Aula 1

mamancio

20/07/2021

Monitoria 1: Introdução ao R

O que é o R?

```
#Addition: +
#Subtraction: -
#Multiplication: *
#Division: /
#Exponentiation: ^
#Modulo: %%
#log (base =, x =)
#rnorm(2)
```

Assigment -> <-

chamar um elemento # operações com elementos

Tipos de dados (integer, logical, numeric/doubles, characters/string)

testes de class(x)

```
class(1)

## [1] "numeric"

class(FALSE)

## [1] "logical"

class("Texto")

## [1] "character"
```

operações com dados diferentes

```
1 + 2
```

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

```
TRUE + 1
```

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

```
TRUE + FALSE
```

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

1 + “dois” retorna um erro

#Vetores ## Vetores carregam um tipo de dado

```
numeric_vector <- c(1, 10, 49)
character_vector <- c("a", "b", "c")
boolean_vector <- c(TRUE, FALSE, TRUE)
# A classe de um vetor é seu primeiro elemento
  class(boolean_vector)
```

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

```
what_vector <- c("a", 5, TRUE)
  class(what_vector)
```

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

chamar um elemento de um vetor [], renomear o elemento

sample, sort, 1:X

```
sample(c(1, 5, 78, 2, 3))
```

```
## [1]  1 78  2  5  3
```

```
sort(c(1, 5, 78, 2, 3))
```

```
## [1]  1  2  3  5 78
```

```
sort(c(1, 5, 78, 2, 3), decreasing = TRUE)
```

```
## [1] 78  5  3  2  1
```

```
#qual diferença? Vamos olhar a função sort
?sort
```

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

operação com vetores

```
trade_vector <- c(140, -50, 20, -120, 240)
stocks_vector <-  c(-24, -50, 100, -350, 10)
```

```
# Qual foi o saldo?
trade_vector + stocks_vector

## [1]  116 -100  120 -470  250
```

nomeando um

modo 1

```
notas <- c("Camila" = 5, "Eduardo" = 7)
notas

##  Camila Eduardo
##       5       7
```

#modo 2

```
notas_gennins <- c(1:5, 5, 8)
  nomes_gennins <- c("Naruto", "Kiba", "Rock Lee", "Hinata", "Sakura",
"Sasuke")
  names(notas_gennins) <- nomes_gennins
  notas_gennins

##   Naruto     Kiba Rock Lee   Hinata   Sakura   Sasuke     <NA>
##        1        2        3        4        5        5        8
```

Opa! Falta um nome (NA no 7° elemento)

podemos usar [] para nos referir a um elemento de um vetor

```
# chamando o segundo elemento
notas_gennins[2]

## Kiba
##    2

#chamando os 4 primeiros elementos
notas_gennins[1:4]

##   Naruto     Kiba Rock Lee   Hinata
##        1        2        3        4

# chamando os 4 últimos elementos
notas_gennins[4:7]

## Hinata Sakura Sasuke   <NA>
##      4      5      5      8
```

podemos usar a seleção para nomear o elemento que está com Na [7]

```
names(notas_gennins)[7] <- "Shikamaru"
notas_gennins
```

```
##    Naruto      Kiba  Rock Lee    Hinata    Sakura    Sasuke Shikamaru
##         1         2         3         4         5         5         8
```

Dando uma olhada nas nossas notas

```
summary(notas_gennins)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     2.5     4.0     4.0     5.0     8.0
```

```
max(notas_gennins) -> mais_alta
min(notas_gennins) -> mais_baixa
print(mais_alta)
```

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

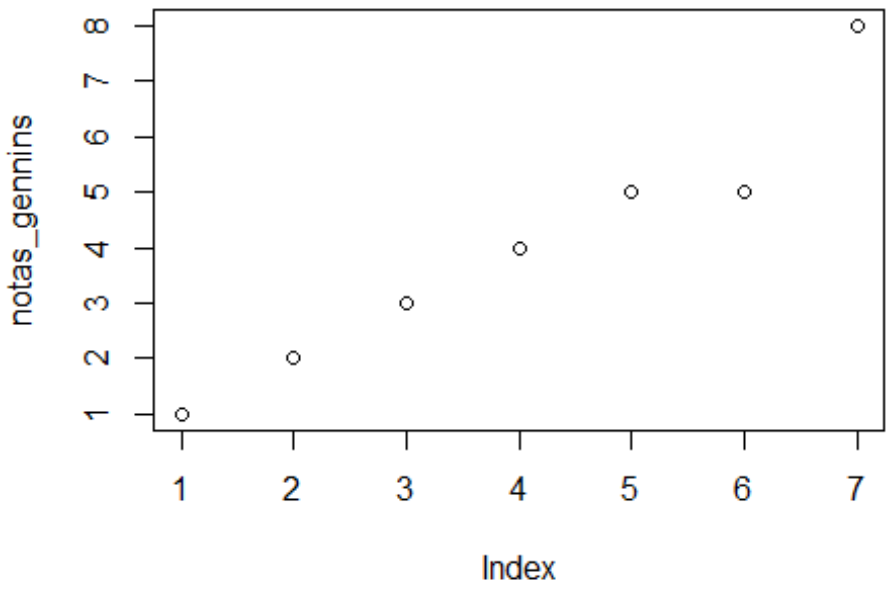
```
mais_baixa
```

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

```
#observe que o a função "print()" tem o mesmo efeito de chamar o elemento
```

podemos também fazer um gráfico

```
plot(notas_gennins)
```



Comparações

```
# <  menor que
10 < 5
```

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

```
# >  maior que
10 > 5
```

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

```
# <= menor ou igual a
10 <= 5
```

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

```
# >=  maior ou igual a
10 >= 5
```

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

```
# ==  igual a
10 == 5+5
```

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

```
10 == (100/50) * 5
```

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

```
TRUE == 1
```

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

```
# != diferente de
TRUE != 1
```

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

```
FALSE != 1
```

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

```
2 != 8/4
```

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

#Podemos usar comparações em vetores!

```
# Gennins com notas maiores ou igual a 4
notas_gennins >= 4
```

```
##    Naruto      Kiba  Rock Lee    Hinata    Sakura    Sasuke Shikamaru
##     FALSE     FALSE     FALSE      TRUE      TRUE      TRUE      TRUE
```

```
# Gennins com notas iguais a 4
notas_gennins == 4
```

```
##    Naruto      Kiba  Rock Lee    Hinata    Sakura    Sasuke Shikamaru
##     FALSE     FALSE     FALSE      TRUE     FALSE     FALSE     FALSE
```

```
# Gennins com notas diferentes de 4
notas_gennins != 4
```

```
##    Naruto      Kiba  Rock Lee    Hinata    Sakura    Sasuke Shikamaru
##      TRUE      TRUE      TRUE     FALSE      TRUE      TRUE      TRUE
```

Podemos usar a lógica para selecionar elementos dos vetores!

```
melhores_gennins <- notas_gennins > 4
notas_gennins[melhores_gennins]
```

```
##    Sakura    Sasuke Shikamaru
##         5         5         8
```

agora chamando os alunos com notas < 4, usando a negação

```
notas_gennins[!melhores_gennins]
```

```
##   Naruto     Kiba Rock Lee   Hinata
##        1        2        3        4
```

Algumas operações com vetores

#Quantos pontos totais eles tiveram?

```
sum(notas_gennins)
```

```
## [1] 28
```

#Média

```
mean(notas_gennins)
```

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

Variância

```
var(notas_gennins)
```

```
## [1] 5.333333
```

#Desvio Padrão

```
sd(notas_gennins)
```

```
## [1] 2.309401
```

O desvio padrão é a raiz quadrada da variância, confere?

```
sd(notas_gennins) == (sqrt(var(notas_gennins)))
```

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

#Quantos pontos fez o time 7?

```
sum(notas_gennins[c("Sasuke", "Sakura", "Naruto")])
```

```
## [1] 11
```

ou

sum(notas_gennins[c(1, 5)])

```
sum(notas_gennins[1], notas_gennins[5], notas_gennins["Sasuke"])
```

```
## [1] 11
```

ou ainda

```
sum(notas_gennins[c(1, 5, 6)])
```

```
## [1] 11
```

ou seja, podemos nos referenciar ao n° do elemento ou seu nome

Naruto colou no teste e foi eliminado! E agora?

```
notas_gennins["Naruto"] <- NA
notas_gennins
```

```
##    Naruto      Kiba  Rock Lee    Hinata    Sakura    Sasuke Shikamaru
##        NA         2         3         4         5         5         8
```

```
mean(notas_gennins)
```

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

se fizermos a média (ou a maioria das outras funções) com um elemento NA, o R nos retorna NA. Vamos olhar para a função mean

```
?mean
```

a função é configurada para receber na.rm = FALSE, ou seja, ela não elimina os NA. Toda função que tem elementos definidos (com um "="), recebe esses elementos como padrão. Mas podemos mudar isso.

```
mean(notas_gennins, na.rm = TRUE)
```

```
## [1] 4.5
```

Agora a nota do naruto é ignorada! Veja que temos os mesmos resultados que simplesmente ignorar o Narut (elemento [1] do nosso vetor)

```
mean(notas_gennins[2:7])

## [1] 4.5
```

#R entende os nomes O time 7 fez 3 testes, vamos olhar suas notas.

```
time7notas1 <- c(5, 4, 1)
time7notas2 <- c(5, 5, 2)
time7notas3 <- c(5, 3, 3)
time7nomes <- c("Sakura", "Sasuke", "Naruto")
names(time7notas1) <- time7nomes
names(time7notas2) <- time7nomes
names(time7notas3) <- time7nomes
time7notas1

## Sakura Sasuke Naruto
##      5      4      1

time7notas2

## Sakura Sasuke Naruto
##      5      5      2

time7notas3

## Sakura Sasuke Naruto
##      5      3      3
```

Vamos ver a nota total de cada um, e a nota de todos juntos

```
(time7notas1 + time7notas2 + time7notas3[na.rm = TRUE])

## Sakura Sasuke Naruto
##     15     12      6

sum(time7notas1, time7notas2, time7notas3, na.rm = TRUE)

## [1] 33
```

Comparando elementos de vetores diferentes

```
time7 <- c("Sakura" = 5, "Sasuke" = 4, "Naruto" = 3)
time10 <- c("Ino" = 4, "Shikamaru" = 6, "Chouji" = 4)
time8 <- c("Hinata" = 4, "Shino" = 4, "Kiba" = 3)

#Sakura tirou uma nota maior que Ino?

time7["Sakura"] > time10["Ino"]
```

```
## Sakura
##   TRUE
```

indo um pouco além

Um data.frame com todas as notas do time 7 ?data.frame

```
data.frame("Genjutsu" = time7notas1, "Ninjutsu" = time7notas2, "Taijutsu"
= time7notas3) -> dftime7
dftime7

##        Genjutsu Ninjutsu Taijutsu
## Sakura        5        5        5
## Sasuke        4        5        3
## Naruto        1        2        3
```

Adicionando uma linha Queremos ver a soma dos resultados de cada um

```
Total <- c(time7notas1 + time7notas2 + time7notas3)
Total

## Sakura Sasuke Naruto
##     15     12      6

cbind(dftime7, Total) -> dftime7n
dftime7n

##        Genjutsu Ninjutsu Taijutsu Total
## Sakura        5        5        5    15
## Sasuke        4        5        3    12
## Naruto        1        2        3     6
```

Adicionar o total em cada categoria

```
rbind(dftime7n, "Total" = c(sum(dftime7n[1]), sum(dftime7n[2]),
sum(dftime7n[3]), sum(dftime7n[4])))

##        Genjutsu Ninjutsu Taijutsu Total
## Sakura        5        5        5    15
## Sasuke        4        5        3    12
## Naruto        1        2        3     6
## Total        10       12       11    33
```

Um data.frame com todos os times

```
data.frame(time7, time8, time10, row.names = NULL) -> dfnotas_folha
dfnotas_folha

##   time7 time8 time10
## 1     5     4      4
## 2     4     4      6
## 3     3     3      4
```

Chamando elementos de um data.frame

um data frame tem linhas e colunas. Continuaremos a usar [], sendo o primeiro elemento a linha e o segundo a coluna

```
## busque a segunda nota do time 8 (coluna [2])

dfnotas_folha[2, 2]

## [1] 4

# todas as notas do time 8
dfnotas_folha[, 2]

## [1] 4 4 3
```

se deixamos um elemento em branco, significa que queremos todos. Ex: um data frame de l linhas e c colunas: Todas as linhas [, c] Todas as coluinas [l,]

```
# sem o time 7 (sem a coluna 1, todas as linhas)
dfnotas_folha[, 2:3]

##   time8 time10
## 1     4      4
## 2     4      6
## 3     3      4

# apenas o primeiro membro de cada time
dfnotas_folha [1,]

##   time7 time8 time10
## 1     5     4      4

# Podemos fazer comparações também
dfnotas_folha >= 4

##      time7 time8 time10
## [1,]  TRUE  TRUE   TRUE
## [2,]  TRUE  TRUE   TRUE
## [3,] FALSE FALSE   TRUE
```

Um pouco mais do que podemos fazer...

```
palmerpenguins::penguins -> penguins
library(tidyverse)

## -- Attaching packages --------------------------------------
tidyverse 1.3.1 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.1     v dplyr   1.0.6
```

```
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ---------------------------------------
tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

head(penguins)

## # A tibble: 6 x 8
##   species island bill_length_mm bill_depth_mm flipper_length_~
body_mass_g sex
##   <fct>   <fct>           <dbl>         <dbl>            <int>
<int> <fct>
## 1 Adelie  Torge~           39.1          18.7              181
3750 male
## 2 Adelie  Torge~           39.5          17.4              186
3800 fema~
## 3 Adelie  Torge~           40.3          18                195
3250 fema~
## 4 Adelie  Torge~           NA            NA                 NA
NA <NA>
## 5 Adelie  Torge~           36.7          19.3              193
3450 fema~
## 6 Adelie  Torge~           39.3          20.6              190
3650 male
## # ... with 1 more variable: year <int>
```

mostrar o .pdf do penguins

quantos de cada espécie?

```
penguins %>%
  count(species)

## # A tibble: 3 x 2
##   species       n
##   <fct>     <int>
## 1 Adelie      152
## 2 Chinstrap    68
## 3 Gentoo      124
```

as médias de cada espécie

```
penguins %>%
  group_by(species) %>%
  summarize(across(where(is.numeric), mean, na.rm = TRUE))
```

```
## # A tibble: 3 x 6
##   species   bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
year
##   <fct>              <dbl>         <dbl>             <dbl>       <dbl>
<dbl>
## 1 Adelie              38.8          18.3              190.       3701.
2008.
## 2 Chinstrap           48.8          18.4              196.       3733.
2008.
## 3 Gentoo              47.5          15.0              217.       5076.
2008.
```

#Encontre quantos penguins de cada espécie vivem em cada ilha

```
penguins %>%
  group_by(species, island) %>%
  summarise(contagem = n())

## `summarise()` has grouped output by 'species'. You can override using
the `.groups` argument.

## # A tibble: 5 x 3
## # Groups:   species [3]
##   species   island    contagem
##   <fct>     <fct>        <int>
## 1 Adelie    Biscoe          44
## 2 Adelie    Dream           56
## 3 Adelie    Torgersen       52
## 4 Chinstrap Dream           68
## 5 Gentoo    Biscoe         124
```

encontre quantos penguins de cada sexo são acima do peso da própria espécie

```
penguins %>%
  group_by(species) %>%
  filter(body_mass_g > mean(body_mass_g, na.rm = TRUE)) %>%
  group_by(sex, species) %>%
  summarise(cont = n())

## `summarise()` has grouped output by 'sex'. You can override using the
`.groups` argument.

## # A tibble: 7 x 3
## # Groups:   sex [3]
##   sex    species    cont
##   <fct>  <fct>     <int>
## 1 female Adelie        8
## 2 female Chinstrap     6
## 3 female Gentoo        4
```

```
## 4 male   Adelie       61
## 5 male   Chinstrap    25
## 6 male   Gentoo       54
## 7 <NA>   Adelie        1
```