# Métodos Computacionais

Departamento de Estatística e Matemática Aplicada

Ronald Targino, Rafael Braz, Juvêncio Nobre e Manoel Santos-Neto 2025-09-21

# Índice

Prefácio								
1 Introdução								
2	Uma pequena introdução ao R							
	2.1 Introdução	6						
	Instalação	6						
	Windows							
	macOS							
	Linux	7						
	2.2 O que é o R	11						
	2.3 Números, Aritmética, Atribuição e Vetores	25						
	Exercícios							
	2.4 Matrizes	32						
	Exercícios	35						
	2.5 Operações com matrizes	36						
	Exercícios	40						
	2.6 Laços e repetições	41						
	2.7 Escrevendo funções no R	44						
	2.7.1 Média Áritmética							
	2.7.2 Mediana							
	Exercícios							
3	Motivação	47						
	Da teoria à simulação	47						
	Um atalho analítico útil	48						
	O papel da simulação	49						
	Atividade: Problema do Aniversário (22 jogadores)	50						
	Exercícios	51						
4	Números Uniformes	52						
	4.1 Geração de sequências $U(0,1)$	52						
	Gerador Congruencial Linear							
	Por que o CGL funciona?							
	Critérios para bons parâmetros							
	Visualização	55						

Referêr	cias	62
4.2	Uso de Números Aleatórios na Avaliação de Integrais	57
	Limitações	55

# 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).

Todo estudante da Universidade Federal do Ceará conhece a cena. É hora do almoço no Restaurante Universitário (RU). A fila se alonga pelo pátio, colegas conversam, alguns reclamam da espera, outros aproveitam o tempo para revisar o conteúdo da próxima prova. O tempo parece correr de forma diferente quando estamos na fila. Para alguns, são apenas alguns minutos. Para outros, parece uma eternidade.

Agora, pense um pouco. Quanto tempo, em média, um aluno passa esperando para se servir? Qual a chance de alguém que chega por volta das 12h30 esperar mais de vinte minutos? Por que em alguns dias a fila anda rápido e em outros parece não ter fim?

Essas perguntas podem parecer simples, mas abrem caminho para um universo fascinante. Elas revelam como a **Probabilidade e a Estatística** estão presentes em situações que vivemos todos os dias. A fila do RU não é apenas um detalhe da rotina estudantil. Ela é um retrato de como os fenômenos aleatórios acontecem ao nosso redor. Cada chegada de estudante, cada tempo de atendimento, cada variação de um dia para o outro forma um sistema dinâmico que pode ser estudado e compreendido.

É esse o convite deste livro. Explorar como traduzir a realidade em modelos, como usar simulações para observar padrões e como a Estatística pode ajudar a responder perguntas sobre o nosso cotidiano. Mais do que fórmulas, ela é uma maneira de olhar o mundo e encontrar nele sentido.

Aprender Estatística é aprender a lidar com a incerteza. É descobrir que até na fila do almoço existe conhecimento escondido, esperando para ser revelado.

# 2 Uma pequena introdução ao R

## 2.1 Introdução

O R é peça-chave em inúmeros trabalhos de pesquisa e análise de dados porque reúne, de forma prática, um conjunto amplo de técnicas estatísticas atuais, das mais básicas às mais sofisticadas, e facilita seu uso no dia a dia. Quem começa no R, porém, muitas vezes também está dando os primeiros passos em programação. Assim, além de aprender as ferramentas do R para seus objetivos, é preciso desenvolver a "cabeça" de programador. Essa fase inicial ajuda a explicar a fama de que o R é "difícil". Mesmo assim, com prática e uma boa orientação, ele se revela bem mais acessível do que parece.

### Instalação

Este guia mostra como baixar e instalar o R a partir do CRAN (Comprehensive R Archive Network), com orientações específicas para Windows, macOS e Linux. Ao final, você testará a instalação e configurará um espelho (mirror) brasileiro para baixar pacotes mais rápido.

#### i Nota

## O que é o CRAN?

É a rede oficial de servidores que distribui o R e seus pacotes. Você pode usar o endereço inteligente <a href="https://cloud.r-project.org">https://cloud.r-project.org</a> ou definir um espelho no Brasil (ex.: C3SL/UFPR).

Você pode simplesmente usar:

- Cloud CRAN (recomendado): https://cloud.r-project.org (redireciona para um espelho próximo).
- Brasil (ex.: UFPR/C3SL): https://cran-r.c3sl.ufpr.br

Mais adiante, mostraremos como fixar o mirror no R permanentemente.

#### Windows

- 1. Acesse a página **Download R for Windows**  $\rightarrow$  **base** e baixe o instalador.
- 2. Execute o instalador e avance com as opções padrão (recomendado para iniciantes).
- 3. (Opcional) Se pretende compilar pacotes a partir do código-fonte, instale o Rtools compatível com a sua versão do R.



Dica

Rtools: após instalar, reinicie o R/RStudio. Em geral, o Rtools adiciona as ferramentas ao PATH automaticamente.

#### macOS

- 1. Acesse R for macOS no CRAN e baixe o arquivo .pkg da versão atual.
- 2. Abra o .pkg e conclua a instalação.
- 3. (Opcional) Para compilar pacotes, instale também as Command Line Tools do Xcode:

```
xcode-select --install
```

#### Linux

#### Ubuntu/Debian

Opção rápida (repositório da distribuição):

```
sudo apt update
sudo apt install -y r-base
```

Para obter versões mais novas (repositório do CRAN), siga as instruções do CRAN para adicionar o repositório oficial e então:

```
sudo apt update
sudo apt install -y r-base r-base-dev
```

#### Verificar a instalação

No terminal/Prompt:

```
R --version R
```

#### Dentro do R:

#### version

```
x86_64-pc-linux-gnu
platform
arch
               x86_64
os
               linux-gnu
               x86_64, linux-gnu
system
status
               4
major
               5.1
minor
               2025
year
               06
month
day
               13
               88306
svn rev
language
               R
version.string R version 4.5.1 (2025-06-13)
nickname
               Great Square Root
```

Se o R abriu no console, a instalação está ok!

### Teste rápido de pacotes

No R:

```
#install.packages("tidyverse") # teste de instalação/espelho
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------
                                             ----- tidyverse 2.0.0 --
v dplyr 1.1.4
                  v readr 2.1.5
v forcats 1.0.0
                    v stringr
                               1.5.1
v ggplot2 4.0.0
                  v tibble
                               3.3.0
v lubridate 1.9.3
                    v tidyr
                               1.3.1
v purrr
           1.0.2
-- Conflicts ----- tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag() masks stats::lag()
i Use the conflicted package (<a href="http://conflicted.r-lib.org/">http://conflicted.r-lib.org/</a>) to force all conflicts to become
```

```
tibble(x = 1:5, y = x^2)
```

Se o pacote instalou e carregou sem erros, está tudo certo.

#### Fixar um mirror brasileiro do CRAN

Defina o espelho apenas nesta sessão:

```
options(repos = c(CRAN = "https://cran-r.c3sl.ufpr.br"))
install.packages("ggplot2")
```

Para tornar permanente, adicione a linha abaixo ao seu arquivo ~/.Rprofile:

```
options(repos = c(CRAN = "https://cloud.r-project.org"))
# ou, se preferir, o espelho da UFPR:
# options(repos = c(CRAN = "https://cran-r.c3sl.ufpr.br"))
```

#### Como editar o ~/.Rprofile

• Linux/macOS:

```
echo 'options(repos = c(CRAN = "https://cloud.r-project.org"))' >> ~/.Rprofile
```

• Windows: o ~ normalmente aponta para C:\\Users\\SEU\_USUARIO\\Documents.

Você pode criar/editar C:\\Users\\SEU\_USUARIO\\Documents\\.Rprofile com um editor de texto.

#### (Opcional) IDE recomendada: RStudio

Após instalar o R, instale o RStudio Desktop (Posit) para um ambiente de desenvolvimento mais amigável:

- Criação/edição de scripts
- Gerenciamento de projetos
- Visualização de plots e help integrados

#### Dicas e solução de problemas

- Permissões de administrador: em ambientes corporativos, pode ser necessário pedir para TI instalar o R.
- Firewall/Proxy: se a instalação de pacotes falhar, verifique configurações de proxy e tente trocar o mirror.
- Compatibilidade de versões: ao compilar pacotes, garanta que as ferramentas (Rtools no Windows; CLT/Xcode no macOS) correspondam à sua versão do R.
- Atualização do R: ao atualizar o R, alguns pacotes precisarão ser reinstalados; use install.packages() novamente.
- Testes mínimos:

#### sessionInfo()

```
R version 4.5.1 (2025-06-13)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 24.04.3 LTS
```

Matrix products: default

BLAS: /usr/lib/x86\_64-linux-gnu/openblas-pthread/libblas.so.3

LAPACK: /usr/lib/x86\_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.26.so; LAPACK version

#### locale:

```
[1] LC_CTYPE=pt_BR.UTF-8 LC_NUMERIC=C
```

[3] LC\_TIME=pt\_BR.UTF-8 LC\_COLLATE=pt\_BR.UTF-8
[5] LC\_MONETARY=pt\_BR.UTF-8 LC\_MESSAGES=pt\_BR.UTF-8

[7] LC\_PAPER=pt\_BR.UTF-8 LC\_NAME=C

[9] LC\_ADDRESS=C LC\_TELEPHONE=C

[11] LC\_MEASUREMENT=pt\_BR.UTF-8 LC\_IDENTIFICATION=C

```
time zone: America/Recife
tzcode source: system (glibc)
attached base packages:
[1] stats
              graphics grDevices utils
                                             datasets methods
                                                                 base
other attached packages:
 [1] lubridate_1.9.3 forcats_1.0.0
                                      stringr_1.5.1
                                                      dplyr_1.1.4
 [5] purrr_1.0.2
                     readr_2.1.5
                                      tidyr_1.3.1
                                                      tibble_3.3.0
 [9] ggplot2_4.0.0
                     tidyverse_2.0.0
loaded via a namespace (and not attached):
 [1] gtable_0.3.6
                        jsonlite_1.8.8
                                            compiler_4.5.1
                                                                tidyselect_1.2.1
                                                                fastmap_1.1.1
 [5] dichromat_2.0-0.1
                        scales_1.4.0
                                            yaml_2.3.8
 [9] R6_2.6.1
                        generics_0.1.3
                                            knitr_1.50
                                                               pillar_1.11.0
[13] RColorBrewer_1.1-3 tzdb_0.4.0
                                            rlang_1.1.6
                                                                stringi_1.8.3
[17] xfun_0.53
                        S7_0.2.0
                                            timechange_0.3.0
                                                                cli_3.6.5
[21] withr_3.0.2
                        magrittr_2.0.3
                                            digest_0.6.34
                                                                grid_4.5.1
[25] rstudioapi_0.15.0 hms_1.1.3
                                            lifecycle_1.0.4
                                                                vctrs_0.6.5
                        glue 1.8.0
[29] evaluate 1.0.5
                                            farver 2.1.2
                                                                rmarkdown 2.29
                        pkgconfig_2.0.3
                                            htmltools_0.5.8.1
[33] tools_4.5.1
```

	44 -	· l			
capabilities()	# C	neca	recursos	gráficos,	etc.

jpeg	png	tiff	tcltk	X11	aqua
TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
http/ftp	sockets	libxml	fifo	cledit	iconv
TRUE	TRUE	FALSE	TRUE	FALSE	TRUE
NLS	Rprof	profmem	cairo	ICU	long.double
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
libcurl					
TRUE					

## 2.2 O que é o R

R é uma implementação moderna da linguagem S, voltada à computação estatística e à visualização de dados. Ele reúne, no mesmo lugar, um ambiente interativo para análise e criação de gráficos e uma linguagem de programação completa.

#### Nota

Em uma frase: R é um ambiente estatístico + uma linguagem, criada para trabalhar bem com dados, gráficos e métodos modernos.

#### Principais características

- Interativo e interpretado, com suporte a JIT/bytecode via pacote compiler.
- Orientado a objetos (S3, S4 e R6) e com forte base funcional.
- Modelo "tudo é objeto": números, vetores, data frames, funções, ambientes e modelos ajustados.
- Vetorização nativa e operações matriciais eficientes.
- Ecossistema de pacotes amplo (CRAN, Bioconductor) para estatística e ciência de dados.
- Extensível com C/C++/Fortran e integração com Python, SQL e serviços externos.
- Multiplataforma (Windows, macOS, Linux) e foco em reprodutibilidade (scripts, Quarto/R Markdown).

### Exemplo - "tudo é objeto"

```
x <- 1:5  # vetor (objeto)
media <- mean(x)  # função aplicada ao objeto

f <- function(z) z^2
classe_f <- class(f)  # "function" - funções também são objetos

attr(x, "nota") <- "exemplo de atributo"
lista <- list(x = x, media = media, classe_f = classe_f)

str(lista)  # inspeciona a estrutura</pre>
```

```
List of 3
$ x : int [1:5] 1 2 3 4 5
..- attr(*, "nota")= chr "exemplo de atributo"
$ media : num 3
$ classe_f: chr "function"
```

#### lista

```
$x
[1] 1 2 3 4 5
attr(,"nota")
[1] "exemplo de atributo"

$media
[1] 3

$classe_f
[1] "function"
```

```
Dica
```

Dica: use str(obj) para entender a estrutura de qualquer objeto no R. Outras funções úteis: class(), attributes(), typeof(), methods(class = ...).

Aqui vai como mudar o prompt do R para R>, na sessão atual usando:

```
options(prompt = "R> ", continue = "+")
```

Para adicionar comentários no R, como você observou nos exemplos acima, basta usar #, como no exemplo abaixo:

```
#Este é um comentário!
#Que o semestre seja leve para todos nós!
```

Uma sessão ativa do R sempre está "apontando" para um diretório de trabalho. Se você não informar um caminho completo ao salvar ou importar arquivos, o R usará esse diretório por padrão. Para descobrir onde ele está:

```
getwd()
```

[1] "/home/manoel/Documentos/Book\_Computacional\_UFC"

Se você deseja modificar o diretório de trabalho, isso pode ser feito de maneira simples:

```
# Windows
setwd("C:/Users/SeuUsuario/Documents/meu-projeto")

# macOS / Linux
setwd("/Users/seuusuario/meu-projeto")
```

O R não vem com todos os pacotes adicionais que a comunidade desenvolve. Para usá-los, você precisa baixar e instalar a partir de um repositório, em geral, o CRAN.

```
# define um espelho (mirror) do CRAN para esta sessão
options(repos = c(CRAN = "https://cloud.r-project.org"))

# instala um ou mais pacotes
install.packages("tidyverse")
install.packages(c("ggplot2", "dplyr", "readr"))

# carrega para usar
library(tidyverse)
```

A instalação pode ser feita também facilmente usando o RStudio.

Pacotes da comunidade são atualizados com frequência para corrigir erros e acrescentar recursos. Periodicamente, vale checar se há novas versões para o que você já tem instalado. A partir do console do R, você pode verificar e atualizar com um único comando.

```
# defina um mirror (opcional, mas recomendável)
options(repos = c(CRAN = "https://cloud.r-project.org"))
# procura versões mais novas e atualiza automaticamente
update.packages(ask = FALSE, checkBuilt = TRUE)
```

- ask = FALSE atualiza sem perguntar pacote a pacote.
- checkBuilt = TRUE recompila pacotes se sua versão do R mudou.

#### Ver o que está desatualizado antes:

```
old.packages() # retorna uma tabela com pacotes que têm versão mais nova

Package LibPath
ggiraph "ggiraph" "/home/manoel/R/x86_64-pc-linux-gnu-library/4.5"
ggplotify "ggplotify" "/home/manoel/R/x86_64-pc-linux-gnu-library/4.5"
```

```
"pillar"
                                "/home/manoel/R/x86_64-pc-linux-gnu-library/4.5"
pillar
askpass
               "askpass"
                                "/usr/lib/R/site-library"
                                "/usr/lib/R/site-library"
backports
               "backports"
               "bit"
                                "/usr/lib/R/site-library"
bit
                                "/usr/lib/R/site-library"
bit64
               "bit64"
                                "/usr/lib/R/site-library"
broom
               "broom"
bslib
               "bslib"
                                "/usr/lib/R/site-library"
cachem
               "cachem"
                                "/usr/lib/R/site-library"
callr
               "callr"
                                "/usr/lib/R/site-library"
cli
               "cli"
                                "/usr/lib/R/site-library"
               "colorspace"
                                "/usr/lib/R/site-library"
colorspace
                                "/usr/lib/R/site-library"
commonmark
               "commonmark"
               "cpp11"
                                "/usr/lib/R/site-library"
cpp11
crayon
               "crayon"
                                "/usr/lib/R/site-library"
curl
               "curl"
                                "/usr/lib/R/site-library"
                                "/usr/lib/R/site-library"
data.table
               "data.table"
DBI
               "DBI"
                                "/usr/lib/R/site-library"
               "dbplyr"
                                "/usr/lib/R/site-library"
dbplyr
               "digest"
                                "/usr/lib/R/site-library"
digest
               "dtplyr"
                                "/usr/lib/R/site-library"
dtplyr
evaluate
               "evaluate"
                                "/usr/lib/R/site-library"
               "fansi"
                                "/usr/lib/R/site-library"
fansi
farver
               "farver"
                                "/usr/lib/R/site-library"
               "fastmap"
                                "/usr/lib/R/site-library"
fastmap
fontawesome
               "fontawesome"
                                "/usr/lib/R/site-library"
               "fs"
                                "/usr/lib/R/site-library"
fs
                                "/usr/lib/R/site-library"
gargle
               "gargle"
generics
               "generics"
                                "/usr/lib/R/site-library"
                                "/usr/lib/R/site-library"
ggplot2
               "ggplot2"
               "glue"
                                "/usr/lib/R/site-library"
glue
               "googledrive"
                                "/usr/lib/R/site-library"
googledrive
googlesheets4
               "googlesheets4"
                                "/usr/lib/R/site-library"
               "gtable"
gtable
                                "/usr/lib/R/site-library"
               "haven"
                                "/usr/lib/R/site-library"
haven
                                "/usr/lib/R/site-library"
               "highr"
highr
htmltools
               "htmltools"
                                "/usr/lib/R/site-library"
httpuv
               "httpuv"
                                "/usr/lib/R/site-library"
jsonlite
               "jsonlite"
                                "/usr/lib/R/site-library"
               "knitr"
                                "/usr/lib/R/site-library"
knitr
later
               "later"
                                "/usr/lib/R/site-library"
               "lubridate"
                                "/usr/lib/R/site-library"
lubridate
               "magrittr"
                                "/usr/lib/R/site-library"
magrittr
mime
               "mime"
                                "/usr/lib/R/site-library"
```

```
"munsell"
                                "/usr/lib/R/site-library"
munsell
openssl
               "openssl"
                                "/usr/lib/R/site-library"
               "pillar"
                                "/usr/lib/R/site-library"
pillar
                                "/usr/lib/R/site-library"
pkgKitten
               "pkgKitten"
                                "/usr/lib/R/site-library"
processx
               "processx"
                                "/usr/lib/R/site-library"
promises
               "promises"
               "ps"
                                "/usr/lib/R/site-library"
ps
               "purrr"
                                "/usr/lib/R/site-library"
purrr
               "R6"
                                "/usr/lib/R/site-library"
R6
ragg
               "ragg"
                                "/usr/lib/R/site-library"
               "Rcpp"
                                "/usr/lib/R/site-library"
Rcpp
                                "/usr/lib/R/site-library"
readxl
               "readxl"
               "reprex"
                                "/usr/lib/R/site-library"
reprex
rlang
               "rlang"
                                "/usr/lib/R/site-library"
rmarkdown
               "rmarkdown"
                                "/usr/lib/R/site-library"
               "rstudioapi"
                                "/usr/lib/R/site-library"
rstudioapi
               "rvest"
                                "/usr/lib/R/site-library"
rvest
               "sass"
                                "/usr/lib/R/site-library"
sass
                                "/usr/lib/R/site-library"
scales
               "scales"
               "shiny"
                                "/usr/lib/R/site-library"
shiny
               "stringi"
stringi
                                "/usr/lib/R/site-library"
                                "/usr/lib/R/site-library"
stringr
               "stringr"
sys
               "sys"
                                "/usr/lib/R/site-library"
               "systemfonts"
                                "/usr/lib/R/site-library"
systemfonts
               "textshaping"
                                "/usr/lib/R/site-library"
textshaping
                                "/usr/lib/R/site-library"
               "tibble"
tibble
                                "/usr/lib/R/site-library"
tidyselect
               "tidyselect"
tinytex
               "tinytex"
                                "/usr/lib/R/site-library"
               "tzdb"
                                "/usr/lib/R/site-library"
tzdb
utf8
               "utf8"
                                "/usr/lib/R/site-library"
               "uuid"
                                "/usr/lib/R/site-library"
uuid
vroom
               "vroom"
                                "/usr/lib/R/site-library"
withr
               "withr"
                                "/usr/lib/R/site-library"
               "xfun"
                                "/usr/lib/R/site-library"
xfun
                                "/usr/lib/R/site-library"
xm12
               "xm12"
yaml
               "yaml"
                                "/usr/lib/R/site-library"
               "boot"
boot
                                "/usr/lib/R/library"
lattice
               "lattice"
                                "/usr/lib/R/library"
                                "/usr/lib/R/library"
mgcv
               "mgcv"
               "spatial"
                                "/usr/lib/R/library"
spatial
               Installed Built
                                  ReposVer
                         "4.5.1" "0.9.1"
               "0.9.0"
ggiraph
ggplotify
               "0.1.2"
                         "4.5.0" "0.1.3"
```

```
"1.11.0"
                         "4.5.1" "1.11.1"
pillar
               "1.2.0"
                         "4.3.1" "1.2.1"
askpass
                         "4.1.2" "1.5.0"
               "1.4.1"
backports
               "4.0.5"
                         "4.2.2" "4.6.0"
bit
                         "4.0.2" "4.6.0-1"
              "4.0.5"
bit64
               "1.0.5"
                         "4.3.0" "1.0.10"
broom
bslib
               "0.6.1"
                         "4.3.2" "0.9.0"
                         "4.3.0" "1.1.0"
cachem
               "1.0.8"
               "3.7.5"
                         "4.3.3" "3.7.6"
callr
               "3.6.2"
                         "4.3.2" "3.6.5"
cli
              "2.1-0"
                         "4.2.2" "2.1-1"
colorspace
              "1.9.1"
                         "4.3.2" "2.0.0"
commonmark
               "0.4.7"
                         "4.3.2" "0.5.2"
cpp11
              "1.5.2"
                         "4.2.2" "1.5.3"
crayon
               "5.2.0"
                         "4.3.3" "7.0.0"
curl
              "1.14.10" "4.3.2" "1.17.8"
data.table
DBI
               "1.2.2"
                         "4.3.2" "1.2.3"
               "2.4.0"
                         "4.3.2" "2.5.1"
dbplyr
               "0.6.34"
                         "4.3.2" "0.6.37"
digest
                         "4.3.0" "1.3.2"
              "1.3.1"
dtplyr
               "0.23"
                         "4.3.2" "1.0.5"
evaluate
               "1.0.5"
                         "4.3.2" "1.0.6"
fansi
                         "4.2.1" "2.1.2"
farver
              "2.1.1"
               "1.1.1"
                         "4.2.2" "1.2.0"
fastmap
               "0.5.2"
                         "4.3.1" "0.5.3"
fontawesome
                         "4.3.3" "1.6.6"
               "1.6.3"
fs
               "1.5.2"
                         "4.3.1" "1.6.0"
gargle
                         "4.2.1" "0.1.4"
generics
              "0.1.3"
                         "4.3.1" "4.0.0"
               "3.4.4"
ggplot2
               "1.7.0"
                         "4.3.2" "1.8.0"
glue
               "2.1.1"
                         "4.3.1" "2.1.2"
googledrive
googlesheets4 "1.1.1"
                         "4.3.1" "1.1.2"
                         "4.3.1" "0.3.6"
gtable
               "0.3.4"
               "2.5.4"
                         "4.3.2" "2.5.5"
haven
              "0.10"
                         "4.2.2" "0.11"
highr
               "0.5.7"
                         "4.3.2" "0.5.8.1"
{\tt htmltools}
               "1.6.14"
                         "4.3.3" "1.6.16"
httpuv
               "1.8.8"
                         "4.3.2" "2.0.0"
jsonlite
              "1.45"
                         "4.3.2" "1.50"
knitr
               "1.3.2"
                         "4.3.2" "1.4.4"
later
               "1.9.3"
                         "4.3.1" "1.9.4"
lubridate
                         "4.1.2" "2.0.4"
              "2.0.3"
magrittr
               "0.12"
                         "4.1.1" "0.13"
mime
```

```
"0.5.0"
                         "4.0.1" "0.5.1"
munsell
              "2.1.1"
                         "4.3.3" "2.3.3"
openssl
                         "4.3.0" "1.11.1"
              "1.9.0"
pillar
              "0.2.3"
                         "4.2.2" "0.2.4"
pkgKitten
                         "4.3.2" "3.8.6"
              "3.8.3"
processx
              "1.2.1"
                         "4.3.1" "1.3.3"
promises
              "1.7.6"
                         "4.3.2" "1.9.1"
ps
                         "4.3.1" "1.1.0"
              "1.0.2"
purrr
              "2.5.1"
                         "4.1.1" "2.6.1"
R6
              "1.2.7"
                         "4.3.3" "1.5.0"
ragg
              "1.0.12"
                         "4.3.2" "1.1.0"
Rcpp
              "1.4.3"
                         "4.3.1" "1.4.5"
readxl
              "2.1.0"
                         "4.3.2" "2.1.1"
reprex
              "1.1.3"
                         "4.3.2" "1.1.6"
rlang
              "2.25"
                         "4.3.2" "2.29"
rmarkdown
                         "4.3.1" "0.17.1"
              "0.15.0"
rstudioapi
              "1.0.3"
                         "4.2.1" "1.0.5"
rvest
              "0.4.8"
                         "4.3.2" "0.4.10"
sass
              "1.3.0"
                         "4.3.2" "1.4.0"
scales
                         "4.3.2" "1.11.1"
              "1.8.0"
shiny
              "1.8.3"
stringi
                         "4.3.2" "1.8.7"
              "1.5.1"
                         "4.3.2" "1.5.2"
stringr
              "3.4.2"
                         "4.3.1" "3.4.3"
sys
              "1.0.5"
                         "4.3.1" "1.2.3"
systemfonts
              "0.3.7"
                         "4.3.1" "1.0.3"
textshaping
              "3.2.1"
                         "4.3.1" "3.3.0"
tibble
              "1.2.0"
                         "4.2.2" "1.2.1"
tidyselect
              "0.49"
                         "4.3.2" "0.57"
tinytex
              "0.4.0"
                         "4.3.1" "0.5.0"
tzdb
utf8
              "1.2.4"
                         "4.3.1" "1.2.6"
              "1.2-0"
                         "4.3.2" "1.2-1"
uuid
                         "4.3.2" "1.6.6"
              "1.6.5"
vroom
                         "4.1.2" "3.0.2"
withr
              "2.5.0"
              "0.41"
                         "4.3.2" "0.53"
xfun
                         "4.3.2" "1.4.0"
xml2
              "1.3.6"
                         "4.3.2" "2.3.10"
yaml
              "2.3.8"
              "1.3-31"
                         "4.4.2" "1.3-32"
boot
                         "4.3.3" "0.22-7"
              "0.22-5"
lattice
              "1.9-1"
                         "4.3.2" "1.9-3"
mgcv
              "7.3-17" "4.4.0" "7.3-18"
spatial
              Repository
              "https://cloud.r-project.org/src/contrib"
ggiraph
ggplotify
               "https://cloud.r-project.org/src/contrib"
```

```
"https://cloud.r-project.org/src/contrib"
pillar
              "https://cloud.r-project.org/src/contrib"
askpass
backports
              "https://cloud.r-project.org/src/contrib"
              "https://cloud.r-project.org/src/contrib"
bit
              "https://cloud.r-project.org/src/contrib"
bit64
              "https://cloud.r-project.org/src/contrib"
broom
bslib
              "https://cloud.r-project.org/src/contrib"
cachem
              "https://cloud.r-project.org/src/contrib"
callr
              "https://cloud.r-project.org/src/contrib"
              "https://cloud.r-project.org/src/contrib"
cli
              "https://cloud.r-project.org/src/contrib"
colorspace
              "https://cloud.r-project.org/src/contrib"
commonmark
              "https://cloud.r-project.org/src/contrib"
cpp11
              "https://cloud.r-project.org/src/contrib"
crayon
curl
              "https://cloud.r-project.org/src/contrib"
              "https://cloud.r-project.org/src/contrib"
data.table
DBI
              "https://cloud.r-project.org/src/contrib"
              "https://cloud.r-project.org/src/contrib"
dbplyr
digest
              "https://cloud.r-project.org/src/contrib"
              "https://cloud.r-project.org/src/contrib"
dtplyr
evaluate
              "https://cloud.r-project.org/src/contrib"
              "https://cloud.r-project.org/src/contrib"
fansi
farver
              "https://cloud.r-project.org/src/contrib"
              "https://cloud.r-project.org/src/contrib"
fastmap
fontawesome
              "https://cloud.r-project.org/src/contrib"
              "https://cloud.r-project.org/src/contrib"
fs
              "https://cloud.r-project.org/src/contrib"
gargle
              "https://cloud.r-project.org/src/contrib"
generics
              "https://cloud.r-project.org/src/contrib"
ggplot2
              "https://cloud.r-project.org/src/contrib"
glue
              "https://cloud.r-project.org/src/contrib"
googledrive
googlesheets4 "https://cloud.r-project.org/src/contrib"
gtable
              "https://cloud.r-project.org/src/contrib"
              "https://cloud.r-project.org/src/contrib"
haven
              "https://cloud.r-project.org/src/contrib"
highr
              "https://cloud.r-project.org/src/contrib"
htmltools
              "https://cloud.r-project.org/src/contrib"
httpuv
isonlite
              "https://cloud.r-project.org/src/contrib"
knitr
              "https://cloud.r-project.org/src/contrib"
              "https://cloud.r-project.org/src/contrib"
later
              "https://cloud.r-project.org/src/contrib"
lubridate
              "https://cloud.r-project.org/src/contrib"
magrittr
mime
              "https://cloud.r-project.org/src/contrib"
```

```
munsell
              "https://cloud.r-project.org/src/contrib"
              "https://cloud.r-project.org/src/contrib"
openssl
              "https://cloud.r-project.org/src/contrib"
pillar
              "https://cloud.r-project.org/src/contrib"
pkgKitten
              "https://cloud.r-project.org/src/contrib"
processx
              "https://cloud.r-project.org/src/contrib"
promises
              "https://cloud.r-project.org/src/contrib"
ps
purrr
              "https://cloud.r-project.org/src/contrib"
R6
              "https://cloud.r-project.org/src/contrib"
ragg
              "https://cloud.r-project.org/src/contrib"
              "https://cloud.r-project.org/src/contrib"
Rcpp
              "https://cloud.r-project.org/src/contrib"
readxl
              "https://cloud.r-project.org/src/contrib"
reprex
              "https://cloud.r-project.org/src/contrib"
rlang
rmarkdown
              "https://cloud.r-project.org/src/contrib"
              "https://cloud.r-project.org/src/contrib"
rstudioapi
rvest
              "https://cloud.r-project.org/src/contrib"
              "https://cloud.r-project.org/src/contrib"
sass
              "https://cloud.r-project.org/src/contrib"
scales
              "https://cloud.r-project.org/src/contrib"
shiny
stringi
              "https://cloud.r-project.org/src/contrib"
              "https://cloud.r-project.org/src/contrib"
stringr
sys
              "https://cloud.r-project.org/src/contrib"
systemfonts
              "https://cloud.r-project.org/src/contrib"
textshaping
              "https://cloud.r-project.org/src/contrib"
              "https://cloud.r-project.org/src/contrib"
tibble
              "https://cloud.r-project.org/src/contrib"
tidyselect
              "https://cloud.r-project.org/src/contrib"
tinytex
              "https://cloud.r-project.org/src/contrib"
tzdb
utf8
              "https://cloud.r-project.org/src/contrib"
              "https://cloud.r-project.org/src/contrib"
uuid
vroom
              "https://cloud.r-project.org/src/contrib"
withr
              "https://cloud.r-project.org/src/contrib"
              "https://cloud.r-project.org/src/contrib"
xfun
              "https://cloud.r-project.org/src/contrib"
xm12
              "https://cloud.r-project.org/src/contrib"
yaml
              "https://cloud.r-project.org/src/contrib"
boot
lattice
              "https://cloud.r-project.org/src/contrib"
              "https://cloud.r-project.org/src/contrib"
mgcv
              "https://cloud.r-project.org/src/contrib"
spatial
```

#### Atualizar apenas alguns pacotes:

```
update.packages(oldPkgs = c("gamlss", "ggmap"), ask = FALSE)
```

O R traz um conjunto amplo de arquivos de ajuda que permitem:

- Buscar funcionalidades por nome ou por tema.
- Entender como usar uma função e quais argumentos ela recebe.
- Esclarecer o papel de cada argumento e o que a função retorna.
- Ver exemplos de uso prontos para rodar.
- Saber como citar o R, pacotes e conjuntos de dados em publicações.

Agora iremos ver algumas funções úteis:

• Ajuda com função específica:

```
?lm # atalho
help("lm") #abre a documentação da função
help(package = "stats") # indice de um pacote
```

• Buscar por tema (se você não sabe o nome exato):

```
??"linear model"  # busca por tópicos
help.search("normal")  # alternativa
apropos("plot")  # lista objetos cujo nome contém "plot"
```

• Ver argumentos e formas de chamada

```
args(lm) # nomes dos argumentos (se aplicável)
formals(lm) # valores-padrão dos argumentos
```

• Exemplos, retorno e estrutura

• Vignettes (tutoriais)

```
vignette() # lista geral
vignette(package = "ggplot2") # vignettes de um pacote
browseVignettes(package = "dplyr") # abre no navegador
```

```
• Como citar R/pacotes/dados
citation() # como citar o R
To cite R in publications use:
  R Core Team (2025). _R: A Language and Environment for Statistical
  Computing_. R Foundation for Statistical Computing, Vienna, Austria.
  <https://www.R-project.org/>.
Uma entrada BibTeX para usuários(as) de LaTeX é
  @Manual{,
    title = {R: A Language and Environment for Statistical Computing},
    author = {{R Core Team}},
    organization = {R Foundation for Statistical Computing},
    address = {Vienna, Austria},
    year = \{2025\},\
    url = {https://www.R-project.org/},
  }
We have invested a lot of time and effort in creating R, please cite it
when using it for data analysis. See also 'citation("pkgname")' for
citing R packages.
citation("ggplot2")
                      # como citar um pacote
To cite ggplot2 in publications, please use
  H. Wickham. ggplot2: Elegant Graphics for Data Analysis.
  Springer-Verlag New York, 2016.
Uma entrada BibTeX para usuários(as) de LaTeX é
  @Book{,
    author = {Hadley Wickham},
    title = {ggplot2: Elegant Graphics for Data Analysis},
```

```
publisher = {Springer-Verlag New York},
  year = {2016},
  isbn = {978-3-319-24277-4},
  url = {https://ggplot2.tidyverse.org},
}
```

```
toBibtex(citation("ggplot2")) # em BibTeX
```

```
@Book{,
  author = {Hadley Wickham},
  title = {ggplot2: Elegant Graphics for Data Analysis},
  publisher = {Springer-Verlag New York},
  year = {2016},
  isbn = {978-3-319-24277-4},
  url = {https://ggplot2.tidyverse.org},
}
```

• Conjuntos de dados

```
data() # abre uma lista com os datasets disponíveis
# ?mtcars # ajuda de um dataset específico
```

Console vs. editor: Você pode digitar comandos diretamente no console ou escrever um script no editor e então executá-lo. Em materiais didáticos, distinguimos os dois formatos para evitar confusão.

- 1) Código mostrado com prompt (R>) = digitado no console
- Em livros/notas, comandos de console costumam aparecer com o prompt na frente. Exemplo (divisão de 14 por 6):

```
R> options(prompt = "R> ", continue = "+ ")
R> 14/6
```

- Ao copiar para o R, remova o R>, o console já mostra o seu próprio prompt.
- 2) Código pensado para script (sem prompt)
- Quando o texto disser "escreva no editor e depois execute", o código aparece sem R>. Exemplo (laço simples):

```
for (myitem in 5:7) {
  cat("--INÍCIO DO BLOCO--\n")
  cat("o item atual é", myitem, "\n")
  cat("--FIM DO BLOCO--\n\n")
}
```

```
--INÍCIO DO BLOCO--
o item atual é 5
--FIM DO BLOCO--
--INÍCIO DO BLOCO--
o item atual é 6
--FIM DO BLOCO--
--INÍCIO DO BLOCO--
o item atual é 7
--FIM DO BLOCO--
```



Dica

Dica de estilo: chaves na mesma linha do for, quebra de linha para o corpo e indentação consistente.

- 3) Linhas longas: uma linha só ou "quebradas"
- Chamadas extensas podem ficar em uma linha ou ser quebradas em pontos naturais (geralmente após vírgulas ou antes de argumentos nomeados).
- Ao quebrar, alinhe com o parêntese de abertura ou indente um nível.

Uma linha:

```
ordfac.vec <- factor(x = c("Small", "Large", "Large", "Regular", "Small"),</pre>
                       levels = c("Small", "Regular", "Large"),
                       ordered = TRUE)
```

Quebrada (equivalente, apenas formatação):

```
ordfac.vec2 <-
 factor(
            = c("Small", "Large", "Large", "Regular", "Small"),
    levels = c("Small", "Regular", "Large"),
```

```
ordered = TRUE
)
identical(ordfac.vec, ordfac.vec2)
```

[1] TRUE

# 2.3 Números, Aritmética, Atribuição e Vetores

1) Números (numerics)

Em R, o tipo numérico padrão é **double** (ponto flutuante). Você também pode ter **inteiros** (sufixo L) e **complexos**.

[1] -Inf

-1/0 # -Inf

```
0/0 # NaN
```

[1] NaN

```
is.finite(c(Inf, 3.14)) # checa finitude
```

[1] FALSE TRUE



Precisão numérica: comparações com == podem falhar por arredondamento binário. Prefira all.equal(x, y).

```
# 1) Exemplo clássico
x < -0.1 + 0.2
y < -0.3
                              # pode dar FALSE
x == y
```

[1] FALSE

```
identical(x, y)
                             # compara "bit a bit": quase sempre FALSE aqui
```

[1] FALSE

```
# Ver a diferença real (use mais dígitos)
old <- options(digits = 17)</pre>
                              # diferença minúscula de arredondamento binário
x; y; x - y
```

- [1] 0.30000000000000004
- [1] 0.299999999999999
- [1] 5.5511151231257827e-17

```
options(old)
# Comparação com tolerância
[1] 0.3
[1] 0.3
all.equal(x, y)
                             # Se "quase iguais", retorna TRUE. Se diferem, retorna uma stri:
[1] TRUE
isTRUE(all.equal(x, y))
                             # retorna um lógico (TRUE/FALSE)
[1] TRUE
# 2) Outros casos que sofrem com ponto flutuante
sin(pi)
                             # ~ 1.224646799e-16 (não é 0 exato)
[1] 1.224647e-16
sin(pi) == 0
                             # FALSE
[1] FALSE
all.equal(sin(pi), 0)
                             # TRUE
[1] TRUE
# 3) Ajustando a tolerância manualmente (quando precisar ser mais/menos rígido)
all.equal(x, y, tolerance = 1e-12) # ainda TRUE
[1] TRUE
```

```
all.equal(x, y, tolerance = 1e-20) # agora acusa diferença
```

[1] "Mean relative difference: 1.850372e-16"

```
# 4) Estratégia base sem all.equal: comparar o |erro| com um limite abs((sqrt(2))^2 - 2) < 1e-12 # TRUE (aceita "igualdade" numérica com folga)
```

- [1] TRUE
  - 2) Aritmética

Operadores básicos: + - \* / ^ e, ainda, divisão inteira %/% e módulo %%.

```
14/6
```

[1] 2.333333

```
14 %/% 6 # quociente inteiro
```

[1] 2

```
14 %% 6 # resto
```

[1] 2

```
2<sup>3</sup> # potência
```

- [1] 8
  - Precedência:  $^{\hat{}} \rightarrow *$  / %/% %//  $\rightarrow *$  -. Use parênteses para deixar claro!
  - 3) Atribuição

O padrão recomendado é <-. = também atribui, mas use-o preferencialmente para argumentos de função. -> é atribuição para a direita. <<- afeta o ambiente pai (use com parcimônia).

```
x <- 10
y = 5  # ok, mas prefira <- fora de chamadas de função
z <- x + y

z -> resultado  # atribuição para a direita (menos comum)
resultado
```

[1] 15

```
# Exemplo de <<- (evite, pode dificultar depuração)
contador <- local({</pre>
  n <- 0
  function() { n <<- n + 1; n }</pre>
})
contador(); contador(); contador()
```

- [1] 1
- [1] 2
- [1] 3



🔔 Aviso

Boas práticas: Evite <<- sempre que possível; prefira retornar valores e trabalhar com escopos explícitos.

4) Vetores (atômicos)

Vetores são coleções unidimensionais do mesmo tipo: lógico, inteiro, double, caractere, complexo ou raw.

```
v \leftarrow c(2, 4, 6, 8)
length(v); typeof(v)
```

- [1] 4
- [1] "double"

```
seq(1, 5, by = 2)
                       # sequência
```

[1] 1 3 5

```
1:5
                        # atalho
```

[1] 1 2 3 4 5

```
# repetição
rep(3, times = 4)
```

#### [1] 3 3 3 3

• Coerção automática: ao misturar tipos, R promove para um tipo comum.

```
c(1, TRUE, "a") # tudo vira character
```

```
[1] "1"
           "TRUE" "a"
```

#### Aviso

Quando você faz c(1, TRUE, "a"), o R precisa escolher um tipo comum para todos os elementos. A regra de coerção (promoção de tipos) segue, a grosso modo: logical integer double complex character

• Indexação e filtragem:

```
x \leftarrow c(10, 20, 30, 40, 50)
x[3]
                     # posição
```

[1] 30

```
x[-1]
                      # tudo, exceto o 1^{\circ}
```

[1] 20 30 40 50

```
x[x > 25]
                  # filtragem lógica
```

[1] 30 40 50

```
which(x > 25)
                 # posições TRUE
```

[1] 3 4 5

```
names(x) <- letters[1:5]
x["c"]</pre>
```

с 30

#### 4.1. Vetorização e reciclagem

A maioria das operações é vetorizada (aplica-se elemento a elemento). Quando os comprimentos diferem, R tenta reciclar o menor vetor.

```
a <- 1:5
b <- 10
a + b # soma escalar-vetor (b é reciclado)</pre>
```

[1] 11 12 13 14 15

```
a + c(1, 2) # reciclagem com aviso (5 não é múltiplo de 2)
```

Warning in a + c(1, 2): comprimento do objeto maior não é múltiplo do comprimento do objeto menor

[1] 2 4 4 6 6

4.2. Nomes  $\times$  Objetos e copy-on-modify

```
x <- 1:3
y <- x
y[1] <- 99
x # permanece 1 2 3
```

[1] 1 2 3

```
y # 99 2 3
```

[1] 99 2 3

Aviso

Exceção: ambientes e estruturas por referência (p.ex., R6) não seguem copy-on-modify.

#### **Exercícios**

- 1. Crie um vetor com os números de 5 a 15 e calcule a média.
- 2. Use seq() para gerar 0, 0.5, 1,..., 5.
- 3. Mostre com código a diferença entre %/% e %% usando 14 e 6.
- 4. Atribua x <- 1:4 e some com o vetor c(10, 20). O que acontece?
- 5. Verifique o tipo de 1, 1L e 1+0i. Explique a diferença entre tipo (typeof) e classe (class).
- 6. Crie e armazene uma sequência de valores de 5 a -11, progredindo em passos de 0.3.
- 7. Reescreva o objeto de (6) com a mesma sequência invertida.
- 8. Repita o vetor c(-1, 3, -5, 7, -9) duas vezes, com cada elemento repetido 10 vezes. Apresente os dados ordenados do maior para o menor.
- 9. Crie e armazene um vetor contendo, em qualquer ordem:
- a) inteiros de 6 a 12 (inclusive);
- b) 5.3 repetido 3 vezes;
- c) o número -3;
- d) Uma sequência de nove valores começando em 102 e terminando no número que é o comprimento total do vetor criado em (8).
- 10. Confirme que o comprimento do vetor criado em (9) é 20.

#### 2.4 Matrizes

A matriz é uma construção matemática importante e é essencial para muitos métodos estatísticos. Normalmente descreve-se uma matriz A como uma matriz  $m \times n$ ; isto é, A terá exatamente m linhas e n colunas. Isso significa que A terá um total de mn entradas, com cada entrada a i, j ocupando uma posição única dada por sua linha específica (i = 1, 2, ..., m)e coluna (j = 1, 2, ..., n).

Para criar uma matriz no R, use o comando, apropriadamente chamado, matrix, fornecendo as entradas da matriz ao argumento data como um vetor:

```
A <- matrix(
    data = c(-3, 2, 893, 0.17),
    nrow = 2, #linha
    ncol = 2 #colunas
)</pre>
```

```
[,1] [,2]
[1,] -3 893.00
[2,] 2 0.17
```

É importante estar ciente de como o R preenche a matriz usando as entradas de data. Observando o exemplo anterior, você pode ver que a matriz A foi preenchida coluna por coluna ao ler as entradas de dados da esquerda para a direita. Você pode controlar como o R preenche os dados usando o argumento byrow, como mostrado nos exemplos a seguir:

```
matrix(data = c(1,2,3,4,5,6), nrow = 2, ncol = 3, byrow = FALSE)
```

```
[,1] [,2] [,3]
[1,] 1 3 5
[2,] 2 4 6
```

Qual a diferença na construção da matriz ao usar byrow = TRUE?

Também é possível construir matrizes usando os comandos cbind e rbind. Veja os exemplos abaixo:

```
cbind(c(1,4), c(2,5), c(3,6))
```

```
[,1] [,2] [,3]
[1,] 1 2 3
[2,] 4 5 6
```

```
rbind(1:3,4:6)
```

```
[,1] [,2] [,3]
[1,] 1 2 3
[2,] 4 5 6
```

Para saber a dimensão, número de linhas e colunas de uma matriz você pode usar os seguintes comandos:

```
dim(A); nrow(A); ncol(A)
```

- [1] 2 2
- [1] 2
- [1] 2

Considere a seguinte matriz:

```
A <- matrix(c(0.3, 4.5, 55.3, 91, 0.1, 105.5, -4.2, 8.2, 27.9), nrow = 3, ncol = 3)
A
```

```
[,1] [,2] [,3]
[1,] 0.3 91.0 -4.2
[2,] 4.5 0.1 8.2
[3,] 55.3 105.5 27.9
```

Para dizer ao R para "olhar para a terceira linha de A e me dar o elemento da segunda coluna", você executa o seguinte:

```
A[3, 2]
```

[1] 105.5

Como esperado, você recebe o elemento na posição [3, 2].

Você também pode identificar os valores ao longo da diagonal de uma matriz quadrada (isto é, uma matriz com igual número de linhas e colunas) usando o comando diag.

```
diag(A)
```

[1] 0.3 0.1 27.9

#### **Exercícios**

(a) Construa e armazene uma matriz  $4 \times 2$  preenchida **por linhas** com os valores: 4.3, 3.1, 8.2, 8.2, 3.2, 0.9, 1.6 e 6.5 (nessa ordem).

```
A <- matrix(c(4.3, 3.1, 8.2, 8.2, 3.2, 0.9, 1.6, 6.5), nrow = 4, byrow = TRUE); A

[,1] [,2]
[1,] 4.3 3.1
[2,] 8.2 8.2
[3,] 3.2 0.9
[4,] 1.6 6.5
```

(b) Confirme que as dimensões da matriz do item (a) são  $3 \times 2$  se você remover qualquer uma das linhas.

```
matrix_test_32 <- function(data){

v_saida <- NULL
for(i in 1:nrow(data)){

v_saida[i] <- all(dim(data[-i, , drop = FALSE]) == c(3,2))
}

sprintf( "A matriz é 3x2? %s", ifelse(all(v_saida) == TRUE, "Sim", "Não" ) )
}

matrix_test_32(A)</pre>
```

[1] "A matriz é 3x2? Sim"

```
B \leftarrow matrix(c(1,2,3,4), ncol = 2); B
```

```
[,1] [,2]
[1,] 1 3
[2,] 2 4
```

```
matrix_test_32(B)
```

- [1] "A matriz é 3x2? Não"
- (c) Sobrescreva a **segunda coluna** da matriz do item (a) com essa mesma coluna **ordenada** do menor para o maior.
- (d) O que o R retorna se você excluir a quarta linha e a primeira coluna do item (c)? Use matrix para garantir que o resultado seja uma matriz de uma única coluna, e não um vetor.
- (e) Armazene os quatro elementos inferiores do item (c) como uma nova matriz  $2 \times 2$ .
- (f) Sobrescreva, **nesta ordem**, os elementos do item (c) nas posições (4, 2), (1, 2), (4, 1) e (1, 1) com  $-\frac{1}{2}$  dos dois valores na **diagonal do item (e)**.

# 2.5 Operações com matrizes

Em R, a transposta de uma matriz se obtém com t(), ela troca linhas por colunas.

```
A <- rbind(c(2,5,2),c(6,1,4))
t(A)
```

```
[,1] [,2]
[1,] 2 6
[2,] 5 1
[3,] 2 4
```

Você pode criar uma matriz identidade de qualquer dimensão usando a função matrix, mas há uma forma mais rápida usando diag. Antes, usei diag em uma matriz existente para extrair ou sobrescrever seus elementos da diagonal. Você também pode usá-la assim:

```
I4 <- diag(4) # 4x4
I4
```

```
[,1] [,2] [,3] [,4]
[1,]
         1
              0
                    0
[2,]
         0
              1
                    0
                          0
              0
[3,]
         0
                    1
                          0
[4,]
         0
              0
                    0
                          1
```

```
I5 <- diag(1, nrow = 5, ncol = 5) #forma explicita</pre>
I5
```

```
[,1] [,2] [,3] [,4] [,5]
[1,]
                    0
              0
                          0
[2,]
                          0
                               0
        0
              1
                    0
[3,]
              0
                    1
                          0
                               0
[4,]
              0
                    0
                          1
                               0
[5,]
                               1
```

O R realizará essa multiplicação de maneira elemento a elemento, como você poderia esperar. A multiplicação escalar de uma matriz é realizada usando o operador aritmético padrão \*.

```
A <- rbind(c(2,5,2), c(6,1,4))
a <- 2

a*A
```

```
[,1] [,2] [,3]
[1,] 4 10 4
[2,] 12 2 8
```

Você pode somar ou subtrair quaisquer duas matrizes de mesmo tamanho usando os símbolos padrão + e -.

```
A <- cbind(c(2,5,2), c(6,1,4))
A
```

```
[,1] [,2]
[1,] 2 6
[2,] 5 1
[3,] 2 4
```

```
B <- cbind(c(-2,3,6), c(8.1,8.2,-9.8))

B
```

```
[,1] [,2]
[1,] -2 8.1
[2,] 3 8.2
[3,] 6 -9.8
```

### A - B

```
[,1] [,2]
[1,] 4 -2.1
[2,] 2 -7.2
[3,] -4 13.8
```

### A + B

```
[,1] [,2]
[1,] 0 14.1
[2,] 8 9.2
[3,] 8 -5.8
```

Ao contrário da adição, subtração e multiplicação escalar, a multiplicação de matrizes não é um cálculo elemento a elemento, e o operador padrão \* não pode ser usado. Em vez disso, você deve usar o operador de produto matricial do R, escrito com símbolos de porcentagem como %\*%. Antes de tentar esse operador, vamos primeiro armazenar as duas matrizes de exemplo e verificar se o número de colunas na primeira matriz corresponde ao número de linhas na segunda matriz usando dim.

```
A <- rbind(c(2,5,2), c(6,1,4))
dim(A)
```

[1] 2 3

```
B <- cbind(c(3,-1,1), c(-3,1,5))

dim(B)
```

[1] 3 2

Isso confirma que as duas matrizes são compatíveis para a multiplicação, então você pode prosseguir.

### A%\*%B

Você pode mostrar que a multiplicação de matrizes é **não comutativa** usando as mesmas duas matrizes. Inverter a ordem da multiplicação produz um resultado completamente diferente.

### B%\*%A

Matrizes que não são invertíveis são chamadas de singulares. Inverter uma matriz é frequentemente necessário ao resolver sistemas de equações e tem implicações práticas importantes. Há vários métodos para inversão, e o custo computacional cresce muito à medida que o tamanho da matriz aumenta. Não entraremos em detalhes aqui; para discussões formais, veja Golub e Van Loan (1989). Por ora, veja a função solve do R como uma opção para inverter matrizes.

```
A <- matrix(data = c(3, 4, 1, 2), nrow = 2, ncol = 2)

A
```

### solve(A)

Você também pode verificar que o produto dessas duas matrizes (usando as regras de multiplicação de matrizes) resulta na matriz identidade  $2\times 2$ 

### **Exercícios**

(a) Calcule

$$\frac{2}{7} \left( \begin{bmatrix} 1 & 2 \\ 2 & 4 \\ 7 & 6 \end{bmatrix} - \begin{bmatrix} 10 & 20 \\ 30 & 40 \\ 50 & 60 \end{bmatrix} \right).$$

(b) Armazene as duas matrizes abaixo:

$$A = \begin{bmatrix} 1 \\ 2 \\ 7 \end{bmatrix}, \quad B = \begin{bmatrix} 3 \\ 4 \\ 8 \end{bmatrix}.$$

Quais das multiplicações a seguir são possíveis? Para as que forem, calcule o resultado.

i. 
$$A \cdot B$$
;

ii. 
$$A^{\top} \cdot B$$
;

iii. 
$$B^{\top} \cdot (A \cdot A^{\top});$$

iv. 
$$(A \cdot A^{\top}) \cdot B^{\top}$$
;

$$\text{v. } \left[ (B \cdot B^\top) + (A \cdot A^\top) - 100 \, I_3 \right]^{-1}.$$

(c) Para

$$A = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix},$$

confirme que  $A^{-1}\cdot A - I_4$  fornece uma matriz  $4\times 4$  de zeros.

### 2.6 Laços e repetições

O laço clássico, ao estilo Fortran, está disponível no R. A sintaxe é um pouco diferente, mas a ideia é idêntica: você pede que um índice, i, assuma uma sequência de valores, e que uma ou mais linhas de comandos sejam executadas tantas vezes quantos forem os valores distintos de i. A seguir, um laço executado cinco vezes com i de 1 a 5; imprimimos o quadrado de cada valor:

```
for(i in 1:5) print(i^2)
```

- [1] 1
- [1] 4
- [1] 9
- [1] 16
- [1] 25

Para várias linhas de código, você usa chaves {} para incluir o material sobre o qual o laço deve atuar. Note que a "quebra de linha" (pressionar a tecla Enter) ao final de cada comando é uma parte essencial da estrutura (você pode substituir as quebras de linha por ponto e vírgula, se preferir, mas a clareza melhora se cada comando for colocado em uma linha separada).

```
j <- k <- 0
for(i in 1:5){
j <- j + 1
k <- k + i*j
print(i + j + k)
}</pre>
```

- [1] 3
- [1] 9
- [1] 20
- [1] 38
- [1] 65

O fatorial de um número inteiro x (escrito x!) é o produto de todos os inteiros positivos de 1 até x. Por definição:

$$x! = x \times (x-1) \times (x-2) \times (x-3) \cdots \times 2 \times 1.$$

Por exemplo,  $4! = 4 \times 3 \times 2 \times 1 = 24$ . Aqui está a função:

```
fac1 <- function(x){</pre>
f <- 1
if(x < 2) return(1)
  for(i in 2:x){
    f <- f*i
  }
f
}
```

Parece complicado para uma tarefa simples. Mas vamos testar de 0 a 5:

```
sapply(0:5, fac1)
```

```
[1]
               2
                   6 24 120
      1
           1
```

Existem outras duas funções de laço no R: repeat e while. Vamos demonstrar seu uso para fins de ilustração, mas podemos fazer bem melhor ao escrever uma função compacta para calcular fatoriais (veja abaixo). Primeiro, a função while:

```
fac2 <- function(x){</pre>
  f <- 1
  t <- x
  while(t > 1){
    f <- f*t
    t <- t - 1
  }
  return(f)
}
```

Agora iremos testar a função para os números inteiros de 0 a 5:

```
sapply(0:5, fac2)
```

[1] 2 6 24 120

O ponto central

o while sozinho não gera a sequência; você precisa configurar e atualizar manualmente uma variável de controle (aqui t). Isso torna o while menos compacto do que for, mas útil quando você não sabe de antemão quantas iterações serão necessárias.

Por fim, vamos mostrar o uso da função repeat:

```
fac3 <- function(x){
    f <- 1
    t <- x
    repeat{
        if(t<2) break
        f <- f*t
        t <- t - 1
    }
    return(f)
}</pre>
```

O repeat em R não tem um fim explícito, ele vai rodar para sempre a menos que você coloque uma condição de saída que use break.

```
sapply(0:5,fac3)
```

```
[1] 1 1 2 6 24 120
```

É uma boa prática de programação em R evitar o uso de laços sempre que possível. O uso de funções vetorizadas torna isso particularmente simples em muitos casos. Suponha que você queira substituir todos os valores negativos de um array por zeros. Você poderia ter escrito algo assim:

```
for(i in 1:length(y)){
  if(y[i] < 0) y[i] <- 0
}</pre>
```

No entanto, você pode simplesmente fazer assim:

```
y[y<0] <- 0
```

Às vezes você quer fazer uma coisa se uma condição for verdadeira e outra diferente se a condição for falsa (em vez de não fazer nada, como no exemplo anterior). A função ifelse permite fazer isso em vetores inteiros sem precisar usar laços com for. Por exemplo, podemos querer substituir qualquer valor negativo de y por -1 e qualquer valor positivo ou zero por +1:

```
y < -c(-5, -1, 0, 2, 7)
ifelse(y < 0, -1, 1)
```

```
[1] -1 -1 1 1 1
```

### 2.7 Escrevendo funções no R

Normalmente você escreve funções em R para realizar operações que exigem duas ou mais linhas de código para serem executadas, e que você não quer digitar várias vezes. Podemos querer escrever funções simples para calcular medidas de tendência central, calcular fatoriais e coisas do tipo.

Funções em R são objetos que realizam operações sobre argumentos que lhes são fornecidos e retornam um ou mais valores. A sintaxe para escrever uma função é

```
function(lista_de_argumentos){
    # corpo da função
    # instruções
    return(valor)
}
```

O primeiro componente da declaração de função é a palavra-chave function, que indica ao R que você quer criar uma função.

Uma lista de argumentos é uma lista separada por vírgulas de argumentos formais. Um argumento formal pode ser:

- um símbolo (isto é, um nome de variável como x ou y),
- uma declaração do tipo símbolo = expressão (por exemplo, pch = 16),
- ou o argumento especial ... (três pontos), que permite passar múltiplos argumentos adicionais.

O corpo pode ser qualquer expressão válida em R ou um conjunto de expressões em uma ou mais linhas. Em geral, o corpo é um grupo de expressões dentro de chaves { }, com cada expressão em uma linha separada (se o corpo couber em uma linha, as chaves não são necessárias).

Funções normalmente são atribuídas a símbolos (variáveis), mas isso não é obrigatório. Esse conceito só começa a fazer sentido depois de ver vários exemplos em funcionamento.

#### 2.7.1 Média Aritmética

A média é a soma dos valores  $y_i$  dividida pela quantidade de valores n (somando ao longo do número de elementos do vetor y). Em R, n = length(y) e  $\sum y = sum(y)$ . Assim, uma função para calcular a média aritmética é:

```
media_aritmetica <- function(x) sum(x)/length(x)</pre>
```

Vamos testar a função com dados em que sabemos de antemão qual deve ser o resultado.

```
y <- c(3, 3, 4, 5, 5)
media_aritmetica(y)
```

[1] 4

#### 2.7.2 Mediana

A mediana (ou 50º percentil) é o valor do meio dos valores ordenados de um vetor de números:

```
sort(y)[ceiling(length(y)/2)]
```

Há, é claro, um pequeno problema aqui, porque se o vetor contém um número par de números, então não existe valor do meio. A lógica aqui é que precisamos calcular a média aritmética dos dois valores de y em cada lado do meio. Surge agora a questão de como sabemos, em geral, se o vetor contém um número ímpar ou par de números, de modo que possamos decidir qual dos dois métodos usar. O truque aqui é usar o módulo 2. Agora temos todas as ferramentas de que precisamos para escrever uma função geral para calcular medianas. Vamos chamar a função de mediana e defini-la assim:

```
mediana <- function(x){
  impar_par <- length(x)%2
  if(impar_par == 0) (sort(x)[length(x)/2] + sort(x)[1 + length(x)/2])/2
  else sort(x)[ceiling(length(x)/2)]
}</pre>
```

Vamos agora testar a função:

```
mediana(y)
```

[1] 4

### Exercícios

- 1. Crie sua própria função que calcula a variância, o desvio-padrão e o coeficiente de variação.
- 2. Crie uma função para listar os elementos de uma série de Fibonacci (1, 1, 2, 3, 5, 8, ...).
- 3. Crie uma função para calcular fatoriais usando o comando cumprod.
- 4. Crie sua própria função para calcular a mediana.
- 5. Crie sua própria função para calcular a média.

# 3 Motivação

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

### •

### Enuciado

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

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

### Da teoria à simulação

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

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

Logo, a probabilidade de pelo menos uma coincidência é

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

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

### Um atalho analítico útil

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

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

Aplicando ao produto,

$$\ln(1-p_r) = \sum_{i=1}^{r-1} \ln\left(1 - \frac{i}{365}\right)$$

$$\approx -\sum_{i=1}^{r-1} \frac{i}{365} = -\frac{1+2+\dots+(r-1)}{365} = -\frac{r(r-1)}{2\cdot 365}.$$

Exponentiando e isolando  $p_r$ , obtemos a aproximação

$$p_r \; \approx \; 1 - \exp\biggl\{ -\frac{r(r-1)}{730} \biggr\} \, . \label{eq:pr}$$

Essa fórmula tem três virtudes didáticas:

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

### ▲ Exemplo Rápido

• Para 23 pessoas:

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

alinhando-se ao resultado clássico de que  ${\bf 23}$  pessoas já superam 50% de chance de coincidência.

### O papel da simulação

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

Por exemplo, em  $\mathbf{R}$ :

```
r <- 23
birthdays <- sample(1:365, r, replace = TRUE)
any(duplicated(birthdays))</pre>
```

[1] TRUE

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

```
set.seed(123) #reprodutibilidade

r <- 23
B <- 10000

acertos <- 0
i <- 0

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

p_hat <- acertos / B
p_hat</pre>
```

[1] 0.5073

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

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

Agora iremos considerar o seguinte problema:



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

### **Objetivos**

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

### Hipóteses

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

#### Materiais

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

### **Exercícios**

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

## 4 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).

### **4.1** 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. Um dos métodos mais antigos e influentes para isso é o **Gerador Congruencial Linear (GCL)**.

### **Gerador Congruencial Linear**

O GCL produz uma sequência de inteiros  $x_0, x_1, x_2, \dots$  segundo a regra:

$$x_n = (a x_{n-1} + c) \mod m$$

em que, m > 0 é o **módulo**, a é o **multiplicador**, c é o **incremento** e  $x_0$  é a **semente** (seed), escolhida pelo usuário.

O resultado final é obtido normalizando os valores:

$$u_n = \frac{x_n}{m}, \quad \text{com } u_n \in (0, 1).$$

### i Tipos

- Multiplicativo: c = 0.
- Misto:  $c \neq 0$ .

### Por que o CGL funciona?

- 1. Simplicidade: apenas uma multiplicação, uma soma e uma operação de módulo.
- 2. Velocidade: pode ser implementado de forma extremamente eficiente.
- 3. Controle: dependendo da escolha de a, c, m, é possível obter um período longo, ou seja, muitos números gerados antes de a sequência se repetir.

### Critérios para bons parâmetros

- O módulo m deve ser **grande**, de preferência próximo da capacidade da máquina.
- Para GCL multiplicativo, se m for primo, existe a possibilidade de alcançar período máximo m-1.
- Valores históricos:
  - Park-Miller (1988):  $m = 2^{31} 1$ , a = 16807, c = 0.
  - Numerical Recipes (1992):  $m = 2^{32}$ , a = 1664525, c = 1013904223.

### **Exemplos**

```
# Gerador Congruencial Linear em Python

def lcg(seed, a, c, m, n=10):
    "Gera n números pseudoaleatórios normalizados em (0,1)."
    x = seed
    seq = []
    for _ in range(n):
        x = (a * x + c) % m
        seq.append(x / m)
    return seq

# Exemplo: Park-Miller (multiplicativo)
m = 2**31 - 1
a = 16807
c = 0
seed = 12345

lcg(seed, a, c, m, n=10)
```

```
# Gerador Congruencial Linear em R

lcg <- function(seed, a, c, m, n=10) {
    x <- seed
    seq <- numeric(n)
    for (i in 1:n) {
        x <- (a * x + c) %% m
        seq[i] <- x / m
    }
    seq
}

# Exemplo: Park-Miller
m <- 2^31 - 1
a <- 16807
c <- 0
seed <- 12345</pre>
```

```
lcg(seed, a, c, m, n=10)
```

- [1] 0.09661653 0.83399463 0.94770250 0.03587859 0.01154585 0.05115522
- [7] 0.76578717 0.58492974 0.91413005 0.78380039

### Visualização

Uma forma simples de verificar se um gerador se comporta bem é observar os valores em um gráfico de dispersão  $(u_n, u_{n+1})$ .

- Um bom gerador mostra pontos espalhados de forma quase aleatória.
- Um gerador ruim forma padrões visíveis (linhas ou grades).

### Limitações

Apesar de sua importância histórica, os LCGs apresentam limitações:

- O período, mesmo que longo, é finito.
- Podem apresentar correlações indesejadas em dimensões mais altas.
- Não são recomendados para aplicações de alta segurança (como criptografia).

Hoje, geradores como o Mersenne Twister substituíram o LCG em muitas linguagens (por exemplo, é o padrão no R e no NumPy). Ainda assim, o LCG é fundamental para entender os princípios da geração de números pseudoaleatórios.

#### Exercícios

- 1. Implemente um LCG (multiplicativo ou misto) em Python ou R.
- 2. Gere 1000 números pseudoaleatórios e faça:
- Um histograma para verificar se a distribuição se parece com a uniforme (0,1).
- Um gráfico de dispersão de pares consecutivos  $(u_n,u_{n+1})$ .
- 3. Compare o comportamento quando:
- Usa os parâmetros clássicos de Park-Miller  $(m=2^{31}-1, a=16807, c=0)$ .
- Usa parâmetros pequenos, por exemplo m = 17, a = 5, c = 1.

**Pergunta para reflexão:** O que acontece com a qualidade dos números gerados quando usamos parâmetros pequenos?

### i Park–Miller hoje: ainda vale a pena?

O gerador congruencial linear clássico de **Park–Miller** foi proposto em 1988, com parâmetros:

$$m = 2^{31} - 1 \approx 2,147,483,647, \quad a = 16807, \quad c = 0.$$

Na época, esses valores eram ideais para computadores de **32 bits**, embora a implementação precisasse de alguns cuidados para evitar **overflow**.

Nos computadores modernos de **64 bits**, esse problema desaparece: o produto  $a \times x$  cabe confortavelmente nos registradores, e a implementação é direta e eficiente.

### Vantagens atuais:

- Simples e rápido.
- Fácil de implementar em qualquer linguagem.
- Período máximo de mais de 2 bilhões de números.

### Limitações:

- Hoje, esse período pode ser considerado curto para simulações muito extensas.
- Geradores modernos, como o **Mersenne Twister** (período  $2^{19937} 1$ ) ou a família **PCG** (**Gerador Congruencial Permutado**), oferecem propriedades estatísticas superiores.

**Em resumo:** O Park-Miller ainda é excelente para fins didáticos e pequenas simulações, mas para aplicações científicas de grande escala recomenda-se usar geradores mais robustos.

Existem no R vários algoritmos para geradores de números pseudoaleatórios. Para ver quais são, basta:

#### help(Random)

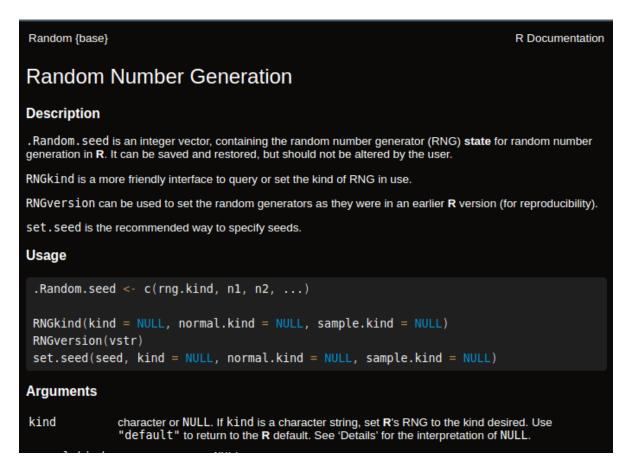


Figura 4.1: Imagem da documentação.

## 4.2 Uso de Números Aleatórios na Avaliação de Integrais

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

$$\theta = \int_{0}^{1} g(x)dx.$$

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

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

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

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

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

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

$$\theta = \int_{a}^{b} g(x)dx.$$

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

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

o que nos permite reescrevê-la como

$$\theta = \int_{a}^{b} g(x)dx = (b-a)\int_{0}^{1} g(a+(b-a)u)du.$$

Definindo

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

temos

$$\theta = \int_{0}^{1} h(u)du.$$

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

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

Agora, se nosso objetivo é calcular a integral:

$$\theta = \int_{0}^{\infty} g(x)dx.$$

Fazendo a mudança de variável

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

Logo,

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

e a integral resultante é

$$\theta = \int_{0}^{1} h(u)du,$$

com

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

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

$$\theta = \int\limits_0^1 \int\limits_0^1 \ldots \int\limits_0^1 g(x_1,x_2,\ldots,x_n) dx_1 dx_2 \ldots dx_n.$$

Observe que  $\theta$  pode ser expresso como o seguinte valor esperado:

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

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

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

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

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

#### Exemplo

Nosso objetivo é encontrar o valor aproximado da integral:  $\int_0^1 x^3 dx = 0.25$ .

```
set.seed(2025) #fixando a semente n \leftarrow 100000 #quantidade de valores uniformes gerados u \leftarrow runif(n) #numeros uniformes (0,1) mean(u^3)
```

#### [1] 0.2503221

```
import numpy as np
np.random.seed(2025)
n = 100000
u = np.random.uniform(0, 1, n)
resultado = np.mean(u**3)
print(resultado)
```

#### 0.2508972398766188

#### **Exercícios**

- 1. Se  $x_0$ e  $x_n=3x_{n-1} \, \mathrm{mod} \, 150,$ encontre  $x_1,x_2,\ldots,x_{30}.$
- 2. Se  $x_3$ e  $x_n=(5x_{n-1}+7)\,\mathrm{mod}\,200,$ encontre  $x_1,x_2,\ldots,x_{10}.$
- 3. Compare sua estimativa com o valor exato:

- $\int_0^1 \exp\{e^x\} dx.$
- $\int_0^1 (1-x^2)^{3/2} dx$ .
- $\int_{-2}^{2} \exp\{x + x^2\} dx$ .
- $\int_0^\infty x(1+x^2)^{-2}dx$ .
- $\int_{-\infty}^{\infty} \exp\{-x^2\} dx.$
- $\int_0^1 \int_0^1 \exp\{(x+y)^2\} dy dx$ .
- $\int_0^\infty \int_0^x \exp\{x+y\}dydx$ .
- 4. Use simulação para encontrar o valor aproximado de  $Cov(U,e^U)$ , em que U é uniforme em (0,1). Compare seu resultado com o valor exato.

## Referências

- 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.