Introdução ao R

Willber Nascimento

2022-06-19

Aula 1

Apresentação básica do R e RStudio

Gerenciamento de projetos com RStudio

- Caso você se interesse por isso veja o minicurso de gerenciamento de projetos com R e Rstudio https://odysee.com/@willbernascimento:8/Git-Github-e-RStudio-Gerenciamento-de-projetos-de-analise-de-dados

Diretórios e caminhos

No geral, nossa utilização do R envolve sua linguagem e a manipulação de arquivos. Esses arquivos, frequentemente estão no nosso computador e estão organizados de uma forma especifica. Seja para carregar dados ou exportá-lo é importante entender como o R pode ser usado para a tarefa.

Para se localizar você pode usar o comando `getwd()`.

```
getwd()
```

```
## [1] "C:/Users/Willber/Documents/introducaoR/aulas"
```

Para mudar de localização você pode usar `setwd()`

```
setwd('meu/novo/endereco')
```

Contudo, quase nunca seria necesário mudar de localização, sendo preferível movimentar-se entre diretórios usando os caminhos relativos - mas mantendo a sua sessão R no mesmo local de partida.

- Caminhos Absolutos

Os caminhos absolutos envolvem todo o endereço de um determinando arquivo ou diretório no seu computador. E todos eles começam a partir da raiz de diretórios do sistema operacional. No Windows é o 'C:', no Linux ou Mac é '/'. Você pode pensar esse caminho absoluto como o endereço formal de uma casa (com nome da rua, cep, etc). Se uma pessoa não conhece esses dados, eles dificilmente encontrarão a casa. Abaixo vejamos diferentes caminhos absolutos no computador.

```
"C:\\Users\\Willber\\Videos"

"C:/Users/Willber/Downloads"

"C:\\Users\\Willber Documents"
```

- Caminhos Relativos

Indicam uma localização com base em um diretório especifico. Esse diretório em questão serve como a nova raiz, ou o ponto de partida. Imaginando o exemplo de uma casa, um endereço relativo poderia ser, a partir de sua posição atual, "a terceira casa depois da padaria".

```
"\\Willber\\Videos"

"\\Downloads\\dados\\ufpe"

"\\curso"
```

Note que no Windows, as barras invertidas precisam ser duplicadas ou então usar o padrão 'Linux/Mac' de barras para frente - o que eu sugiro.

Gerenciamento de pacotes

Tudo que fazemos no R gira em torno de pacotes, sejam eles o que já vem com a instalação padrão ou aqueles que precisarão ser instalados sob demanda. Desse modo, é importante entendermos o mínimo sobre como gerenciá-los.

Há pelo menos duas formas de instalar novos pacotes no R. A primeira, e mais comum, é através da função (ou comando) `install.packages`.

```
install.packages("ggplot2")
```

Esse comando vai receber o nome do pacote que você quer instalar, entre aspas. Esse comando irá buscar esse pacote nos servidores do CRAN e, se ele existir, instalar em seu computador. Mais especificamente, ele será instalado na biblioteca do R em seu computador. Contudo, para utilizar os comandos ou códigos disponíveis nesse novo pacote, será necessário carregá-los na memória do seu computador. O comando seguinte carrega os comandos disponíveis no pacote `ggplot2` na sua sessão atual.

```
library(ggplot2)
```

Esse comando chama o pacote para sua sessão atual e aumenta a quantidade de códigos prontos disponíveis. Agora você pode usar essas ferramentas para fazer um gráfico.

```
ggplot(data.frame(rnorm(300)), aes(rnorm.300.))+ geom_histogram()
```

Se você descarregar o pacote e pedir o gráfico novamente o resultado será outro. Tente você mesmo.

```
detach("package:ggplot2", unload = TRUE)
ggplot(data.frame(rnorm(300)), aes(rnorm.300.))+ geom_histogram()
```

Há outras formas de instalar pacotes, manualmente e pacotes do github. Que veremos posteriormente .

R como calculadora

A maneira mais simples de usar o R é como se ele fosse uma calculadora. Digite os exemplos abaixo para se familiarizar.

```
1+1
```

```
## [1] 2
```

```
30-10
```

```
## [1] 20
```

```
2*4
```

```
## [1] 8
```

```
66/4
```

```
## [1] 16.5
```

```
4^4
```

```
## [1] 256
```

Vejamos uma operação simples com um número decimal.

```
1,3 + 1
```

```
## Error: <text>:1:2: ',' inesperado
## 1: 1,
##      ^
```

O padrão de decimal no R (e em praticamente toda linguagem de programação) é o `.`. A vírgula é usada para separar argumentos em funções. Então neste contexto não é uma sintaxe válida.

Lembre-se de seguir corretamente a ordem das operações.

```
2 * 2 + 1
```

```
## [1] 5
```

```
2 * (2 + 1)
```

```
## [1] 6
```

Objetos e Variáveis

Ojetos são úteis para guardar valores que vamos usar novamente no futuro. As vezes, esse objeto pode ser chamado de variável também. Imagine que você precisa, várias vezes, converter reais para dólares. Hoje, o dolár está em R$ 5,18. Se você tem R$ 250,00, quantos doláres você terá na conversão?

```r
250.00/ 5.18
```

```
## [1] 48.26255
```

Você pode salvar esse valor em uma variável e o utilizar diversas vezes.

```r
dolar <- 5.18

# a mesma operacao
235.00 / dolar
```

```
## [1] 45.3668
```

Quantos reais você terá com $ 50 dólares?

```r
dolar * 50
```

```
## [1] 259
```

O objeto dolar representa aqui um valor de dolar em um dia específico, que pode variar no dia seguinte, logo ele também pode ser pensado como uma variável no contexto da programação.

Tipos básicos

Números

Todo objeto no R, possui um tipo específico. No R os números são alocados frequentemente em doubles ou inteiros.

```r
x <- 10
typeof(x)
```

```
## [1] "double"
```

```r
y <- 10.0
typeof(y)
```

```
## [1] "double"
```

```r
z <- 10L
typeof(z)
```

```
## [1] "integer"
```

```r
ci <- 1e2
typeof(ci)
```

```
## [1] "double"
```

No R nós podemos averiguar a classe de um objeto usando a função `class`.

```
class(x)
```

```
## [1] "numeric"
```

```
class(y)
```

```
## [1] "numeric"
```

```
class(z)
```

```
## [1] "integer"
```

```
class(1e2)
```

```
## [1] "numeric"
```

Texto/Carácteres

A entrada de texto no R exige que a informação esteja entre aspas, seja elas duplas ou não.

```
duplas <- "sabão"
typeof(duplas)
```

```
## [1] "character"
```

```
simples <- 'sabão'
typeof(simples)
```

```
## [1] "character"
```

```
numero <- "10"
typeof(numero)
```

```
## [1] "character"
```

```
class(numero)
```

```
## [1] "character"
```

Cuidado ao usar caracteres que possuem algum tipo de aspa literal. Nesses casos é necessário usar aspas diferentes daquelas literais no texto.

```
texto <- 'dinner's'
```

```
## Error: <text>:1:18: unexpected symbol
## 1: texto <- 'dinner's
##                     ^
```

Usamos aspas duplas quando o texto apresenta aspas simples e vice versa.

```
texto <- "dinner's"
```

Lógico

Para operações lógicas o R possui um tipo específico, `logical`.

```
verdadeiro <- TRUE
typeof(verdadeiro)
```

```
## [1] "logical"
```

```
falso <- FALSE
typeof(falso)
```

```
## [1] "logical"
```

Podemos abreviar para `T` e `F` respectivamente.

Operações Lógicas

O R salva informações lógicas como um tipo booleano: `TRUE`para verdadeiro e `TRUE` para falso. Esse tipo de variável é criado automaticamente quando você faz algum tipo de comparação, por exemplo:

```
10 > 5
```

```
## [1] TRUE
```

Agora, salve essa mesma operação em uma variável e a execute no console:

```
maior <- 10 > 5
maior
```

```
## [1] TRUE
```

Experimente essa linha de código:

```
maior <- 10 > 5
maior == (5 > 5)
```

```
## [1] FALSE
```

O código compara o valor de `maior` com o resultado da comparação `(5 > 5)`. Em português: `TRUE` é igual à cinco é maior que cinco?; isto é, `TRUE` é igual à `FALSE`? Você pode ver os operadóres lógicos do R, na tabela abaixo.

Operador	Descrição
==	É igual à

Operador	Descrição
!=	É diferente de
&	E (and)
|	Ou (or)

Verificação e coerção de tipos

NO R podemos verificar um tipo de um objeto com em um conjunto de funções específicas que iniciam com o prefixo `is.`.

```
numero <- "10"
is.numeric(numero)
```

```
## [1] FALSE
```

```
is.logical(T)
```

```
## [1] TRUE
```

```
is.character('texto')
```

```
## [1] TRUE
```

Como vemos, o retorno dessas funções é sempre do tipo lógico. Mais à frente podemos tirar vantagem dessa característica na manipulação de dados.

Além disso, as funções com prefixo `as.` podem ser usadas para transformar o tipo de dados de um objeto. Essa transformação é em si uma coerção de tipos.

```
numero <- '10'
typeof(numero)
```

```
## [1] "character"
```

```
numero <- '10'

numero <- as.numeric(numero)
typeof(numero)
```

```
## [1] "double"
```

```
numero
```

```
## [1] 10
```

Podemos fazer o contrário, transformar um objeto de texto em número.

```r
numero <- 12345
typeof(numero)
```

```
## [1] "double"
```

```r
texto <- as.character(numero)
typeof(texto)
```

```
## [1] "character"
```

Tome cuidado com essas operações de coerção visto que muitas vezes não faz sentido a transformação do dados e isso poderá resultar em perdas de dados.

```r
a1 <- as.numeric('a')
```

```
## Warning: NAs introduzidos por coerção
```

```r
a2 <- as.numeric('3')
a1
```

```
## [1] NA
```

```r
a2
```

```
## [1] 3
```

Como pedir ajuda?

Todo o ecossistema do R conta com uma ampla documentação. Você pode consultá-la direto do RStudio com `? alguma coisa`.

```r
?is.logical
```

```
## starting httpd help server ... done
```

```r
?as.numeric
```

Porém, nem sempre nossa dúvida será tirada consultando a documentação ofical. Muitas vezes vamos precisar consultar os universitários. Eles estão no StackOverflow. Lá, centenas de dúvidas e erros comuns já possuem resposta e você pode se juntar a comunidade para fazer consultas e para ajudar nas respostas. Normalmente, usamos o Google para encontrar as respostas que precisamos usando `como fazer isso aqui in r`. É importante usar a pesquisa em inglês visto que a grande maioria de soluções estará nessa linguagem.