

Ambiente R

Primeiros Conceitos

1. Introdução
2. Agregação de Dados
3. Ordenação de Dados
4. Leitura de bases de dados

Introdução

Sobre o R

- É uma linguagem de programação.
- É uma ferramenta gráfica e volta para soluções estatísticas.
- É um software livre.
- Disponível para Windows, OS e Linux.
- Há aplicações na internet que executam código R.
- É *case sensitive*.
- Execução por **linha de comando**.
- Executa **script** com vários comandos.
- Há funcionalidades disponíveis em pacotes.

Aparência do R

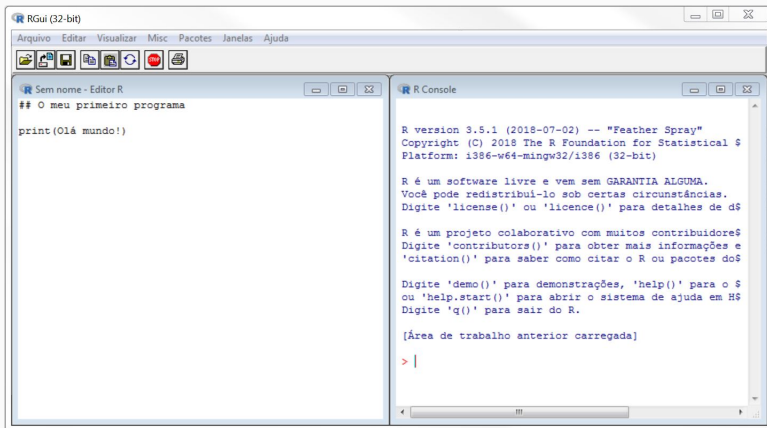


Figura 1: Visão do ambiente R

Ambiente R - Linha de Comando

- O terminal do R está pronto para receber comandos a partir do prompt `>`.
- Para executar um comando use a tecla **ENTER**

Ambiente R - Linha de Comando

Exemplo: digite $1+2+3$ em frente ao cursor $>$ e então tecla **ENTER**. O efeito dessa execução será:

```
> 1+2+3  
[1] 6  
>
```

Exemplo: digite $1+2*3$ em frente ao cursor $>$ e então tecla **ENTER**. Depois digite $(1+2)*3$, pressione **ENTER**. O resultado será

```
> 1+2*3  
[1] 7  
> (1+2)*3  
[1] 9  
>
```

IMPORTANTE :

- Os resultados anteriores são precedidos por `[1]`, indicando que o 1º elemento da linha é a resposta.
- Se a resposta ocupar mais de uma linha, esse índice serve de suporte para compreender a resposta. Por exemplo:

```
> 1:24
```

O resultado será:

```
[1]  1  2  3  4  5  6  7  8  9 10 11 12 13
```

```
[14] 14 15 16 17 18 19 20 21 22 23 24
```

```
>
```

`[14]` indica que a 14ª resposta ocupa a 1ª posição da segunda linha.

O script é um arquivo texto com extensão **R**, com uma série de comandos que serão executados sequencialmente, do primeiro ao último.

Definição (Pacote)

É um conjunto de dados, comandos e funções.

- Aumenta a capacidade de análise do R.
- Otimiza o uso da memória.
- Evita conflito de nomes de funções.
- Há pacotes disponíveis para download na internet.
- Os pacotes recomendados pelo comitê do R são acessíveis por meio do CRAN.

Um pacote pode ser instalado por meio do comando
`install.packages()`

Agregação de Dados

Definição

A operação de agregar é aplicar uma função a um grupo específico de dados

Agregação de Dados - função aggregate.

A função aggregate divide a base de dados em grupos e calcula uma determinada estatística a cada um deles.

```
> aux <- aggregate( x =iris$"Sepal.Length",  
+                   by = list(iris$"Species"),  
+                   FUN= mean)  
> aux
```

	Group.1	x
1	setosa	5.006
2	versicolor	5.936
3	virginica	6.588

Agregação de Dados - função aggregate.

Uma chamada básica para a função aggregate é:

```
aggregate(x, by, FUN)
```

Onde os argumentos são:

- x - um objeto R, os dados a serem agregados.
- by - uma **lista**, são os valores que formarão os grupos.
- FUN - a função que será aplicada em cada grupo.

A função indicada em FUN deve retornar apenas um valor.

Agregação de Dados - função tapply.

A função tapply divide a base de dados em grupos e calcula uma determinada estatística a cada um deles.

```
> aux <- tapply( X =iris[,1],  
+               INDEX = iris[5],  
+               FUN= mean)  
> aux  
Species  
  setosa versicolor  virginica  
   5.006      5.936      6.588
```


Agregação de Dados - função `tapply`.

Uma chamada básica para a função `tapply` é:

```
tapply(X, INDEX, FUN)
```

Onde os argumentos são:

- `X` - um objeto R, os dados a serem agregados.
- `INDEX` - uma **lista**, são os valores que formarão os grupos.
- `FUN` - a função que será aplicada em cada grupo.

Agregação de dados - função apply.

A função apply aplica uma função sobre um conjunto de dados.

```
> # media das colunas
> aux <- apply( X =iris[,1:4],
+              MARGIN = 2,
+              FUN= mean)
> aux
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
5.843333	3.057333	3.758000	1.199333

Agregação de Dados - função apply.

Uma chamada básica para a função `tapply` é:

```
apply(x, MARGIN, FUN)
```

Onde os argumentos são:

- `X` - um objeto R, os dados a serem agregados.
- `MARGIN` - um vetor que indica como a função será aplicada. Para um data frame, 1 indica as linhas, 2 as colunas e `c(1,2)` sobre as linhas e as colunas.
- `FUN` - a função que será aplicada em cada grupo.

Outras funções úteis: `colMeans`, `rowMeans`, `colSums`, `rowSums`.

Ordenação de Dados

Ordenação de Dados - função sort.

A função sort ordena um vetor ou um fator de forma crescente ou decrescente.

```
> x <- iris[1:4, "Sepal.Length"]  
> sort(x)  
[1] 4.6 4.7 4.9 5.1  
> sort(x,decreasing = TRUE)  
[1] 5.1 4.9 4.7 4.6
```

Ordenação de Dados - função sort.

Uma chamada básica para a função sort é:

```
sort(x, decreasing=FALSE)
```

Onde os argumentos são:

- `x` - um vetor ou fator, os dados a serem ordenados.
- `decreasing` - Se `FALSE` os dados são ordenados de forma crescente, se `TRUE`, de ordem decrescente.

Ordenação de dados - função order.

A função `order` gera os índices para que os valores originais fiquem em ordem crescente ou decrescente.

```
> x <- iris[1:4, "Sepal.Length"]  
> x  
[1] 5.1 4.9 4.7 4.6  
> order(x)  
[1] 4 3 2 1  
> order(x,decreasing = TRUE)  
[1] 1 2 3 4
```

Ordenação de Dados - função order.

Uma chamada básica para a função sort é:

```
order(x, decreasing=FALSE)
```

Onde os argumentos são:

- `x` - um vetor ou fator, os dados a serem organizados.
- `decreasing` - Se `FALSE` os dados são ordenados de forma crescente, se `TRUE`, de ordem decrescente.

Ordenação de Dados - função order.

Ordenando um data frame.

```
> x <- c(1,1,3:1,1:4,3) ; y <- c(9,9:1); z <- c(2,1:9)
> dados <- data.frame(x,y,z)
> dados[order(x,y,z),]
      x y z
[1,]  1 5 5
[2,]  1 6 4
[3,]  1 9 1
[4,]  1 9 2
[5,]  2 4 6
[6,]  2 7 3
[7,]  3 1 9
[8,]  3 3 7
[9,]  3 8 2
[10,] 4 2 8
```

Leitura de bases de dados

Uma chamada básica para a função save é:

```
save( ..., file=)
```

Onde os argumentos são:

- ... - nome dos objetos a serem salvos.
- file - arquivo onde serão salvos os objetos.

A função `save` gera o arquivo especificado em `file` que armazena os objetos especificados.

```
> x <- c(1,1,3:1,1:4,3) ; y <- c(9,9:1); z <- c(2,1:9)
> dados <- data.frame(x,y,z)
> save (x,y,z,dados,file="teste.RData")
```

Uma chamada básica para a função load é:

```
load(file, verbose=TRUE)
```

Onde os argumentos são:

- `file` - arquivo onde os objetos R estão armazenados.
- `verbose` - Se `FALSE` a leitura do arquivo não gera nenhuma mensagem. Se `TRUE`, é apresentada uma lista com o nome dos objetos lidos.

Leitura de bases de dados - função load.

A função load recupera do arquivo especificado em file todos os objetos R que o forma.

```
> rm(list=ls())  
> x <- c(1,1,3:1,1:4,3) ; y <- c(9,9:1); z <- c(2,1:9)  
> dados <- data.frame(x,y,z)  
> save (x,y,z,dados,file="teste.RData")  
> load("teste.RData", verbose = T)
```

Loading objects:

x

y

z

dados

```
> ls()
```

```
[1] "dados" "x"      "y"      "z"
```

Leitura de bases de dados - função load.

O arquivo `imoveis.csv` armazena informações sobre: bairro, localização, **preço** (reais), **número de quartos**, **tamanho** (m^2), **bairro**, **código do corretor**, **data do anúncio** e preço por m^2 (reais). Esses dados foram observados em 3782 imóveis anunciados em um site nos anos de 2013 e 2014. Com essas informações faça as tarefas abaixo:

1. Crie em sua área de trabalho a pasta teste e copie o arquivo `imoveis.csv` para essa pasta.
2. Com a função `setwd()` defina a pasta teste como a sua área de trabalho.
3. Com a função `read.csv2()` crie o data frame `dados` a partir do arquivo `imoveis.csv`.
4. Com a função `aggregate` calcule o preço médio do imóvel por bairro e o preço médio do número de quartos em cada bairro.
5. Crie o data frame `asa_norte` só com as informações do bairro `asa norte`, mas excluindo as variáveis **localização**, **código do corretor**, **data do anúncio** e preço por m^2 .

O pacote dplyr apresenta seis funções que permitem tratar as principais tarefas relacionadas a manipulação de bases de dados:

- `filter()` - Seleciona observações.
- `arrange()` - Ordena observações.
- `select()` - Seleciona variáveis.
- `mutate()` - Cria novas variáveis
- `group_by()` - Permite ações por grupos.
- `summarize()` - Agrega valores.

A função `filter()` escolhe as as observações que atendam determinada condição, as observações que retornam NA são desconsideradas.

```
dados <- filter(iris, Species == "setosa")
```

A função `arrange()` ordena as observações em ordem crescente.

```
dados <- arrange(iri, Sepal.Length)
# use a função desc() para ordenar em ordem decrescente
dados <- arrange(iri, desc(Sepal.Length))
```

Pacote dplyr - select()

A função `select()` seleciona variáveis de um data frame. E aceita funções que facilita o trabalho com data frames com muitas variáveis: `starts_with()`, `ends_with()`, `contains()`, `matches()`, `num_range()`, `one_of()`, `everything()`.

```
select(iris, starts_with("Petal"))  
select(iris, ends_with("Width"))
```

Pacote dplyr - mutate()

A função `mutate()` é útil para criar novas variáveis.

```
dados <- mutate(iri, x=Sepal.Length*100)
```

A maioria das operações com dados são executadas em grupos de observações definidos por determinadas variáveis. A função `group_by()` toma uma `tbl` e a converte em uma `tbl` agrupada, na qual as operações são executadas por grupos.

```
dados <- group_by (iris, Species)
```

A função `summarize()` é usada para agregar os valores de uma variável, isto é, transforma um conjunto de valores em um. Se a variável estiver agrupada (`group_by`) é apresentado um valor para cada grupo.

```
dados <- group_by (iris, Species)
summarize(dados, mean(Sepal.Length))
```

Com base nos arquivos `votacao_candidato_munzona_2018_DF.csv` e `leiamme_votacao.pdf` calcule a quantidade de votos obtidos por zona eleitoral de cada candidato a uma vaga de senador. Utilize o pacote `dplyr`.