R para Economia

Lucas Mendes

24/03/2020

O que veremos nesse curso

O que veremos nesse curso

- Introdução ao R

O que veremos nesse curso

- Introdução ao R
- Manipulação de dados

O que veremos nesse curso

- Introdução ao R
- Manipulação de dados
- Visualização de dados

O que veremos nesse curso

- Introdução ao R
- Manipulação de dados
- Visualização de dados
- Regressão Linear e Logística

O que veremos nesse curso

- Introdução ao R
- Manipulação de dados
- Visualização de dados
- Regressão Linear e Logística
- Séries Temporais

O que veremos nesse curso

- Introdução ao R
- Manipulação de dados
- Visualização de dados
- Regressão Linear e Logística
- Séries Temporais
- Relatórios

Avisos

Avisos

- R não é excel

Avisos

- R não é excel
- Em um primeiro momento pode não fazer sentido

Avisos

- R não é excel
- Em um primeiro momento pode não fazer sentido
- Você vai errar muito

Avisos

- R não é excel
- Em um primeiro momento pode não fazer sentido
- Você vai errar muito
- Muito mesmo

Avisos

- R não é excel
- Em um primeiro momento pode não fazer sentido
- Você vai errar muito
- Muito mesmo
- Busque Ajuda

Quem irá te ajudar?

Quem irá te ajudar?

- Karmendes

Quem irá te ajudar?

- Karmendes
- Se a aula for presencial, seu colega ao lado

Quem irá te ajudar?

- Karmendes
- Se a aula for presencial, seu colega ao lado
- **Stackoverflow**

Quem irá te ajudar?

- Karmendes
- Se a aula for presencial, seu colega ao lado
- **Stackoverflow**
- Páginas aleatórias do google sobre R

Quem irá te ajudar?

- Karmendes
- Se a aula for presencial, seu colega ao lado
- **Stackoverflow**
- Páginas aleatórias do google sobre R
- Blogs de R

Quem irá te ajudar?

- Karmendes
- Se a aula for presencial, seu colega ao lado
- **Stackoverflow**
- Páginas aleatórias do google sobre R
- Blogs de R
- A bíblia do R

Introdução ao R

O que veremos agora?

- Operadores

O que veremos agora?

- Operadores
- Objetos Atômicos

O que veremos agora?

- Operadores
- Objetos Atômicos
- Vetores

O que veremos agora?

- Operadores
- Objetos Atômicos
- Vetores
- Matrizes

O que veremos agora?

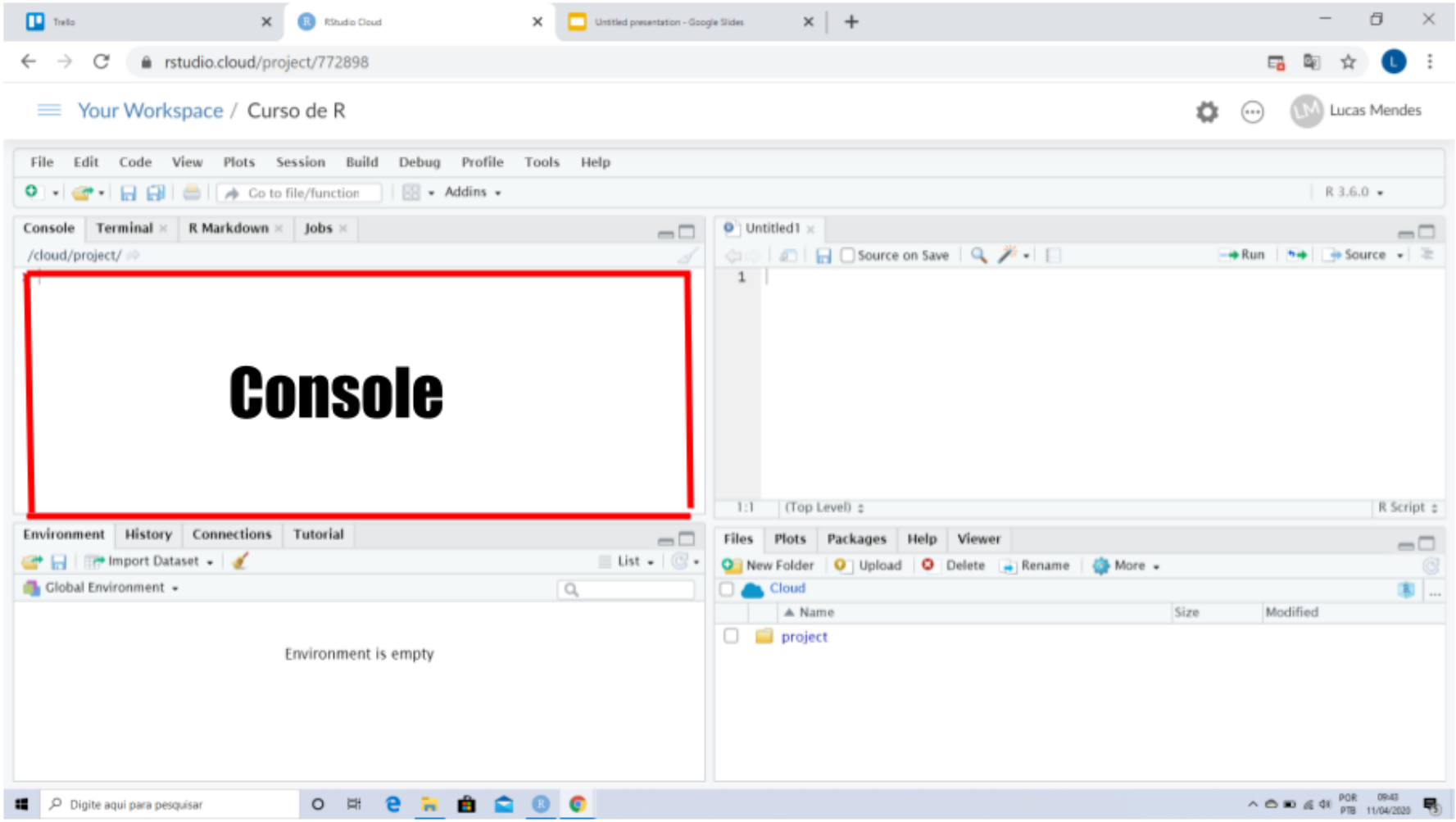
- Operadores
- Objetos Atômicos
- Vetores
- Matrizes
- Data Frames

O que veremos agora?

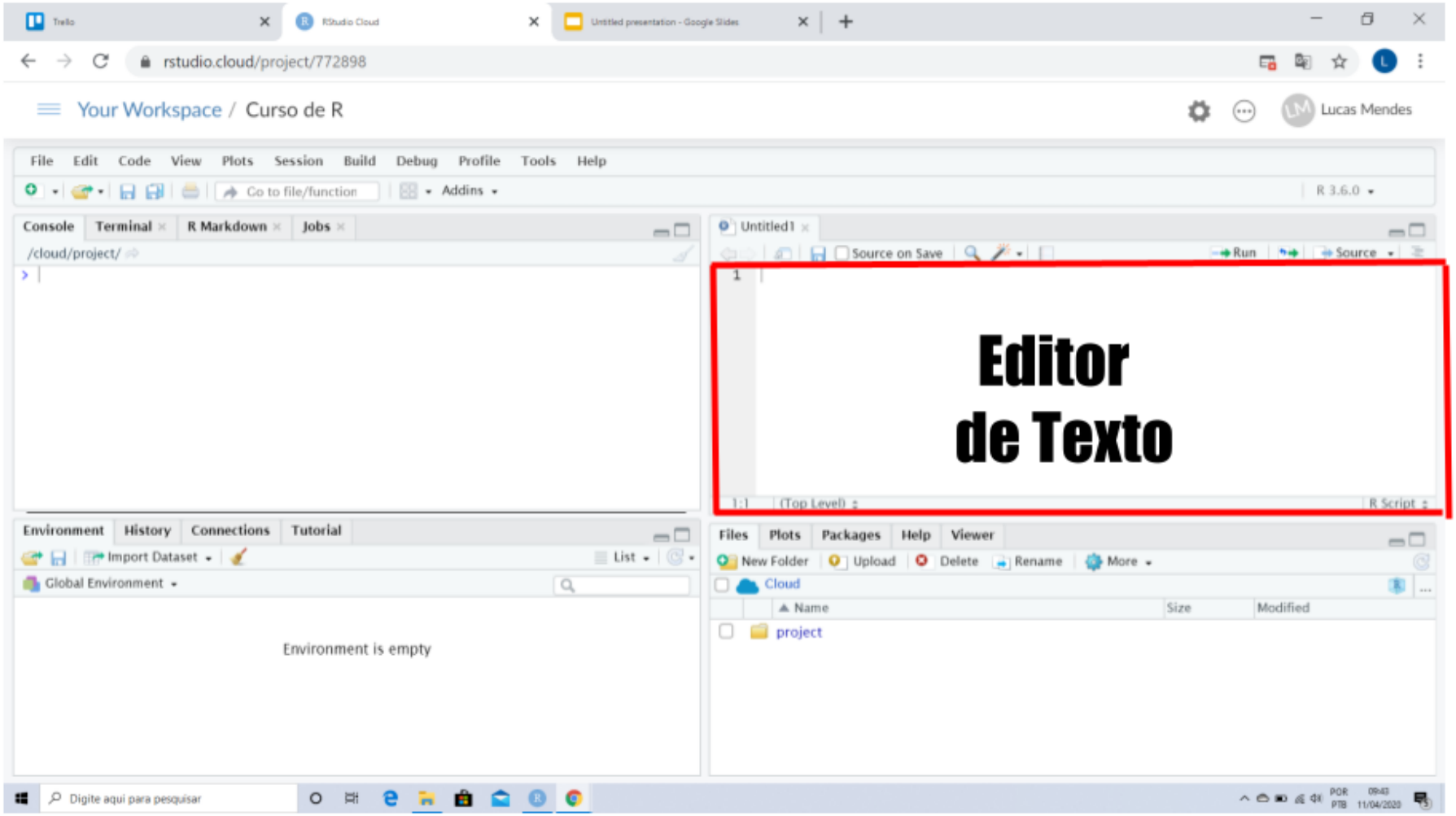
- Operadores
- Objetos Atômicos
- Vetores
- Matrizes
- Data Frames
- Listas

Layout do Rstudio

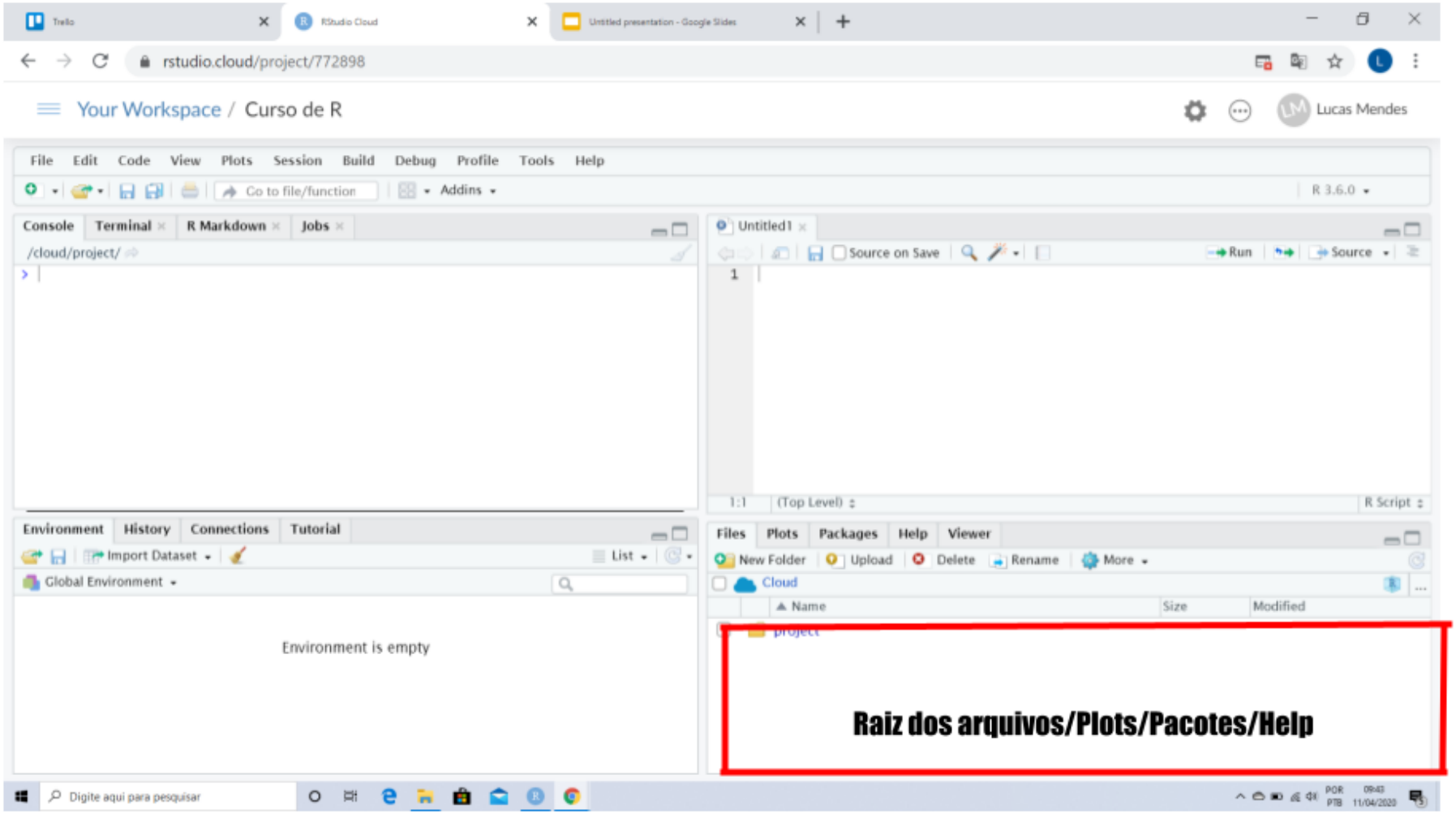
Layout do Rstudio



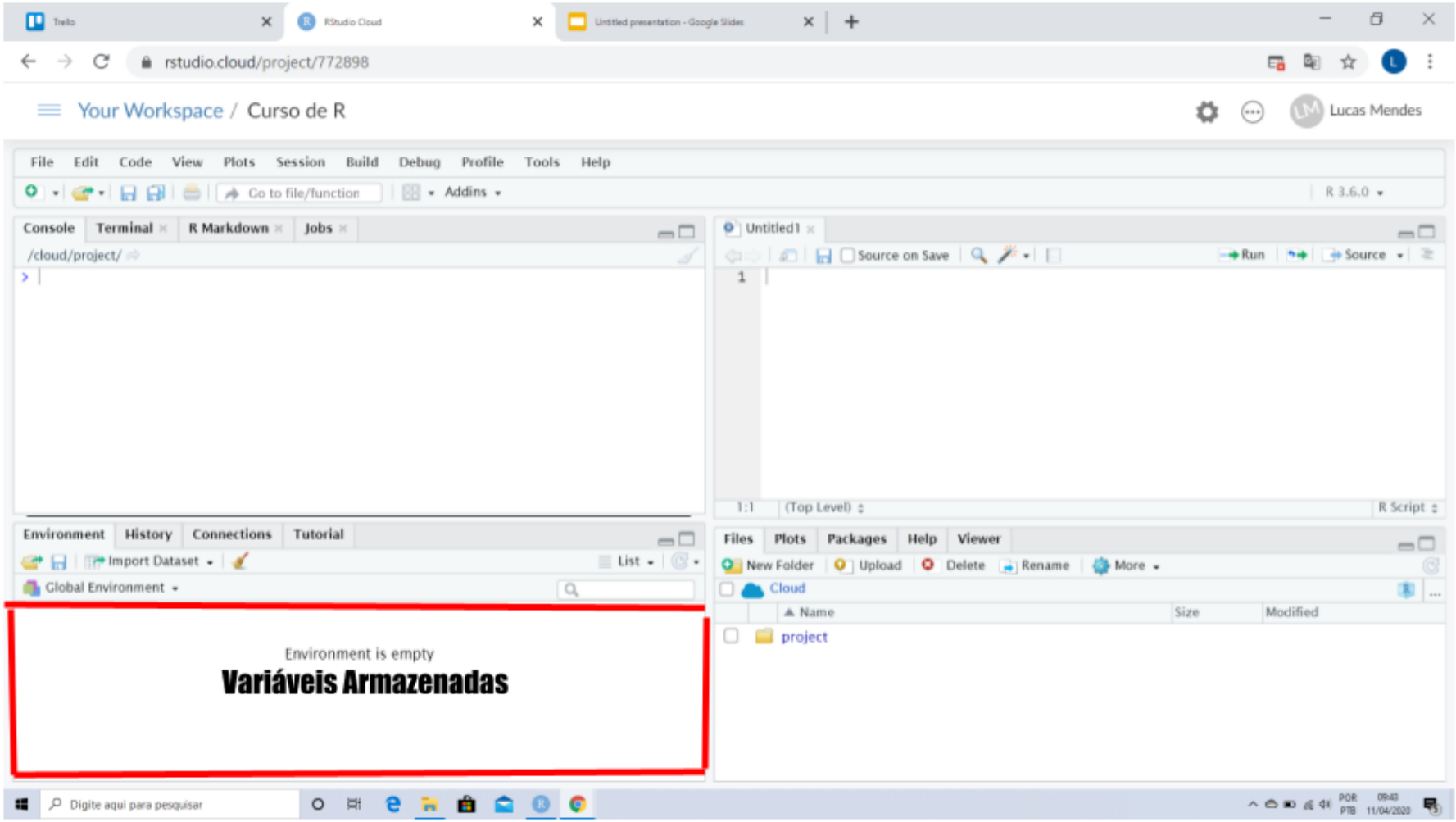
Layout do Rstudio



Layout do Rstudio



Layout do Rstudio



Disclaimer

Disclaimer

Nós temos que diferenciar duas coisas nesse curso:

Variáveis != Objetos

Variável = Serve para armazenar objetos. Objeto = Coisas que manipulamos no R (Vetor, Matriz, Tabelas, Gráficos. . .)

Regra da criação de variáveis

1° Não começa com Números: **1x** não é possivel. **x1** sim

2° Há palavras que não podem se tornar variaveis: **TRUE** e **FALSE** por exemplo.

3° Há diferenciação de minúscula para maiúscula: **X** != **x**

Operadores

Operadores

Nós iremos abordar dois tipos de operadores:

- Os operadores de atribuição

Operadores

Nós iremos abordar dois tipos de operadores:

- Os operadores de atribuição
- Os operadores relacionais

Operadores

Nós iremos abordar dois tipos de operadores:

- Os operadores de atribuição
- Os operadores relacionais

- **Operadores de atribuição**: Atribuem um valor ou objeto a uma variável

Operadores

Nós iremos abordar dois tipos de operadores:

- Os operadores de atribuição
- Os operadores relacionais

- **Operadores de atribuição**: Atribuem um valor ou objeto a uma variável
- **Operadores relacionais**: Perguntam se dada afirmação é *Falsa* ou *Verdadeira*

Operador de atribuição

Podemos usar tanto o sinal = como o <-, sendo mais comum o último.

Exemplo 1

```
x = 10
```

Exemplo 2

```
x <- 12
```

O que está ocorrendo

Quando você roda uma linha de código, como :

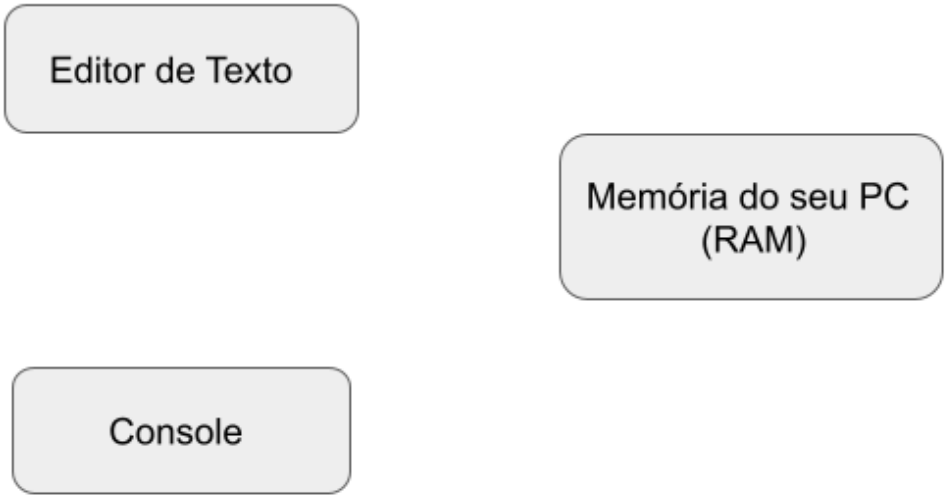
```
x <- 10
```

O que acontece é que o computador aloca na memoria dele um espaço para armazenar a variavel `x` que consequentemente tem o valor 10.

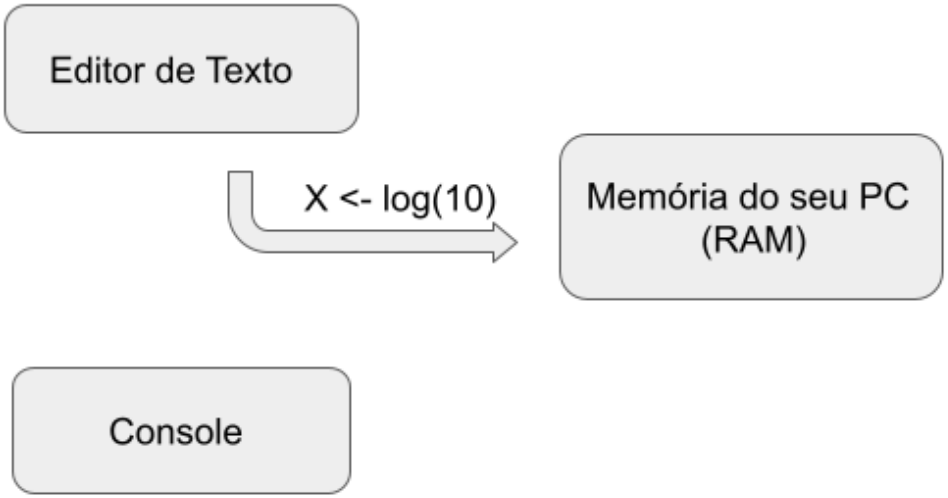
Se você passar qualquer outro valor para `x` e rodar o código, o computador ira tirar da memória o valor anterior e gravar o novo.

Em imagens

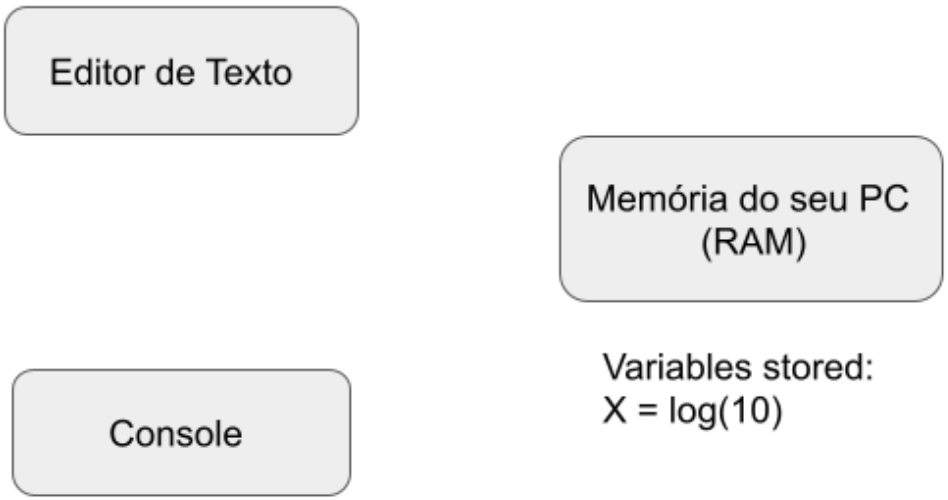
Gravando uma variavel



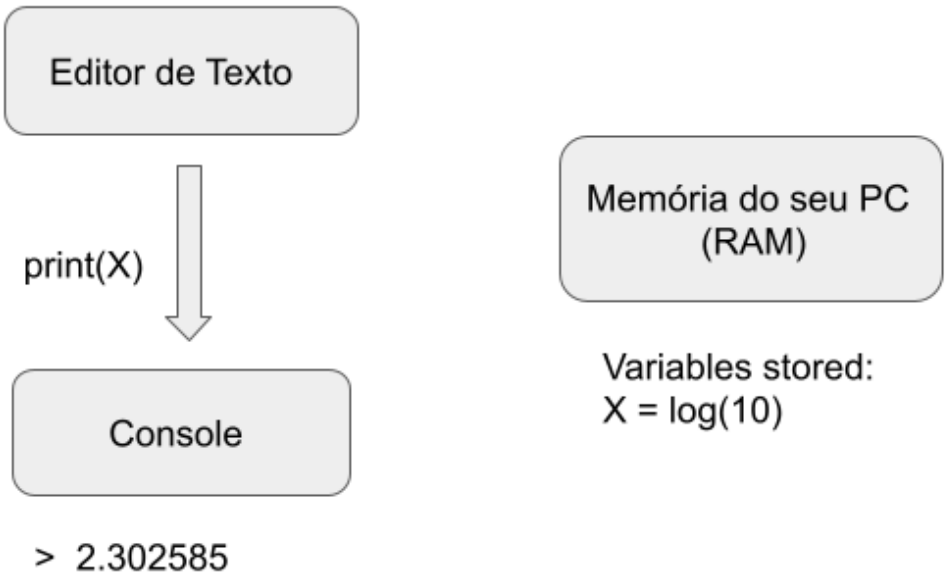
Gravando uma variavel



Gravando uma variavel

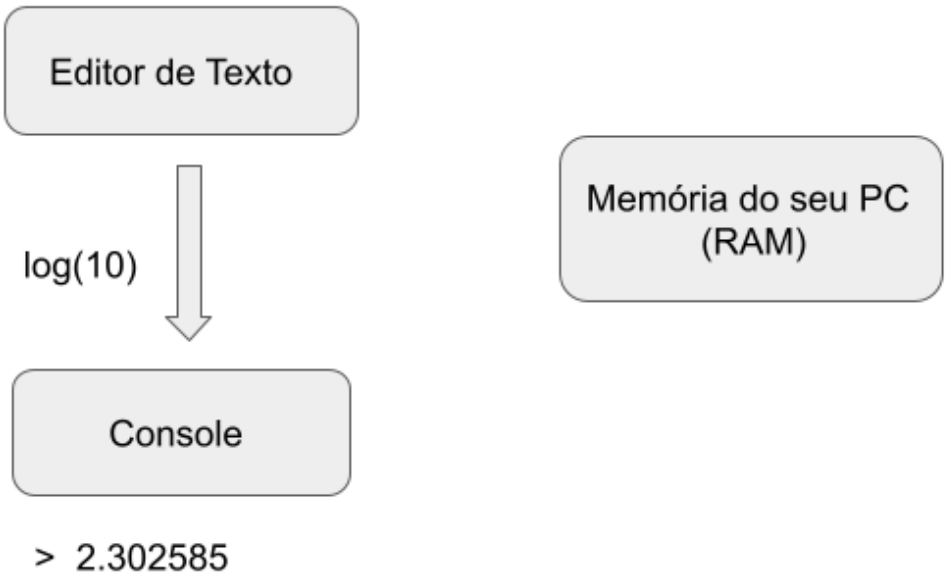


Gravando uma variavel

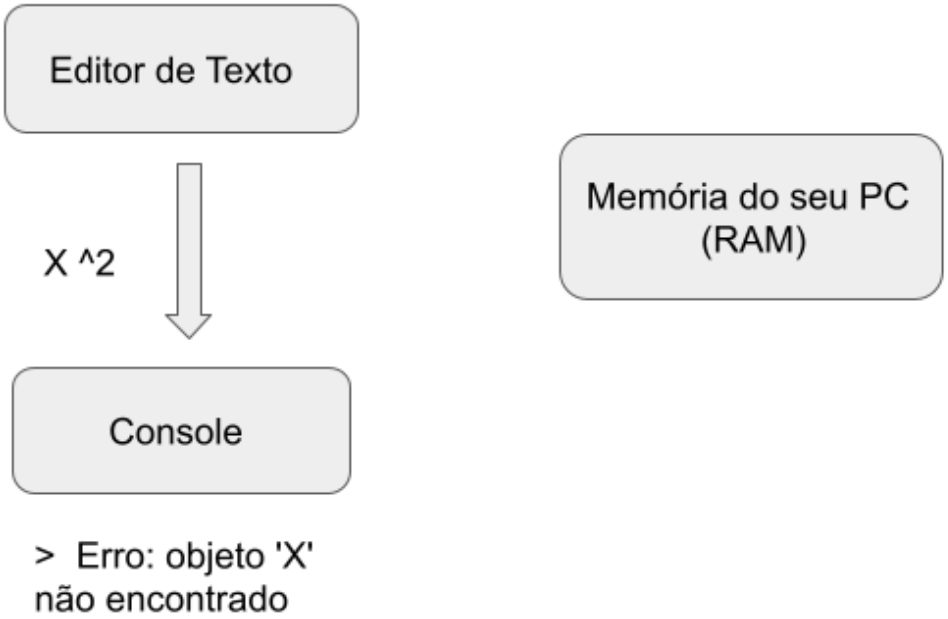


Modo Errado

Gravando uma variavel

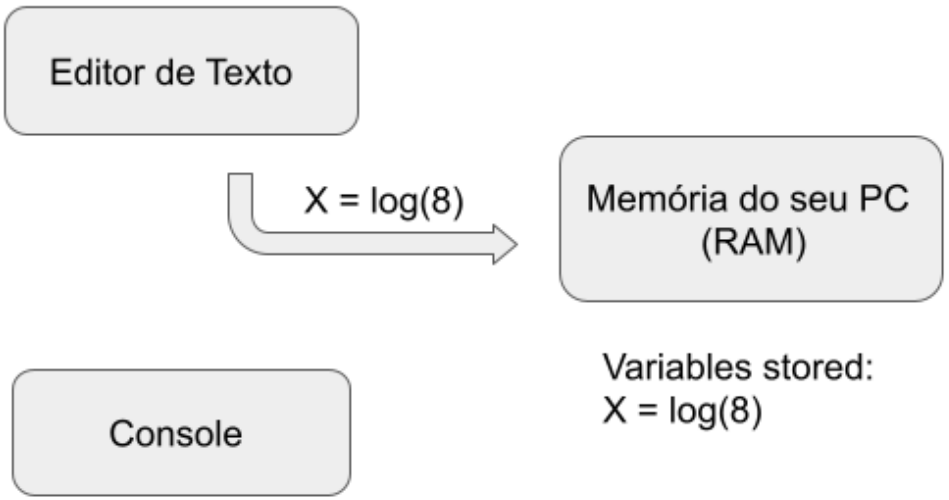


Gravando uma variavel



Gravando um valor em cima

Gravando uma variavel



Gravando uma variavel

O computador lê uma sequencia de comandos

- Portanto, fica armazenada na memória a última linha rodada.

Operador Lógico

Há uma pequena lista bem intuitiva de operadores lógicos que iremos destrinchar, mas para resumir: Esses operadores fazem perguntas na qual são respondidas com **True** ou **False**

- Operador de **Maior que**: >

Operador Lógico

Há uma pequena lista bem intuitiva de operadores lógicos que iremos destrinchar, mas para resumir: Esses operadores fazem perguntas na qual são respondidas com **True** ou **False**

- Operador de **Maior que**: >
- Operador de **Menor que**: <

Operador Lógico

Há uma pequena lista bem intuitiva de operadores lógicos que iremos destrinchar, mas para resumir: Esses operadores fazem perguntas na qual são respondidas com **True** ou **False**

- Operador de **Maior que**: >
- Operador de **Menor que**: <
- Operador de **Maior igual**: >=

Operador Lógico

Há uma pequena lista bem intuitiva de operadores lógicos que iremos destrinchar, mas para resumir: Esses operadores fazem perguntas na qual são respondidas com **True** ou **False**

- Operador de **Maior que**: >
- Operador de **Menor que**: <
- Operador de **Maior igual**: >=
- Operador de **Menor igual**: <=

Operador Lógico

Há uma pequena lista bem intuitiva de operadores lógicos que iremos destrinchar, mas para resumir: Esses operadores fazem perguntas na qual são respondidas com **True** ou **False**

- Operador de **Maior que**: >
- Operador de **Menor que**: <
- Operador de **Maior igual**: >=
- Operador de **Menor igual**: <=
- Operador **Igual a** : ==

Operador Lógico

Há uma pequena lista bem intuitiva de operadores lógicos que iremos destrinchar, mas para resumir: Esses operadores fazem perguntas na qual são respondidas com **True** ou **False**

- Operador de **Maior que**: >
- Operador de **Menor que**: <
- Operador de **Maior igual**: >=
- Operador de **Menor igual**: <=
- Operador **Igual a** : ==
- Operador **Diferente de**: !=

Exemplos

```
# Inicializando valores nas variaveis
a <- 2
b <- 3
```

```
# Maior que
a > b
```

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

Exemplos

```
# Menor que
a < b
```

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

```
# Maior Igual que
a >= b
```

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

Exemplos

```
# Menor igual que
a <= b
```

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

```
# Igual que
a == b
```

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

```
# Diferente de
a != b
```

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

Exercicio com a ajuda do professor

Objetos Atômicos

Objetos Atômicos

O R possui muitos objetos atômicos. Nós iremos ver agora os **4 principais**:

- Numerics (Double)

Objetos Atômicos

O R possui muitos objetos atômicos. Nós iremos ver agora os **4 principais**:

- Numerics (Double)
- Inteiros (Double)

Objetos Atômicos

O R possui muitos objetos atômicos. Nós iremos ver agora os **4 principais**:

- Numerics (Double)
- Inteiros (Double)
- Caracters (Characters)

Objetos Atômicos

O R possui muitos objetos atômicos. Nós iremos ver agora os **4 principais**:

- Numerics (Double)
- Inteiros (Double)
- Caracters (Characters)
- Lógicos (Booleanos,Binários)

Numerics

Os objetos `Numerics` representam números,obviamente.

Qualquer número que você digite no R, vai ser da **Classe** dos númericos previamente

```
# Criando objeto atômico numérico
x <- 5.3

# Observando sua classe
class(x)
```

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

Funções

Como você viu no ultimo slide, o R é feito de funções, no caso usamos a função `class()`.

Todas as funções no R possuem esse formato

`nome_da_função(param1,param2,param3,...)`

As vezes a função tem apenas 1 parâmetro obrigatório e outros são opcionais

Caso você queria ver como usar a função, rode no seu console `?nome_da_função`

Inteiros

Explicando péssimamente os objetos inteiros são como fosse objetos `Numerics`, só que **arredondados**.

Como todos os números que botamos no R são da classe `Numeric` previamente, temos que usar a função `as.intenger()` para transformar um objeto em `inteiro`

```
z <- 4.3
class(z)
```

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

```
# Transformando
z <- as.integer(z)
z
```

```
## [1] 4
```

Caracter

Os objetos `caracter` representam um texto no R.

Textos são normalmente formados por letras, porém números também podem aparecer.

`caracter` é caracterizado pelo uso de **aspas** (") envolvendo - os

```
t <- 'd'
class(t)
```

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

Observe que o código abaixo produz um `caracter` e não um `numeric`, pelo uso das aspas

```
w <- '2'
class(w)
```

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

Lógicos

Objetos do tipo `lógicos` são objetos que retornam `TRUE` ou `FALSE` como vimos anteriormente.

```
t <- TRUE
f <- FALSE
```

```
class(t)
```

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

```
class(f)
```

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

Exercicios

Exercicios

Vetores

Vetores

Vetor é um dos objetos mais importantes no R. Ele é a base para a construção dos outros 2 objetos que iremos estudar.

Caracteristicas de um vetor:

- O vetor é um objeto unidimensional

Vetores

Vetor é um dos objetos mais importantes no R. Ele é a base para a construção dos outros 2 objetos que iremos estudar.

Caracteristicas de um vetor:

- O vetor é um objeto unidimensional
- O vetor só aceita uma classe de objetos atômicos

Vetores

Vetor é um dos objetos mais importantes no R. Ele é a base para a construção dos outros 2 objetos que iremos estudar.

Caracteristicas de um vetor:

- O vetor é um objeto unidimensional
- O vetor só aceita uma classe de objetos atômicos
- A classe dos elementos que compôe o vetor é a classe do mesmo

Vetores

Vetor é um dos objetos mais importantes no R. Ele é a base para a construção dos outros 2 objetos que iremos estudar.

Caracteristicas de um vetor:

- O vetor é um objeto unidimensional
- O vetor só aceita uma classe de objetos atômicos
- A classe dos elementos que compôe o vetor é a classe do mesmo
- A função `c()` cria um vetor

Criando um vetor de caracteres

```r
# Criando
v1 <- c("MC","Kevin","o","Chris")
print(v1)
```

```
## [1] "MC"    "Kevin" "o"     "Chris"
```

```r
# Verificando a classe
class(v1)
```

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

Criando um vetor de numerics

```r
# Criando
v2 <- c(1,2,3,4)
print(v2)
```

```
## [1] 1 2 3 4
```

```r
# Verificando a classe
class(v2)
```

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

Combinando Vetores

Podemos combinar dois vetores em um

```
# Vetores
v1 <- c("MC","Kevin","o","Chris")
v2 <- c(1,2,3,4)

# Combinando

v3 <- c(v1,v2)
print(v3)
```

```
## [1] "MC"    "Kevin" "o"     "Chris" "1"     "2"     "3"
```

Dominancia de classes

Caracter > Numeric > Inteiro > Lógico

Operações com vetores

Podemos fazer operações básicas com os vetores

```
a <- c(1,2)
b <- c(3,4)
```

```
# Subtração
a - b
```

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

Operações com vetores

```
# Soma
a + b
```

```
## [1] 4 6
```

```
# Divisão
a / b
```

```
## [1] 0.3333333 0.5000000
```

```
# Multiplicação
a * b
```

```
## [1] 3 8
```

```
# Potenciação
a^b
```

```
## [1]  1 16
```

E se os vetores tiverem tamanhos diferentes?

```
a <- c(1,2)
b <- c(3,4,5,6)
```

```
# Soma
a + b
```

```
## [1] 4 6 6 8
```

- Sim, repete - se o vetor menor até o fim do vetor maior

E se os vetores tiverem tamanhos diferentes?

```
a <- c(1,2)
b <- c(3,4,5,6)
```

```
# Soma
a + b
```

```
## [1] 4 6 6 8
```

- Sim, repete - se o vetor menor até o fim do vetor maior
- Essa operação se dá o nome de **Reciclagem**

Mas e se os tamanhos não forem múltiplos?

```r
a <- c(1,2)
b <- c(3,4,5,6,7)
```

```r
# Soma
a + b
```

```
## [1] 4 6 6 8 8
```

- A mesma coisa acontece, só que o vetor menor não irá ser somado todos os seus elementos na ultima repetição

Exercicios

Matriz

Matriz

Matrizes são muito similares aos vetores, a única diferença é que são objetos bidimensionais.

Um vetor poderia ser composto por uma linha ou uma coluna.

Já matrizes são compostas obrigatóriamente por linhas e colunas

resumindo:

- A matriz é um objeto composto por linhas e colunas

Matriz

Matrizes são muito similares aos vetores, a única diferença é que são objetos bidimensionais.

Um vetor poderia ser composto por uma linha ou uma coluna.

Já matrizes são compostas obrigatóriamente por linhas e colunas

resumindo:

- A matriz é um objeto composto por linhas e colunas
- Ela só aceita uma classe de objetos atômicos

Matriz

Matrizes são muito similares aos vetores, a única diferença é que são objetos bidimensionais.

Um vetor poderia ser composto por uma linha ou uma coluna.

Já matrizes são compostas obrigatóriamente por linhas e colunas

resumindo:

- A matriz é um objeto composto por linhas e colunas
- Ela só aceita uma classe de objetos atômicos
- A classe dos elementos que compôe a matriz é a classe da mesma

Matriz

Matrizes são muito similares aos vetores, a única diferença é que são objetos bidimensionais.

Um vetor poderia ser composto por uma linha ou uma coluna.

Já matrizes são compostas obrigatóriamente por linhas e colunas

resumindo:

- A matriz é um objeto composto por linhas e colunas
- Ela só aceita uma classe de objetos atômicos
- A classe dos elementos que compôe a matriz é a classe da mesma
- A função `matrix()` cria uma matriz

Matriz

Matrizes são muito similares aos vetores, a única diferença é que são objetos bidimensionais.

Um vetor poderia ser composto por uma linha ou uma coluna.

Já matrizes são compostas obrigatóriamente por linhas e colunas

resumindo:

- A matriz é um objeto composto por linhas e colunas
- Ela só aceita uma classe de objetos atômicos
- A classe dos elementos que compôe a matriz é a classe da mesma
- A função `matrix()` cria uma matriz
- Para preencher uma matriz você terá que passar um vetor dentro dela

Matriz

Matrizes são muito similares aos vetores, a única diferença é que são objetos bidimensionais.

Um vetor poderia ser composto por uma linha ou uma coluna.

Já matrizes são compostas obrigatóriamente por linhas e colunas

resumindo:

- A matriz é um objeto composto por linhas e colunas
- Ela só aceita uma classe de objetos atômicos
- A classe dos elementos que compôe a matriz é a classe da mesma
- A função `matrix()` cria uma matriz
- Para preencher uma matriz você terá que passar um vetor dentro dela
- **This way**: `matrix(c())`

Criando uma matriz com 4 elementos

```
matrix(c(1,2,3,4))
```

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

- O padrão de uma matriz é ser preenchida por coluna

Criando uma matriz com 4 elementos

```
matrix(c(1,2,3,4))
```

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

- O padrão de uma matriz é ser preenchida por coluna
- Podemos especificar quantas colunas queremos em uma matriz com o parâmetro `nc`

Criando uma matriz com 4 elementos

```
matrix(c(1,2,3,4))
```

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

- O padrão de uma matriz é ser preenchida por coluna
- Podemos especificar quantas colunas queremos em uma matriz com o parâmetro `nc`
- Podemos fazer o mesmo procedimento so que por linhas com o parâmetro `nr`

Criando uma matriz de 4 elementos e 2 colunas

```
matrix(c(1,2,3,4),nc = 2)
```

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

Criando uma matriz de 4 elementos e 4 linhas

```
matrix(c(1,2,3,4),nr = 4)
```

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

Criando uma matriz de 4 elementos preenchida por linha

Para preencher uma matriz por linha, podemos usar o parâmetro `byrow = TRUE`

```
matrix(c(1,2,3,4),
       nc = 2,
       byrow = TRUE)
```

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

Exercicio

Data Frames

Data Frames

Data frames são como matrizes, porém não possuem restriçoes de classe.

São os objetos que se parecem com planilhas ou tabelas do excel

Resumindo:

- Um Data Frame é um objeto composto por linhas e colunas

Data Frames

Data frames são como matrizes, porém não possuem restriçoes de classe.

São os objetos que se parecem com planilhas ou tabelas do excel

Resumindo:

- Um Data Frame é um objeto composto por linhas e colunas
- Ela aceita mais de uma classe de objetos atômicos

Data Frames

Data frames são como matrizes, porém não possuem restriçoes de classe.

São os objetos que se parecem com planilhas ou tabelas do excel

Resumindo:

- Um Data Frame é um objeto composto por linhas e colunas
- Ela aceita mais de uma classe de objetos atômicos
- A classe do objeto data frame é data.frame

Data Frames

Data frames são como matrizes, porém não possuem restriçoes de classe.

São os objetos que se parecem com planilhas ou tabelas do excel

Resumindo:

- Um Data Frame é um objeto composto por linhas e colunas
- Ela aceita mais de uma classe de objetos atômicos
- A classe do objeto data frame é data.frame
- A função `data.frame()` cria uma data frame

Data Frames

Data frames são como matrizes, porém não possuem restriçoes de classe.

São os objetos que se parecem com planilhas ou tabelas do excel

Resumindo:

- Um Data Frame é um objeto composto por linhas e colunas
- Ela aceita mais de uma classe de objetos atômicos
- A classe do objeto data frame é data.frame
- A função `data.frame()` cria uma data frame
- Para criar as colunas de um data frame você terá que passar um vetor

Data Frames

Data frames são como matrizes, porém não possuem restriçoes de classe.

São os objetos que se parecem com planilhas ou tabelas do excel

Resumindo:

- Um Data Frame é um objeto composto por linhas e colunas
- Ela aceita mais de uma classe de objetos atômicos
- A classe do objeto data frame é data.frame
- A função `data.frame()` cria uma data frame
- Para criar as colunas de um data frame você terá que passar um vetor
- **This way**: `data.frame(Alunos = c(),Escola = c())`

Data frame

```r
# Criando Data Frame
df1 <- data.frame(Times = c("Flamengo","Palmeiras"),
                  Libertadores = c(2,1),
                  Mundial = c(TRUE,FALSE))
```

```
##       Times Libertadores Mundial
## 1  Flamengo             2    TRUE
## 2 Palmeiras             1   FALSE
```

Data Frame

```r
# Informações
str(df1)
```

```
## 'data.frame':    2 obs. of  3 variables:
##  $ Times        : Factor w/ 2 levels "Flamengo","Palmeiras":
##  $ Libertadores: num  2 1
##  $ Mundial      : logi  TRUE FALSE
```

```r
# Nº Colunas
ncol(df1)
```

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

```r
# Nº Linhas
nrow(df1)
```

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

Data Frame

```
mtcars
```

```
##                    mpg cyl  disp  hp drat    wt  qsec vs
## Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0
## Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0
## Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1
## Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0
## Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1
## Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0
## Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1
## Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1
## Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1
## Merc 280C         17.8   6 167.6 123 3.92 3.440 18.90  1
## Merc 450SE        16.4   8 275.8 180 3.07 4.070 17.40  0
## Merc 450SL        17.3   8 275.8 180 3.07 3.730 17.60  0
## Merc 450SLC       15.2   8 275.8 180 3.07 3.780 18.00  0
```

Data Frame

```r
# Observando as primeiras 6 linhas
head(mtcars)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0
```

Data Frame

```
# Observando as ultimas 6 linhas
tail(mtcars)
```

```
##                  mpg cyl  disp  hp drat    wt qsec vs am gea
## Porsche 914-2   26.0   4 120.3  91 4.43 2.140 16.7  0  1
## Lotus Europa    30.4   4  95.1 113 3.77 1.513 16.9  1  1
## Ford Pantera L 15.8   8 351.0 264 4.22 3.170 14.5  0  1
## Ferrari Dino    19.7   6 145.0 175 3.62 2.770 15.5  0  1
## Maserati Bora   15.0   8 301.0 335 3.54 3.570 14.6  0  1
## Volvo 142E      21.4   4 121.0 109 4.11 2.780 18.6  1  1
```

Data Frame

```r
# Nome das colunas
names(df1)
```

```
## [1] "Times"        "Libertadores" "Mundial"
```

```r
# Renomeando as colunas
names(df1) <- c("Clube","Libertadores","Tem_Mundial")
```

```
##       Clube Libertadores Tem_Mundial
## 1  Flamengo            2        TRUE
## 2 Palmeiras            1       FALSE
```

Exercicios

Listas

Listas

Listas são um dos objetos mais complexos do R. Em um primeiro momento ele pode confundir, mas não passa de um vetor ampliado.

Resumindo:

- Listas armazenam qualquer objeto nelas mesmas (vetores,data frames,matrizes,listas)

Listas

Listas são um dos objetos mais complexos do R. Em um primeiro momento ele pode confundir, mas não passa de um vetor ampliado.

Resumindo:

- Listas armazenam qualquer objeto nelas mesmas (vetores,data frames,matrizes,listas)
- Elas são dividas pelo que chamamos de nós

Listas

Listas são um dos objetos mais complexos do R. Em um primeiro momento ele pode confundir, mas não passa de um vetor ampliado.

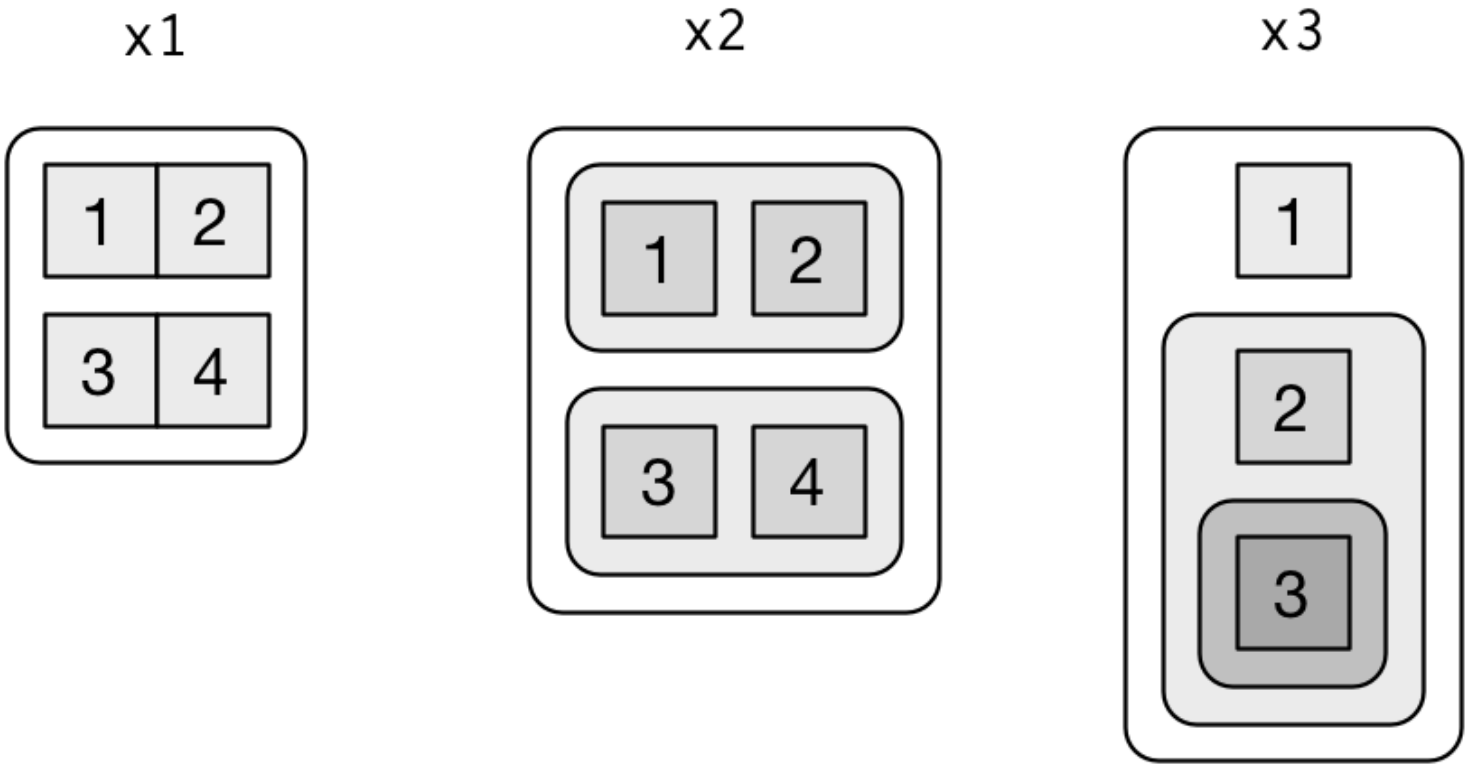
Resumindo:

- Listas armazenam qualquer objeto nelas mesmas (vetores,data frames,matrizes,listas)
- Elas são dividas pelo que chamamos de nós
- Para criar uma lista nós chamamos a função `list()`

Exemplo

```
x1 <- list(c(1, 2), c(3, 4))
x2 <- list(list(1, 2), list(3, 4))
x3 <- list(1, list(2, list(3)))
```

Estrutura de uma lista



Exemplo de lista não nomeada

Criando uma lista com matriz e data frame

```
df <- data.frame(coluna1 = c(1,2,3,4),
                 coluna2 = c(42,5,2,5))
m1 <- matrix(c(1:8),nc = 2)

list1 <- list(df, # 1º Nó
              m1) # 2º Nó
```

Exemplo

```
## [[1]]
##   coluna1 coluna2
## 1       1      42
## 2       2       5
## 3       3       2
## 4       4       5
##
## [[2]]
##      [,1] [,2]
## [1,]    1    5
## [2,]    2    6
## [3,]    3    7
## [4,]    4    8
```

Exemplo de lista nomeada

Vetor e data frame

```
df1 <- data.frame(x = c("Start","End"),
                  y = c(FALSE,TRUE))
v1 <- c(1,2,3,4,5)

list2 <- list(primeiro = df1,
              segundo = v1)
```

Exemplo de lista nomeada

Vetor e data frame

Onde estudar listas?

Acesse aqui

O exemplo acima foi tirado de lá.

Exercicios