GUIA PRÁTICO

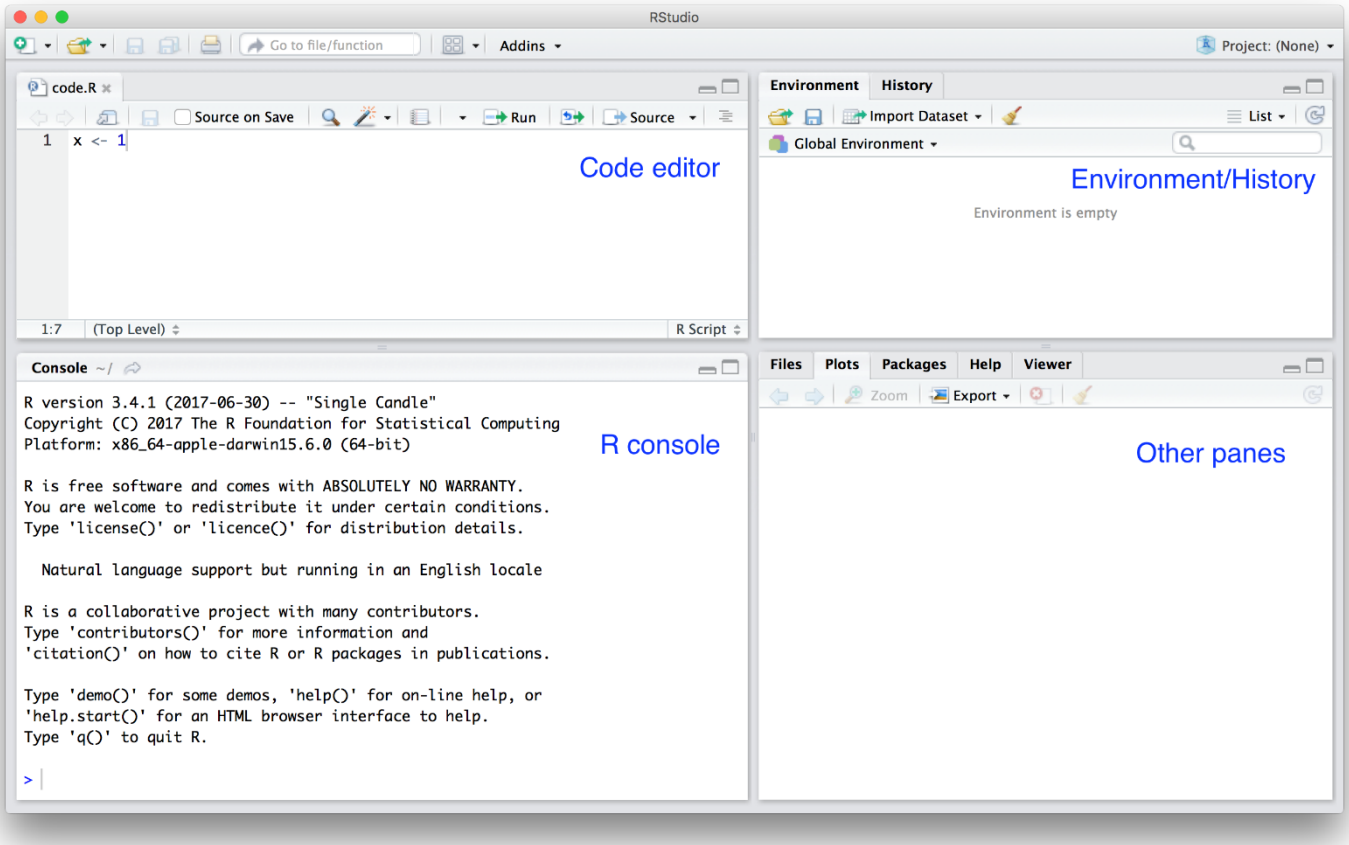
# INTRODUÇÃO AO R

UNIVERSIDADE
FEDERAL DO CEARÁ

O R não é uma linguagem de programação como C ou Java. O R não foi criado por engenheiros de software para desenvolvimento de software. Ao invés disso, foi desenvolvido por estatísticos como um ambiente interativo para análise de dados. Você pode ler a história completa no artigo A Brief History of S (Uma breve história de S)5. A interatividade é um recurso indispensável na ciência de dados porque, como você aprenderá em breve, a capacidade de explorar rapidamente os dados é necessária para o sucesso neste campo. No entanto, assim como em outras linguagens de programação, no R, você pode gravar o seu trabalho como scripts, que podem ser facilmente executados a qualquer momento. Esses scripts servem como um registro da análise que você realizou, uma peça-chave que facilita o trabalho reprodutível. Se você for um programador profissional, não espere que o R siga as convenções às quais está acostumado, pois você ficará desapontado. Se você for paciente, apreciará a grande vantagem do R quando se trata de análise e, principalmente, visualização de dados.

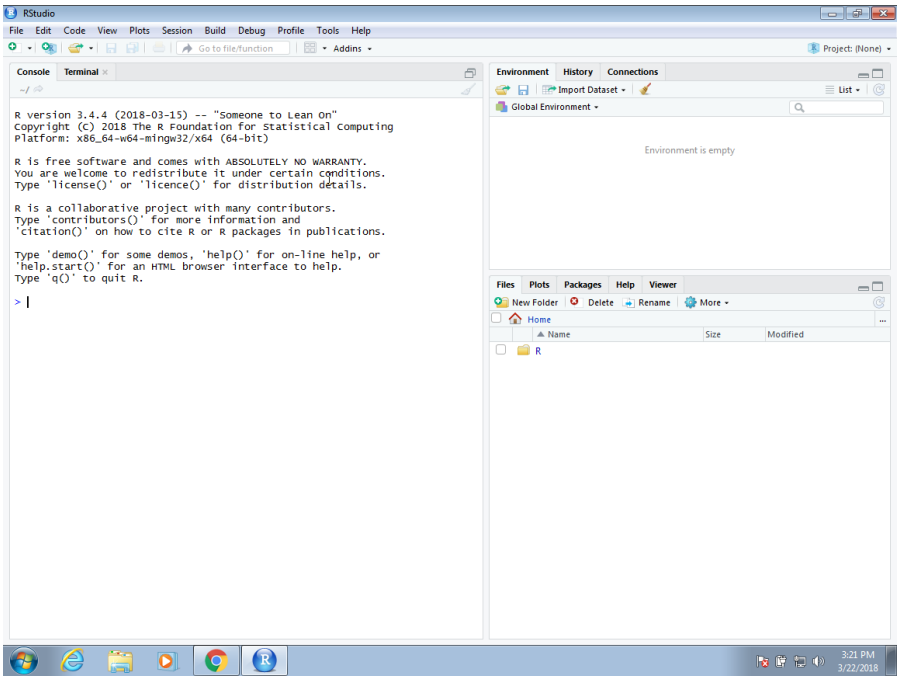
## 1. Scripts

O RStudio inclui um editor com muitos recursos específicos de R, um console para executar seu código e outros painéis úteis, incluindo um para exibir imagens. Você pode salvar seu trabalho como scripts, que podem ser editados e salvos com um editor de texto.

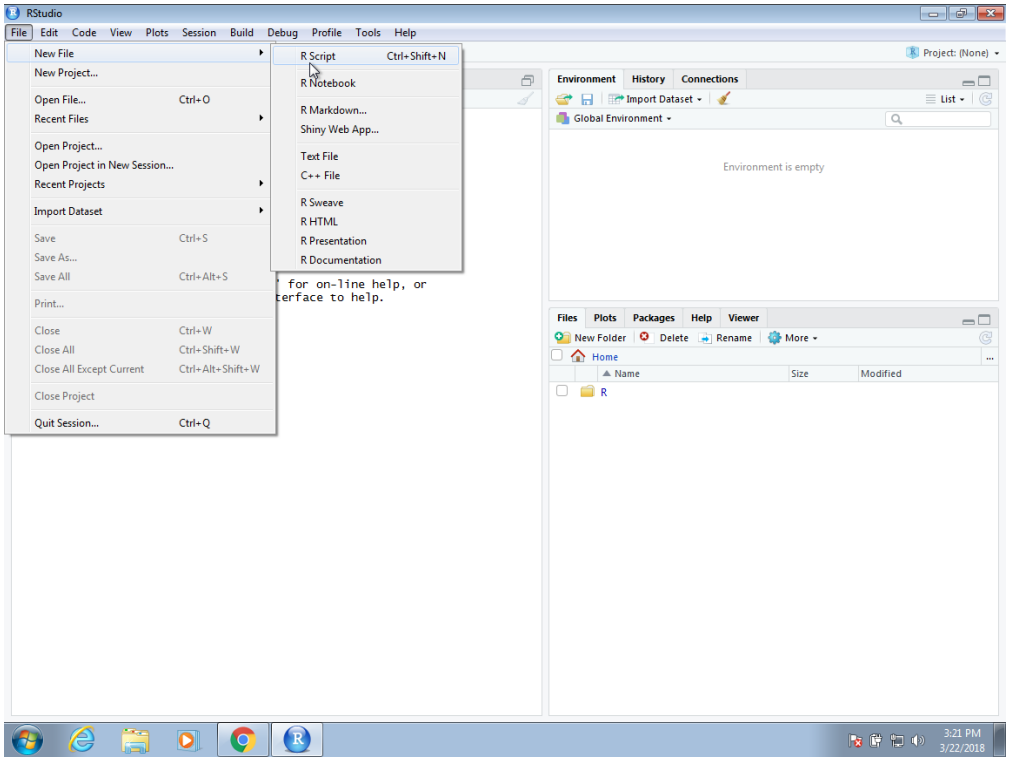


## 2. Painéis

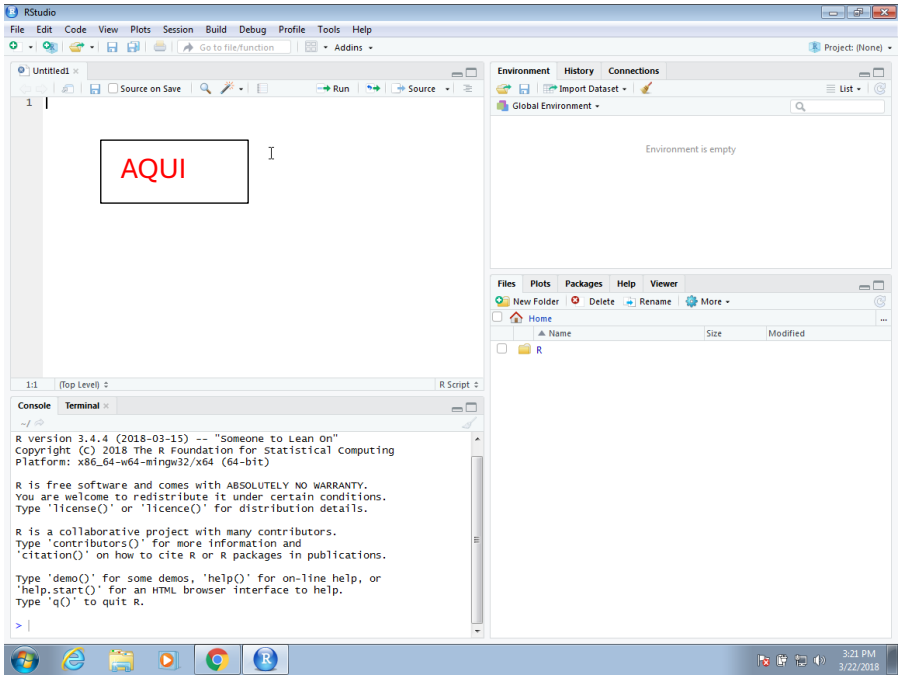
Quando iniciar o RStudio pela primeira vez, você verá três painéis. O painel esquerdo mostra o console do R. À direita, o painel superior inclui guias como Environment (ambiente) e History (histórico), enquanto o painel inferior mostra cinco guias: Files (arquivos), Plots (gráficos), Packages (pacotes), Help (ajuda) e Viewer (visualizador). Essas guias podem ser diferentes nas novas versões do RStudio. Você pode clicar em cada guia para percorrer as diferentes opções.



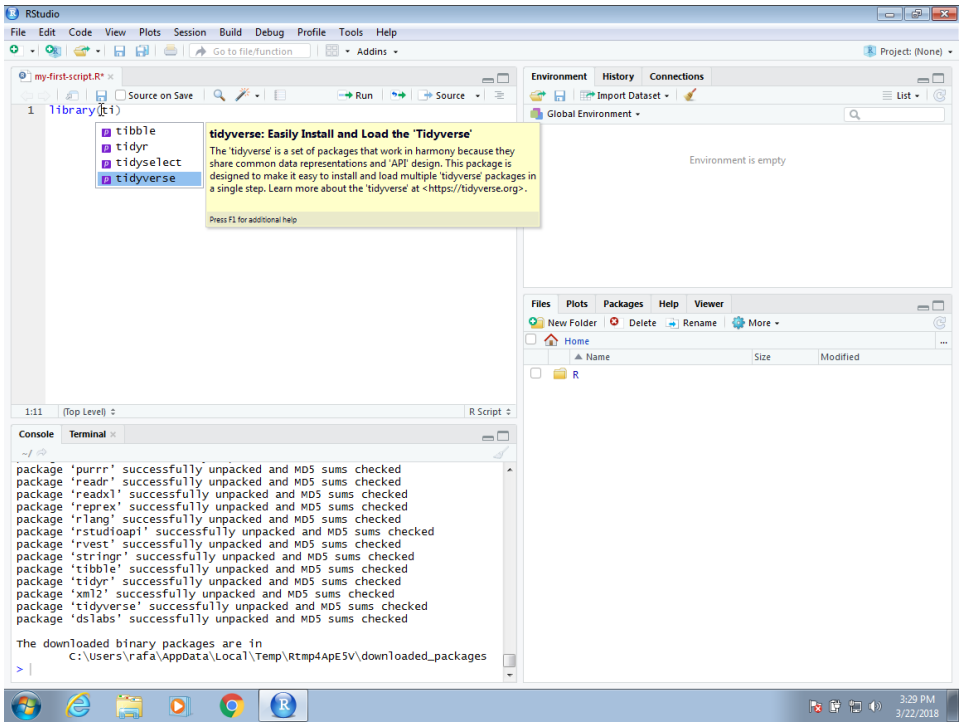

Para iniciar um novo *script*, clique em *File*, depois em *New File* e, em seguida, em *R Script*.



Isso inicia um novo painel à esquerda e é aqui que você pode começar a digitar seu *script*.




As primeiras linhas de código em um script R são dedicadas ao carregamento dos pacotes que iremos usar. Outro recurso útil do RStudio é que, uma vez que digitamos library(), o RStudio começa a auto-completar o que estamos escrevendo com as bibliotecas que instalamos.




## 3. Instalando pacotes

O R facilita a instalação de pacotes dentro do próprio R. Por exemplo, para instalar o pacote **dslabs**, você deve escrever:

```
install.packages("dslabs")
```

No RStudio, você pode navegar para a guia *Tools* e selecionar *install packages* (instalar pacotes). Em seguida, podemos carregar o pacote em nossas sessões R usando a função library:

```
library(dslabs)
```

Podemos instalar mais de um pacote por vez, fornecendo uma lista de nomes para esta função:

```
install.packages(c("tidyverse", "dslabs"))
```

Você pode ver todos os pacotes que já instalou usando a seguinte função:

```
installed.packages()
```

Nós usamos **<-** para atribuir valores a variáveis.
Também podemos atribuir valores usando = ao invés de <-, mas recomenda-se não usar **=** para evitar confusão. Ex:

```
a <- 1
b <- 1
c <- -1
```

Para ver o valor armazenado em uma variável, simplesmente pedimos que R avalie a e isso mostra o valor armazenado:
Esse valor após o #> é o resultado.

```
a
#> [1] 1
```

Uma maneira mais explícita de pedir ao R para mostrar o valor armazenado em a é usar print desta forma:

```
print(a)
#> [1] 1
```

## 4. Nomes de variáveis

Anteriormente, usamos as letras a, b e c como nomes de variáveis, mas nomes de variáveis podem ser quase qualquer coisa. Algumas regras básicas em R definem apenas que os nomes de variáveis devem começar com uma letra, não podem conter espaços e não devem ser variáveis predefinidas em R. Por exemplo, não nomeie uma de suas variáveis install.packages escrevendo algo como: install.packages <- 2.

Uma boa convenção a seguir é usar palavras significativas que descrevam o que é armazenado, use apenas letras minúsculas e sublinhados como substitutos de espaços. Para equações quadráticas, podemos usar algo como:

```
solucao_1 <- (-b + sqrt(b^2 - 4*a*c))/ (2*a)
solucao_2 <- (-b - sqrt(b^2 - 4*a*c))/ (2*a)
```

Para comentar uma linha de código basta utilizar o símbolo “#” e a linha não será executada.

## 5. Tipos de dados

Variáveis em R podem ser de tipos diferentes. Por exemplo, precisamos distinguir números de sequências de caracteres, e tabelas de simples lista de números. A função class nos ajuda a determinar que tipo de objeto temos:

```
a <- 2
class(a)
#> [1] "numeric"
```

## 6. Data Frames

Até agora, as variáveis que definimos são apenas numéricas. Isso não é muito útil para armazenar dados. A maneira mais comum de armazenar um conjunto de dados em R é usar um *data frame*. Conceitualmente, podemos pensar em um *data frame* como uma tabela com linhas que representam observações e com colunas que representam as diferentes variáveis coletadas para cada observação. *Data frames* são particularmente úteis para *datasets* pois permitem combinar diferentes tipos de dados em um único objeto.

Uma grande proporção dos desafios da análise de dados começa com os dados armazenados em um *data frame*. Por exemplo, armazenamos os dados do nosso exemplo motivador em um *data frame*. Você pode acessar esse *dataset* carregando a biblioteca **dslabs** e, em seguida, usando a função `data` carregar o *dataset* `murders`:

```
library(dslabs)
data(murders)
```

Para verificar se esse objeto é um data frame utilizamos a função class:

```
class(murders)
#> [1] "data.frame"
```

Podemos exibir as seis primeiras linhas usando a função head:

```
head(murders)
#>        state abb region population total
#> 1    Alabama  AL  South    4779736   135
#> 2     Alaska  AK   West     710231    19
#> 3    Arizona  AZ   West    6392017   232
#> 4   Arkansas  AR  South    2915918    93
#> 5 California  CA   West   37253956  1257
#> 6   Colorado  CO   West    5029196    65
```

## 7. O operador de acesso $

Muitas vezes precisaremos acessar as diferentes variáveis representadas pelas colunas incluídas neste data frame. Para fazer isso, usamos o operador de acesso $ da seguinte maneira:

```
murders$population
#>  [1]  4779736   710231  6392017  2915918 37253956  5029196
3574097
#>  [8]   897934   601723 19687653  9920000  1360301  1567582
12830632
#> [15]  6483802  3046355  2853118  4339367  4533372  1328361
5773552
#> [22]  6547629  9883640  5303925  2967297  5988927   989415
1826341
#> [29]  2700551  1316470  8791894  2059179 19378102  9535483
672591
#> [36] 11536504  3751351  3831074 12702379  1052567  4625364
814180
#> [43]  6346105 25145561  2763885   625741  8001024  6724540
1852994
#> [50]  5686986   563626
```

## 8. Criando vetores

Podemos criar vetores usando a função c, que significa *concatenate* (concatenar). Nós usamos c para concatenar entradas da seguinte maneira:

```
codes <- c(380, 124, 818)
codes
#> [1] 380 124 818
```

Também podemos criar vetores de caracteres. Usamos aspas para indicar que as entradas são caracteres e não nomes de variáveis.

```
country <- c("italy", "canada", "egypt")
```

## 9. Algumas operações:

```
2+2 # Soma
[1] 4
8-3 # Subtração
[1] 5
3*8 # Multiplicação
[1] 24
8/2 # Divisão
[1] 4
2^8 # Potências
[1] 256
(2+4)/7 # Prioridade de solução
[1] 0.8571429
```

## 10. Ajuda

A função mais importante do R é a função *help* ou *?*. Com essa função é possível encontrar o arquivo de ajuda das funções. Esse arquivo geralmente contêm uma descrição do que a função exatamente faz, dos argumentos necessários, descrição dos resultados, referências e exemplos.

```
help(log) # Abre a ajuda da função log, ou então...
?log # ... também abre a ajuda
```

Os principais itens do arquivo de ajuda da função são:

- Description - Apresenta um resumo sobre o que a função faz.
- Usage - Mostra todos os argumentos, ordem e as opções preestabelecidas de cada um.
- Arguments - Explica cada argumento.
- Details - Detalhes sobre o uso da função, métodos e aplicação.
- Value - Explica cada um dos os resultados.
- References – Referências dos métodos.
- See also - Funções relacionadas.
- Examples - Mostra alguns exemplos que podem ser executados.

Os exemplos são bastante úteis pois mostram com as funções podem ser executadas. Geralmente eles funcionam com conjuntos de dados que já estão disponíveis dentro do R. Como isso, é possível verificar o funcionamento das funções antes mesmo de organizar seu conjunto de dados.

## 11. Guia de ajuda rápida

```
# - Adicionar comentário
? - Obter ajuda de função
?? - Relizar buscar
<- ou = - Atribuir objeto (direita para esquerda)
-> - Atribuir objeto (esquerda para direita)
+ - Somar
- - Subtrair
* - Multiplicar
/ - Dividir
^ - Potêncializar (direita para esquerda)
%*% - Multiplicar matrizes
< - Comparar, menor
> - Comparar, maior
<= - Comparar, menor ou igual
>= - Comparar, maior ou igual
== - Comparar, exatamente igual
!= - Comparar, diferente
! - Lógico, NÃO. Inverter resultado de teste lógico
& - Lógico, critério aditivo E. Operação elementar
```

| - Lógico, critério aditivo OU. Operação elementar

&& - Lógico, E

|| - Lógico, OU

~ - Fórmula estatística

FALSE ou F - Argumento lógico falso

TRUE ou T - Argumento lógico verdadeiro

NA - Indeterminado (Not Avaiable)

NaN - Indeterminado (Not a Number)

Inf - Infinito

NULL - Objeto nulo

c() - Concatenar valores em vetor

factor() - Criar fator

ordered() - Criar fator ordenado

data.frame() - Criar tabela de dados (data.frames)

matrix() - Criar matriz

list() - Criar lista

rbind() - Combinar vetores por linhas

cbind() - Combinar vetores por colunas

paste() - Concatenar caracteres em sequências regulares

class() - Conferir/Atribuir classe do objeto

str() - Conferir estrutura do objeto

: - Gerar sequência numérica contínua

$ - Indexar vetores/listas pelo nome das variáveis/listas

@ - Indexar na classe S4

[] - Indexar vetores/listas

[ , , drop = FALSE] - Indexar data.frames/matrizes. Primeiro valor linha, segundo colun
a

[[]] - Indexar listas

names() - Conferir/Atribuir nomes a vetores/listas

colnames() - Conferir/Atribuir nomes as linhas de data.frames/matrizes

rownames() - Conferir/Atribuir nomes as colunas de data.frames/matrizes

length() - Conferir comprimento de vetores/listas

dim() ou nrow() e ncol() - Conferir dimensões de data.frames/matrizes

head() e tail() - Conferir início e fim de data.frames/matrizes

t() - Transpor data.frames/matrizes

seq() - Obter sequência regular

rep() - Repetir valores

ifelse() - Aplicar teste condicional

read.csv() ou read.table() - Importar tabelas

write.csv() ou write.table() - Exportar tabelas

```
ls() - Listar objetos da área de trabalho
rm() - Remover objetos da área de trabalho
save.image() - Salvar área de trabalho
load() - Carregar área de trabalho
max() - Obter máximo
min() - Obter mínimo
sum() - Obter soma
sqrt() - Obter raiz quadrada
log() - Obter logaritmo
exp() - Obter função exponential
abs() - Obter valores absoluto
round() - Arredondar valores
factorial() - Obter fatorial
mean() - Obter média
median() - Obter mediana
var() - Obter variância
sd() - Obter desvio padrão
cor() - Obter correlação
cov() - Obter covariação
runif() - Gerar distribuição uniforme
rnorm() - Gerar distribução normal
rpois() - Gerar distribuição de Poisson
sample() - Amostrar valores
sort() - Ordenar valores
order() - Obter posição da order dos valores
which() - Procurar posição (índice) conforme teste lógico
sapply() - Aplicar função em data.frames/listas
tapply() - Aplicar função em vetores
apply() - Aplicar função em data.frames/matrizes
replicate() - Replicar expressões
table() - Produzir tabelas de contingência
```

# REFERÊNCIAS

**Introdução ao R**. Disponível em: <https://vanderleidebastiani.github.io/tutoriais/Introducao_ao_R.html>.

IRIZARRY, R. A. **Capítulo 1 Introdução ao R e RStudio | Introdução à Ciência de Dados**. Disponível em: <https://rafalab.dfci.harvard.edu/dslivro/getting-started.html>. Acesso em: 23 out. 2024.