

# **Métodos Computacionais**

**Departamento de Estatística e Matemática Aplicada**

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

2026-03-08

# Índice

<b>Prefácio</b>	<b>3</b>
<b>1 Introdução</b>	<b>4</b>
<b>2 Motivação</b>	<b>5</b>
2.1 Atividade: Problema do Aniversário (22 jogadores) . . . . .	5
2.2 Solução Analítica . . . . .	6
2.3 Solução Aproximada . . . . .	6
2.4 Comparação entre as soluções exata e aproximada . . . . .	6
2.5 Possíveis dúvidas . . . . .	6
<b>3 Números Uniformes</b>	<b>7</b>
3.1 Geração de sequências $U(0, 1)$ . . . . .	7
3.2 Geradores Congruenciais Lineares . . . . .	8
3.2.1 Exemplo . . . . .	8
3.2.2 Implementação em R . . . . .	8
3.3 Geradores Congruenciais Lineares Mistos . . . . .	9
3.3.1 Questão de estouro e aritmética modular . . . . .	10
3.3.2 Implementação em R (com segurança de overflow) . . . . .	10
3.4 Geradores Congruenciais Lineares Multiplicativos . . . . .	11
3.4.1 Características e restrições . . . . .	11
3.4.2 Definição de raiz primitiva . . . . .	11
3.4.3 Exemplo de implementação em R . . . . .	12
<b>4 Otimização Numérica</b>	<b>13</b>
<b>5 Métodos de Reamostragem</b>	<b>14</b>
<b>6 Métodos de Monte Carlo</b>	<b>15</b>
<b>7 Algoritmo EM</b>	<b>16</b>
<b>8 Métodos Adicionais</b>	<b>17</b>
<b>References</b>	<b>18</b>

# Prefácio

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

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

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

# 1 Introdução

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

## 2 Motivação

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

Para iniciar a discussão, propõe-se o seguinte problema:

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

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

### 2.1 Atividade: Problema do Aniversário (22 jogadores)

#### Objetivos

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

#### Hipóteses

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

#### Materiais

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

## **2.2 Solução Analítica**

## **2.3 Solução Aproximada**

## **2.4 Comparação entre as soluções exata e aproximada**

## **2.5 Possíveis dúvidas**

Algumas dúvidas podem surgir, entre elas:

-

## 3 Números Uniformes

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

### 3.1 Geração de sequências $U(0, 1)$

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

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

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

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

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

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

## 3.2 Geradores Congruenciais Lineares

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

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

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

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

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

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

### 3.2.1 Exemplo

Seja o gerador:

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

Escolhendo  $X_0 = 3$ :

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

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

e assim por diante.

A sequência  $R_i = X_i/16$  gerada terá período  $\ell = 16$ .

### 3.2.2 Implementação em R



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

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

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

```
resultado$R
```

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

### 3.3 Geradores Congruenciais Lineares Mistos

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

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

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

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

### 3.3.1 Questão de estouro e aritmética modular

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

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

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

### 3.3.2 Implementação em R (com segurança de overflow)

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

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

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

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

```
g1$R
```

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

## 3.4 Geradores Congruenciais Lineares Multiplicativos

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

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

### 3.4.1 Características e restrições

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

### 3.4.2 Definição de raiz primitiva

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

Matematicamente,  $a$  satisfaz:

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

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

### 3.4.3 Exemplo de implementação em R

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

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

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

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

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

```
g2$R
```

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

## 4 Otimização Numérica

## 5 Métodos de Reamostragem

## 6 Métodos de Monte Carlo

## 7 Algoritmo EM



## 8 Métodos Adicionais

## References

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