O ***R*** é uma linguagem e ferramenta usada principalmente para análise estatística e gráfica de dados. Seu ambiente oferece uma vasta coleção de funções, além de diversas bibliotecas desenvolvidas por terceiros que podem ser facilmente instaladas e utilizadas. Usaremos esta ferramenta durante os laboratórios do curso de Fundamentos de Analytics.

Neste laboratório, iremos aprender os comandos básicos do R que vão servir de base para os próximos labs. Inicialmente, veremos quais as estruturas de dados e tipos básicos usados no R. Em seguida, veremos comandos para: entrada e saída de dados usando arquivos; acessar a documentação e ajuda de funções; implementar loops, iteradores e condicionais; além de aprender como escrever e executar scripts.

O R pode ser usado tanto na execução de scripts quanto em modo interativo, onde o usuário pode realizar operações em um console e obter resultados na tela. Após instalar a ferramenta, execute o comando "R" em um terminal linux para iniciá-lo no modo interativo. Também veremos neste lab como executar scripts.

Tipos básicos

Os tipos de dados mais comuns em R são os seguintes: - ***numeric***: representa números, podendo ser inteiro (**integer**) ou decimal (**double**).

- ***character:*** representa elementos textuais, como as strings em outras linguagens. "character" em R representa toda uma sequência de caracteres, ao invés de representar apenas um caracter como no tipo "char" de outras linguagens.

- ***logical:*** expressão lógica booleana, como o "boolean" de outras linguagens, que pode assumir um desses 2 valores: TRUE ou FALSE.

Estruturas de dados básicas

As estruturas de dados mais comuns em R são as seguintes:

- ***vector:*** o vetor é uma sequência de uma só dimensão e com

elementos de um único tipo, que podem ser acessados por um índice inteiro. Em R, dado um vetor "v", podemos acessar o primeiro elemento do vetor usando "v[1]" e o último elemento usando "v[length(v)]".

Exemplo:

```
v = vector(mode = "integer", length = 10) # atribuição v # imprime conteúdo da variável na tela is.vector(v)

v = c(1, 2, 3, 4, 5)
v
v = 1:5
v
v = seq(1, 5)
print(v)
```

- ***array:*** ao contrário do vector, o array pode ter mais de uma dimensão, mas todos os elementos também devem ser do mesmo tipo.

Exemplo:

```
a = array(1:10, dim = c(5, 2))
a
is.array(a)
a[1,2] # retorna o elemento da linha 1, coluna 2 a[1,] # retorna todas as colunas da linha 1 a[,1] # retorna todas as linhas da coluna 1
```

- ***matrix:*** é um tipo específico de array que contém duas dimensões.

Exemplo:

```
m = matrix(1:10, nrow = 5, ncol = 2) m
```

- ***data.frame***: o *dataframe* é um estrutura bastante usada no R, que representa tabelas onde cada coluna pode ter um tipo diferente, enquanto as linhas de uma mesma coluna devem ser do mesmo tipo. Nos dataframes, todas as colunas devem possuir a mesma quantidade de linhas.

Exemplo:

```
df = data.frame(Coluna1 = c("A", "B", "C"),

                Coluna2 = c(1, 2, 3),
                Coluna3 = c(TRUE, FALSE, TRUE))
df df$Coluna1 # retorna primeira coluna


_____ _____ _____
df2 = data.frame(Coluna1 = c("D", "E", "F"),

df2
df2$Coluna2
Coluna2 = c(4, 5, 6),
Coluna3 = c(FALSE, TRUE, FALSE))
df = rbind(df, df2) # Combina dataframes por linhas, mantendo
colunas is.data.frame(df) df df$Coluna3
```

- ***list***: estrutura parecida com o *dataframe*, porém cada atributo da lista pode ter quantidades de elementos diferentes.

Exemplo:

```
l = list(Atributo1 = c("A", "B", "C"),
    Atributo2 = c(1, 2, 3, 4, 5),
    Atributo3 = c(TRUE, FALSE))
l
```

Entrada e saída de dados usando arquivos

Os arquivos de dados de entrada do R podem ter diversos formatos. Dependendo do formato utilizado, podemos usar diferentes comandos para carregar os dados no ambiente. Os comandos mais comuns estão listados abaixo:

- ***scan***: lê um arquivo texto e transforma o conteúdo em um vector, assumindo que cada elemento do arquivo está separado por espaço ou por nova linha.

- ***read.csv***: lê um arquivo texto no formato CSV e transforma o conteúdo em uma tabela do tipo dataframe.

- ***read.table***: lê um arquivo texto em que cada elemento é separado por um determinado caracter separador e transforma o

conteúdo em uma tabela do tipo dataframe. Por default o comando vai tentar identificar automaticamente qual o separador dos elementos, sendo os espaço " " e tab "\t" os mais comuns.

- ***write*:** Comando simples de escrita de dados em arquivos. - ***write.table:*** Escreve tabelas em um arquivo de saída, com vários opções de formatação. - ***write.csv:*** Escreve tabelas em um arquivo de saída em formado CSV.

Comandos básicos

Ajuda

Um comando bastante útil no R é o de ajuda, que apresenta a documentação do comando em questão, como uma descrição da funcionalidade, parâmetros de entrada, valores de saída, além de exemplos do uso do comando. As formas de acessar a ajuda são:

- **help(comando)** ou **?comando**: apresenta a documentação do comando em questão. - **example(comando)**: executa o exemplo contido na documentação do comando.

Exemplo:

help(rbind)
?data.frame
?c
?print
Usem e abusem do comando de documentação. Outra fonte de ajuda são os fóruns da comunidade do R, que são bastante movimentados e possuem uma grande quantidade de questões solucionadas. Geralmente, fazendo uma busca no Google por *R <dúvida>* se obtém bons links. Também pode filtrar apenas os resultados do *site:stat.ethz.ch*, que é um dos fóruns de R mais utilizados.

Atribuição

Como já vimos em alguns exemplos, o símbolo "=" é usado para atribuição. Um outro símbolo bastante usado no R para atribuição é o "<-", que tem efeito igual ao do "=". A seta no

sentido inverso "->" também pode ser usado.

Exemplo:

```
# mesmo resultado para os três casos
x = 10
x <- 10
10 -> x
```

Condicional

A sintaxe da operação básica condicional do R é semelhante à linguagens mais conhecidas:

```
if (condição lógica) {
...
} else {
...
}
```

_____ _____

Uma outra alternativa é o comando *ifelse*, que permite operações condicionais em vetores:

Exemplo:

```
x = -5:5
y = ifelse(x > 0, x*2, x*5)
y
```

Entre os operadores condicionais, temos: • |:or

• &:and • ==:iguala

Filtro

Podemos aplicar um filtro nos dados baseado em condições. Dependendo da estrutura de dados usadas, podemos usar diferentes comandos. No caso de vetores, podemos usar:

```
x.filtrado = x[condição]
```

No caso de estruturas de dados com 2 dimensões, podemos usar:

```
x.filtrado <- x[condição, ] # filtra linhas onde a condição é TRUE
x.filtrado <- x[ ,condição] # filtra colunas onde a condição é
```

TRUE

Outra opção é usar o comando *subset*, como podemos ver no exemplo abaixo.

Exemplo:

```
x = -5:5
x.positivos = x[x > 0]
df = data.frame(X = -10:10, Y = 0:20) df df.filtrado = df[df$X > 0, ] df.filtrado = subset(df, X > 0) # igual ao anterior df.filtrado
```

Um outro comando útil de filtro é o ***which***, que retorna o índice dos elementos que satisfazem a condição.

Exemplo:

```
x = c(1, -4, 3, 5, -2, -5) which(x < 0) # deve retornar os índices 2, 5 e 6
```

_____ _____ _____

Loops e iteradores

O R possui comandos básicos de loops como nas linguagens mais conhecidas, como ***for*** e ***while***. As sintaxes são:

```
for (i in <vetor>) {
... }
```

```
while (condição) {
... }
```

Outros comandos bastante usados para iterar sobre os elementos de uma estrutura de dados e aplicar funções neles são: ***apply***, ***aggregate***, ***tapply***, ***by***. Verifiquem a documentação de cada comando usando o *help(...)* ou *?comando*.

Executando scripts

Como dito anteriormente, além do modo interativo, o R também pode ser usado na execução de scripts. Para executar um arquivo de script, rode o seguinte comando em um terminal:

Rscript <*arquivo do script*> <*argumentos*> Você também pode

usar um arquivo de script dentro do ambiente interativo do R, usando:

source('<arquivo do script>')