejeição

No caso contínuo o método de aceitação-rejeição é conduzido de forma análoga. O algoritmo segue ilustrado graficamente.

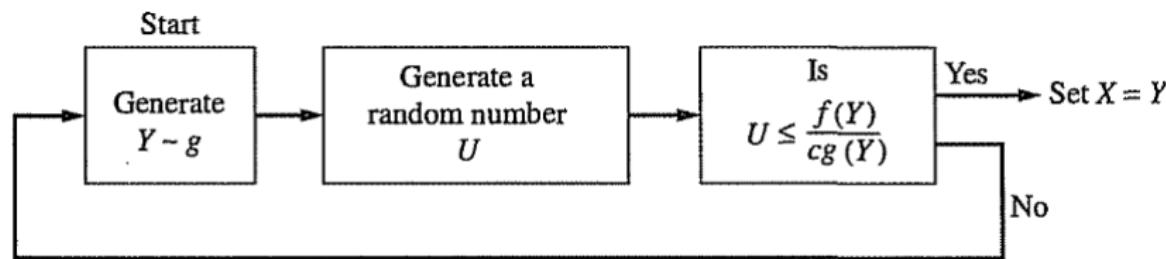
Método da Aceitação-Rejeição

No caso contínuo o método de aceitação-rejeição é conduzido de forma análoga. O algoritmo segue ilustrado graficamente.



Método da Aceitação-Rejeição

No caso contínuo o método de aceitação-rejeição é conduzido de forma análoga. O algoritmo segue ilustrado graficamente.



em que c é uma constante tal que $f(y)/g(y) \leq c$ e U é uma variável aleatória uniforme no $(0,1)$. Além disso, temos que $f(y)$ é a densidade de X no ponto y e $g(y)$ é a densidade de Y no mesmo ponto.

Método da Aceitação-Rejeição

Exercício: Utilize o método da aceitação-rejeição para gerar observações da variável aleatória X que possui função densidade de probabilidade abaixo:

$$f_X(x) = 20x(1-x)^3, \quad 0 < x < 1.$$

Otimização Não-Linear

Séries de Potência:

Uma **série de potência** é uma série da forma

$$\sum_{n=0}^{\infty} c_n x^n = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \dots,$$

em que x é uma variável e c_n são constantes chamadas **coeficientes** da série. Em geral, poderemos construir uma série chamada **série de potência em $(x - a)$** ou **série de potência centrada em a** (**série de potência em torno de a**).

Otimização Não-Linear

A série de potência em torno do ponto a é dada por:

$$\sum_{n=0}^{\infty} c_n(x - a)^n = c_0 + c_1(x - a) + c_2(x - a)^2 + \dots$$

Nota: Observe que quando $x = a$, todos os termos são 0 para $n \geq 1$ e assim a série de potência em torno do ponto a sempre converge quando $x = a$.

Otimização Não-Linear

Nota: Utilizamos séries de potência como uma **aproximação** à uma função que não tem antiderivadas elementares, para resolver equações diferenciais e para aproximar funções por polinômios. Cientistas fazem isso para simplificar expressões que eles utilizam. Tal aproximação é bastante útil para representar funções complicadas em computadores ou calculadoras.

Séries de Taylor e Maclaurin:

Otimização Não-Linear

Nota: Utilizamos séries de potência como uma **aproximação** à uma função que não tem antiderivadas elementares, para resolver equações diferenciais e para aproximar funções por polinômios. Cientistas fazem isso para simplificar expressões que eles utilizam. Tal aproximação é bastante útil para representar funções complicadas em computadores ou calculadoras.

Séries de Taylor e Maclaurin:

As **séries de Taylor e Maclaurin** são séries de potências em torno do ponto a tal que

$$c_n = \frac{f^{(n)}(a)}{n!}.$$

Otimização Não-Linear

Assim, tem-se que:

$$\begin{aligned}f(x) &= \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \\&= f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \dots,\end{aligned}\tag{1}$$

em que $f^{(n)}(a)$ é a n -ésima derivada da função aplicada no ponto a . Para $a = 0$ temos a **série de Maclaurin**.

Otimização Não-Linear

Assim, tem-se que:

$$\begin{aligned}f(x) &= \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \\&= f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \dots,\end{aligned}\tag{1}$$

em que $f^{(n)}(a)$ é a n -ésima derivada da função aplicada no ponto a . Para $a = 0$ temos a **série de Maclaurin**.

Nota: A série de Maclaurin nada mais é que uma série de Taylor em torno do ponto $a = 0$.

Otimização Não-Linear

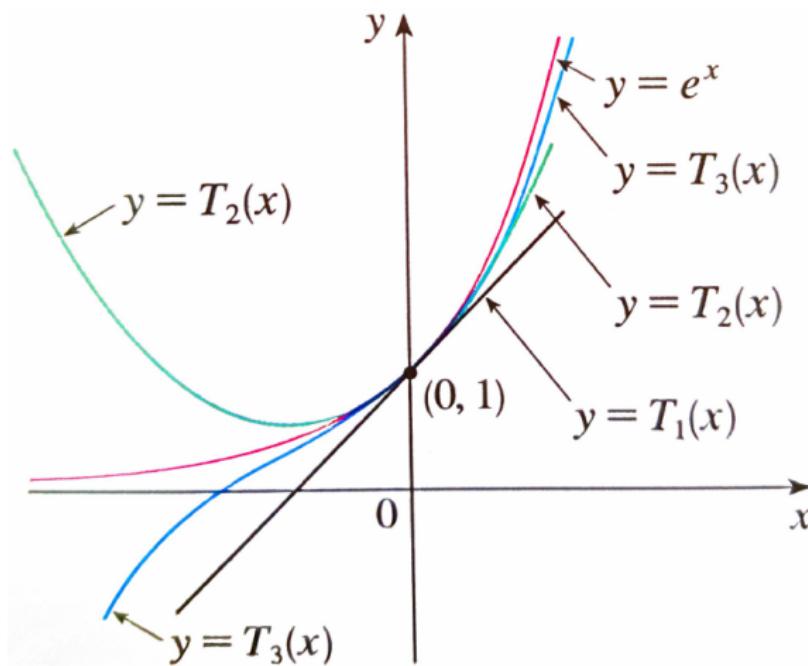


Figure: Aproximação por série de Taylor em torno de $a = 0$ (Maclaurin) da função $f(x) = e^x$

Otimização Não-Linear

Importante: Observe que a precisão de aproximação é melhorada a medida que aumentamos o número de termos da série. Em outras palavras, aproximaremos mais pontos em torno de um ponto a ao considerarmos polinômios de alta ordem.

Otimização Não-Linear

Na estatística e em diversas áreas, é comum o interesse em se obter os pontos de máximo e/ou mínimo de uma função. Normalmente, e especialmente na estatística, é comum o interesse em otimizar uma função $f : \mathbb{R}^n \rightarrow \mathbb{R}$, como é o caso das funções de log-verossimilhanças.

Otimização Não-Linear

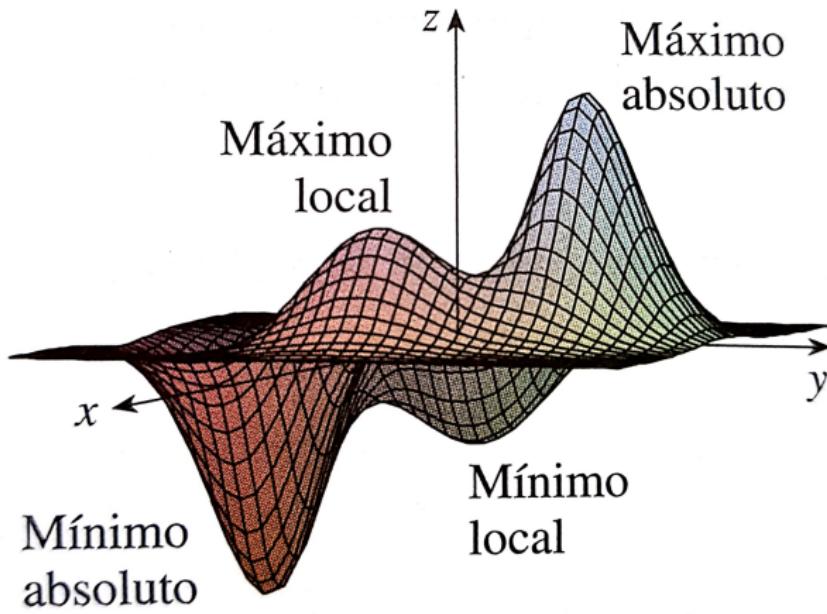


Figure: Pontos críticos de interesse ao se otimizar uma função (caso multivariado).

Otimização Não-Linear

No caso univariado como no caso multivariado, o **método de Newton** também chamado de **Newton-Raphson** para encontrar raízes de uma função f é utilizado para encontrar pontos críticos de uma função, podendo esses pontos serem pontos de máximo e mínimo (locais ou globais), pontos de inflexão ou pontos de sela.

Otimização Não-Linear

No caso univariado como no caso multivariado, o **método de Newton** também chamado de **Newton-Raphson** para encontrar raízes de uma função f é utilizado para encontrar pontos críticos de uma função, podendo esses pontos serem pontos de máximo e mínimo (locais ou globais), pontos de inflexão ou pontos de sela.

Importante

Os pontos críticos de uma função são raízes do vetor gradiente de f , isto é, $f'(\cdot)$ aplicada à um ponto crítico é igual à 0. Por isso que o método de Newton-Raphson para aproximação do zero de uma função poderá ser utilizado para obtenção de pontos críticos de interesse.

Otimização Não-Linear

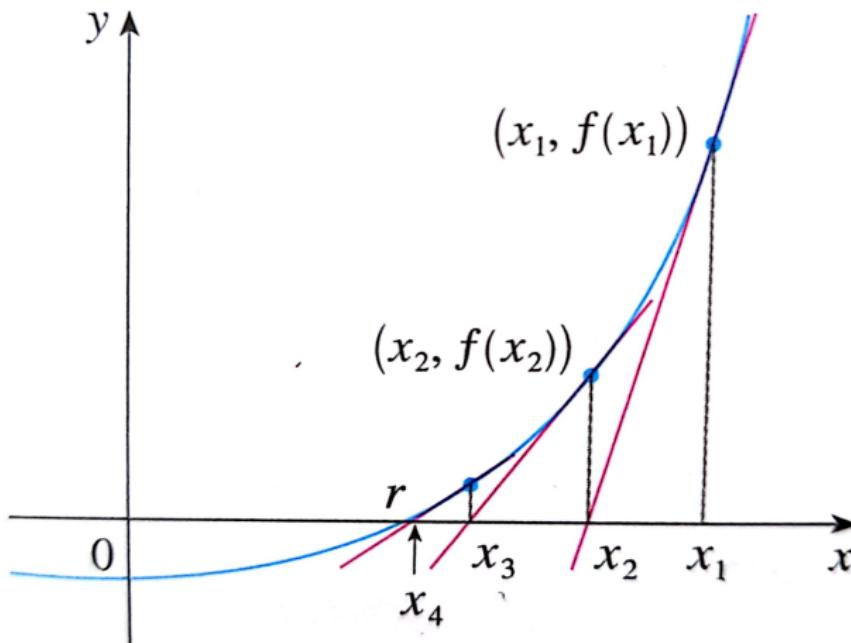


Figure: Interpretação geométrica do método de Newton-Raphson, em que a raiz é o ponto r .

Otimização Não-Linear

Utilizando uma aproximação de primeira ordem para $f(x)$ por série de Taylor em torno de um ponto inicial x_1 , temos que

$$\begin{aligned}f(x) &= f(x_1) + f'(x_1)(x - x_1) \\f(x) - f(x_1) &= f'(x_1)(x - x_1)\end{aligned}$$

Otimização Não-Linear

Utilizando uma aproximação de primeira ordem para $f(x)$ por série de Taylor em torno de um ponto inicial x_1 , temos que

$$\begin{aligned}f(x) &= f(x_1) + f'(x_1)(x - x_1) \\f(x) - f(x_1) &= f'(x_1)(x - x_1)\end{aligned}$$

Fazendo $f(x) = 0$, temos um novo ponto x no domínio de f candidato a ser o zero da função (raiz da função). Assim, temos que

Otimização Não-Linear

Utilizando uma aproximação de primeira ordem para $f(x)$ por série de Taylor em torno de um ponto inicial x_1 , temos que

$$\begin{aligned}f(x) &= f(x_1) + f'(x_1)(x - x_1) \\f(x) - f(x_1) &= f'(x_1)(x - x_1)\end{aligned}$$

Fazendo $f(x) = 0$, temos um novo ponto x no domínio de f candidato a ser o zero da função (raiz da função). Assim, temos que

$$x = x_1 - \frac{x_1}{f'(x_1)}.$$

Assim, partimos de um chute inicial x_1 e obtemos um novo valor $x_2 = x$ da forma apresentada logo acima.

Otimização Não-Linear

De forma iterativa, temos que

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Poderemos parar o método interativo de novos candidatos à aproximação da raiz da função f quando $|x_{n+1} - x_n| \leq \varepsilon$, com $\varepsilon > 0$ (pequeno).

Exercício: Por meio do chute inicial $x_1 = 2$, construa uma tabela com as 7 primeiras aproximações para a raiz da função $f(x) = x^3 - 2x - 5$ mostrando o melhoramento da aproximação do zero da função em cada passo.

Otimização Não-Linear

Exercício: Implemente, em R, o método de Newton para obtenção numérica do zero de uma função univariada.

Otimização Não-Linear

Exercício: Implemente, em R, o método de Newton para obtenção numérica do zero de uma função univariada.

REVISANDO ALGUNS RESULTADOS MATEMÁTICOS:

Otimização Não-Linear

Exercício: Implemente, em R, o método de Newton para obtenção numérica do zero de uma função univariada.

REVISANDO ALGUNS RESULTADOS MATEMÁTICOS:

Teorema de Fermat: Se f tiver um máximo ou mínimo local em c e se $f'(c)$ existir, então $f'(c) = 0$.

Otimização Não-Linear

Exercício: Implemente, em R, o método de Newton para obtenção numérica do zero de uma função univariada.

REVISANDO ALGUNS RESULTADOS MATEMÁTICOS:

Teorema de Fermat: Se f tiver um máximo ou mínimo local em c e se $f'(c)$ existir, então $f'(c) = 0$.

Observe que c é raiz da primeira derivada (gradiente) da função objetivo f ao qual queremos otimizar.

Otimização Não-Linear

Definição (Ponto Crítico): Um **número crítico** de uma função é o número c no domínio de f tal que ou $f'(c) = 0$ ou $f'(c)$ não existe. Assim, se c é um número crítico de f , então o ponto crítico da função f é dado por $(c, f(c))$.

Otimização Não-Linear

Definição (Ponto Crítico): Um **número crítico** de uma função é o número c no domínio de f tal que ou $f'(c) = 0$ ou $f'(c)$ não existe. Assim, se c é um número crítico de f , então o ponto crítico da função f é dado por $(c, f(c))$.

Nota: Se f tiver um máximo ou mínimo local em c , então c é um número crítico de f .

Otimização Não-Linear

Definição: Seja c um número no domínio D de f . Então $f(c)$ é o

Otimização Não-Linear

Definição: Seja c um número no domínio D de f . Então $f(c)$ é o

- 1 valor de máximo absoluto** de f em D se
 $f(c) \geq f(x), \forall x \in D;$

Otimização Não-Linear

Definição: Seja c um número no domínio D de f . Então $f(c)$ é o

- 1 valor de máximo absoluto** de f em D se
 $f(c) \geq f(x), \forall x \in D;$

- 2 valor de mínimo absoluto** de f em D se
 $f(c) \leq f(x), \forall x \in D.$

Otimização Não-Linear

Definição: O número $f(c)$ é um

Otimização Não-Linear

Definição: O número $f(c)$ é um

- 1 valor de máximo local** de f se $f(c) \geq f(x)$ quando x é próximo de c ;

Otimização Não-Linear

Definição: O número $f(c)$ é um

- 1 valor de máximo local** de f se $f(c) \geq f(x)$ quando x é próximo de c ;

- 2 valor de mínimo local** de f se $f(c) \leq f(x)$ quando x é próximo de c .

Otimização Não-Linear

Lembre-se das disciplinas de cálculo:

- 1 Se $f'(x) > 0$ em um intervalo, então f é crescente nesse intervalo;
- 2 Se $f'(x) < 0$ em um intervalo, então f é decrescente nesse intervalo;

Observação: Se f' não muda de sinal em c (isto é, se em ambos os lados de c , f' for positivo ou negativo), então f não tem máximo ou mínimo locais em c .

Otimização Não-Linear

Definição: Se o gráfico de f estiver acima de todas as suas tangentes no intervalo I do domínio de f , então f é **côncava para cima** em I . Se o gráfico de f estiver abaixo de todas as suas tangentes em I , então diremos que f é **concava para baixo** em I .

Otimização Não-Linear

Teste de Concavidade:

Otimização Não-Linear

Teste de Concavidade:

- 1 Se $f''(x) > 0 \forall x \in I$, então o gráfico de f é côncavo para cima em I ;

Otimização Não-Linear

Teste de Concavidade:

- 1 Se $f''(x) > 0 \forall x \in I$, então o gráfico de f é côncavo para cima em I ;
- 2 Se $f''(x) < 0 \forall x \in I$, então o gráfico de f é côncavo para baixo em I .

Otimização Não-Linear

Teste de Concavidade:

- 1 Se $f''(x) > 0 \forall x \in I$, então o gráfico de f é côncavo para cima em I ;
- 2 Se $f''(x) < 0 \forall x \in I$, então o gráfico de f é côncavo para baixo em I .

Definição (Ponto de Inflexão): Um ponto P na curva $y = f(x)$ é chamado de **ponto de inflexão** se f é contínua no ponto e a curva mudar de concava para cima para concava para baixo ou vice-versa.

Otimização Não-Linear

Teste da Segunda Derivada: Suponha que f'' seja contínua na proximidade de c .

- 1 Se $f'(c) = 0$ e $f''(c) > 0$, então f tem mínimo local em c ;
- 2 Se $f'(c) = 0$ e $f''(c) < 0$, então f tem máximo local em c .

Importante

No caso univariado, podemos utilizar o método de Newton-Raphson para obter críticos de f' . E utilizando os resultados matemáticos apresentados, podemos dizer se os pontos críticos de f' são pontos de máximo ou mínimo locais de f .

Otimização Não-Linear

O método de Newton-Raphson acima considerou a função objetivo $f : \mathbb{R} \rightarrow \mathbb{R}$. Porém, o método de Newton-Raphson poderá ser generalizado para o caso de $f : \mathbb{R}^p \rightarrow \mathbb{R}$.

Otimização Não-Linear

O método de Newton-Raphson acima considerou a função objetivo $f : \mathbb{R} \rightarrow \mathbb{R}$. Porém, o método de Newton-Raphson poderá ser generalizado para o caso de $f : \mathbb{R}^P \rightarrow \mathbb{R}$.

Nosso interesse consiste em:

$$\hat{\theta}^* = \underset{\hat{\theta} \in \mathbb{R}^P}{\operatorname{argmax}} f(\hat{\theta}).$$

Otimização Não-Linear

O método de Newton-Raphson acima considerou a função objetivo $f : \mathbb{R} \rightarrow \mathbb{R}$. Porém, o método de Newton-Raphson poderá ser generalizado para o caso de $f : \mathbb{R}^P \rightarrow \mathbb{R}$.

Nosso interesse consiste em:

$$\hat{\theta}^* = \underset{\hat{\theta} \in \mathbb{R}^P}{\operatorname{argmax}} f(\hat{\theta}).$$

O método de Newton-Raphson é apenas um método iterativo de um conjunto de métodos iterativos na classe de métodos de Newton. De forma mais geral e no caso multivariado, temos que os métodos iterativos tem a seguinte estrutura:

$$\hat{\theta}_{t+1} = \hat{\theta}_t + \text{alguma correção}, \quad t = 0, 1, 2, \dots$$

Otimização Não-Linear

De forma mais específica, tem-se que:

$$\tilde{\theta}_{t+1} = \tilde{\theta}_t + \lambda_t \tilde{\Delta}_t \quad t = 0, 1, 2, \dots,$$

em que $\tilde{\theta}_{t+1}$ é um vetor $p \times 1$, $\lambda_t > 0$ é o tamanho do passo e $\tilde{\Delta}_t$ é o vetor direcional de dimensão $p \times 1$ (vetor gradiente).

Nota: Utilizamos a notação $\tilde{\theta}$ ao invés de \tilde{x} uma vez que no contexto estatístico, queremos otimizar a função objetivo com respeito aos parâmetros da função de log-verossimilhança $\ell(\tilde{\theta})$ e não em termo das observações.

Otimização Não-Linear

A grande sacada

Perceba que atualizamos $\tilde{\theta}_t$ para $\tilde{\theta}_{t+1}$ por meio de uma escolha de um número $\lambda_t > 0$. Sendo assim, não precisamos otimizar cada componente de $\tilde{\theta}$. Basta apenas fazer uma busca em linha otimizando λ_t .

Otimização Não-Linear

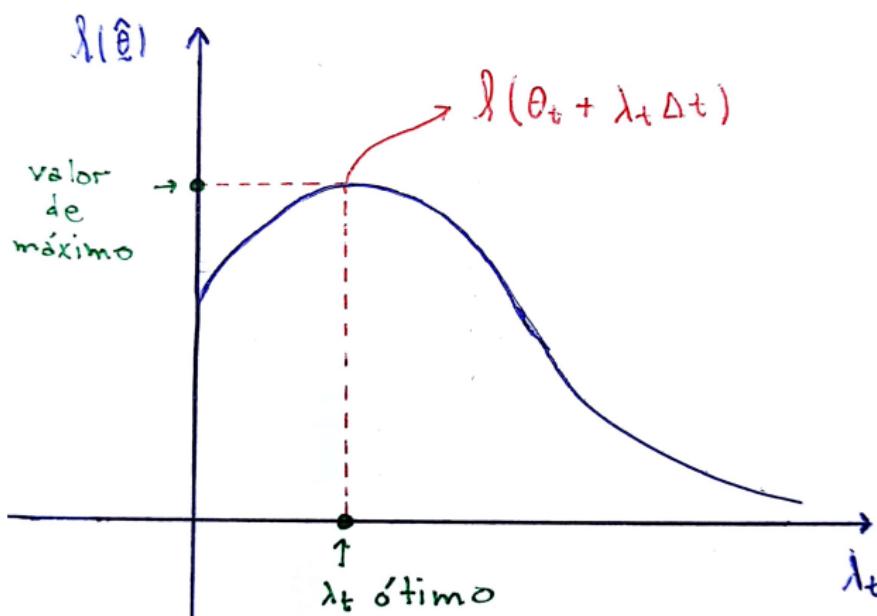


Figure: Buscamos um valor ótimo para λ_t que maximiza $\ell(\hat{\theta})$.

Otimização Não-Linear

Assim,

$$\frac{\partial \ell(\tilde{\theta}_t + \lambda_t \Delta_t)}{\partial \lambda_t} = g(\tilde{\theta}_t + \lambda_t \Delta_t)' \Delta_t = 0,$$

em que $g \equiv g(\tilde{\theta}) = \frac{\partial \ell(\tilde{\theta})}{\partial \tilde{\theta}}$ é o vetor gradiente de $\ell(\tilde{\theta})$. Além disso, temos que

$$H \equiv H(\tilde{\theta}) = \frac{\partial^2 \ell(\tilde{\theta})}{\partial \tilde{\theta} \partial \tilde{\theta}'}$$

é a matriz Hessiana de $\ell(\tilde{\theta})$, sendo esta uma matriz $p \times p$.

Otimização Não-Linear

Expandindo $\ell(\tilde{\boldsymbol{\theta}}_t + \lambda_t \tilde{\Delta}_t)$ em torno do ponto $\lambda_t = 0$ em séries de Taylor de primeira ordem, temos que

$$\begin{aligned}\ell(\tilde{\boldsymbol{\theta}}_t + \lambda_t \tilde{\Delta}_t) &\approx \ell(\tilde{\boldsymbol{\theta}}_t) + \lambda_t g(\tilde{\boldsymbol{\theta}}_t)' \tilde{\Delta}_t \\ \ell(\tilde{\boldsymbol{\theta}}_t + \lambda_t \tilde{\Delta}_t) - \ell(\tilde{\boldsymbol{\theta}}_t) &\approx \lambda_t g(\tilde{\boldsymbol{\theta}}_t)' \tilde{\Delta}_t\end{aligned}\quad (2)$$

Métodos Gradientes

A classe mais utilizada de algoritmos iterativos é conhecida como **classe de métodos gradiente**. Nessa classe, a direção do passo (vetor direcional) é dada por

$$\tilde{\Delta}_t = W_t g_t,$$

em que $g_t = g(\tilde{\theta}_t)$ e W_t é uma matriz positiva-definida de dimensão $p \times p$, ambos na iteração t . Assim,

$$\ell(\tilde{\theta}_t + \lambda_t \tilde{\Delta}_t) - \ell(\tilde{\theta}_t) \approx \lambda_t g_t' W_t g_t.$$

Métodos Gradientes

Steepest Ascent:

O algoritmo iterativo mais simples é o da **subida mais inclinada** também conhecido como algoritmo **steepest ascent**. Nesse método usa-se

$$W_t = I_p,$$

em que I_p é a matriz identidade $p \times p$. Nesse caso, temos que $\Delta_t = g_t$.

Nota: Esse algoritmo é normalmente pouco utilizado uma vez que tende a ter convergência lenta.

Métodos Gradientes

Método de Newton-Raphson (multidimensional):

O método de **Newton** ou **Newton-Raphson** pode ser expresso pela equação de atualização abaixo:

$$\hat{\theta}_{t+1} = \hat{\theta}_t - H_t^{-1} g_t,$$

em que H^{-1} é a inversa da matriz hessiana.

Nota: Nesse método, temos que $W_t = -H^{-1}$ e $\lambda_t = 1$, para todo t .

Métodos Gradientes

Prova: Tome uma expansão de série de Taylor para a função $\partial\ell(\tilde{\theta})/\partial\tilde{\theta}$ em torno de um ponto arbitrário, por exemplo o ponto $\tilde{\theta}_0$. Assim,

$$\frac{\partial\ell(\tilde{\theta})}{\partial\tilde{\theta}} \approx g(\tilde{\theta}_0) + H(\tilde{\theta} - \tilde{\theta}_0).$$

O resultado segue ao fazer $\partial\ell(\tilde{\theta})/\partial\tilde{\theta} \approx 0$ e isolarmos $\tilde{\theta}$ aplicando H^{-1} .

Métodos Gradientes

Método BHHH (Berndt, Hall, Hall, Hausman):

Em muitas otimizações a avaliação de $-H^{-1}$ poderá vir a ser custosa. O método BHHH é semelhante ao método de Newton-Raphson. A diferença entre os métodos consiste em que no método de Newton-Raphson é avaliado a matriz hessiana e no método BHHH substituímos H por $g_t g_t'$ que tem dimensão $p \times p$. O produto dos vetores gradientes é chamado de *outer product of the gradient* (**produto exterior do gradiente**). Assim, utilizamos

$$-H^* = -H^*(\tilde{\theta}) = -\frac{\partial \ell(\tilde{\theta}_t)}{\partial \tilde{\theta}_t} \frac{\partial \ell(\tilde{\theta}_t)}{\partial \tilde{\theta}'}$$

Métodos Gradientes

Método BHHH (Berndt, Hall, Hall, Hausman):

Em muitas otimizações a avaliação de $-H^{-1}$ poderá vir a ser custosa. O método BHHH é semelhante ao método de Newton-Raphson. A diferença entre os métodos consiste em que no método de Newton-Raphson é avaliado a matriz hessiana e no método BHHH substituímos H por $g_t g_t'$ que tem dimensão $p \times p$. O produto dos vetores gradientes é chamado de *outer product of the gradient* (**produto exterior do gradiente**). Assim, utilizamos

$$-H^* = -H^*(\tilde{\theta}) = -\frac{\partial \ell(\tilde{\theta}_t)}{\partial \tilde{\theta}_t} \frac{\partial \ell(\tilde{\theta}_t)}{\partial \tilde{\theta}'}$$

Nota: O método BHHH não precisa avaliar matrizes de segundas derivadas e é bastante utilizado em várias aplicações econometria.

Métodos Gradientes

Método Escore de Fisher:

O método escore de Fisher (*Fisher's scoring*) também é semelhante ao método de Newton-Rapshon. A diferença reside no fato de que as componentes da matriz hessiana, no caso estatístico são variáveis aleatórias. Sendo assim, é possível tomar o valor esperado da matriz hessiana, tendo assim o que chamamos de matriz de informação de Fisher que é dada por:

$$K = K(\boldsymbol{\theta}) = E \left(\frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_t \partial \boldsymbol{\theta}'_t} \right).$$

Sendo assim, $W = K^{-1}$ é a variância assintótica de \sqrt{n} vezes o estimador de máxima verossimilhança de $\boldsymbol{\theta}$.

Métodos quasi-Newton

Todos métodos de otimização que fazem uso de segundas derivadas são chamados de métodos de Newton. Porém, há diversos métodos muito eficientes de otimização de uma função que elimina a necessidade do cálculo de segundas derivadas que são a classe dos algoritmos **quasi-Newton**, também conhecido como **métodos de métrica variável**.

Ao invés de utilizar $-H^{-1}$, utiliza-se uma sequência de matrizes que converge para $-H^{-1}$ quanto o $t \rightarrow \infty$ (o número de iterações dente ao infinito).

Nota: Todos os elementos da sequência são matrizes positivas definida.

Métodos quasi-Newton

Na classe de métodos quasi-Newton, usa-se a sequência de matrizes:

$$W_{t+1} = W_t + N_t,$$

Métodos quasi-Newton

Na classe de métodos quasi-Newton, usa-se a sequência de matrizes:

$$W_{t+1} = W_t + N_t,$$

em que N_t é uma matriz positiva definida. A ideia básica dos algoritmos quasi-Newton é construir iterativamente uma boa aproximação para $-H(\theta)^{-1}$, ou seja, utilizar uma sequência de matrizes tal que

Métodos quasi-Newton

Na classe de métodos quasi-Newton, usa-se a sequência de matrizes:

$$W_{t+1} = W_t + N_t,$$

em que N_t é uma matriz positiva definida. A ideia básica dos algoritmos quasi-Newton é construir iterativamente uma boa aproximação para $-H(\theta)^{-1}$, ou seja, utilizar uma sequência de matrizes tal que

$$\lim_{t \rightarrow \infty} W_t = -H^{-1}.$$

Métodos quasi-Newton

A ideia central dos métodos quasi-Newton é apresentada no artigo:

Métodos quasi-Newton

A ideia central dos métodos quasi-Newton é apresentada no artigo:

W. C. Davison. **Variable metric method for minimization.** Technical Report ANL-5990 (revised), Argonne National Laboratory, 1959.

Métodos quasi-Newton

A ideia central dos métodos quasi-Newton é apresentada no artigo:

W. C. Davison. **Variable metric method for minimization**. Technical Report ANL-5990 (revised), Argonne National Laboratory, 1959.

Atualmente há diversos algoritmos que pertence à essa classe, por exemplo, o algoritmo DFP (**D**avison, **F**letcher e **P**owell) utiliza

Métodos quasi-Newton

A ideia central dos métodos quasi-Newton é apresentada no artigo:

W. C. Davison. **Variable metric method for minimization**. Technical Report ANL-5990 (revised), Argonne National Laboratory, 1959.

Atualmente há diversos algoritmos que pertence à essa classe, por exemplo, o algoritmo DFP (**D**avison, **F**letcher e **P**owell) utiliza

$$W_{t+1} = W_t + \frac{\tilde{\delta}_t \tilde{\delta}_t'}{\tilde{\delta}_t' \nu_t} + \frac{W_t \nu_t \nu_t' W_t}{\nu_t' W_t \tilde{\delta}_t},$$

Métodos quasi-Newton

A ideia central dos métodos quasi-Newton é apresentada no artigo:

W. C. Davison. **Variable metric method for minimization**. Technical Report ANL-5990 (revised), Argonne National Laboratory, 1959.

Atualmente há diversos algoritmos que pertence à essa classe, por exemplo, o algoritmo DFP (**D**avison, **F**letcher e **P**owell) utiliza

$$W_{t+1} = W_t + \frac{\tilde{\delta}_t \tilde{\delta}_t'}{\tilde{\delta}_t' \nu_t} + \frac{W_t \nu_t \nu_t' W_t}{\nu_t' W_t \tilde{\delta}_t},$$

em que $\tilde{\delta}_t = \tilde{\theta}_{t+1} - \tilde{\theta}_t$ e $\nu_t = f(\tilde{\theta}_{t+1}) - f(\tilde{\theta}_t)$.

Métodos quasi-Newton

Método BFGS (Broyden, Fletcher, Goldfarb e Shannon):

O algoritmo quasi-Newton mais utilizado é o BFGS. Este algoritmo tem geralmente desempenho melhor que o algoritmo DFP, sendo este o motivo do método BFGS ser mais utilizado em aplicações práticas. Aqui temos:

$$\begin{aligned} W_{t+1} = & \quad W_t + \frac{\delta_t \delta_t'}{\delta_t' \nu_t} + \frac{W_t \nu_t \nu_t' W_t}{\nu_t' W_t \delta_t} - \nu_t' W_t \nu_t \left(\frac{\delta_t}{\delta_t' \nu_t} - \frac{W_t \nu_t}{\nu_t' W_t \nu_t} \right) \times \\ & \left(\frac{\delta_t}{\delta_t' \nu_t} - \frac{W_t \nu_t}{\nu_t' W_t \nu_t} \right)' . \end{aligned}$$

Maximizando/Minimizando no R

Na linguagem R, podemos fazer uso da função `optim()` para minimizar uma função objetivo. A função `optim()` está disponível no pacote **stats** que vem por padrão em qualquer instalação da linguagem R.

A função `optim()` por padrão minimiza uma função. Caso o interesse seja maximizar uma função objetivo, devemos lembrar de multiplicar a função objetivo por -1 . Assim, passaremos a função esta função como argumento para a função `optim()`. Dessa forma, sairemos facilmente de um problema de minimização para um problema de maximização.

Maximizando/Minimizando no R

Na linguagem R, podemos fazer uso da função `optim()` para minimizar uma função objetivo. A função `optim()` está disponível no pacote **stats** que vem por padrão em qualquer instalação da linguagem R.

A função `optim()` por padrão minimiza uma função. Caso o interesse seja maximizar uma função objetivo, devemos lembrar de multiplicar a função objetivo por -1 . Assim, passaremos a função esta função como argumento para a função `optim()`. Dessa forma, sairemos facilmente de um problema de minimização para um problema de maximização.

Exercício: Estude a documentação da função `optim()`.

Maximizando/Minimizando no R

Exercício: Utilize a função `optim()` para **minimizar** pelo método **BFGS** a função **Rosenbrock function** introduzida por Howard H. Rosenbrock em 1960 utilizada para testar a performance de algoritmos de otimização. A função de Rosenbrock é definida como:

Maximizando/Minimizando no R

Exercício: Utilize a função `optim()` para **minimizar** pelo método **BFGS** a função **Rosenbrock function** introduzida por Howard H. Rosenbrock em 1960 utilizada para testar a performance de algoritmos de otimização. A função de Rosenbrock é definida como:

$$f(x, y) = (a - x)^2 + b(y - x^2)^2,$$

em que usualmente $a = 1$ e $b = 100$.

Maximizando/Minimizando no R

Exercício: Utilize a função `optim()` para **minimizar** pelo método **BFGS** a função **Rosenbrock function** introduzida por Howard H. Rosenbrock em 1960 utilizada para testar a performance de algoritmos de otimização. A função de Rosenbrock é definida como:

$$f(x, y) = (a - x)^2 + b(y - x^2)^2,$$

em que usualmente $a = 1$ e $b = 100$.

Solução: Como temos uma função $f : \mathbb{R}^p \rightarrow \mathbb{R}$, com $p = 2$, parece conveniente construir o gráfico da função objetivo $f(x, y)$.

Poderemos produzir gráficos de superfícies em R de diversas formas. Apresentaremos a seguir duas formas de se obter tais gráficos.

Maximizando/Minimizando no R

Nota: Instale o pacote **plot3D** para que seja possível produzir a segunda forma de gráfico de superfície apresentada adiante.

Maximizando/Minimizando no R

Nota: Instale o pacote **plot3D** para que seja possível produzir a segunda forma de gráfico de superfície apresentada adiante.

```
16 # Substitua o diretorio de trabalho abaixo para um diretorio
17 # de interesse em seu computador.
18 setwd("diretorio")
19
20 rosenbrock <- function(x,y) {
21 100 * (y - x * x)^2 + (1 - x)^2
22 }
23 # Forma 1
24 x <- seq(-100, 100, length = 30)
25 y <- x
26 z <- outer(X = x, Y = y, rosenbrock)
27
28 pdf(file="rosenbrock1.pdf", width=9, height=9, paper="
29     special", family="Bookman", pointsize=14)
30 persp(x, y, z, theta = 3, phi = 40, expand = 0.5, col =
31     "lightblue")
32 dev.off()
```

Maximizando/Minimizando no R

```
31 #Forma 2
32
33 library(plot3D)
34 M <- mesh(seq(-100, 100, length.out = 500) ,
35 seq(-100, 100, length.out = 500))
36 x <- M$x ; y <- M$y
37 z <- rosenbrock(x=x,y=y)
38
39 pdf(file="rosenbrock2.pdf",width=9,height=9, paper="special"
      , family="Bookman",pointsize=14)
40 surf3D(x,y,z,inttype=1,bty="b2",phi=40, theta=3)
41 dev.off()
```

Maximizando/Minimizando no R

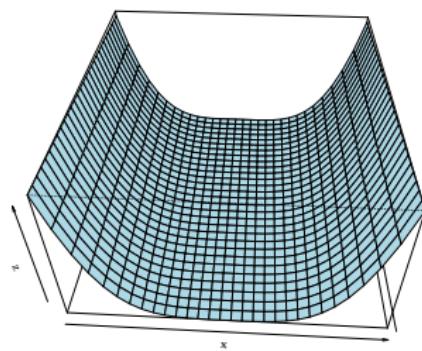


Figure: Forma 1 usando a função `persp()`.

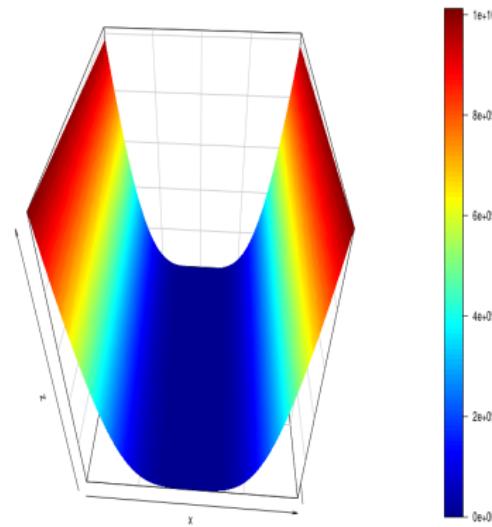


Figure: Forma 2 usando a função `surf3D()` do pacote **plot3D**.

Maximizando/Minimizando no R

Após uma ideia visual do gráfico da função $f(x, y)$, procederemos com o processo de minimização de $f(x, y)$ pelo método **BFGS** utilizando a função `optim()`. Assim, continuando o programa logo acima, temos:

```
42 f_rosenbrock <- function(par) {  
43   x <- par[1]  
44   y <- par[2]  
45   100 * (y - x * x)^2 + (1 - x)^2  
46 }  
47 # Utilizamos como chutes iniciais x = 1 e y = 1.  
48 resultado <- optim(par = c(3.51997,12.40602), fn = f_  
        rosenbrock, method = "BFGS")  
49 resultado
```

Maximizando/Minimizando no R

```
50 #> $par
51 #> [1] 0.9999297 0.9998591
52 #>
53 #> $value
54 #> [1] 4.951026e-09
55 #>
56 #> $counts
57 #> function gradient
58 #> 249      52
59 #>
60 #> $convergence
61 #> [1] 0
62 #>
63 #> $message
64 #> NULL
```

Maximizando/Minimizando no R

Importante: Ao utilizarmos uma função de algum software ou linguagem de programação que implementa um método de otimização, não necessariamente iremos obter convergência. Normalmente o retorno das funções assinalam a convergência ou não convergência de um método aplicado à uma função objetivo.

Entendendo os retornos da função optim():

Maximizando/Minimizando no R

Importante: Ao utilizarmos uma função de algum software ou linguagem de programação que implementa um método de otimização, não necessariamente iremos obter convergência. Normalmente o retorno das funções assinalam a convergência ou não convergência de um método aplicado à uma função objetivo.

Entendendo os retornos da função `optim()`:

- `$par`: retornam os últimos valores encontrados dos parâmetros da função objetivo candidatos à ponto de mínimo global;

Maximizando/Minimizando no R

Importante: Ao utilizarmos uma função de algum software ou linguagem de programação que implementa um método de otimização, não necessariamente iremos obter convergência. Normalmente o retorno das funções assinalam a convergência ou não convergência de um método aplicado à uma função objetivo.

Entendendo os retornos da função `optim()`:

- `$par`: retornam os últimos valores encontrados dos parâmetros da função objetivo candidatos à ponto de mínimo global;
- `$value`: valor de mínimo, isto é, valor da função aplicada aos parâmetros estimados retornados por `$par`;

Maximizando/Minimizando no R

- \$counts: número de chamadas à função objetivo e ao vetor gradiente realizadas pelo algoritmo;

Maximizando/Minimizando no R

- \$counts: número de chamadas à função objetivo e ao vetor gradiente realizadas pelo algoritmo;
- \$convergence: se retornado 0, **houve** convergência do algoritmo;

Maximizando/Minimizando no R

- \$counts: número de chamadas à função objetivo e ao vetor gradiente realizadas pelo algoritmo;
- \$convergence: se retornado 0, **houve** convergência do algoritmo;
- \$message: em alguns casos, poderá sair algum retorno com mensagens adicionais sobre o processo de otimização.

Maximizando/Minimizando no R

- \$counts: número de chamadas à função objetivo e ao vetor gradiente realizadas pelo algoritmo;
- \$convergence: se retornado 0, **houve** convergência do algoritmo;
- \$message: em alguns casos, poderá sair algum retorno com mensagens adicionais sobre o processo de otimização.

Importante: Se \$convergence retornar zero, houve convergência e qualquer número diferente de zero implica não convergência do algoritmo utilizado e especificado no argumento `methods` da função `optim()`.

Maximizando/Minimizando no R

Notas:

Maximizando/Minimizando no R

Notas:

- Um dos números que podem sair como resultado de \$convergence é o número 1. Nesse caso, temos que o número máximo de iterações do algoritmo foi atingido. O número máximo de iterações poderá ser passado à função optim() por meio da passagem de control = list(maxit = 10000) à função optim().

Maximizando/Minimizando no R

Notas:

- Um dos números que podem sair como resultado de \$convergence é o número 1. Nesse caso, temos que o número máximo de iterações do algoritmo foi atingido. O número máximo de iterações poderá ser passado à função optim() por meio da passagem de control = list(maxit = 10000) à função optim().
- É muito comum que um algoritmo não converja a depender dos chutes iniciais passados ao argumento par da função optim(). Em casos de não convergência é conveniente mudar os chutes iniciais e/ou aumentar o número de iterações do algoritmo ou mesmo mudar de método de otimização.

Maximizando/Minimizando no R

Maximizando/Minimizando no R

- As metodologias de otimização que fazem uso de derivadas (método de Newton e quasi-Newton) podem ter problemas em otimização de funções com regiões aproximadamente planas. Nessas situações poderá ser útil utilizar algoritmos heurísticos de otimização como **genetic algorithm** ou algoritmos baseados em **swarm intelligence** como é o caso do método **PSO Particle Swarm Optimization**.

Maximizando/Minimizando no R

- As metodologias de otimização que fazem uso de derivadas (método de Newton e quasi-Newton) podem ter problemas em otimização de funções com regiões aproximadamente planas. Nessas situações poderá ser útil utilizar algoritmos heurísticos de otimização como **genetic algorithm** ou algoritmos baseados em **swarm intelligence** como é o caso do método **PSO Particle Swarm Optimization**.

O método **PSO** encontra-se implementado no pacote **AdequacyModel** disponível no CRAN da linguagem R.

Maximizando/Minimizando no R

- As metodologias de otimização que fazem uso de derivadas (método de Newton e quasi-Newton) podem ter problemas em otimização de funções com regiões aproximadamente planas. Nessas situações poderá ser útil utilizar algoritmos heurísticos de otimização como **genetic algorithm** ou algoritmos baseados em **swarm intelligence** como é o caso do método **PSO Particle Swarm Optimization**.

O método **PSO** encontra-se implementado no pacote **AdequacyModel** disponível no CRAN da linguagem R.

Exercício: Refaça a otimização da função **Rosenbrock** considerando os chutes iniciais $x_0 = 10$ e $y_0 = 10$. Depois, refaça novamente a mesma otimização considerando os mesmos chutes porém, considere o número de iterações máximo igual à 10000.

Maximizando/Minimizando no R

Exercício: Pesquise sobre método de otimização **Simulated Annealing**. Refaça a otimização da função **Rosenbrock** utilizando o algoritmo **simulated annealing**. **Dica:** É possível fazer uso da função `optim()` da linguagem R para otimizar uma função objetivo utilizando este algoritmo.

Exercício: Pesquise sobre o método **PSO** (estude o algoritmo) e refaça a otimização da função **Rosenbrock** utilizando o algoritmo **PSO** implementado no pacote **AdequacyModel**.

Maximizando/Minimizando no R

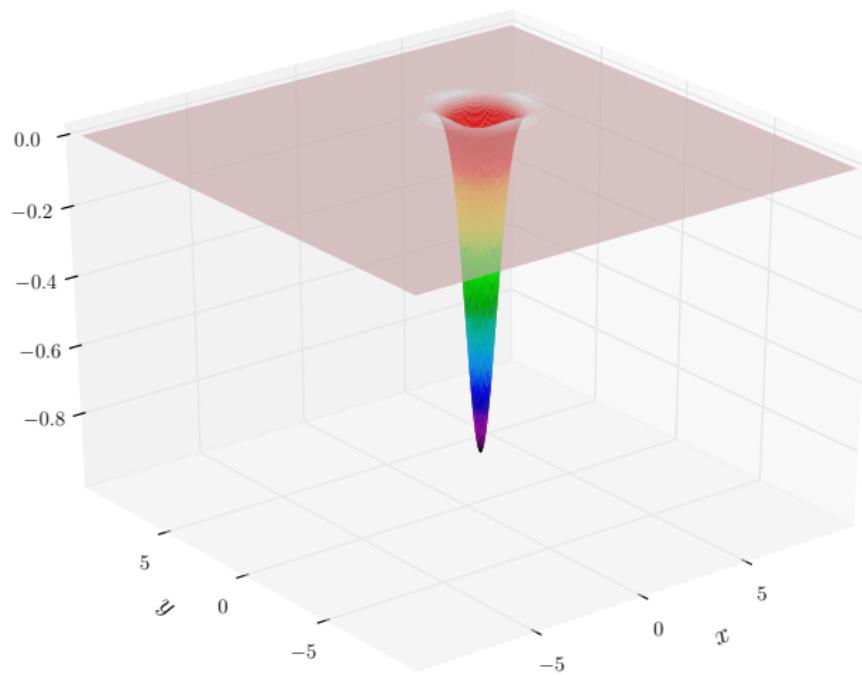
O pacote AdequacyModel versão 2.0.0 apresenta a função pso que permite que seja realizado uma otimização de uma função arbitrária podendo esta função envolver um dado conjunto de dados como nos casos das funções de log-verossimilhanças.

Exemplo: Considere a função Easom

$$f(x, y) = -\cos(x) \cos(y) \exp\{-(x - \pi)^2 + (y - \pi)^2\},$$

e $-10 \leq x, y \leq 10$. A função de Easom é minimizada no ponto $x = y = \pi$, com $f(\pi, \pi) = -1$. O uso da função pso para minimizar esta função é dado abaixo:

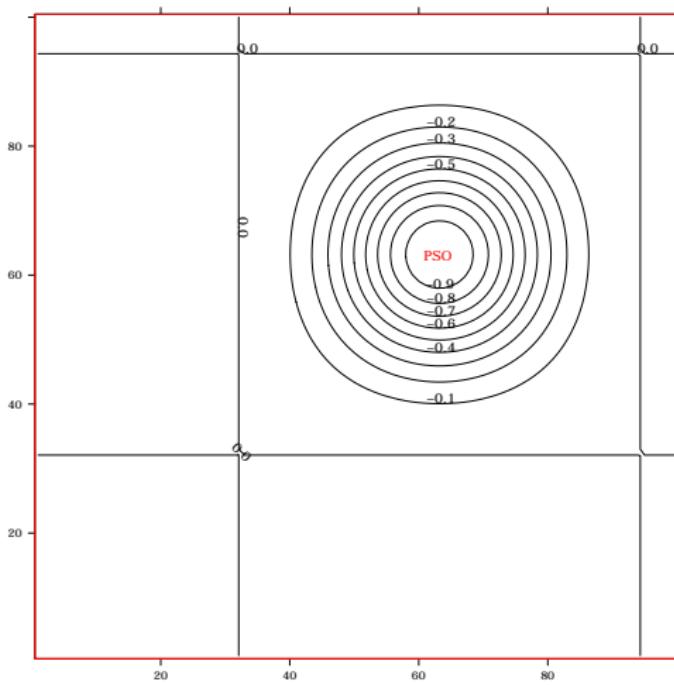
Maximizando/Minimizando no R



Função pso do pacote **AdequacyModel**

(Loading Video...)

Maximizando/Minimizando no R



Maximizando/Minimizando no R

Exemplo: Consideremos o caso da função Hölder, muito peculiar e difícil de ser otimizada. Ela é definida por

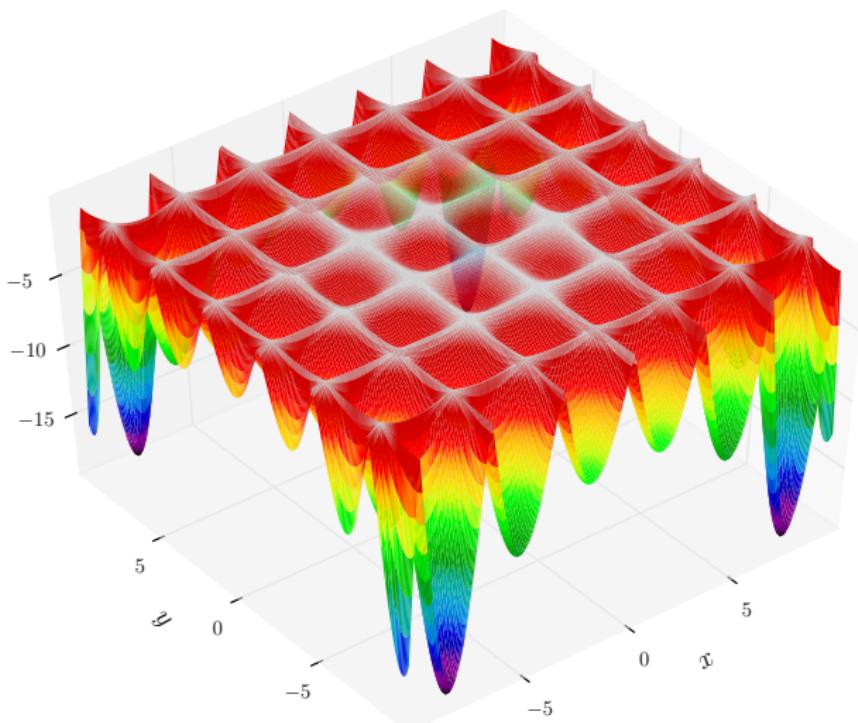
$$f(x, y) = - \left| \sin(x) \cos(y) \exp \left(\left| 1 - \frac{\sqrt{x^2 + y^2}}{\pi} \right| \right) \right|,$$

em que

$$\text{Min} = \begin{cases} f(8.05502, 9.66459) &= -19.2085 \\ f(-8.05502, 9.66459) &= -19.2085 \\ f(8.05502, -9.66459) &= -19.2085 \\ f(-8.05502, -9.66459) &= -19.2085 \end{cases},$$

e $-10 \leq x, y \leq 10$.

Maximizando/Minimizando no R



Função pso do pacote **AdequacyModel**

(Loading Video...)

Projeto Manhattan



Figura: Primeiro teste nuclear *Trinity* em 16 de julho de 1945.

O Projeto Manhattan envolveu uma das maiores colaborações científicas já realizadas. Nesse projeto surgiram inúmeras novas tecnologias, indo muito além do aproveitamento da fissão nuclear.

Foi um projeto de pesquisa e desenvolvimento que produziu as primeiras bombas atômicas durante a Segunda Guerra Mundial.

Projeto Manhattan

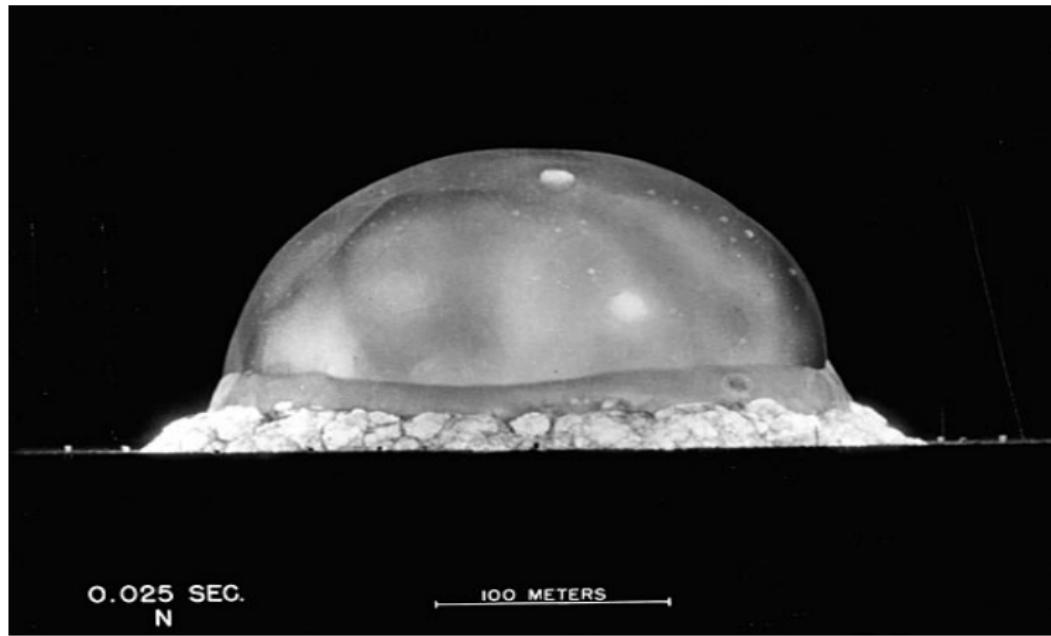


Figure: Primeiro teste nuclear (nome de código *Trinity*), uma das três bombas atômicas produzidas pelo Projeto Manhattan.

Projeto Manhattan

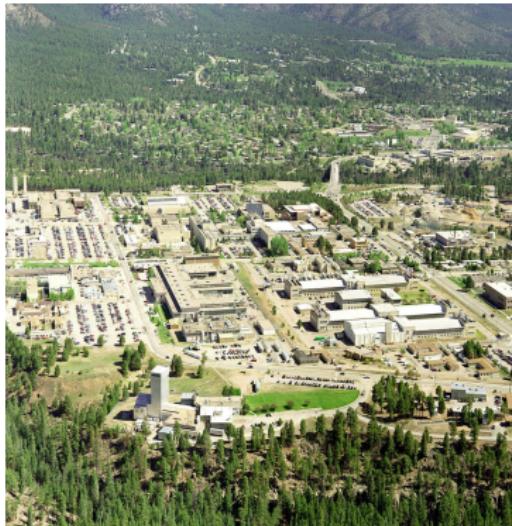


Figura: Laboratório de Los Alamos,
Novo México.

O projeto foi liderado pelos Estados Unidos e teve apoio do Reino Unido e Canadá e iniciou-se em 1939.

O Projeto Manhattan foi conduzido no Laboratório Nacional de Los Alamos, construído para essa finalidade. Atualmente o laboratório trata-se de uma das maiores instituições científicas multidisciplinares.

Método de Monte Carlo

Os métodos de Monte Carlo surgiu durante o projeto Manhattan na Segunda Guerra Mundial e trata-se de uma ampla classe de metodologias que dependem de amostragem aleatória repetida para obter resultados numéricos. No projeto, foi utilizado o método de Monte-Carlo para simular a difusão de nêutrons em certos materiais ao explodir uma bomba atômica.

Por que Monte Carlo?

Por se tratar de uma metodologia que faz uso de geração de números aleatórios, um dos integrantes do projeto Manhattan (matemático polaco Stanislaw Ulam) batizou a metodologia com o nome do famoso cassino de Monte Carlo em Mônaco, cassino este em que seu tio gostava de jogar.

Método de Monte Carlo

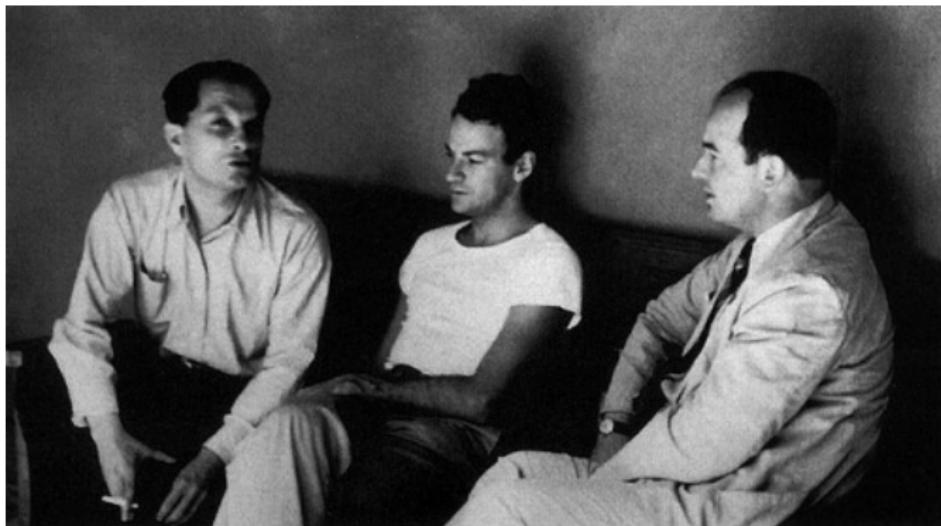


Figure: Stanislaw Ulam, Richard Feynman e John von Neumann em 1944 durante o Projeto Manhattan.

Método de Monte Carlo

O método de Monte-Carlo consiste observar um fenômeno aleatório de interesse em um cenário virtual, isto é, em observar o comportamento de um dado fenômeno sob um conjunto aleatório de possíveis cenários que são obtidos pela geração de números pseudo-aleatórios (números aleatórios gerados em um computador).

As metodologias de Monte Carlo são amplamente utilizadas na estatística quando se quer estudar um determinado modelo ou estimador em diferentes cenários (amostras) distintas. Dessa forma, o modelo ou o estimador proposto são testados exaustivamente em diferentes cenários e os resultados observados são registrados para cada um dos cenários. Espera-se que um bom modelo ou estimador apresente bons resultados (**em média**) nos cenários ao qual foram submetidos.

Método de Monte Carlo

Um exemplo do uso do método de Monte-Carlo é o cálculo aproximado da constante $\pi \approx 3.141593\dots$ que poderia ser obtido calculando a proporção de pontos aleatórios gerados em um quadrado definido em $[-1, 1] \times [-1, 1]$ que caem dentro de um círculo de raio 1 centrado no ponto $(0,0)$ com o total de pontos gerados no quadrado.

O próximo slide mostra uma animação do método de Monte Carlo para uma aproximação da constante π .

Método de Monte Carlo

(Loading Video...)

Método de Monte Carlo

Para uma circunferência de raio 1 centrada no ponto $(0, 0)$, temos que a área da circunferência é igual à π .

Método de Monte Carlo

Para uma circunferência de raio 1 centrada no ponto $(0, 0)$, temos que a área da circunferência é igual à π .

Perceba que se gerarmos pontos no interior do quadrado de área 4, $P(X^2 + Y^2 \leq 1) = \frac{\pi}{4}$, uma vez que a área da circunferência é uma proporção da área do quadrado. Além disso, podemos estimar $P(X^2 + Y^2 \leq 1)$ pela proporção de pontos no interior do círculo de raio unitário pelo número total de pontos gerado. Assim,

Método de Monte Carlo

Para uma circunferência de raio 1 centrada no ponto $(0, 0)$, temos que a área da circunferência é igual à π .

Perceba que se gerarmos pontos no interior do quadrado de área 4, $P(X^2 + Y^2 \leq 1) = \frac{\pi}{4}$, uma vez que a área da circunferência é uma proporção da área do quadrado. Além disso, podemos estimar $P(X^2 + Y^2 \leq 1)$ pela proporção de pontos no interior do círculo de raio unitário pelo número total de pontos gerado. Assim,

$$P(X^2 + Y^2 \leq 1) = \frac{\pi}{4} \approx \frac{\#C}{n},$$

Método de Monte Carlo

Para uma circunferência de raio 1 centrada no ponto $(0, 0)$, temos que a área da circunferência é igual à π .

Perceba que se gerarmos pontos no interior do quadrado de área 4, $P(X^2 + Y^2 \leq 1) = \frac{\pi}{4}$, uma vez que a área da circunferência é uma proporção da área do quadrado. Além disso, podemos estimar $P(X^2 + Y^2 \leq 1)$ pela proporção de pontos no interior do círculo de raio unitário pelo número total de pontos gerado. Assim,

$$P(X^2 + Y^2 \leq 1) = \frac{\pi}{4} \approx \frac{\#C}{n},$$

em que $\#C$ é o número de pontos gerados no interior da circunferência e n é o total de pontos gerado no interior do quadrado.

Método de Monte Carlo

Logo, temos que uma estimativa para o valor de π por meio de Monte Carlo é:

Método de Monte Carlo

Logo, temos que uma estimativa para o valor de π por meio de Monte Carlo é:

$$\pi \approx 4 \times \frac{\#C}{n}.$$

Nota: A aproximação da constante π é melhorada a medida que $n \rightarrow +\infty$ (a medida que o número de iterações de MC cresce).

Método de Monte Carlo

Logo, temos que uma estimativa para o valor de π por meio de Monte Carlo é:

$$\pi \approx 4 \times \frac{\#C}{n}.$$

Nota: A aproximação da constante π é melhorada a medida que $n \rightarrow +\infty$ (a medida que o número de iterações de MC cresce).

Exercício: Implemente uma função de nome `mc_pi` que realiza uma simulação de Monte Carlo (MC) para aproximação da constante π . A função deverá ter como argumento o número de iterações de MC que será considerada.

Método de Monte Carlo

Solução:

```
1 plot_circ <- function(raio = 1, origem = c(0,0)){
2   a <- origem[1]
3   b <- origem[2]
4   x <- seq(a - raio, a + raio, by = 0.0001)
5   y1 <- b + sqrt(raio^2 - (x-a)^2) # Parte superior.
6   y2 <- b - sqrt(raio^2 - (x-a)^2) # Parte inferior.
7
8   plot.new()
9   plot.window(xlim = c(a-raio,a+raio), ylim = c(b-raio,b+raio))
10  axis(1)
11  axis(2)
12  points(c(x,x), c(y1,y2), cex = 0.2)
13  lines(c(min(x), max(x)), c(origem[2],origem[2]), lwd = 1,
14        lty = 2)
15  lines(c(origem[1], origem[1]), c(min(y2),max(y1)), lwd = 1,
16        lty = 2)
17 }
```

Método de Monte Carlo

```
16 mc_pi <- function(n = 30000, graphic = FALSE){  
17   if(graphic != FALSE){  
18     plot_circ(raio = 1, origem = c(0,0))  
19     lines(c(-1,-1),c(1,-1))  
20     lines(c(1,1),c(-1,1))  
21     lines(c(-1,1),c(1,1))  
22     lines(c(-1,1),c(-1,-1))  
23   }  
24   # Tente fazer tem usar loop.  
25   count <- 0  
26   for(i in 1:n){  
27     coord <- runif(n = 2, min = -1, max = 1)  
28     if(coord[1]^2 + coord[2]^2 <= 1){  
29       count <- count + 1  
30     if(graphic != FALSE) points(coord[1], coord[2], pch = 16,  
31       col = "red")  
32   }else{  
33     if(graphic != FALSE) points(coord[1], coord[2], pch = 16,  
34       col = "blue")  
35   }  
36 }; 4*count/n  
37 }; mc_pi(n = 10000, graphic = T)
```



Método de Monte Carlo

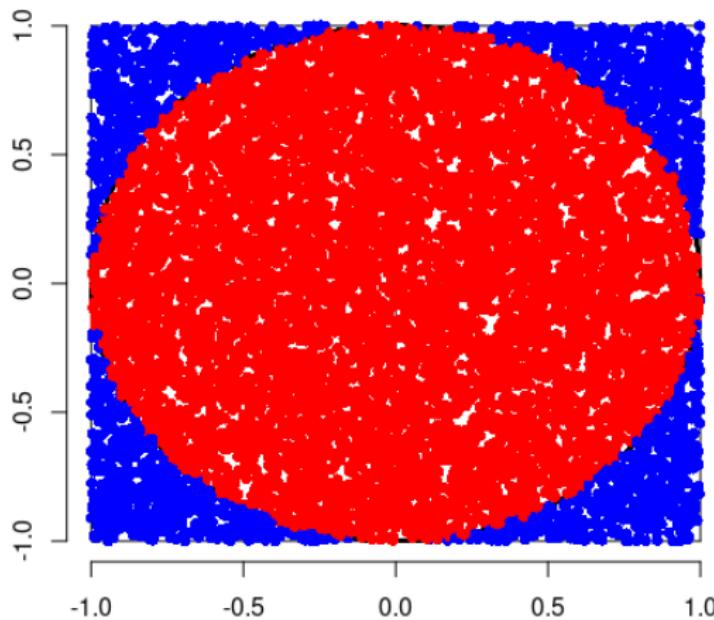


Figure: Pontos gerados pela simulação de Monte Carlo do programa anterior para aproximação da constante π .

Método de Monte Carlo

Dessa forma, o procedimento poderá ser generalizado para o cálculo da área de uma região com formas não bem comportada, como a área da região abordada por um lago.

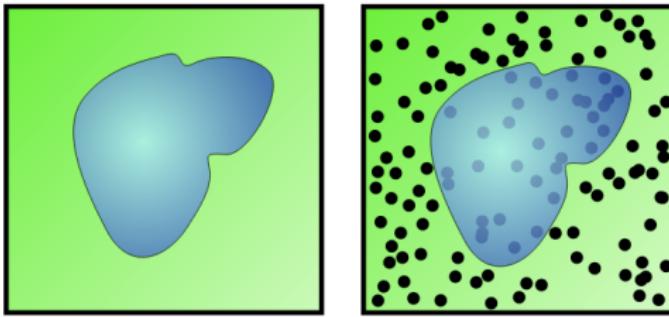


Figure: Uso de método de Monte Carlo para o cálculo da área de um lago.

Método de Monte Carlo

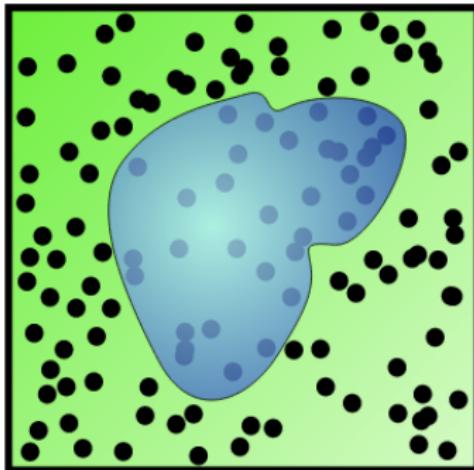


Figura: Área de um lago estimada pelo método de Monte Carlo.

A área do lago (região em azul) pode ser facilmente estimada contando o número de pontos aleatórios contidos na região do lago e dividindo pelo total de pontos em toda região. O problema consiste em arrumar uma forma de contar os pontos na região azul. Quanto mais pontos se considera, melhor a qualidade da aproximação da área real do lago.

Método de Monte Carlo

Uma das inúmeras utilizações do método de Monte Carlo é na estimativa de integrais definidas. Seja $h(x)$ uma função contínua em todo ponto no intervalo (a, b) . Estamos interessados em calcular

$$I = \int_a^b h(x)dx.$$

Note que poderemos reescrever a integral acima como

$$I = \int_a^b h(x)dx = \int_a^b (b-a)h(x)\frac{1}{b-a}dx.$$

Método de Monte Carlo

Note também que se $X \sim U(a, b)$, então, $f_X(x) = \frac{1}{b-a}$, $\forall x \in (a, b)$ e $f_X(x) = 0$ para todo ponto fora de (a, b) . Assim,

$$I = (b - a) \int_a^b h(x)f_X(x)dx = (b - a)E[h(X)].$$

Dessa forma, o problema para calcular a integral I reduz ao problema de se calcular $E[h(X)]$. Se dispomos de uma amostra aleatória de tamanho n , com X_1, \dots, X_n , com $X_i \sim U(a, b)$, também temos que $h(X_1), \dots, h(X_n)$ também é uma amostra aleatória, com h conhecida e definida em (a, b) . Denotemos μ como sendo

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

Método de Monte Carlo

Um estimador para μ que é não viesado e que goza de boas propriedades (consistência e suficiência) é definido por

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

Daí, temos que uma estimativa para $I = \int_a^b h(x)dx$ é dada por:

$$\hat{I} = (b - a)\hat{\mu}.$$

Método de Monte Carlo

Observe que o estimador \hat{I} é não viesado para I . De fato,

$$E(\hat{I}) = \frac{(b-a)}{n} \sum_{i=1}^n E[h(X_i)] = \int_a^b h(x)dx.$$

O estimador \hat{I} também goza de outras propriedades como suficiência e consistência I . Dessa forma, é razoável estimar I por meio de \hat{I} .

Método de Monte Carlo

Algoritmo de integração por Monte Carlo:

Método de Monte Carlo

Algoritmo de integração por Monte Carlo:

- 1 Gere uma sequência de números pseudo-aleatórios x_1, \dots, x_n da distribuição $U(a, b)$;

Método de Monte Carlo

Algoritmo de integração por Monte Carlo:

- 1 Gere uma sequência de números pseudo-aleatórios x_1, \dots, x_n da distribuição $U(a, b)$;
- 2 Obtenha $h(x_1), \dots, h(x_n)$;

Método de Monte Carlo

Algoritmo de integração por Monte Carlo:

- 1 Gere uma sequência de números pseudo-aleatórios x_1, \dots, x_n da distribuição $U(a, b)$;
- 2 Obtenha $h(x_1), \dots, h(x_n)$;
- 3 Calcule $\bar{h} = n^{-1} \sum_{i=1}^n h(x_i)$;

Método de Monte Carlo

Algoritmo de integração por Monte Carlo:

- 1 Gere uma sequência de números pseudo-aleatórios x_1, \dots, x_n da distribuição $U(a, b)$;
- 2 Obtenha $h(x_1), \dots, h(x_n)$;
- 3 Calcule $\bar{h} = n^{-1} \sum_{i=1}^n h(x_i)$;
- 4 Obtenha a estimativa da integral I por $\hat{I} = (b - a)\bar{h}$.

Método de Monte Carlo

Algoritmo de integração por Monte Carlo:

- 1 Gere uma sequência de números pseudo-aleatórios x_1, \dots, x_n da distribuição $U(a, b)$;
- 2 Obtenha $h(x_1), \dots, h(x_n)$;
- 3 Calcule $\bar{h} = n^{-1} \sum_{i=1}^n h(x_i)$;
- 4 Obtenha a estimativa da integral I por $\hat{I} = (b - a)\bar{h}$.

Como \hat{I} é um estimador consistente para I , espera-se melhores estimativas de I a medida que aumentamos o valor de n .

Método de Monte Carlo

Exercício: Utilizando o método de Monte Carlo para integração numérica, construa uma função de nome `int_mc` para aproximação de uma integral simples de uma função em um intervalo $[a, b]$ do domínio da função. O usuário deverá passar como argumento para a função `int_mc()` a função ao qual deseja-se uma aproximação à integral definida.

Bootstrap



Bootstrap

O nome **bootstrap** foi utilizado para se referir à uma área da estatística que é empregada em diversos estudos e artigos científicos. O uso na estatística da palavra **bootstrap** teve origem de uma frase da obra literária de Rudolf Erich Raspe (1736 à 1794) que foi um escritor e cientista alemão. Em uma de suas mais conhecidas obra intitulada **The Surprising Adventures of Baron Munchausen** é dito a seguinte frase:

Bootstrap

O nome **bootstrap** foi utilizado para se referir à uma área da estatística que é empregada em diversos estudos e artigos científicos. O uso na estatística da palavra **bootstrap** teve origem de uma frase da obra literária de Rudolf Erich Raspe (1736 à 1794) que foi um escritor e cientista alemão. Em uma de suas mais conhecidas obra intitulada **The Surprising Adventures of Baron Munchausen** é dito a seguinte frase:

“The Baron had fallen to the bottom of a deep lake. Just when it looked like all was lost, he thought to pick himself up by his own bootstraps.” (“O Barão tinha caído no fundo de um lago profundo. Justo quando parecia que tudo estava perdido, ele pensou em se retirar por seus próprios bootstraps.”)

Bootstrap

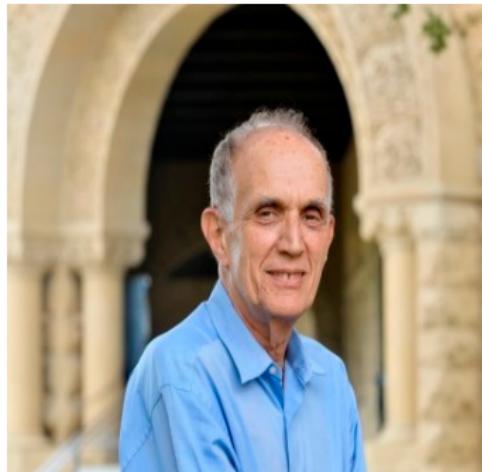


Figura: Bradley Efron.

Bootstrap em estatística é um método de **reamostragem** proposto por **Bradley Efron** em **1979**.

Bootstrap é uma das metodologias estatísticas mais utilizadas quando se carece de resultados analíticos, o que é bastante comum quando não conhecemos a distribuição dos dados ou de uma estatística de interesse.

Bootstrap

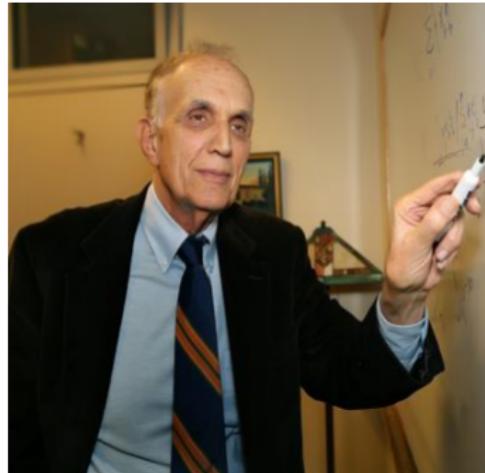


Figura: Bradley Efron.

Efron criou o **bootstrap** inspirado no método **jackknife** desenvolvido por Maurice Quenouille (1949, 1956) que é uma teoria mais antiga de reamostragem que mesmo sendo um caso particular do **bootstrap** ainda continua sendo utilizada, principalmente na correção de viés de estimadores.

Pode-se dizer que **jackknife** é uma aproximação ao **bootstrap**.

Bootstrap

Bootstrap é largamente utilizado para se obter:

- Estimativa do erro-padrão de um estimador;
- Construção de intervalos aleatórios para um parâmetro populacional;
- Testar hipóteses;
- Correção de viés de um estimador;

Bootstrap

Bootstrap é largamente utilizado para se obter:

- Estimativa do erro-padrão de um estimador;
- Construção de intervalos aleatórios para um parâmetro populacional;
- Testar hipóteses;
- Correção de viés de um estimador;

Cuidado: **Bootstrap** não serve para estimar pontualmente um parâmetro populacional. A metodologia **bootstrap** aproxima a forma da distribuição do estimador porém não estima a locação da distribuição.

Bootstrap

A referência do primeiro artigo sobre **bootstrap** encontra-se logo abaixo:

Bootstrap

A referência do primeiro artigo sobre **bootstrap** encontra-se logo abaixo:

Efron, B. (1979). **Bootstrap methods: Another look at the jackknife.** *The Annals of Statistics*. 7, 1-26.

Bootstrap

A referência do primeiro artigo sobre **bootstrap** encontra-se logo abaixo:

Efron, B. (1979). **Bootstrap methods: Another look at the jackknife.** *The Annals of Statistics*. 7, 1-26.

Anos depois, Bradley Efron e Robert Tibshirani publicam um artigo de revisão cuja referência encontra-se abaixo:

Bootstrap

A referência do primeiro artigo sobre **bootstrap** encontra-se logo abaixo:

Efron, B. (1979). **Bootstrap methods: Another look at the jackknife.** *The Annals of Statistics*. 7, 1-26.

Anos depois, Bradley Efron e Robert Tibshirani publicam um artigo de revisão cuja referência encontra-se abaixo:

Efron, B., Tibshirani, R. **Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy.** *Statistical Science* (1986), 1, 54-75.

Bootstrap

Inicialmente houve ceticismo sobre a metodologia **bootstrap**, tendo sido tal ceticismo superado à medida em que estudos acumularam evidências de que o bootstrap pode ser consideravelmente mais eficaz que metodologias tradicionais.

A ideia de substituir aproximações complicadas e muitas vezes imprecisas por métodos de simulação baseados em reamostragem tem atraído diversos pesquisadores a desenvolver metodologias baseadas em **bootstrap** para os mais variados fins. Com a popularização do método **bootstrap**, alguns pesquisadores começaram a estabelecer condições matemáticas sob as quais o **bootstrap** é justificável.

Bootstrap

As metodologias **bootstrap** apresentam dois paradigmas, sendo eles o **bootstrap** paramétrico e o **bootstrap** não-paramétrico. **Bootstrap** paramétrico refere-se ao caso em que a reamostragem é feita com base em uma distribuição $F(\hat{\theta})$ conhecida ou estabelecida, em que $\hat{\theta}$ é um estimador para θ . Em contrapartida, no **bootstrap** não-paramétrico há o desconhecimento da distribuição F verdadeira. A reamostragem é feita com base na função de distribuição empírica \hat{F}_n . Reamostrar de \hat{F}_n equivale a reamostrar dos dados com reposição.

Notação:

Bootstrap

As metodologias **bootstrap** apresentam dois paradigmas, sendo eles o **bootstrap** paramétrico e o **bootstrap** não-paramétrico. **Bootstrap** paramétrico refere-se ao caso em que a reamostragem é feita com base em uma distribuição $F(\hat{\theta})$ conhecida ou estabelecida, em que $\hat{\theta}$ é um estimador para θ . Em contrapartida, no **bootstrap** não-paramétrico há o desconhecimento da distribuição F verdadeira. A reamostragem é feita com base na função de distribuição empírica \hat{F}_n . Reamostrar de \hat{F}_n equivale a reamostrar dos dados com reposição.

Notação:

- Parâmetro: $\theta = T(F)$, em que F é a distribuição verdadeira (desconhecida) dos dados.

Bootstrap

As metodologias **bootstrap** apresentam dois paradigmas, sendo eles o **bootstrap** paramétrico e o **bootstrap** não-paramétrico. **Bootstrap** paramétrico refere-se ao caso em que a reamostragem é feita com base em uma distribuição $F(\hat{\theta})$ conhecida ou estabelecida, em que $\hat{\theta}$ é um estimador para θ . Em contrapartida, no **bootstrap** não-paramétrico há o desconhecimento da distribuição F verdadeira. A reamostragem é feita com base na função de distribuição empírica \hat{F}_n . Reamostrar de \hat{F}_n equivale a reamostrar dos dados com reposição.

Notação:

- Parâmetro: $\theta = T(F)$, em que F é a distribuição verdadeira (desconhecida) dos dados.
- Estimador: $\hat{\theta} = T_n$ que obtido por meio de \hat{F}_n .

Bootstrap

Observação Importante

Bootstrap

Observação Importante

A **função de distribuição empírica** denotada por \hat{F}_n é a função de distribuição dos dados $\tilde{\mathbf{X}} = (\tilde{X}_1, \dots, \tilde{X}_n)$. Sendo assim, podemos utilizar $\tilde{\mathbf{X}}$ como uma aproximação à distribuição verdadeira F de \mathbf{X} . Sendo assim, suponhamos que X_1, \dots, X_n é uma sequência de variáveis aleatórias independentes e identicamente distribuídas. Então,

Bootstrap

Observação Importante

A **função de distribuição empírica** denotada por \hat{F}_n é a função de distribuição dos dados $\tilde{\mathbf{X}} = (\tilde{X}_1, \dots, \tilde{X}_n)$. Sendo assim, podemos utilizar $\tilde{\mathbf{X}}$ como uma aproximação à distribuição verdadeira F de \mathbf{X} . Sendo assim, suponhamos que X_1, \dots, X_n é uma sequência de variáveis aleatórias independentes e identicamente distribuídas. Então,

$$\hat{F}_n(t) = \frac{1}{n} \sum_{i=1}^n \{x_i \leq t\} = \frac{\#\{x_i \leq t\}}{n}.$$

Bootstrap

Exercício: Seja $X \sim \text{Weibull}(\alpha, \beta)$, em que

$$F_X(x) = 1 - \exp\{-(x/\beta)^\alpha\},$$

com $x > 0$ e $\alpha, \beta \in (0, \infty)$. Utilize o método da inversão para gerar uma amostra de 250 observações da distribuição $F_X(x)$. Depois, em R, construa um gráfico da função de distribuição $F_X(x)$ com a distribuição empírica dos dados gerados. **Dica:** Observe o gráfico para diferentes tamanhos de amostra, por exemplo, $n = 10, 20, 50, 100, 1000, 10000$.

Bootstrap

Solução:

```
1 random_weibull <- function(n, alpha, beta){  
2   beta * (-log(1-runif(n = n, min = 0,max = 1)))^(1/alpha)  
3 }  
4  
5 # Fixando uma semente.  
6 set.seed(1)  
7 data <- random_weibull(n = 10000, alpha = 1.7, beta = 2.3)  
8  
9 # Construindo um grafico.  
10 plot.new()  
11 plot.window(xlim = c(0, max(data)), ylim = c(0,1))  
12 axis(1); axis(2)  
13 title(main = "Real Distribution x Empirical Distribution",  
14 xlab = "x", ylab = "Distributions")
```

Bootstrap

```
15 # Funcao de distribuicao verdadeira.  
16 x <- seq(0, max(data), length.out = 1500)  
17 lines(x, pweibull(q = x, scale = 2.3, shape = 1.7), lwd = 2)  
18  
19 # Funcao de distribuicao empirica.  
20 empirical <- function(t, sample){  
21 length(sample[sample <= t])/length(sample)  
22}  
23  
24 # Adicionando a funcao de distribuicao empirica.  
25 x_emp <- seq(0,max(data), length.out = 1500)  
26 y_emp <- sapply(X = x_emp, FUN = empirical, sample = data)  
27 lines(x = x_emp, y = y_emp, col = "red", lwd = 2)  
28  
29 # Adicionando uma legenda ao grafico.  
30 legend("topleft", legend = c(expression(F[X]), expression(  
    hat(F)[n])),  
31 lwd=c(2.5,2.5), inset = 0.03, lty = c(1,1), cex=1.1,  
32 col=c("black","red"), box.lty = 0)
```

Bootstrap

Exercício: Estude os comandos das soluções dos exercícios anteriores para construção de gráficos em R. Tente correr o código dos exercícios e entender as funções específicas para construção gráfica.

Bootstrap

Enfoque Bootstrap:

Bootstrap

Enfoque Bootstrap:

Tratamos a amostra como se fosse a população, amostramos da amostra como se estivéssemos amostrando da população e extraímos informações dessas “pseudo-amostras”.

Bootstrap

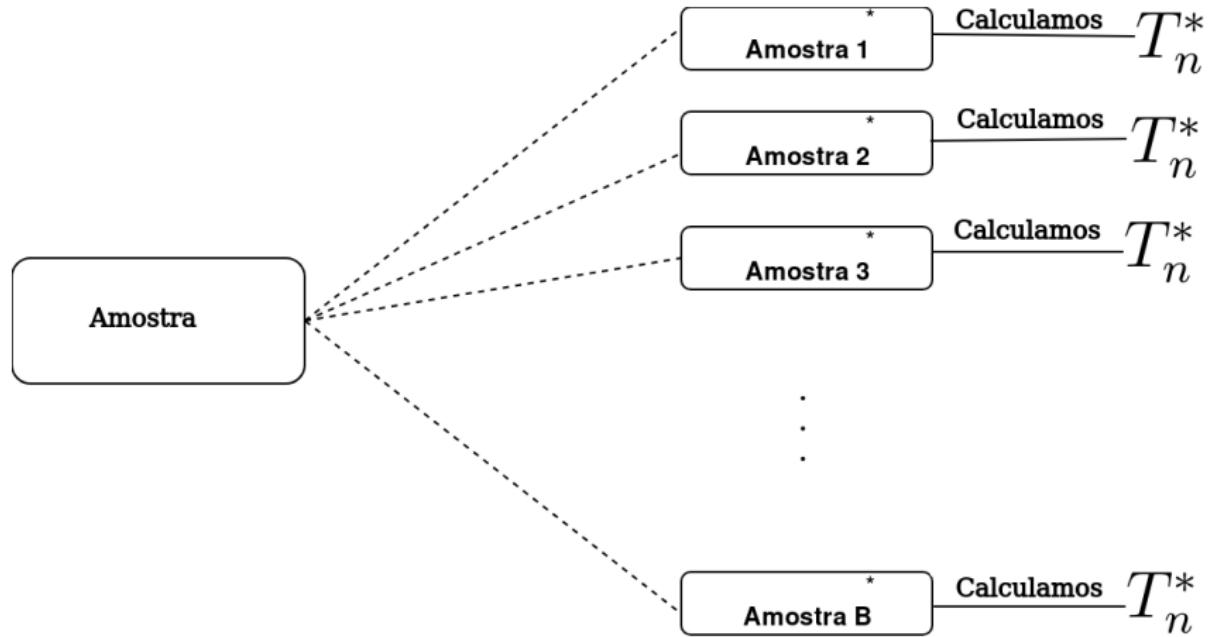


Figure: Esquema bootstrap em que $\hat{\theta}_n^* = T_n^*$.

Bootstrap

Sendo assim, temos uma amostra aleatória $\tilde{\mathbf{X}} = \{\tilde{X}_1, \dots, \tilde{X}_n\}$ que é tratada como uma população e amostraremos B amostras de mesmo tamanho da amostra original \mathbf{X} , gerando assim as “pseudo-amostras” $\tilde{\mathbf{X}}_1^*, \dots, \tilde{\mathbf{X}}_B^*$.

Bootstrap

Sendo assim, temos uma amostra aleatória $\tilde{\mathbf{X}} = \{\tilde{X}_1, \dots, \tilde{X}_n\}$ que é tratada como uma população e amostraremos B amostras de mesmo tamanho da amostra original $\tilde{\mathbf{X}}$, gerando assim as “pseudo-amostras” $\tilde{\mathbf{X}}_1^*, \dots, \tilde{\mathbf{X}}_B^*$.

Seja T_n uma estatística para θ calculada com base na amostra original $\tilde{\mathbf{X}}$ e T_n^* a mesma estatística calculada em alguma das B amostras obtidas de $\tilde{\mathbf{X}}$ (amostral original).

Bootstrap

Sendo assim, temos uma amostra aleatória $\tilde{\mathbf{X}} = \{\tilde{X}_1, \dots, \tilde{X}_n\}$ que é tratada como uma população e amostraremos B amostras de mesmo tamanho da amostra original \mathbf{X} , gerando assim as “pseudo-amostras” $\tilde{\mathbf{X}}_1^*, \dots, \tilde{\mathbf{X}}_B^*$.

Seja T_n uma estatística para θ calculada com base na amostra original \mathbf{X} e T_n^* a mesma estatística calculada em alguma das B amostras obtidas de $\tilde{\mathbf{X}}$ (amostral original).

Notação: Denotamos por t a estimativa obtida pela estatística T_n sob a amostra original e t^* a estimativa obtida pela estatística T_n^* sob alguma das B amostras bootstrap.

Bootstrap

Como mencionado anteriormente, o **bootstrap** possui dois enfoques:

Bootstrap

Como mencionado anteriormente, o **bootstrap** possui dois enfoques:

- 1 **Paramétrico:** Amostramos de uma distribuição $F(\hat{\theta})$, em que F é uma distribuição imposta ou conhecida.

Bootstrap

Como mencionado anteriormente, o **bootstrap** possui dois enfoques:

- 1 **Paramétrico**: Amostramos de uma distribuição $F(\hat{\theta})$, em que F é uma distribuição imposta ou conhecida.
- 2 **Não-Paramétrico**: Amostramos de $\hat{F} \equiv \hat{F}_n$.

Importante: No **bootstrap** não-paramétrico, amostrar de \hat{F} equivale a amostar dos dados com reposição. Sendo assim, obtemos as B amostras **bootstrap** extraíndo as observações da amostra original com reposição.

Bootstrap

Algoritmo bootstrap para a estimação do erro-padrão de um estimador:

Seja T_n um estimador para θ , em que normalmente T_n é o estimador de máxima verossimilhança de θ , em que θ é um parâmetro de uma distribuição de probabilidade ou modelo estatístico. Em situações em que não conhecemos a distribuição verdadeira F_X , podemos estimar o erro-padrão de T_n por meio de **bootstrap** $\hat{\text{se}}(T_n)_{\text{boot}}$.

Bootstrap

- 1 Gere **com reposição**, a partir das observações da amostra original ($x_n = \{x_1, \dots, x_n\}$), uma amostra

Bootstrap

- 1 Gere **com reposição**, a partir das observações da amostra original ($x_n = \{x_1, \dots, x_n\}$), uma amostra **bootstrap** $x_n^* = \{x_1^*, \dots, x_n^*\}$ e calcule T_n^* guardando os resultados.

Bootstrap

- 1 Gere **com reposição**, a partir das observações da amostra original ($x_n = \{x_1, \dots, x_n\}$), uma amostra **bootstrap** $x_n^* = \{x_1^*, \dots, x_n^*\}$ e calcule T_n^* guardando os resultados.
- 2 Repita o passo anterior B vezes.

Bootstrap

- 1 Gere **com reposição**, a partir das observações da amostra original ($x_n = \{x_1, \dots, x_n\}$), uma amostra **bootstrap** $x_n^* = \{x_1^*, \dots, x_n^*\}$ e calcule T_n^* guardando os resultados.
- 2 Repita o passo anterior B vezes.
- 3 Estime o erro-padrão por:

$$\widehat{se}(T_n)_{boot} = \sqrt{\frac{1}{B-1} \sum_{b=1}^B (t_b^* - \bar{t}^*)^2},$$

Bootstrap

- 1 Gere **com reposição**, a partir das observações da amostra original ($x_n = \{x_1, \dots, x_n\}$), uma amostra **bootstrap** $x_n^* = \{x_1^*, \dots, x_n^*\}$ e calcule T_n^* guardando os resultados.
- 2 Repita o passo anterior B vezes.
- 3 Estime o erro-padrão por:

$$\widehat{se}(T_n)_{boot} = \sqrt{\frac{1}{B-1} \sum_{b=1}^B (t_b^* - \bar{t}^*)^2},$$

em que t_b^* é a estimativa obtida por T_n^* sob a b -ésima amostra **bootstrap** e $\bar{t}^* = 1/B \sum_{b=1}^B t_b^*$.

Bootstrap

Dicas:

Bootstrap

Dicas:

- Para a estimativa do erro padrão de um estimador T_n , considerar $B \geq 250$.

Bootstrap

Dicas:

- Para a estimativa do erro padrão de um estimador T_n , considerar $B \geq 250$.
- Em geral, consideramos T_n como sendo o estimador de máxima verossimilhança para θ . Em muitas situações, T_n não tem forma fechada. Sendo assim, as estimativas de T_n deverão ser obtidas numericamente por métodos iterativos.

Bootstrap

Importante: No uso de métodos iterativos para obtenção da estimativa de máxima verossimilhança de θ , deve-se prestar atenção à convergência do algoritmo utilizado para maximização da função de verossimilhança. Em caso de não convergência, a amostra deverá ser descartada e substituída por uma nova amostra **bootstrap** em que haja convergência. Sendo assim, precisamos que nas B amostras **bootstrap** tenha-se convergência.

Bootstrap

Exercício: Consideremos novamente $X \sim \text{Weibull}(\alpha, \beta)$, em que

$$F_X(x) = 1 - \exp\{-(x/\beta)^\alpha\},$$

com $x > 0$ e $\alpha, \beta \in (0, \infty)$. Obtenha os estimadores de máxima verossimilhança de α e β . Com os estimadores de máxima verossimilhança para os parâmetros, calcule o erro padrão de $\hat{\alpha}$ e $\hat{\beta}$ usando **bootstrap** para uma amostra de tamanho 500 com distribuição Weibull ($\alpha = 2.3, \beta = 1.7$).

Bootstrap

Exercício: Consideremos novamente $X \sim \text{Weibull}(\alpha, \beta)$, em que

$$F_X(x) = 1 - \exp\{-(x/\beta)^\alpha\},$$

com $x > 0$ e $\alpha, \beta \in (0, \infty)$. Obtenha os estimadores de máxima verossimilhança de α e β . Com os estimadores de máxima verossimilhança para os parâmetros, calcule o erro padrão de $\hat{\alpha}$ e $\hat{\beta}$ usando **bootstrap** para uma amostra de tamanho 500 com distribuição Weibull ($\alpha = 2.3, \beta = 1.7$).

Exercício: Utilize simulações de Monte Carlo para estudar o comportamento dos estimadores $\widehat{se}(\hat{\alpha})_{boot}$ e $\widehat{se}(\hat{\beta})_{boot}$ em diferentes tamanhos de amostra ($n = 10, 60, 100, 500$ e 1000).

Bootstrap

Uma outra situação em que o uso de **bootstrap** é frequente é na correção de viés de um estimador. Sendo assim, seja T_n um estimador viesado para θ . Então,

Bootstrap

Uma outra situação em que o uso de **bootstrap** é frequente é na correção de viés de um estimador. Sendo assim, seja T_n um estimador viesado para θ . Então,

$$\text{bias}(T_n) = \text{E}(T_n) - \theta.$$

Bootstrap

Uma outra situação em que o uso de **bootstrap** é frequente é na correção de viés de um estimador. Sendo assim, seja T_n um estimador viesado para θ . Então,

$$\text{bias}(T_n) = \text{E}(T_n) - \theta.$$

Logo, em uma situação ideal teremos:

$$T_n^{ideal} = T_n - \text{bias}(T_n).$$

Bootstrap

Uma outra situação em que o uso de **bootstrap** é frequente é na correção de viés de um estimador. Sendo assim, seja T_n um estimador viesado para θ . Então,

$$\text{bias}(T_n) = \text{E}(T_n) - \theta.$$

Logo, em uma situação ideal teremos:

$$T_n^{ideal} = T_n - \text{bias}(T_n).$$

Porém, note que T_n^{ideal} não é um estimador, uma vez que T_n^{ideal} depende de θ .

Bootstrap

Então, teremos que estimar $\text{bias}(T_n)$, ou seja, queremos $\widehat{\text{bias}}(T_n)$.

Bootstrap

Então, teremos que estimar $\text{bias}(T_n)$, ou seja, queremos $\widehat{\text{bias}}(T_n)$. Assim temos que

$$T_n^{\text{quase ideal}} = T_n - \widehat{\text{bias}}(T_n).$$

Agora sim, temos que $T_n^{\text{quase ideal}}$ é um estimador corrigido por viés de θ .

Bootstrap

Então, teremos que estimar $\text{bias}(T_n)$, ou seja, queremos $\widehat{\text{bias}}(T_n)$. Assim temos que

$$T_n^{\text{quase ideal}} = T_n - \widehat{\text{bias}}(T_n).$$

Agora sim, temos que $T_n^{\text{quase ideal}}$ é um estimador corrigido por viés de θ .

Queremos construir algo semelhante à $T_n^{\text{quase ideal}}$.

Bootstrap

Daí, temos que

$$\begin{aligned} T_n^{\text{corrigido}} &= \widehat{T_n} - \text{bias}(\widehat{T_n}) = T_n - (\widehat{\text{E}(T_n)} - T_n) \\ &= 2T_n - \widehat{\text{E}(T_n)}. \end{aligned}$$

Bootstrap

Daí, temos que

$$\begin{aligned} T_n^{\text{corrigido}} &= T_n - \widehat{\text{bias}}(T_n) = T_n - (\widehat{\text{E}(T_n)} - T_n) \\ &= 2T_n - \widehat{\text{E}(T_n)}. \end{aligned}$$

Obtemos $\widehat{\text{E}(T_n)}$ por **bootstrap**. Estimaremos $\widehat{\text{E}(T_n)}$ por:

Bootstrap

Daí, temos que

$$\begin{aligned} T_n^{\text{corrigido}} &= T_n - \widehat{\text{bias}}(T_n) = T_n - (\widehat{\text{E}(T_n)} - T_n) \\ &= 2T_n - \widehat{\text{E}(T_n)}. \end{aligned}$$

Obtemos $\widehat{\text{E}(T_n)}$ por **bootstrap**. Estimaremos $\widehat{\text{E}(T_n)}$ por:

$$\widehat{\text{E}(T_n)} = \frac{1}{B} \sum_{i=1}^B T_{n,i}^*,$$

em que $T_{n,i}^*$ significa o estimador T_n na i -ésima amostra **bootstrap**.

Bootstrap

Atenção, isso é muito importante

O estimador $\widehat{E(T_n)}$ apresentado anteriormente poderá ser utilizado na correção de viés do estimador T_n . Porém, $\widehat{E(T_n)}$ não é um bom estimador de θ . Como mencionado anteriormente, o **bootstrap** fornece boas aproximações para a **forma** e **amplitude** da distribuição de T_n mas **não** para a sua **locação**. Sendo assim, jamais tente estimar T_n pela média das estimativas de T_n^* .

Bootstrap

Atenção, isso é muito importante

O estimador $\widehat{E(T_n)}$ apresentado anteriormente poderá ser utilizado na correção de viés do estimador T_n . Porém, $\widehat{E(T_n)}$ não é um bom estimador de θ . Como mencionado anteriormente, o **bootstrap** fornece boas aproximações para a **forma** e **amplitude** da distribuição de T_n mas **não** para a sua **locação**. Sendo assim, jamais tente estimar T_n pela média das estimativas de T_n^* .

Assim, o estimador $T_n^{corrigido}$ para θ corrigido por viés via **bootstrap** é dado por:

Bootstrap

Atenção, isso é muito importante

O estimador $\widehat{E(T_n)}$ apresentado anteriormente poderá ser utilizado na correção de viés do estimador T_n . Porém, $\widehat{E(T_n)}$ não é um bom estimador de θ . Como mencionado anteriormente, o **bootstrap** fornece boas aproximações para a **forma** e **amplitude** da distribuição de T_n mas **não** para a sua **locação**. Sendo assim, jamais tente estimar T_n pela média das estimativas de T_n^* .

Assim, o estimador $T_n^{corrigido}$ para θ corrigido por viés via **bootstrap** é dado por:

$$T_n^{corrigido} = 2T_n - \overline{T}_n^*.$$

Bootstrap

Bootstrap Duplo para correção de viés:

Nota: Podemos ampliar o **bootstrap** utilizando outros níveis de reamostragem para o melhoramento das estimativas que seriam obtidas considerando apenas um nível de **bootstrap**. Aqui discutiremos o uso de outro nível de reamostragem para o melhoramento da correção do viés de um estimador T_n por meio de **bootstrap duplo**.

Bootstrap

Bootstrap Duplo para correção de viés:

Nota: Podemos ampliar o **bootstrap** utilizando outros níveis de reamostragem para o melhoramento das estimativas que seriam obtidas considerando apenas um nível de **bootstrap**. Aqui discutiremos o uso de outro nível de reamostragem para o melhoramento da correção do viés de um estimador T_n por meio de **bootstrap duplo**.

Lembre-se que:

$$T_n^{\text{corrigido}} = 2T_n - \widehat{\text{E}}(T_n).$$

Bootstrap

Observe que $\widehat{E}(T_n)$ é uma estimativa que obtemos via **bootstrap** por meio da média das estimativas T_n calculadas nas pseudo-amostras (amostras **bootstrap**). Por se tratar de uma estimativa que muito provavelmente venha conter algum viés, poderemos então corrigir essa estimativa por meio de **bootstrap**, o que fará necessário considerar o outro nível de reamostragem.

Nota: A ideia de que $\widehat{E}(T_n)$ contém algum viés é razoável e se deve ao fato de que o **bootstrap** é um bom estimador para a forma e amplitude da distribuição de T_n , não sendo um bom estimador da locação da distribuição de T_n .

Bootstrap

Note que melhorar mais ainda a correção de $T_n^{\text{corrigido}}$ equivale à corrigir a estimativa de $\widehat{E(T_n)}$. Sendo assim, seja $\widehat{E(T_n)}^{\text{corrigido}} = \widehat{\delta}$ o estimador corrigido de $E(T_n)$ (usaremos **bootstrap** para isso).

Bootstrap

Note que melhorar mais ainda a correção de $T_n^{\text{corrigido}}$ equivale à corrigir a estimativa de $\widehat{E(T_n)}$. Sendo assim, seja $\widehat{E(T_n)}^{\text{corrigido}} = \widehat{\delta}$ o estimador corrigido de $E(T_n)$ (usaremos **bootstrap** para isso). Então,

$$\widehat{\delta} = \frac{1}{B} \sum_{i=1}^B T_{n,i}^* - \left\{ E \left(\frac{1}{B} \sum_{i=1}^B T_{n,i}^* \right) - \frac{1}{B} \sum_{i=1}^B T_{n,i}^* \right\}$$

Bootstrap

Note que melhorar mais ainda a correção de $T_n^{\text{corrigido}}$ equivale à corrigir a estimativa de $\widehat{E(T_n)}$. Sendo assim, seja $\widehat{E(T_n)}^{\text{corrigido}} = \widehat{\delta}$ o estimador corrigido de $E(T_n)$ (usaremos **bootstrap** para isso). Então,

$$\begin{aligned}\widehat{\delta} &= \frac{1}{B} \sum_{i=1}^B T_{n,i}^* - \left\{ E\left(\frac{1}{B} \sum_{i=1}^B T_{n,i}^*\right) - \frac{1}{B} \sum_{i=1}^B T_{n,i}^* \right\} \\ &= \frac{2}{B} \sum_{i=1}^B T_{n,i}^* - \frac{1}{B} \sum_{i=1}^B E(T_{n,i}^*).\end{aligned}$$

Bootstrap

Note que não conhecemos $E(T_{n,i}^*)$. Dessa forma, estimamos $E(T_{n,i}^*)$ por **bootstrap**, assim como fizemos para obter $\widehat{E(T_n)}$.

Bootstrap

Note que não conhecemos $E(T_{n,i}^*)$. Dessa forma, estimamos $E(T_{n,i}^*)$ por **bootstrap**, assim como fizemos para obter $\widehat{E(T_n)}$. Daí,

$$\hat{\delta} = \frac{2}{B} \sum_{i=1}^B T_{n,i}^* - \frac{1}{BM} \sum_{i=1}^B \sum_{j=1}^M T_{n,i,j}^{**}.$$

em que $T_{n,i,j}^{**}$ é a estatística T_n calculada na pseudo-amostra j do segundo nível de **bootstrap** obtida da pseudo-amostra i do primeiro nível de **bootstrap**.

Nota: Assim como anteriormente, B e M são números inteiros de pseudo-amostras no primeiro e segundo nível de **bootstrap**, respectivamente.

Bootstrap

Nota: Não há uma escolha ideal para os valores de B e M . Como sugestão, $B = 500$ e $M = 250$ é uma escolha razoável.

Portanto, o estimador T_n de θ com uma correção melhorada via **bootstrap duplo** (denotaremos por T_n^{duplo}) é obtido por

$$T_n^{duplo} = 2T_n - \widehat{\delta}.$$

Bootstrap

Exercício: Nadarajah e Haghghi em 2011 propuseram a distribuição distribuição de probabilidade Nadarajah-Haghghi (NH) em que se $X \sim NH(\alpha, \lambda)$, então

$$F_X(x) = 1 - \exp\{1 - (1 + \lambda x)^\alpha\}, \text{ com } x > 0,$$

em que $\lambda > 0$ e $\alpha > 0$. Por meio de simulações de Monte-Carlo, obtenha um estudo dos estimadores de máxima verossimilhança de α e λ para diferentes tamanhos de amostra $n = 20, 60, 100, 500, 1000$ e 10000 . Obtenha a média das estimativas de máxima verossimilhança e a média das estimativas corrigidas por **bootstrap**. Além disso, obtenha a média das estimativa do erro-padrão também estimado por **bootstrap**.

Bootstrap

Observação: O esquema duplo de **bootstrap** poderá ser aplicado não apenas na correção de viés de um estimador. Por exemplo, é possível utilizar **bootstrap duplo** para obtenção de intervalos aleatórios para um parâmetro desconhecido.

Exercício: Refaça o exercício anterior obtendo as estimativas de máxima verossimilhanças dos parâmetros da distribuição NH corrigida via **bootstrap duplo**.

Bootstrap

Um intervalo de confiança (IC) é uma estimativa intervalar para um parâmetro de interesse de uma população. Em vez de estimar o parâmetro por um único valor (estimativa pontual), o intervalo de confiança fornece um conjunto de estimativas possíveis para esse parâmetro de interesse através de um intervalo aleatório. Estimativas intervalares são realizadas sob um nível de confiança $1 - \alpha$, com $\alpha \in (0, 1)$, em que α é o nível de significância adotado e fixado pelo pesquisador. Um intervalo I_γ (intervalo de nível γ) para o parâmetro θ é tal que

$$\Pr\{I_\gamma \text{ conter } \theta\} = \gamma = 1 - \alpha. \quad (3)$$

Bootstrap

Uma abordagem frequentemente utilizada na construção de intervalos de confiança paramétricos é considerar um estimador $\hat{\theta}$ para um parâmetro θ da população, em que $\hat{\theta}$ é usualmente um estimador de máxima verossimilhança de θ . Queremos encontrar um intervalo que contenha θ com $100\gamma\%$ de confiança. Seja T_n um estimador do escalar θ baseado em n observações e t sua estimativa. Por simplicidade, suponhamos que T_n seja uma variável aleatória contínua. Denotando-se o p -ésimo quantil da distribuição da variável aleatória $T_n - \theta$ por a_p , temos que

$$\Pr \left\{ T_n - \theta \leq a_{\frac{\alpha}{2}} \right\} = \frac{\alpha}{2} = \Pr \left\{ T_n - \theta \geq a_{1-\frac{\alpha}{2}} \right\}. \quad (4)$$

Bootstrap

Como a quantidade $Q = T_n - \theta$ é inversível em θ e T_n depende apenas da amostra, podemos construir o intervalo de confiança para θ reescrevendo os eventos em (4), ou seja, podemos reescrever os eventos $T_n - \theta \leq a_{\frac{\alpha}{2}}$ e $T_n - \theta \geq a_{1-\frac{\alpha}{2}}$ como $\theta > T_n - a_{\frac{\alpha}{2}}$ e $\theta < T_n - a_{1-\frac{\alpha}{2}}$, respectivamente. Assim, o intervalo de confiança de nível γ é dado pelos limites

$$\ell_{\alpha/2} = t - a_{1-\frac{\alpha}{2}}, \quad \ell_{1-\alpha/2} = t - a_{\frac{\alpha}{2}}. \quad (5)$$

Bootstrap

Em um grande número de aplicações práticas a distribuição de $T_n - \theta$ é desconhecida, o que impossibilita calcularmos a_p . Este fato leva os pesquisadores a considerar vários métodos alternativos aproximados para construção de intervalos de confiança.

A maioria das metodologias propõe formas de estimar os quantis verdadeiros da distribuição desconhecida da variável aleatória $T_n - \theta$. A inferência estatística se preocupa em estabelecer metodologias para realização de testes hipóteses e construção de métodos para realizar estimativas pontuais e intervalares.

Bootstrap

Essas metodologias normalmente fazem uso de rigor matemático e se apoiam em suposições que precisam ser perfeitamente atendidas. Em se tratando de estimação intervalar, métodos baseados em bootstrap são amplamente utilizados. Entre tais metodologias, podemos citar os intervalos de confiança normais aproximados, o método bootstrap percentil e o bootstrap- t .

Bootstrap

Intervalos Normais Aproximados

Um enfoque usual é utilizar a distribuição $\mathcal{N}(0, v)$ como uma aproximação assintótica da distribuição da variável aleatória $T_n - \theta$. Aqui suponha que T_n é um estimador assintoticamente gaussiano de θ . Um caso comum é quando T_n é o estimador de máxima verossimilhança de θ . Essa aproximação fornece os limites de confiança aproximados

$$\ell_{\alpha/2}, \ell_{1-\alpha/2} = t \mp v^{1/2} z_{1-\alpha/2}, \quad (6)$$

em que $z_{1-\alpha/2} = \Phi^{-1}(1 - \alpha/2)$, Φ^{-1} sendo a função quantílica da distribuição normal padrão.

Bootstrap

Se $\hat{\theta}$ é um estimador assintoticamente normal de θ , a variância aproximada v pode ser obtida diretamente da função log-verossimilhança $\ell(\theta)$. Caso não haja parâmetros de incômodo, podemos utilizar o inverso da informação de Fisher observada, $v = -1/\ddot{\ell}(\hat{\theta})$ ou o inverso da informação de Fisher esperada dado por $v = 1/i(\hat{\theta})$, em que $i(\theta) = (-\ddot{\ell}(\theta)) = \text{var}(\dot{\ell}(\theta))$ e

$$\dot{\ell}(\theta) = \frac{\partial \ell(\theta)}{\partial \theta} \quad \text{e} \quad \ddot{\ell} = \frac{\partial^2 \ell(\theta)}{\partial \theta \partial \theta'}$$

A equação (6) é a forma padrão para os limites de confiança construídos pela aproximação assintótica da distribuição da variável aleatória $T_n - \theta$ pela distribuição normal.

Bootstrap

Intervalo Bootstrap Studentizado

Seja $\tilde{\mathbf{X}} = \{\tilde{X}_1, \tilde{X}_2, \dots, \tilde{X}_n\}$ uma amostra aleatória de tamanho n de variáveis aleatórias supostamente independentes e identicamente distribuídas com distribuição F , em que F é desconhecida. Seja também $\tilde{\mathbf{X}}_n^*$ uma nova amostra obtida a partir da distribuição empírica, i.e., de (\hat{F}_n) . Considere T_n^* como sendo a estatística T_n com base na amostra $\tilde{\mathbf{X}}_n^*$ e t^* sua estimativa. Usando a forma geral para intervalos de confiança dada pela equação (5), podemos estimar os quantis $a_{\alpha/2}$ e $a_{1-\alpha/2}$, que correspondem aos quantis de $T_n^* - t$.

Bootstrap

No bootstrap studentizado assumimos a forma de uma aproximação normal para limites de confiança, mas substituímos a aproximação $\mathcal{N}(0, 1)$ por $z = (T_n - \theta)/V^{1/2}$ por uma aproximação bootstrap. Aqui não conhecemos v pois não conhecemos F , em que $V = \text{var}(T_n|F)$. Cada amostra simulada é utilizada para calcular t^* , uma estimativa consistente de V (v^*) e a versão bootstrap de z dada por $z^* = (t^* - t)/v^{*1/2}$. São calculados B (número de amostras bootstrap) valores z^* , que são posteriormente ordenados.

Bootstrap

O quantil p da distribuição de z é estimado por $z_{((B+1)p)}^*$, i.e, o valor na posição $(B + 1)p$ dos valores ordenados de $z_1^*, z_2^*, \dots, z_B^*$. Em seguida, os limites de confiança dados em (6) são

$$\ell_{\alpha/2} = t - v^{1/2} z_{((B+1)(1-\alpha/2))}^*, \quad \ell_{1-\alpha/2} = t - v^{1/2} z_{((B+1)\alpha/2)}^*, \quad (7)$$

sendo estes os limites de confiança do bootstrap studentizado.

Bootstrap

O quantil p da distribuição de z é estimado por $z_{((B+1)p)}^*$, i.e, o valor na posição $(B + 1)p$ dos valores ordenados de $z_1^*, z_2^*, \dots, z_B^*$. Em seguida, os limites de confiança dados em (6) são

$$\ell_{\alpha/2} = t - v^{1/2} z_{((B+1)(1-\alpha/2))}^*, \quad \ell_{1-\alpha/2} = t - v^{1/2} z_{((B+1)\alpha/2)}^*, \quad (7)$$

sendo estes os limites de confiança do bootstrap studentizado.

Nota: e $(B + 1)\alpha/2$ não é inteiro devemos considerar $q = [(B + 1)\alpha/2]$, em que $[\cdot]$ é a função maior inteiro. Assim, os quantis de interesse são dados pelo $(B + 1 - q)$ -ésimo e q -ésimo maior valor inteiro de $z_1^*, z_2^*, \dots, z_B^*$.

Bootstrap

Exercício: Realize uma simulação de Monte Carlo comparando as estimativas intervalares por intervalos normais aproximados e bootstrap studentizado (bootstrap- t) para o parâmetro μ de uma distribuição normal. Considere diferentes tamanhos de amostras $n = 10, 50, 100, 500$ e 1000 . Avalie a cobertura dos intervalos, isto é, obtenha o percentual de intervalos de confianças que contem o parâmetro μ . Se produzimos boas estimativas intervalares, a cobertura deverá ser próxima ao nível de confiança adotado. Discuta os resultados. **Dica:** considere $\mu = 0$ e $\sigma^2 = 1$.

Bootstrap

Bootstrap Percentil

Davison e Hinkley (1997) afirma que existe alguma transformação de T_n , $U = h(T_n)$, tal que U possui uma distribuição simétrica. Suponhamos que sabemos calcular o intervalo de confiança de nível $1 - \alpha$ para $\phi = h(\theta)$. Segundo Davison e Hinkley (1997) podemos utilizar bootstrap para obter uma aproximação da distribuição de $T_n - \theta$ utilizando a distribuição de $T_n^* - t$. Dessa forma, estimamos o p -ésimo quantil de $T_n - \theta$ pelo $(B + 1)p$ -ésimo valor ordenado de $t^* - t$, ou seja, o p -ésimo quantil de $T_n - \theta$ é estimado por $t_{((B+1)p)}^* - t$. Analogamente, o p -ésimo quantil de $h(T_n) - h(\theta) = U - \phi$ poderá ser estimado pelo $(B + 1)p$ -ésimo valor ordenado de $h(T_n^*) - h(t) = u^* - u$. Seja b_p o p -ésimo quantil de $U - \phi$. Como U tem distribuição simétrica, então $U - \phi$ também tem distribuição simétrica, logo é verdade que $b_{\frac{\alpha}{2}} = -b_{1-\frac{\alpha}{2}}$.

Bootstrap

Utilizando a forma geral para intervalos de confiança dada por (5) e a simetria de $U - \phi$, temos que $h(\ell_{\alpha/2}) = u + b_{\alpha/2}$ e $h(\ell_{1-\alpha/2}) = u + b_{1-\alpha/2}$. Como $b_{\alpha/2}$ e $b_{1-\alpha/2}$ são quantis da distribuição de $U - \phi$ e sabemos calcular os quantis dessa distribuição, temos que os limites inferior e superior de confiança são dados por $u + (u^*_{((B+1)\alpha/2)} - u)$ e $u + (u^*_{((B+1)(1-\alpha/2))} - u)$, respectivamente, implicando os limites

$$u^*_{((B+1)\alpha/2)}, \quad u^*_{((B+1)(1-\alpha/2))},$$

cuja transformação para θ é

$$t^*_{(B+1)\alpha/2}, \quad t^*_{(B+1)(1-\alpha/2)}. \tag{8}$$

Bootstrap

Nota: Observe que não precisamos conhecer a transformação h . O intervalo de nível $1 - \alpha$ para o parâmetro θ não envolve h e pode ser calculado sem o conhecimento desta transformação. O intervalo (8) é conhecido como intervalo bootstrap percentil. Segundo Davison e Hinkley (1997) o método percentil poderá ser aplicado a qualquer estatística.

Bootstrap

Nota: Observe que não precisamos conhecer a transformação h . O intervalo de nível $1 - \alpha$ para o parâmetro θ não envolve h e pode ser calculado sem o conhecimento desta transformação. O intervalo (8) é conhecido como intervalo bootstrap percentil. Segundo Davison e Hinkley (1997) o método percentil poderá ser aplicado a qualquer estatística.

Exercício: Refaça o exercício anterior também considerando intervalos de confianças obtidos por meio de bootstrap percentil.

Tópicos especiais em simulação

Bootstrap de Wu (1986)

Tópicos especiais em simulação

Bootstrap de Wu (1986)

Citação da referência: WU, C. F. J. **Jackknife, bootstrap and other resampling methods in regression analysis.** Annals of Statistics, v. 14, p. 1261-1295, 1986.

Tópicos especiais em simulação

Bootstrap de Wu (1986)

Citação da referência: WU, C. F. J. **Jackknife, bootstrap and other resampling methods in regression analysis.** Annals of Statistics, v. 14, p. 1261-1295, 1986.

O uso de bootstrap para o cálculo de estimativas intervalares em modelos de regressão com heteroscedasticidade de forma desconhecida a partir de uma ponderação dos resíduos foi proposto por Wu (1986). Porém, precisaremos introduzir essa nova classe de modelos de regressão.

Tópicos especiais em simulação

Modelos Lineares de Regressão com Heteroscedasticidade de Forma Desconhecida

Os parâmetros dos modelos lineares de regressão são tipicamente estimados pelo Método dos Mínimos Quadrados ou Mínimos Quadrados Ordinários (MQO). MQO é o método de estimação mais amplamente utilizado em econometria. Essa metodologia consiste em minimizar a soma das perdas quadráticas e não requer suposição distribucional sobre eles. Observe que em nenhum momento será necessário conhecer a distribuição de y . Consideremos o modelo linear de regressão, dado por

$$y = X\beta + \varepsilon,$$

Tópicos especiais em simulação

em que y é um vetor $n \times 1$ de observações de interesse (regressando), X é uma matriz fixa e conhecida de variáveis independentes de dimensão $n \times p$ com posto coluna completo, ou seja, $\text{posto}(X) = p < n$, $\beta = (\beta_1, \dots, \beta_p)'$ é um vetor de dimensão $p \times 1$ de parâmetros lineares e $\varepsilon = (\varepsilon_1, \dots, \varepsilon_n)'$ é um vetor $n \times 1$ de erros aleatórios.

Tópicos especiais em simulação

em que y é um vetor $n \times 1$ de observações de interesse (regressando), X é uma matriz fixa e conhecida de variáveis independentes de dimensão $n \times p$ com posto coluna completo, ou seja, $\text{posto}(X) = p < n$, $\beta = (\beta_1, \dots, \beta_p)'$ é um vetor de dimensão $p \times 1$ de parâmetros lineares e $\varepsilon = (\varepsilon_1, \dots, \varepsilon_n)'$ é um vetor $n \times 1$ de erros aleatórios.

A solução de mínimos quadrados é alcançada minimizando $\varepsilon^2 = \varepsilon' \varepsilon = \sum_{i=1}^n \varepsilon_i^2$. Ou seja, $\widehat{\beta} = \arg \min_{\beta \in \mathbb{R}^p} \sum_{i=1}^n \varepsilon_i^2$.

Tópicos especiais em simulação

A estimação do vetor de parâmetros β se dará da seguinte forma:

Tópicos especiais em simulação

A estimação do vetor de parâmetros β se dará da seguinte forma:

$$\hat{\beta} = (X'X)^{-1}X'y. \quad (9)$$

Tópicos especiais em simulação

A estimação do vetor de parâmetros β se dará da seguinte forma:

$$\hat{\beta} = (X'X)^{-1}X'y. \quad (9)$$

Em se tratando de modelos lineares de regressão, as suposições que tipicamente são feitas são:

S1: O modelo $y = X\beta + \varepsilon$ é, de fato, o modelo verdadeiro;

Tópicos especiais em simulação

A estimação do vetor de parâmetros β se dará da seguinte forma:

$$\hat{\beta} = (X'X)^{-1}X'y. \quad (9)$$

Em se tratando de modelos lineares de regressão, as suposições que tipicamente são feitas são:

S1: O modelo $y = X\beta + \varepsilon$ é, de fato, o modelo verdadeiro;

S2: $\mathbb{E}(\varepsilon_i) = 0, i = 1, \dots, n;$

Tópicos especiais em simulação

A estimação do vetor de parâmetros β se dará da seguinte forma:

$$\hat{\beta} = (X'X)^{-1}X'y. \quad (9)$$

Em se tratando de modelos lineares de regressão, as suposições que tipicamente são feitas são:

- S1:** O modelo $y = X\beta + \varepsilon$ é, de fato, o modelo verdadeiro;
- S2:** $\mathbb{E}(\varepsilon_i) = 0, i = 1, \dots, n;$
- S3:** $\mathbb{E}(\varepsilon_i^2) = \text{var}(\varepsilon_i) = \sigma_i^2, i = 1, \dots, n;$

Tópicos especiais em simulação

S3': $\text{var}(\varepsilon_i) = \sigma^2, \ i = 1, \dots, n \ (0 < \sigma^2 < \infty);$

Tópicos especiais em simulação

S3': $\text{var}(\varepsilon_i) = \sigma^2, i = 1, \dots, n$ ($0 < \sigma^2 < \infty$);

S4: $\mathbb{E}(\varepsilon_i \varepsilon_s) = 0, \forall i \neq s;$

Tópicos especiais em simulação

S3': $\text{var}(\varepsilon_i) = \sigma^2, i = 1, \dots, n$ ($0 < \sigma^2 < \infty$);

S4: $\mathbb{E}(\varepsilon_i \varepsilon_s) = 0, \forall i \neq s$;

S5: $\lim_{n \rightarrow \infty} n^{-1}(X'X) = Q$, em que Q é uma matriz positiva-definida.

Sob [S1] e [S2], temos que o estimador $\hat{\beta} = (X'X)^{-1}X'y$ é não viesado para β , ou seja, em média iguala-se ao parâmetro verdadeiro. Sob essas suposições, temos que $\mathbb{E}(y) = X\beta$. Logo, $\mathbb{E}(\hat{\beta}) = (X'X)^{-1}X'\mathbb{E}(y) = (X'X)^{-1}X'X\beta = I_p\beta = \beta$, em que I_p é uma matriz identidade de dimensão $p \times p$. Tal estimador coincide com o estimador de máxima verossimilhança sob normalidade dos erros.

Tópicos especiais em simulação

Quando as suposições [S1], [S2], [S3] e [S4] são válidas, a matriz de covariâncias de ε é dada por

$$\Omega = \text{diag}\{\sigma_i^2, \dots, \sigma_n^2\}.$$

Essa matriz se reduz a $\Omega = \sigma^2 I_n$ quando $\sigma_i^2 = \sigma^2 > 0$ com $i = 1, \dots, n$ (vale [S3']), sendo I_n matriz identidade de dimensão $n \times n$. Esse resultado pode ser visto abaixo. Note que

Tópicos especiais em simulação

$$\mathbb{E}(\varepsilon\varepsilon') = \begin{bmatrix} \mathbb{E}(\varepsilon_1^2) & \mathbb{E}(\varepsilon_1\varepsilon_2) & \cdots & \mathbb{E}(\varepsilon_1\varepsilon_n) \\ \mathbb{E}(\varepsilon_2\varepsilon_1) & \mathbb{E}(\varepsilon_2^2) & \cdots & \mathbb{E}(\varepsilon_2\varepsilon_n) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{E}(\varepsilon_n\varepsilon_1) & \mathbb{E}(\varepsilon_n\varepsilon_2) & \cdots & \mathbb{E}(\varepsilon_n^2) \end{bmatrix}.$$

Usando [S1], [S2], [S3'] e [S4], obtemos

$$\Omega = \mathbb{E}(\varepsilon\varepsilon') = \mathbb{E}(\varepsilon^2) = \sigma^2 I_n.$$

Tópicos especiais em simulação

O vetor de parâmetros β pode ser estimado usando o método de mínimos quadrados ordinários; equação (9). Sem que se faça suposição sobre a distribuição das respostas, minimiza-se a soma dos quadrados dos erros. É importante destacar também que mesmo quando a suposição [S3] é satisfeita o estimador de mínimos quadrados ordinários continua não-viesado. O estimador $\hat{\beta}$ possui as seguintes propriedades:

Tópicos especiais em simulação

- (1) Quando as suposições [S1] e [S2] são válidas, $\hat{\beta}$ é um estimador não-viesado para β . Ou seja, em média o estimador iguala-se ao parâmetro em estimação;

Tópicos especiais em simulação

- (1) Quando as suposições [S1] e [S2] são válidas, $\hat{\beta}$ é um estimador não-viesado para β . Ou seja, em média o estimador iguala-se ao parâmetro em estimação;
- (2) A estrutura de covariância de $\hat{\beta}$ é
$$\text{cov}(\hat{\beta}) = \Psi_{\hat{\beta}} = (X'X)^{-1}X'\Omega X(X'X)^{-1};$$

Tópicos especiais em simulação

- (1) Quando as suposições [S1] e [S2] são válidas, $\hat{\beta}$ é um estimador não-viesado para β . Ou seja, em média o estimador iguala-se ao parâmetro em estimação;
- (2) A estrutura de covariância de $\hat{\beta}$ é $\text{cov}(\hat{\beta}) = \Psi_{\hat{\beta}} = (X'X)^{-1}X'\Omega X(X'X)^{-1}$;
- (3) Quando as suposições [S1], [S2] e [S5] valem, o estimador $\hat{\beta}$ é consistente para β , ou seja, converge em probabilidade para o vetor de parâmetros;

Tópicos especiais em simulação

- (1) Quando as suposições [S1] e [S2] são válidas, $\hat{\beta}$ é um estimador não-viesado para β . Ou seja, em média o estimador iguala-se ao parâmetro em estimação;
- (2) A estrutura de covariância de $\hat{\beta}$ é $\text{cov}(\hat{\beta}) = \Psi_{\hat{\beta}} = (X'X)^{-1}X'\Omega X(X'X)^{-1}$;
- (3) Quando as suposições [S1], [S2] e [S5] valem, o estimador $\hat{\beta}$ é consistente para β , ou seja, converge em probabilidade para o vetor de parâmetros;
- (4) Sob as suposições [S1], [S2], [S3'] e [S4], vale o Teorema de Gauss-Markov, ou seja, $\hat{\beta}$ possui variância mínima na classe dos estimadores lineares e não-viesados de β ;

Tópicos especiais em simulação

- (1) Quando as suposições [S1] e [S2] são válidas, $\hat{\beta}$ é um estimador não-viesado para β . Ou seja, em média o estimador iguala-se ao parâmetro em estimação;
- (2) A estrutura de covariância de $\hat{\beta}$ é $\text{cov}(\hat{\beta}) = \Psi_{\hat{\beta}} = (X'X)^{-1}X'\Omega X(X'X)^{-1}$;
- (3) Quando as suposições [S1], [S2] e [S5] valem, o estimador $\hat{\beta}$ é consistente para β , ou seja, converge em probabilidade para o vetor de parâmetros;
- (4) Sob as suposições [S1], [S2], [S3'] e [S4], vale o Teorema de Gauss-Markov, ou seja, $\hat{\beta}$ possui variância mínima na classe dos estimadores lineares e não-viesados de β ;
- (5) Sob [S1], [S2] e [S3], o estimador $\hat{\beta}$ é assintoticamente gaussiano.

Tópicos especiais em simulação

Importante

É importante destacar que quando a suposição de homoscedasticidade é violada (i.e., os erros seguem padrão heteroscedástico) $\hat{\beta}$ deixa de ser eficiente, ou seja, não vale mais o Teorema de Gauss Markov. Contudo, o estimador continua não-viesado, consistente e assintoticamente gaussiano. Dessa forma, $\hat{\beta}$ mantém muitas das propriedades desejáveis de um bom estimador.

Tópicos especiais em simulação

Sob homoscedasticidade, ou seja, quando as variâncias dos erros são idênticas (suposição [S3']), temos que a estrutura de covariância de $\hat{\beta}$ é dada por $\text{cov}(\hat{\beta}) = \sigma^2(X'X)^{-1}$. Essa matriz pode ser facilmente estimada substituindo σ^2 pelo estimador $\hat{\sigma}^2 = \hat{\varepsilon}'\hat{\varepsilon}/(n - p)$, em que $\hat{\varepsilon}$ é o vetor de dimensão $n \times 1$ contendo os resíduos de mínimos quadrados.

Esses resíduos são dados por $\hat{\varepsilon} = \{I_n - X(X'X)^{-1}X'\}y = (I_n - H)y$. Os erros-padrão fornecidos pela maioria dos softwares estatísticos são as raízes quadradas dos elementos diagonais de $\hat{\sigma}^2(X'X)^{-1}$.

Tópicos especiais em simulação

A matriz $H = X(X'X)^{-1}X'$ é conhecida como **matriz chapéu**, pois $\hat{y} = Hy$. Os elementos da diagonal principal da matriz H assumem valores no intervalo $(0, 1)$ e somam p , i.e., $\text{tr}(H) = p$. Dessa forma, a média dos elementos diagonais de H é igual a p/n .

Tópicos especiais em simulação

A matriz $H = X(X'X)^{-1}X'$ é conhecida como **matriz chapéu**, pois $\hat{y} = Hy$. Os elementos da diagonal principal da matriz H assumem valores no intervalo $(0, 1)$ e somam p , i.e., $\text{tr}(H) = p$. Dessa forma, a média dos elementos diagonais de H é igual a p/n .

Importante

Os elementos diagonais da matriz H são denotados por h_1, \dots, h_n e são utilizados como medidas dos graus de alavancagem, i.e., o i -ésimo elemento diagonal de H (h_i) mede o grau de influência da i -ésima observação sobre o correspondente valor predito (\hat{y}_i).

Tópicos especiais em simulação

Regra bastante utilizada

Uma regra muito usada é concluir que a i -ésima observação é ponto de alavanca se $h_i > 2p/n$ ou $h_i > 3p/n$.

Detalhes sobre a regra acima poderá ser encontrada na referência abaixo:

JUDGE, G.; HILL, R.; GRIFFITHS, W. E.; LUTKEPOHL, H.; LEE, T. **Introduction to the theory and practice of econometrics.** Wiley, 1988.

Tópicos especiais em simulação

Em modelos de regressão heteroscedásticos quando se conhece a matriz Ω de covariâncias dos erros (o que quase nunca ocorre na prática), é possível transformar o modelo $y = \beta X + \varepsilon$ de forma que as suposições necessárias para a validade do Teorema de Gauss-Markov sejam satisfeitas. No contexto em que se conhece a matriz de covariâncias dos erros, o estimador de mínimos quadrados generalizados (EMQG) é dado por $\hat{\beta}_G = (X'\Omega^{-1}X)^{-1}X'\Omega^{-1}y$. Pode-se mostrar que $(\hat{\beta}_G) = \beta$ e

$$\text{cov}(\hat{\beta}_G) = \Psi_{\hat{\beta}_G} = (X'\Omega^{-1}X)^{-1}.$$

Quando a suposição [S3'] é válida, ou seja, sob homoscedasticidade, tem-se que $\hat{\beta}_G = \hat{\beta}$ e $\text{cov}(\hat{\beta}_G) = \text{cov}(\hat{\beta}) = \sigma^2(X'X)^{-1}$.

Tópicos especiais em simulação

Estimação do vetor β pelo método de mínimos quadrados generalizados é inviável, uma vez que os elementos de Ω (i.e., as variâncias dos n erros) são desconhecidos. Uma alternativa é postular um modelo para as variâncias dos erros e usar esse modelo para obter uma estimativa de Ω . Essa estimativa seria então utilizada no lugar da matriz verdadeira no estimador de mínimos quadrados generalizados. O estimador resultante é conhecido como estimador de mínimos quadrados generalizados viável (EMQGV) e é dado por

$$\widehat{\widehat{\beta}} = (X' \widehat{\Omega}^{-1} X)^{-1} X' \widehat{\Omega}^{-1} y.$$

Tópicos especiais em simulação

O Teorema de Gauss Markov, contudo, não vale para o EMQGV. De fato, esse estimador não é linear e não é possível mostrar, em geral, que ele é não-viesado. Adicionalmente, para se usar esse estimador é necessário postular um modelo para as n variâncias. Tipicamente, contudo, é mais difícil modelar variâncias do que efeitos médios.

Lembre-se sempre disso

Deve ser notado que mesmo não valendo o Teorema de Gauss Markov, ou seja, quando a suposição [S3'] não é válida (i.e., os erros seguem um padrão heteroscedástico), $\hat{\beta}$ permanece não-viesado, consistente e assintoticamente gaussiano.

Tópicos especiais em simulação

Dessa forma, podemos basear as estimativas pontuais nesse estimador. Para a obtenção de intervalos de confiança e realização de testes de hipóteses é necessário um estimador consistente da matriz de covariância de $\hat{\beta}$. Para tanto, deve-se usar um estimador $\hat{\Omega}$ de Ω tal que $X'\hat{\Omega}X$ seja consistente para $X'\Omega X$, ou seja, $\text{plim}((X'\Omega X)^{-1}(X'\hat{\Omega}X)) = I_p$, onde plim denota limite em probabilidade.

Tópicos especiais em simulação

White (1980) propôs um estimador da matriz de covariâncias $\hat{\Psi}_\beta$ que é consistente tanto sob homoscedasticidade quanto sob heteroscedasticidade de forma desconhecida. Ele percebeu que não seria preciso estimar a matriz de covariâncias dos erros de forma consistente; note que essa matriz possui n quantidades desconhecidas. Ele notou que basta estimar consistentemente $(X'\Omega X)$, que possui $p(p + 1)/2$ elementos desconhecidos independentemente do tamanho da amostra.

Tópicos especiais em simulação

O estimador de Halbert White é obtido substituindo-se o i -ésimo elemento da diagonal da matriz de covariâncias dos erros (σ_i^2) pelo i -ésimo resíduo MQO ao quadrado, ou seja,

$$\text{HC0} = (X'X)^{-1} X' \hat{\Omega}_0 X (X'X)^{-1},$$

em que $\hat{\Omega}_0 = \text{diag}\{\hat{\varepsilon}_1^2, \dots, \hat{\varepsilon}_n^2\}$.

Tópicos especiais em simulação

O estimador de White pode ser muito viesado em amostras de tamanho pequeno a moderado. O viés desse estimador tende a ser negativo, ou seja, ele é um estimador “otimista”, pois tipicamente subestima as variâncias verdadeiras; ver

CRIBARI-NETO, F.; ZARKOS, S. G. Bootstrap methods for heteroskedastic regression models: evidence on estimation and testing. *Econometric Reviews*, Taylor & Francis, v. 18, n. 2, p. 211-228, 1999.

CRIBARI-NETO, F.; ZARKOS, S. G. Heteroskedasticity-consistent covariance matrix estimation: white's estimator and the bootstrap. *Journal of Statistical Computation and Simulation*, Taylor & Francis, v. 68, n. 4, p. 391-411, 2001.

Tópicos especiais em simulação

Algumas desvantagens a respeito do estimador HC0 são:

- 1 O estimador subestima as variâncias verdadeiras;

Tópicos especiais em simulação

Algumas desvantagens a respeito do estimador HC0 são:

- 1 O estimador subestima as variâncias verdadeiras;
- 2 O viés do estimador é mais acentuado em situações em que os dados incluem pontos de alavanca;

Tópicos especiais em simulação

Algumas desvantagens a respeito do estimador HC0 são:

- 1 O estimador subestima as variâncias verdadeiras;
- 2 O viés do estimador é mais acentuado em situações em que os dados incluem pontos de alavanca;
- 3 Testes sobre os parâmetros que fazem uso desse estimador são tipicamente liberais, i.e., anticonservativos. Dessa forma, a hipótese nula tende a ser rejeitada acima do esperado.

Tópicos especiais em simulação

Algumas desvantagens a respeito do estimador HC0 são:

- 1 O estimador subestima as variâncias verdadeiras;
- 2 O viés do estimador é mais acentuado em situações em que os dados incluem pontos de alavanca;
- 3 Testes sobre os parâmetros que fazem uso desse estimador são tipicamente liberais, i.e., anticonservativos. Dessa forma, a hipótese nula tende a ser rejeitada acima do esperado.

Em busca de corrigir o viés do estimador HC0, uma sequência de estimadores HC0 corrigidos por viés foi obtida por (HC2, HC3, HC4 e HC5).

Tópicos especiais em simulação

Horn, Horn e Duncan (1975), MacKinnon e White (1985) construíram um novo estimador denominado HC2, que usa

$$\widehat{\Omega}_2 = \text{diag}\{\widehat{\varepsilon}_1^2/(1 - h_1), \dots, \widehat{\varepsilon}_n^2/(1 - h_n)\},$$

em que h_i denota o i -ésimo elemento diagonal da **matriz chapeu** $H = X(X'X)^{-1}X'$, $i = 1, \dots, n$. O estimador HC2 é não-viesado sob homoscedasticidade.

Tópicos especiais em simulação

Principais referências para o estimador HC2:

HORN, S. D.; HORN, R. A.; DUNCAN, D. B. Estimating heteroscedastic variances in linear models. *Journal of the American Statistical Association*, v. 70, p. 380-385, 1975.

MACKINNON, J. G.; WHITE, H. Some heteroskedasticity-consistent covariance matrix estimators with improved finite sample properties. *Journal of Econometrics*, v. 29, p. 305-325, 1985.

Tópicos especiais em simulação

O estimador HC3 foi proposto por Davidson e MacKinnon (1993). Ele é uma aproximação ao estimador jackknife, que é obtido a partir da remoção sequencial de cada observação da amostra. O estimador da matriz de covariâncias dos erros utilizado no estimador HC3 é

$$\widehat{\Omega}_3 = \text{diag}\{\widehat{\varepsilon}_1^2/(1 - h_1)^2, \dots, \widehat{\varepsilon}_n^2/(1 - h_n)^2\}.$$

Tópicos especiais em simulação

Principais referências para o estimador HC3:

DAVIDSON, R.; MACKINNON, J. G. Estimation and Inference in Econometrics. New York: Oxford University Press, 1993.

Tópicos especiais em simulação

Um estimador alternativo que também faz uso das medidas de alavancagem e inclui termos de ajustes para amostras finitas foi proposto por Cribari-Neto (2004). Tal estimador, denominado HC4, utiliza

$$\hat{\Omega}_4 = \text{diag}\{\hat{\varepsilon}_1^2/(1-h_1)^{\delta_1}, \dots, \hat{\varepsilon}_n^2/(1-h_n)^{\delta_n}\},$$

em que $\delta_i = \min\{4, h_i/\bar{h}\} = \min\{4, nh_i/p\}$. Note que

$$\bar{h} = n^{-1} \sum_{i=1}^n h_i = \text{tr}(H)/n = p/n.$$

Tópicos especiais em simulação

Principais referências para o estimador HC4:

Asymptotic inference under heteroskedasticity of unknown form.
Computational Statistics & Data Analysis, Elsevier, v. 45, n. 2,
p. 215-233, 2004.

Tópicos especiais em simulação

Um outro estimador foi proposto por Cribari-Neto, Souza e Vasconcellos (2007). Trata-se do estimador HC5, que usa

$$\widehat{\Omega}_5 = \text{diag}\{\widehat{\varepsilon}_1^2/\sqrt{(1-h_1)^{\alpha_1}}, \dots, \widehat{\varepsilon}_n^2/\sqrt{(1-h_n)^{\alpha_n}}\},$$

em que

$$\alpha_i = \min \left\{ \frac{h_i}{\bar{h}}, \max \left\{ \frac{h_i}{\bar{h}}, \frac{k h_{\max}}{\bar{h}} \right\} \right\}.$$

Aqui, k é uma constante cujo valor foi escolhido numericamente a partir de simulações-piloto. Os autores sugerem utilizar $k = 0.7$. Esse estimador leva em conta a alavancagem maximal em todos os n termos de descontos utilizados.

Tópicos especiais em simulação

Principais referências para o estimador HC5:

CRIBARI-NETO, F.; SOUZA, T. C.; VASCONCELLOS, K. L. Inference under heteroskedasticity and leveraged data. Communications in Statistics—Theory and Methods, Taylor & Francis, v. 36, n. 10, p. 1877-1888, 2007, Errata: v. 37, n. 20, p. 3329-3330, 2008.