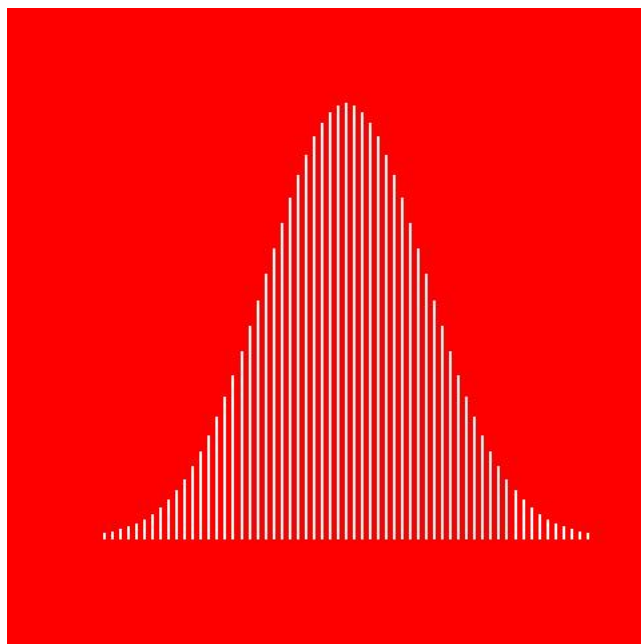


Curso de Análise Estatística

Caderno de Atividades



Brasília - DF
janeiro de 2019

Sumário

Ementa	4
R conceitos básicos	5
R operações básicas	7
Desvio Condicionais e Laços de Repetição	16
Listas	19

Ementa

OBJETIVO: Introduzir as noções básicas de estatística. Capacitar o aluno para ler, interpretar e organizar dados em tabelas e gráficos. Desenvolver a capacidade de interpretação de dados estatísticos e análise crítica de informações. Capacitar o aluno calcular medidas estatísticas e elaborar gráficos com o objetivo de avaliar as informações contidas em grande conjunto de dados, por meio do software R.

PÚBLICO ALVO: Analistas, Gerentes, Alunos e profissionais das mais diversas áreas que trabalham ou que queiram trabalhar com gestão de dados estatísticos.

CONTEÚDO

- Unidade I: Conceitos
 1. Estatística: Descritiva/Inferencial/Probabilística
 2. Variáveis quantitativas e qualitativas
- Unidade II: Software R
 1. Conceitos de vetor e data frame
 2. Funções estatísticas e gráficos
- Unidade III: Frequências:
 1. Dados Brutos e Rol;
 2. Tipo: Simples, Relativa, Acumulada e Relativa acumulada
 3. Histograma
- Unidade IV: Medidas de Tendência Central
 1. Média
 2. Moda
- Unidade V: Medidas de Dispersão
 1. Variância
 2. Desvio Padrão
 3. Coeficiente de Variação
- Unidade VI: Separatrizes
 1. Quartis
 2. Percentis
- Unidade VII: Gráficos
 1. Conceitos e tipos
- Unidade VIII - Correlação e Regressão
 1. Coeficiente de correlação linear.
 2. Retas de Regressão. Proporção de variação explicada.
- Aplicações.

R conceitos básicos

Questão 1 - Qual a função que encerra o R?

- a) `Quit()` b) `quit()` c) `Exit()` d) `exit()`

Questão 2 - Qual a função que apresenta o caminho da pasta usada como “working directory”?

- a) `ls()` b) `rm()` c) `setwd()` d) `getwd()`

Questão 3 - Qual a função que define a pasta a ser usada como “working directory”?

- a) `par()` b) `q()` c) `setwd()` d) `getwd()`

Questão 4 - Qual a expressão que armazena no vetor `y` a média do vetor `x` ?

- a) `y <- sum(x)/length(x)` b) `y <- sum(x)**2/(length(x)-1)`
c) `y <- length(x)/sum(x)` d) `y <- (length(x)-1)/sum(x)`

Questão 5 - Por meio do R foram criados os vetores `x`, `y` e `z`. Qual a expressão que **não** apaga esses vetores ?

- a) `ls(list=rm())` b) `rm("x","y","z")`
c) `rm(list=ls())` d) `rm(list=letters[24:26])`

Questão 6 - Qual a função que retorna o número de elementos de um vetor `x`?

- a) `dim(x)` b) `nrow(x)` c) `ncol(x)` d) `length(x)`

Questão 7 - Em relação a um *vetor*, qual informação é incorreta?

- a) Todos os seus elementos são do mesmo tipo.
b) Cada elemento ocupa uma posição específica.
c) Pode ser criado pela função `vector()`.
d) A função `ncol()` retorna o seu número de elementos.

Questão 8 - Em relação a um *data frame*, qual informação é incorreta?

- a) É organizado em linhas (observações) e colunas (variáveis).
b) As variáveis podem ser de tipos diferentes.
c) As variáveis devem ter o mesmo número de elementos.
d) A função `length()` retorna o seu número de linhas.

Questão 9 - Qual expressão está incorreta?

- a) `x <- y <- 1` b) `1 -> x -> y` c) `x = y = 1` d) `1 = x = y`

Questão 10 - Considere o sistema operacional Windows, qual o comando define a pasta **E:** **X** como a “working directory”?

- a) `setwd("E:\\X")` b) `setwd("E:/X")` c) `getwd("E:/X")` d) `getwd("E:\\X")`

Questão 11 - Sejam os vetores `x <- 1` e `y <- c(2,3)`. É correto afirmar, sobre `z = x + y`:

- a) `z` tem 1 elemento com valor igual a 3.
b) `z` tem 2 elementos, cujos valores são 3.
c) `z` tem 2 elementos, e equivale ao vetor `c(3,4)`.
d) `z` não existe, pois `x` e `y` têm números de elementos diferentes.

Questão 12 - Sejam o vetor `x <- c(1,2)`. O valor de `z = 3*5 + x` é igual a

- a) `c(16,2)` b) `c(1,17)` c) `c(16,17)` d) `c(15,30)`

Questão 13 - Dado `x <- c(2,4,NA)`, qual o resultado de `mean(x)`?

- a) 3 b) 2 c) NA d) 8

Questão 14 - Com `x <- c(3,4,5,NA)`, qual o resultado de `sum(x,na.rm=T)/length(x)`?

- a) 3 b) NA c) 4 d) 12

Questão 15 - Dado `x <- matrix(c(1,2,3,4), byrow=F, nrow = 2)`. É correto afirmar que

- a) $x = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$ b) $x = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ c) `x[1,1]+x[1,2]=3` d) `sum(x[,1])=4`

Questão 16 - Como selecionar os elementos da segunda coluna de uma matriz $X_{3 \times 4}$?

- a) `X[,2]` b) `X[2,]` c) `X[2]` d) `x[, -2]`

Questão 17 - Em relação a uma *matrix*, qual informação é incorreta?

- a) É organizada em linhas e colunas.
b) Os seus elementos podem ser de tipos diferentes.
c) A função `ncol()` retorna o número de colunas da matriz.
d) A função `length()` retorna o número de elementos da matriz.

Questão 18 - Seja o vetor `x <- c(1,2,3)`, qual a opção que retorna o número de elementos do vetor `x`?

- a) `nrow()`
b) `ncol()`
c) `dim()`
d) `length()`

R operações básicas

Início

O R pode ser operado em modo interativo. Assim, todo comando digitado após o `>` (prompt) é executado quando a tecla **ENTER** é pressionada, e os resultados são apresentados nas linhas seguintes.

```
> 5 + 2
[1] 7
> 1:5
[1] 1 2 3 4 5
> seq(1,5,by=1.5)
[1] 1.0 2.5 4.0
```

Detalhes importantes: O interpretador R diferencia uma letra maiúscula de uma minúscula. Logo, o termo 'AB' é diferente de 'Ab', 'aB' e 'ab'. O interpretador pode receber vários comandos por linha, desde que sejam separados por ';' (ponto-e-vírgula). Caso pressione **ENTER** no meio de um comando, será apresentado o símbolo '+' no lugar do '>' permitindo que se prossiga o comando. Toda linha iniciada por '#' (jogo-da-velha) é considerada como comentário e não é executada.

```
> # Atribuindo um conjunto de valores a x
> x <- c(1,3,4,4,9,9,
+       7,6,5,6,2,1)
> # Calculando a média dos valores em x
> mean(x)
[1] 4.75
```

Questão 1 - Corrija e execute o código abaixo:

```
Calcula a soma de x + y
x <- 3 y <- 5
#soma de x + y
X + y
```

Questão 2 - Corrija e execute o código abaixo:

```
# x <- c(2,3,4) vetor quantidade
y <- C(5,2,1) # vetor preco
# o vetor custo é dado por
custo = X * y
```

Questão 3 - Qual o efeito da função `getwd()`?

Questão 4 - Crie no R os objetos `diretorio <- getwd()` e `a = 2`. Qual o efeito da execução da função `ls()` e da expressão `rm(list=ls())`?

Questão 5 - Qual o efeito da função `setwd("C:/")`?

Pacotes ¹

O programa R é composto de 3 partes básicas:

1. O R-base, o “coração” do R, que contém as funções principais disponíveis quando iniciamos o programa,
2. Os pacotes recomendados (*recommended packages*), que são instalados junto com o R-base, mas não são carregados quando iniciamos o programa. Por exemplo, os pacotes `MASS`, `lattice`, `nlme` são pacotes recomendados – e há vários outros. Para usar as funções destes pacotes, deve-se carregá-los antes com o comando `library()`. Por exemplo, o comando `library(MASS)` carrega o pacote `MASS`.
3. Os pacotes contribuídos (*contributed packages*), que não são instalados junto com o R-base. Estes pacotes, disponíveis na página do R são pacotes oficiais. Estes pacotes adicionais fornecem funcionalidades específicas, e para serem utilizados devem ser copiados, instalados e carregados, conforme explicado abaixo. Para ver a lista deste pacotes com uma descrição de cada um deles acesse a página do R e siga os links para CRAN e Package Sources.

Antes de instalar o pacote, você pode ver se ele já está instalado/disponível. Para isto inicie o R e digite o comando:

```
> require(NOME_DO_PACOTE)
```

Se ele retornar T, é porque o pacote já está instalado/disponível e você não precisa instalar. Se retornar F, siga os passos a seguir.

A instalação e uso dos pacotes vai depender do seu sistema operacional e os privilégios que você tem no seu sistema. Na explicação a seguir assume-se que você está em uma máquina conectada à internet.

Instalação em máquinas com Windows ou IOS: Neste caso, basta usar o comando `install.packages()` com o nome do pacote desejado entre aspas. Por exemplo, para instalar o pacote `CircStats` digite:

```
> install.packages('CircStats')
```

O pacote vai ser instalado no sistema e ficar disponível para todos os usuários. Para usar o pacote basta digitar `library(CircStats)` ou `require(CircStats)`.

Questão 1 - Instale o pacote `googlesheets`.

¹<http://www.leg.ufpr.br/paulojus/embrapa/Rembrapa/Rembrapase36.html> visitado em 24/01/2018

Pedindo Ajuda

O R disponibiliza um rico sistema de ajuda (help). Entretanto, ele é escrito para usuários intermediários ou avançados. Seguem as formas mais comuns de consultar a documentação referente a um tópico.

Função	Help retornado	Exemplo
<code>help.start()</code>	página na web	<code>help.start()</code>
<code>help(termo)</code>	relacionado ao termo	<code>help (summary)</code>
<code>?termo</code>	relacionado ao termo	<code>?summary</code>
<code>? "operador"</code>	relacionado ao operador	<code>? " : "</code>
<code>help.search("string")</code>	páginas com a string	<code>help.search("car")</code>
<code>?? "string"</code>	páginas com a string	<code>?? "car"</code>
<code>example(termo)</code>	executa exemplos	<code>example(summary)</code>
<code>help(dataframe)</code>	decreve data frame	<code>help(esoph)</code>

Questão 1 - Consulte o help da função `sum()`. Quais seções são apresentadas?

Questão 2 - Leia o help do data frame `cars`.

Questão 3 - Consulte a seção ‘Values’ retornada pela função `help()` aplicada aos operadores `%`, `%%` e `:` (dois pontos).

Questão 4 - Leia o help para `seq` e execute os códigos R da seção ‘Example’.

Questão 5 - Pesquise o help a função `demo()` e a aplique no pacote *graphics*.

Geradores de Sequências

O operador `:` gera uma sequência regular. A forma de uso é `a:b`, `a` é o valor inicial e `b` é o valor final. Para maiores detalhes `?seq`.

```
> 1:4
[1] 1 2 3 4
> 4:1
[1] 4 3 2 1
```

Questão 1 - Use o operador `:` para gerar as sequências:

a) de -1 a -4

b) de -4 a -1

Questão 2 - Use o operador `:` para gerar as sequências:

a) de 3.1514 a 6

b) de 6 a 3.1514

A função `seq()` gera também uma sequência de valores, mas permite um maior controle do usuário. Detalhes: `?seq`. Uma forma de uso é `seq(a , b , by= c)`, onde `a` é o valor inicial, `b` é o valor final e `c` o incremento.

```
> seq(1, 9, by = 2)
[1] 1 3 5 7 9
> seq(1, 9, by = pi)      # encerra antes de b
[1] 1.000000 4.141593 7.283185
> seq(1, 4, by = 0.5)
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0
```

Questão 3 - Gere um sequência com 10 elementos, que inicia em 1 e termina em 2 com a função `seq()`. Consulte no help o argumento `length.out=`.

Questão 4 - Execute o código abaixo. Consulte o help e mude o tipo de linha , troque a cor para azul e altere o incremento de x para 0.01

```
> x <- seq(-1,1,by=0.05)
> y <- x^2
> plot (x,y,type="b", col="red")
```

Questão 5 - Desenvolva um código que: defina um vetor com valores de -1 a 1, com incremento de 0.05, e um outro vetor y que receba os valores de x elevado ao cubo, por fim, faça um gráfico de dispersão para os dados x e y, com a cor dos pontos em verde.

R como Calculadora

O R pode operar com valores constantes numéricos e ser utilizado como uma calculadora. Os operadores aritméticos são: + (soma), - (subtração), / (divisão), * (produto), ^ (potência), %% (resto da divisão) e %/% (quociente).

Questão 1 - Efetue no R

- a) 5×3

O resultado de uma expressão aritmética pode ser armazenado em vetor por meio dos operadores **<-** (**menor, hífen**) ou **=** (**igual**). Exemplo: `a <- 1 + 2`.

Questão 2 - Dado o comando `a <- 5 * 4 + 3 ^ 3` . Assinale o valor atribuído ao objeto a.

- a) 47 b) 54 c) 29 d) 50

Vetor é um objeto capaz de armazenar vários valores em uma determinada ordem. A forma mais usual de se criar um vetor é por meio da função `c()`. Exemplo: `c(5,6,7)`.

Questão 3 - Assinale a opção correspondente ao código que atribui ao vetor **a** os valores 5, 8, 9 e 1, nessa ordem.

a) `a <- c(1,5,8,9)`b) `a <- c(5,8,9,1)`c) `A <- c(5,8,9,1)`

Operações aritméticas com vetores, com o mesmo número de elementos, são realizadas elemento a elemento. O resultado é um vetor com o mesmo **tamanho** (**length**) dos vetores iniciais.

```
> x1 <- c(2,3); x2 <- c(1,2); x3 <- x1+x2
> x3
[1] 3 5
```

Questão 4 - Escreva um código que gere vetores `x1` e `x2` com 4 elementos aleatórios. E crie um terceiro vetor que contenha o produto de `x1` e `x2`.

Expressões aritméticas que envolvem **vetores com diferentes números de elementos** têm como resultado um **vetor** com o número de elementos igual ao do vetor com o **maior** número de elementos. Os demais vetores têm seus elementos repetidos até alcançarem o mesmo número de elementos do “maior” vetor.

```
x1 <- c(2,3,4,5,6,7); x2 <- c(7,8); x3 <- x1+x2; x3
[1] 9 11 11 13 13 15
>
```

Caso haja, na expressão algum vetor cujo número de elementos não seja divisor do vetor com maior número de elementos serão emitidos alertas, mas a expressão será efetuada.

```
> x1 <- c(2,3,4,5,6); x2 <- c(7,8); x3 <- x1+x2; x3
Warning message:
In x1 + x2 :
  longer object length is not a multiple of shorter object length
[1] 9 11 11 13 13
```

Questão 5 - Escreva um código que gere vetores `x1` e `x2`, com 4 e 2 elementos quaisquer, respectivamente. Após, defina um terceiro vetor com a soma de `x1` e o dobro de `x2`.

Questão 6 - Dado que `qtd <- c(2,4,5)` e `preco <- c(19.5, 15, 20)`, escreva um código para gerar o vetor `custo`, que é definido como o produto entre `qtd` e `preco`.

Questão 7 - O volume de uma esfera de raio r é dado por $\frac{4}{3}\pi r^3$. Use o R para saber qual o volume de uma esfera de raio 5.

Questão 8 - O vetor `elsius <- 1:100` armazena um conjunto de temperaturas na escala Celsius. Converta as temperaturas armazenadas no vetor `celsius` para escala Fahrenheit. A mudança de escala solicitada é dada pela relação $TF = \frac{9}{5}TC + 32$, onde TC é a temperatura em Celsius e TF é a temperatura em Fahrenheit.

O R disponibiliza muitas funções que são úteis para analisar os dados de um vetor. Os comandos `help ()` ou `?` retornam a descrição completa de qualquer função do R. A tabela abaixo apresenta algumas funções.

Função	Valor retornado
<code>length ()</code>	quantidade de elementos
<code>min ()</code>	menor elemento.
<code>max ()</code>	maior elemento.
<code>sum ()</code>	soma dos elementos.
<code>mean ()</code>	média dos elementos.
<code>median ()</code>	mediana dos elementos.
<code>var ()</code>	variância dos elementos.
<code>sd ()</code>	desvio padrão dos elementos.

Questão 9 - Sejam os seguintes dados: { 10, 11, 30, 76, 89, 22 }. Desenvolva um código R que execute

1. Crie um vetor `x` com os dados apresentados.
2. Crie um vetor `x_escala` com os dados ajustados para um escala de 0 (zero) a 10 (dez). Dica:

$$x_{escala} = \frac{x - \text{valor mínimo}}{\text{valor máximo} - \text{valor mínimo}} \times 10$$

Questão 10 - O código `x <- rnorm(100)` cria uma amostra de uma população normal. Desenvolva um código R que execute

1. Crie o vetor `n`, que armazene o número de elementos em `x`.
2. Crie o vetor `S`, que armazene o total dos elementos em `x`.
3. Crie o vetor `x_medio` que armazene a soma dos valores em `x` dividido pelo o número de valores em `x`. Compare o valor de `x_medio` com o de `mean(x)`.

Questão 11 - O código `x <- rnorm(500,mean=5,sd=2)` retira uma amostra de tamanho 500 de uma população normal com média 5 e variância 4. Desenvolva um código R para

1. Usar as funções `mean()` e `var()` para verificar se a média e a variância da amostra em `x` têm valores próximos aos valores populacionais.
2. Criar o vetor `x_padrao` dado por

$$x_{padrao} = \frac{x - \text{media de } x}{\text{raiz quadrada da variância de } x}$$

3. Calcular a média e a variância de `x_padrao`.
4. Gerar uma nova amostra `x` com média = 10 e variância 4 (`x<-rnorm(500,10,2)`). Refaça os itens 1., 2. e 3. O que se observa com a média e a variância dos valores de `x_padrao`?

Dados no R

O R opera sobre estruturas de dados, que incluem os **vetores**, **fatores**, **matrizes**, **arrays**, **listas** e **data frames**.

O vetor é uma estrutura que comporta um conjunto ordenado de dados. O vetor `x`, que consiste dos números 10.4, 05.6, 3.1, 6.4 e 21.7, pode ser gerado pelo código:

```
x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
```

Os valores que constituem os vetores são chamados de elementos. Em função do tipo de elemento que comporta, um vetor é um dos seis tipos básicos: logical, integer, real, complex, character (string) and raw (bruto). Exemplos:

```
> # Vetor logico
> x <- c(TRUE, FALSE, T, F, 5 > 4)
> # Vetor Interiro
> y <- c(1L, 2L)
> # Vetor Real
> z <- c(1, 2)
> # Vetor Complexo
> w <- c(1+2i, 1+0i, 1i)
> # vetor caracter
> k <- c("casa", 'apartamento')
> # Vetor raw armazena seus valores em formato hexadecimal
> y <- charToRaw( "oi tudo bem")
> y
[1] 6f 69 20 74 75 64 6f 20 62 65 6d
```

Todos os elementos de um vetor são ou numéricos (integer, real, complex, raw), ou lógicos ou caracter. Caso se tente gerar um vetor com valores numéricos, ou lógicos, e com valores caracter, os valores numéricos são convertidos para caracter.

Questão 1 - Execute um código **R** que use a função `c ()` para gerar o vetor `y` com os valores {"casa", 3.5, (7 > 2)}.

Questão 2 - Há erros em `x <- c('casa', "ap", 'loja')`? Se houver, explique.

Questão 3 - Execute um código **R** que gere um vetor com todas as letras minúsculas do alfabeto. Dica: `LETTERS`.

Fator é uma estrutura útil para armazenar dados ordinais ou nominais. A função `factor()` converte um vetor em um fator , tem 3 três argumentos:

- `x`- vetor que será convertido em fator.
- `levels` – indicar as categorias ou níveis do fator.
- `labels` – nomeia os níveis.

Exemplo: Cria o fator `area` a partir do vetor numérico `area` para armazenar a informação área de conhecimento.

```
area <- c(1,1,2,1,2,2,1,1)
area <- factor ( area ,
                levels = c(1,2,3),
                labels = c("Humanas", "Exatas", "Saude"))
```

Questão 4 - Experimente os seguintes comandos, onde o fator `area` é o utilizado no texto introdutório.

1 - `mean(area)`

2 - `sum(area)`

3 - `area * 2`

Questão 5 - Crie um fator que contenha os temas: "abacaxi", "maçã", "kiwi" e "uva".

Acesso a valores

Data Frame é uma estrutura de dados retangular, na qual as colunas são as variáveis e as linhas são as observações. A forma mais comum de criar um data frame é por meio da leitura de dados armazenados em outros arquivos, como txt ou xlsx.

A função `data.frame()` cria um data frame a partir de vetores ou fatores.

```
idade <- c(18, 17, 19, 21, 22, 25, 18, NA)
area   <- c(1,1,2,1,2,2,1,1)
genero <- c("f","m","f","f","m","f","m","f")

area <- factor ( area ,
                levels = c(1,2,3),
                labels = c("Humanas", "Exatas", "Saude"))

dados <- data.frame ( area, genero, idade)
```

Pode-se extrair uma variável de um data frame digitando o nome do data frame seguido por `$` e o pelo nome da variável.

```
dados$area
```

Observações e variáveis de um data frame podem ser extraídos com o operador `[linha, coluna]`.

- `dados [x , y]` – elemento na x^a linha e y^o coluna.
- `dados [, y]` – todas as linhas da y^a coluna.
- `dados [x ,]` – todas as colunas da x^a linha.
- `dados [n]` – retorna um data frame formado pela coluna n .

Questão 1 - Digite os comandos que geraram o data frame `dados` do texto introdutório. Calcule a média da variável `area` e da variável `idade`.

Questão 2 - Execute os comandos `help(mtcars)`, `str(mtcars)` e `head(mtcars)`. Descreva os resultados obtidos.

Desvio condicionais e laços de repetição

Desvio Condicional

O comando `if(condição)` permite que um conjunto de comandos (bloco) seja executado se condição for TRUE.

Sintaxe:

```
if ( condição )  
{  
  bloco  
}
```

Exemplo:

```
x <- 4  
if ( x > 0 ) {  
  print("Número Positivo")  
}
```

O comando `if (condição) {bloco 1} else {bloco 2}` permite definir um bloco a ser executado quando condição for FALSE.

Sintaxe:

```
if ( condição )  
{  
  bloco 1  
} else {  
  bloco 2  
}
```

Exemplo:

```
x <- 4  
if ( x > 0 ) {  
  print("Número Positivo")  
} else {  
  print("Número Negativo")  
}
```

O comando `if (condição) {bloco 1} else {bloco 2} admite um outro comando if (condição k) {bloco x} else {bloco y}`.

Sintaxe:

```
if (condição 1){  
  bloco 1  
} else if (condição 2) {  
  bloco 2  
} else if (condição 3) {  
  bloco 3  
} else {  
  bloco 4  
}
```

Exemplo:

```
x <- 4  
if ( x > 0 ) {  
  print("Número Positivo")  
} else if ( x < 0 ) {  
  print("Número Negativo")  
} else {  
  print("Zero")  
}
```

Importante: A condição a ser analisada pode envolver vetores numéricos ou lógicos, mas somente o primeiro elemento do vetor é considerado. Ao ser usado vetores numéricos o valor zero é considerado FALSE e os demais TRUE. Note que o `else` sempre vem na mesma linha do `}` que encerra o bloco de comandos anterior.

Questão 1 - Escreva um programa R que apresente na tela se o valor armazenado em `x` é **par** ou **ímpar**. O valor será **par**, se e somente se, ocorrer `x %% 2 == 0`, caso contrário, o número será classificado como **ímpar**. Dica: O operador `%%` retorna o resto de uma divisão.

A função `ifelse()` tem comportamento semelhante a `if (condição) {bloco 1} else { bloco 2}`, porém a condição é verificada em todos os elementos de um vetor.

```
a = c(5,7,2,9)
ifelse(condição,x,y)      ifelse(a %% 2 == 0,"par","ímpar")
                           [1] "ímpar" "ímpar" "par" "ímpar"
```

O vetor resposta recebe o elemento `x` se a condição for verdadeira e `y` se a condição for falsa. De outra maneira, o elemento `i` do resultado irá ser o elemento `x[i]` se a condição verificada for TRUE, caso contrário o resultado recebe o valor `y[i]`.

Questão 2 - O saldo dos clientes do Banco BSEI estão armazenados no vetor `saldo`. O gerente (você) solicita ao setor de tecnologia a segmentação dos clientes ou na categoria **elite**, saldo superior R\$ 5.000,00, ou na categoria **executivo**, para os demais valores de saldos. Em 15 minutos, a tecnologia lhe retorna o vetor `segmento` para a sua homologação. Use a estatística descritiva para homologar o trabalho da tecnologia. Caso o vetor `segmento` não seja homologado, escreva um código R que gere a segmentação correta.

Importante: Os vetores `saldo` e `segmento` estão armazenados no arquivo `BSEI.RData`.

Dicas: `load()`, `setwd()`, `ifelse()`, `tapply()`, `mean()`, `min()`, `max()` e `boxplot()`.

Questão 3 - O data frame `feijao` em sua variável `m` registra a massa, em gramas, de `n` grãos de um cultivar de feijão. Então,

1. Determine a quantidade de elementos da variável `m`.
2. Faça um histograma e um boxplot para esses dados.
3. Obtenha a média, desvio padrão, o coeficiente de variação e dos decis 10, 50, 70, 90.
4. Classifique os grãos em "A" se a sua massa for maior ou igual a 95% dos outros grãos e em "B" os demais.

Importante: Os dados do data frame `feijão` está no arquivo `feijao.csv`

Dicas: `read.csv2()`, `setwd()`, `ifelse()`, `hist()`, `mean()`, `sd()`, `max()`, `quantile()`, `length()` e `boxplot()`.

Laços de Repetição

O comando `for()` repete um bloco de comandos até que certa variável assuma todos os valores de uma sequência.

Sintaxe:

```
for ( val in vet )  
{  
  bloco  
}
```

Exemplo:

```
# conta pares  
x <- c(2,5,3,9,8,11,6)  
t <- 0  
for (val in x) {  
  if(val %% 2 == 0) t = t + 1  
}  
print(t)
```

A `vet` é um vetor, que pode ser resultado de uma função como a `seq()`, `c()` ou pelo operador `:`.

No exemplo, conta-se a quantidade de elementos do vetor `x` que é um número **par**. O comando `for()` faz a variável `val` assumir todos os valores de `x`, um por vez, e caso o valor em `val` seja **par** a variável `t` é incrementada em 1.

Questão 1 - Crie o vetor `x` com 5 elementos escolhidos aleatoriamente do intervalo de 1 até 100, use da função `sample()`. Dessa amostra, apresente os elementos de `x` e a quantidade de números **ímpares** que contém.

O comando `while()` repete um bloco de comandos até certa condição assuma valor `FALSE`. Em outras palavras o bloco é repetido enquanto a condição for `TRUE`.

Sintaxe:

```
while (condicao) {  
  bloco  
}
```

Exemplo:

```
x <- 1  
while (x < 6) {  
  print(x)  
  x <- x + 1  
}
```

Observe que o valor de `x` foi inicializado com um valor que garanta ao menos uma execução do bloco. Note, também, que deve haver a certeza que `condicao` assumirá o valor `FALSE`, em algum momento.

Questão 1 - O fatorial de número natural $n \geq 1$ é o produto de todos os naturais de 1 a n . O fatorial de 4 é $1 \times 2 \times 3 \times 4 = 24$. Siga os passos para calcular o fatorial de 7 usando o comando `while()`.

1. Inicialize a variável `n` com o valor 7.
2. Inicialize a variável `fatorial` com o valor 1.
3. Inicialize a variável `i` com o valor 1.
4. Enquanto $i \leq n$ faça
 - i Atribua a `fatorial` o valor `fatorial * i`
 - ii Incremente `i`
5. Print o valor de `fatorial`

Listas

Uma lista R é um objeto que consiste em uma coleção ordenada de objetos, conhecidos como **componentes**. Os componentes não precisam ser do mesmo modo ou tipo, por exemplo, uma lista pode consistir de um vetor numérico, um valor lógico, uma matriz, um vetor complexo, uma matriz de caracteres, uma função.

A função `list()` gera uma lista, como no código abaixo.

```
> Lst <- list (nome = "Fred", esposa = "Maria", no.filhos = 3,
              idade.filhos = c (4,7,9))
```

Os componentes são numerados e podem ser referidos como tal. Assim, se `Lst` é o nome de uma lista com quatro componentes, estes podem ser individualmente referidos como `Lst [[1]]`, `Lst [[2]]`, `Lst[[3]]` e `Lst[[4]]`. Se, além disso, `Lst [[4]]` é um array com vetor subscrito, então `Lst[[4]][1]` é sua primeira entrada.

Se `Lst` é uma lista, então a função `length(Lst)` fornece o número de componentes (de nível superior) que possui.

Os componentes de listas também podem ser nomeados e, nesse caso, o componente pode ser referenciado pelo nome do componente, como uma cadeia de caracteres, no lugar do número entre colchetes duplos ou, mais convenientemente, como na expressão.

`nome_lista$nome_do_componente`

têm o mesmo efeito.

Esta é uma convenção muito útil, pois torna mais fácil obter o componente certo se você esquecer o número. Então, no exemplo acima:

- `Lst$nome` é o mesmo que `Lst[[1]]` e é a string "Fred",
- `Lst$esposa` é o mesmo que `Lst[[2]]` e é a string "Maria",
- `Lst$idade.filhos[1]` é o mesmo que `Lst [[4]][1]` e é o número 4.

Além disso, também é possível usar os nomes dos componentes da lista em colchetes duplos, isto é, `Lst [["nome"]]` é o mesmo que `Lst$ nome`. Isso é especialmente útil quando o nome do componente a ser extraído é armazenado em outra variável, como em

```
> x <- "nome"; Lst[[x]]
```

É muito importante distinguir `Lst[[1]]` de `Lst[1]`. `'[[...]]'` é o operador usado para selecionar um único elemento, enquanto `'[...]` é um operador de subscrição geral. Assim, o primeiro é o primeiro objeto na lista `Lst` e, se for uma lista nomeada, o nome não será incluído. Este último é uma sub-lista da lista `Lst`, consistindo apenas na primeira entrada. Se for uma lista nomeada, os nomes são transferidos para a sublista.

Os nomes dos componentes podem ser abreviados até o número mínimo de letras necessárias para identificá-los exclusivamente. Assim, o componente `Lst$coeficientes` podem ser especificados minimamente como `Lst$coe` e `Lst$covariância` como `Lst$cov`.

O vetor `names` é, na verdade, simplesmente um atributo da lista como qualquer outro e pode ser tratado como tal. Outras estruturas além das listas podem, é claro, receber um atributo de nomes similarmente.

Questão 1 - A função `lm()` é usada em análises de modelos lineares, por exemplo, uma regressão linear simples. Os resultados obtidos são organizados em uma lista. Execute o código R abaixo:

```
help("iris")
data("iris")
fit<- lm(iris$Petal.Length~iris$Petal.Width)
summary.lm(fit)
names(fit)
length(fit)
plot (iris$Petal.Width, iris$Petal.Length , pch=16, col = "skyblue")
points(iris$Petal.Width, fit$fitted.values, pch=16 , col="orange")
```

Responda:

- i- Qual é o conteúdo do data frame `iris`?
- ii- Quantos são os componentes lista `fit` e quais os seus nomes?
- iii- Qual o resultado da função `plot()` e `points()`
- iv- Altere os títulos dos eixos x e y para `largura` e `comprimento`, respectivamente. Dica: `?plot`.
- v- Calcule a média dos valores em `fit$residuals` e faça um histograma com esses valores, `hist()`.
- vi- Qual é a soma dos quadrados dos resíduos.

Questão 2 - A expressão `x <- rnorm(1500)` retorna um vetor com 1500 números aleatórios retirados de uma distribuição normal com média 0 e desvio padrão 1.

Faça:

- i- Leitura rápida do help da função `hist()`.
- ii- Calcule a média e variância de `x`, verifique se os valores são próximos aos esperados teoricamente.
- iii- Armazene na variável `minimo` o menor valor observado arredondado para o menor inteiro em `x`, e na variável `maximo` o maior valor observado arredondado para o maior inteiro.
- iv- Armazene em `k` a raiz quadrado do número de elementos em `x` arredondado para menos.
- v- Crie o vetor `limites` que contenha `k` valores equidistantes que iniciam em `minimo` e terminem em `maximo`. Use a função `seq()`
- vi- Faça um histograma com os valores de `x` com as seguintes características:
 - `breaks=limites`.
 - `xlab=' '`, `main=' '`, `ylab=' '` e `axes=F`
- vii- Após a função `hist()` execute o comando `axis(1, at = limites, labels = round(limites,2))`