Minicurso RStudio

PET Estatística UFSCar

Universidade Federal de São Carlos - UFSCar
Centro de Ciências Exatas e Tecnológicas - CCET
Departamento de Estatística - DEs

2 de setembro de 2019

Sumário

O R é uma linguagem e ambiente de computação estatística. Considerado uma variante da linguagem S (laboratórios Bell, desenvolvida por John Chambers e seus colegas), surgiu pela criação da R Foundation for Statistical Computing, com o objetivo de criar uma ferramenta gratuita e de utilização livre para análise de dados e construção de gráficos.

Instalando o R

Para instalar o R no Windows, é necessário entrar no link a seguir para fazer download do instalador com a versão mais atualizada do R: *https://cran.rproject.org/bin/windows/base/* após baixar, salve o arquivo em qualquer pasta do seu computador.

Com o arquivo já salvo no computador, agora vamos instalar o nosso software, clique duas vezes para instalar o arquivo, em seguida em avançar até aparecer a seguinte tela:

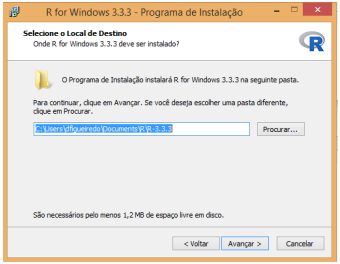


Figura 1: Programa de instalação do R

Continue clicando em "Avançar" até chegar no botão "Concluir".

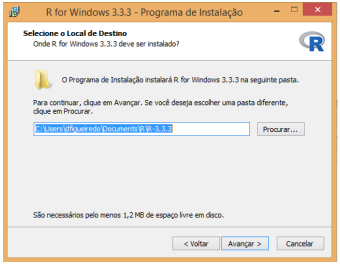


Figura 1: Programa de instalação do R

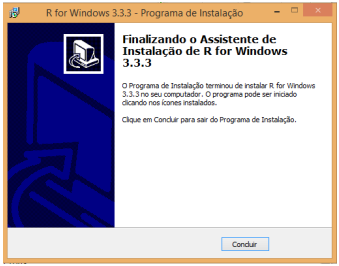


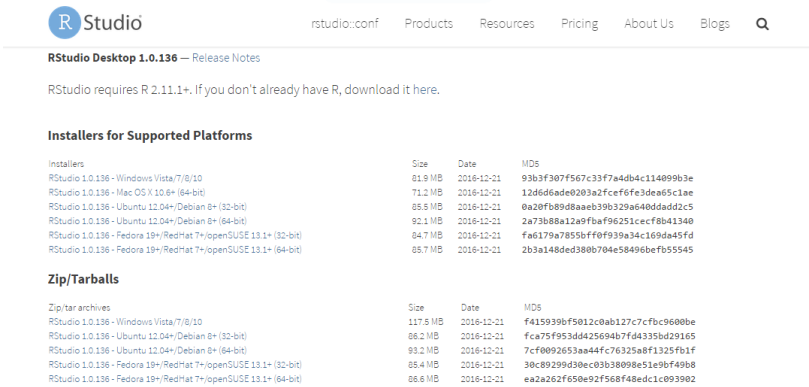
Figura 2: Programa de instalação do R

Continue clicando em "Avançar" até chegar no botão "Concluir".

Assim, já temos o R instalado em nossa máquina.

Instalando do RStudio

Agora vamos instalar o RStudio, que é onde iremos trabalhar com toda a parte de programação. Para fazer o download é necessário acessar o site do RStudio pelo link: ***https://www.rstudio.com/products/rstudio/download/***.



RStudio rstudio::conf Products Resources Pricing About Us Blogs

RStudio Desktop 1.0.136 — Release Notes

RStudio requires R 2.11.1+. If you don't already have R, download it here.

Installers for Supported Platforms

Installers	Size	Date	MD5
RStudio 1.0.136 - Windows Vista/7/8/10	81.9 MB	2016-12-21	93b3f307f567c33f7a4db4c114099b3e
RStudio 1.0.136 - Mac OS X 10.6+ (64-bit)	71.2 MB	2016-12-21	12d6d6ade0203a2fcef6fe3dea65c1ae
RStudio 1.0.136 - Ubuntu 12.04+/Debian 8+ (32-bit)	85.6 MB	2016-12-21	0a20fb89d8aaeb39b329a640ddadd2c5
RStudio 1.0.136 - Ubuntu 12.04+/Debian 8+ (64-bit)	92.1 MB	2016-12-21	2a73b88a12a9fbaf96251cecf8b41340
RStudio 1.0.136 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	84.7 MB	2016-12-21	fa6179a7855bff0f939a34c169da45fd
RStudio 1.0.136 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	85.7 MB	2016-12-21	2b3a148ded380b704e58496befb55545

Zip/Tarballs

Zip/tar archives	Size	Date	MD5
RStudio 1.0.136 - Windows Vista/7/8/10	117.5 MB	2016-12-21	f415939bf5012c0ab127c7cfbc9600be
RStudio 1.0.136 - Ubuntu 12.04+/Debian 8+ (32-bit)	86.2 MB	2016-12-21	fca75f953dd425694b7fd4335bd29165
RStudio 1.0.136 - Ubuntu 12.04+/Debian 8+ (64-bit)	93.2 MB	2016-12-21	7cf0092653aa44fc76325a8f1325fb1f
RStudio 1.0.136 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	85.4 MB	2016-12-21	30c89299d30ec03b38098e51e9bf49b8
RStudio 1.0.136 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	86.6 MB	2016-12-21	ea2a262f650e92f568f48edc1c093902

Source Code

A tarball containing source code for RStudio v1.0.136 can be downloaded from here

Figura 3: Site para download do RStudio

Na imagem acima temos dois possíveis casos:

Na imagem acima temos dois possíveis casos:

1. Se for administrador é só escolher a versão compatível com o sistema operacional na guia *Installers for Supported Platforms*. Em seguida a instalação será bem simples, apenas clicando em "Avançar".

Na imagem acima temos dois possíveis casos:

1. Se for administrador é só escolher a versão compatível com o sistema operacional na guia *Installers for Supported Platforms*. Em seguida a instalação será bem simples, apenas clicando em "Avançar".

2. Caso não seja administrador, será necessário baixar o arquivo compactado na guia *Zip/Tarballs*. Após extrair a pasta baixada, terá que ir na subpasta *bin*, e executar o arquivo rstudio.

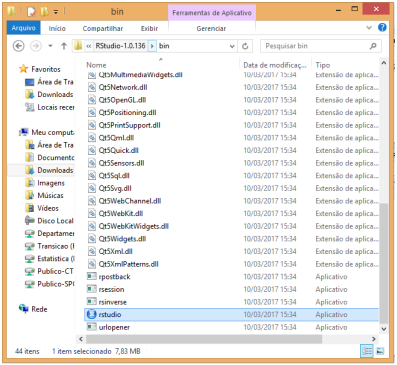


Figura 4: Arquivo de execução do RStudio

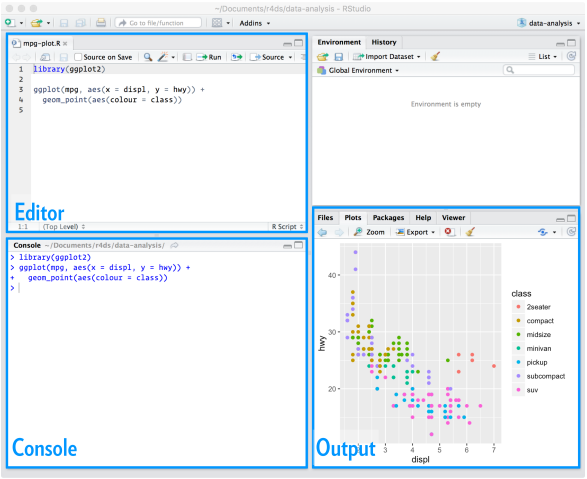


Figura 5: Ambiente de trabalho do RStudio

Vimos a interface com 4 quadrantes, com as seguintes funções:

Vimos a interface com 4 quadrantes, com as seguintes funções:

Editor/Script: É onde escrevemos os códigos.

Vimos a interface com 4 quadrantes, com as seguintes funções:

Editor/Script: É onde escrevemos os códigos.

Console: É onde rodamos os códigos e recebemos as saídas.

Vimos a interface com 4 quadrantes, com as seguintes funções:

Editor/Script: É onde escrevemos os códigos.

Console: É onde rodamos os códigos e recebemos as saídas.

Environment: Painel com todos os objetos criados no R, ao lado tem a aba *History* que contém o histórico com todos os comandos utilizados anteriormente.

Vimos a interface com 4 quadrantes, com as seguintes funções:

Editor/Script: É onde escrevemos os códigos.

Console: É onde rodamos os códigos e recebemos as saídas.

Environment: Painel com todos os objetos criados no R, ao lado tem a aba *History* que contém o histórico com todos os comandos utilizados anteriormente.

Output: Responsável por toda comunicação do R, onde temos a aba *Files* onde mostra os diretórios dos trabalhos. Já a aba *Plots* é toda a saída gráfica, ou seja, qualquer gráfico que for criado na compilação é exibido nesta aba, já o *Help* exibe todas as documentações das funções.

Instalação de pacotes

Para realizarmos algumas tarefas no R, são necessários alguns pacotes específicos, vamos por exemplo instalar o pacote *ggplot2*, que é utilizado para a elaboração de gráficos mais refinados, para instalá-lo basta digitar:

Códigos

```
install.packges("ggplot2")
library(ggplot2)
```

Uma outra maneira de instalar pacotes no R é manualmente. Para isso é só clicar em *Tools* na parte superior do programa e em seguida em *Install Packages*. Com isso irá aparecer uma janela com um espaço disponível para digitar o nome do pacote que se deseja baixar. Após instalado é necessário o carregamento do pacote. Isso é feito por meio da função *library()*, como já visto acima.

Criando conjunto de dados

Existe basicamente duas maneiras diferentes de criar um conjunto de dados no R. Os dados podem ser digitados em uma planilha dentro do próprio R ou ainda podemos importar um arquivo contendo o banco de dados que já esteja salvo no computador.

Criando conjunto de dados

Digitando os Dados

Quando é apenas uma única variável que iremos criar, o comando *scan()* é suficiente para criar o conjunto. Exemplo:

Códigos

```
dados = scan()
    1: 12
    2: 13
    3: 14
    4: 2
    5: 10
    6: 6
    7: 9
    8:
Read 7 items
```

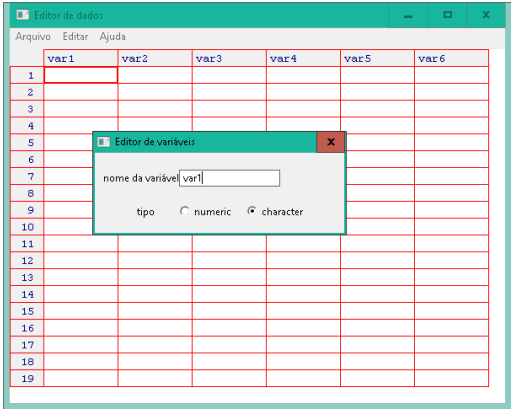


Figura 6: Editor de texto

Se o objetivo é criar um conjunto com várias variáveis faz-se necessário a utilização do comando ***edit(data.frame())***. Esse comando faz com que abra uma planilha para que os dados sejam digitados, inclusive nomeando as variáveis. Exemplo:

Códigos

```
dados = edit(data.frame())
```

Nesta planilha, o nome da variável pode ser alterado clicando duas vezes do nome var1, e assim sucessivamente. Depois de digitar os dados basta fechar a planilha para ter todos os valores salvos dentro do R. Caso seja necessário editar os dados, ou acrescentar variáveis basta utilizar o comando *fix(dados)* que abrirá a tela vista anteriormente para edição dos valores.

Importando os dados

Se existe um conjunto de dados salvo computador, e deseja utilizar para manipulação dentro do R é possível importar este arquivo. O comando utilizado para fazer essa importação é o *read.table()*. O R permite importar arquivos em diferentes formatos entre eles, xlss, csv, txt, entre outros.
Considere um arquivo de dados salvo como "exemplo" em formato txt e que está na pasta Minicurso R da área de trabalho do usuário, o código referente para a importação é:

Códigos

```
exemplo=read.table("C:Documents and Settings\\Administrador\\
                    Meus documentos\\exemplo.txt",header=T)
```

Entendendo um pouco mais como importar um arquivo, dentre os parâmetros que utilizamos na função *read.table()*, o primeiro é o diretório do arquivo entre aspas. Outro parâmetro é o *header=TRUE* se o conjunto de dados contém o nome da variável na primeira linha do banco, caso não tenha basta colocar igual a *FALSE* ou simplesmente não declarar nada.

Outros parâmetros que podemos declarar é, se o conjunto de dados contém dados com casa decimais, podemos especificar se é "."(ponto) ou ","(vírgula). Caso os dados de uma mesma linha estejam separados por tabulação, ponto e vírgula, espaço entre outros também deve ser especificado. Considere agora um arquivo com casas decimais separados por vírgula, e que a separação dos valores esteja por tabulação. O código ficará da seguinte maneira:

Códigos

```
exemplo = read.table(“C:Documents and Settings\\Administrador\\
                     Meus documentos\\exemplo.txt’’,header=T,
                     dec=’’,’’, sep=’’\t’’)
```

Caso o diretório do arquivo seja muito grande, não é interessante escrevê-lo, neste caso podemos utilizar o comando *file.choose()*. Usando este comando aparecerá uma janela, na qual você irá escolher o caminho onde se encontra o arquivo. Exemplo:

Códigos

```
dados = read.table(file.choose(),header=T, dec='',', sep=''\t'')
```

Caso o diretório do arquivo seja muito grande, não é interessante escrevê-lo, neste caso podemos utilizar o comando *file.choose()*. Usando este comando aparecerá uma janela, na qual você irá escolher o caminho onde se encontra o arquivo. Exemplo:

Códigos

```
dados = read.table(file.choose(),header=T, dec='',', sep=''\t'')
```

Alguns outros casos são:

Códigos

```
dados2 = read.csv(file.choose(), header=T, dec=‘,’)
dados3 = read.csv2(file.choose(), header=T, dec=‘.’, sep=''\t'')
```

R como calculadora

Pelo console, é possível fazer inúmeros comandos no R. Temos as operações básicas de matemática como: soma, subtração, divisão, potência, entre outras funções. A seguir temos alguns exemplos:

Operação	Código
Soma	+
Subtração	-
Multiplicação	*
Divisão	/
Parte inteira da divisão	%/%
Resto da divisão	%%
Potência	^
Raiz quadrada	sqrt(n)

Tabela 1: Operações matemáticas

Objetos

Um objeto na linguagem de programação, consiste em qualquer elemento que pode receber um determinado valor, ou característica, como funções, entre outros. O R permite salvar dados dentro de um objeto, para isso é utilizado o operador "=" ou "<-" na qual simboliza atribuição.

Códigos

```
valor = 2
valor <- 2
```

Deste modo o objeto valor passou a possuir o valor 2, ou seja, toda vez que ele executar um código e deparar com o símbolo "valor", automaticamente substituíra por 2.

Vetores

Vetores no R são objetos mais simples no que armazena mais de um valor. Para criarmos um vetor basta utilizar a função c(). No R os valores são separados por vírgula, enquanto que as casa decimais são separadas por ".".

Vetores

Vetores no R são objetos mais simples no que armazena mais de um valor. Para criarmos um vetor basta utilizar a função c(). No R os valores são separados por vírgula, enquanto que as casa decimais são separadas por ".".

Códigos

```
vetor1 = c(1, 2, 3, 4, 5, 6, 7,
           8, 9, 10)
vetor2 = c(1.1, 2.1, 3, 4, 5, 6,
           7.5, 9, 10.1)
```

Vetores

Vetores no R são objetos mais simples no que armazena mais de um valor. Para criarmos um vetor basta utilizar a função c(). No R os valores são separados por vírgula, enquanto que as casa decimais são separadas por ".".

Códigos

```
vetor1 = c(1, 2, 3, 4, 5, 6, 7,
           8, 9, 10)
vetor2 = c(1.1, 2.1, 3, 4, 5, 6,
           7.5, 9, 10.1)
```

Para realizar operações, é necessário que os vetores tenham o mesmo tamanho onde usamos a função *length()*.

Códigos

```
x = c(1,2,3,4,5,6)
y = c(3,4,5,6,7,8)

length(x)
length(y)
```

Matrizes

Matizes são vetores de duas dimensões. Para criarmos matrizes o comando utilizado é *matrix()*. No comando dessa função os argumentos mais importantes são: *nrow* e *ncol*, que especificam o número de linhas e colunas respectivamente. Outro argumento não menos importante é o *byrow*, quando utilizarmos o *byrow=TRUE* o R preencherá a matriz por linha.

Matrizes

Matizes são vetores de duas dimensões. Para criarmos matrizes o comando utilizado é *matrix()*. No comando dessa função os argumentos mais importantes são: *nrow* e *ncol*, que especificam o número de linhas e colunas respectivamente. Outro argumento não menos importante é o *byrow*, quando utilizarmos o *byrow=TRUE* o R preencherá a matriz por linha.

Códigos

```
m = matrix(c(1,2,3,4,5,6,7,8,9), nrow=3, ncol=3, byrow = T)
n = matrix(c(1,2,3,4,5,6,7,8,9), nrow=3, ncol=3, byrow = F)
o = matrix(c(1,2,3,4,5,6,7,8,9), 3, 3, byrow = T)
```

R como calculadora

Vale ressaltar que um banco de dados pode ser representado como uma matriz, e se houver a necessidade de nomear as colunas e linhas temos duas maneira com os seguintes comandos:

R como calculadora

Vale ressaltar que um banco de dados pode ser representado como uma matriz, e se houver a necessidade de nomear as colunas e linhas temos duas maneira com os seguintes comandos:

Primeira maneira

```
dimnames(m) = list(
   vertical = c("obs1","obs2",
                "obs3"),
   horizontal = c("var1", "var2",
                  "var3"))
```

R como calculadora

Vale ressaltar que um banco de dados pode ser representado como uma matriz, e se houver a necessidade de nomear as colunas e linhas temos duas maneira com os seguintes comandos:

Primeira maneira

```
dimnames(m) = list(
   vertical = c("obs1","obs2",
                "obs3"),
   horizontal = c("var1", "var2",
                  "var3"))
```

Segunda maneira

```
rownames(m) = c("obs1","obs2",
                "obs3")
colnames(m) = c("col1","col2",
                "col3")
```

R como calculadora

Vale ressaltar que um banco de dados pode ser representado como uma matriz, e se houver a necessidade de nomear as colunas e linhas temos duas maneira com os seguintes comandos:

Primeira maneira

```
dimnames(m) = list(
   vertical = c("obs1","obs2",
                "obs3"),
   horizontal = c("var1", "var2",
                  "var3"))
```

Segunda maneira

```
rownames(m) = c("obs1","obs2",
                "obs3")
colnames(m) = c("col1","col2",
                "col3")
```

Vale notar que uma matriz não é, de fato, um banco de dados. Alguns comandos só são aplicados convertendo essas matrizes em matriz para estrutura de dados. Para tal é utilizado a seguinte função: matriz1 = as.data.frame(m). Assim matriz1 recebe os elementos da matriz "m"no formato de dados.

Operações com matrizes

Quando trabalhamos com matrizes, podemos nos deparar com a necessidade de realizar alguma operação, a seguir apresentaremos algumas já utilizadas, além daquelas já vista anteriormente.

Para verificar o tamanho da matriz temos alguns comandos como: *dim(m):* retorna o número de linhas e colunas da matriz m, *nrow(m):* retorna o número de linhas da matriz m e *ncol(m):* retorna o número de colunas da matriz m.

- m[3,] – Seleciona a terceira linha.
- m[,4] – Seleciona a quarta coluna.
- m[1,2] – Seleciona o primeiro elemento da segunda coluna.

- m[3,] – Seleciona a terceira linha.
- m[,4] – Seleciona a quarta coluna.
- m[1,2] – Seleciona o primeiro elemento da segunda coluna.
- t(m) – Calcula a matriz transposta da matriz m.

- m[3,] – Seleciona a terceira linha.
- m[,4] – Seleciona a quarta coluna.
- m[1,2] – Seleciona o primeiro elemento da segunda coluna.
- t(m) – Calcula a matriz transposta da matriz m.
- m %*% n – Calcula o produto matricial entre as matrizes m e n.

- m[3,] – Seleciona a terceira linha.
- m[,4] – Seleciona a quarta coluna.
- m[1,2] – Seleciona o primeiro elemento da segunda coluna.
- t(m) – Calcula a matriz transposta da matriz m.
- m %*% n – Calcula o produto matricial entre as matrizes m e n.
- solve(m) – Calcula a inversa da matriz m.

Valores especiais

Existem ainda alguns valores especiais que representam valores faltantes, infinitos, e indefinições matemáticas.

Valores especiais

Existem ainda alguns valores especiais que representam valores faltantes, infinitos, e indefinições matemáticas.

NA (Not Available) representa dados faltantes no conjunto de observação. Pode ser tanto caractere ou numérico.

Valores especiais

Existem ainda alguns valores especiais que representam valores faltantes, infinitos, e indefinições matemáticas.

NA (Not Available) representa dados faltantes no conjunto de observação. Pode ser tanto caractere ou numérico.

NaN (Not a Number) representa indefinições matemáticas, como divisão por zero, logaritmo de número negativo, podemos notar que um NaN é umm NA, mas a volta não é verdadeira.

Inf (Infinito) representa um número muito grande ou um limite matemático. O oposto também existe. Ex: (-Inf).

Inf (Infinito) representa um número muito grande ou um limite matemático. O oposto também existe. Ex: (-Inf).

NULL utilizamos para representar a ausência de observação, temos que a diferença entre um NULL e NA ou NaN é bem leve, enquanto os NA estão relacionados a análise estatística, o que seria a ausência de uma observação no conjunto de dados, o NULL está associado a lógica de programação.

Para testar se o elemento possui algum valor especial, utilizamos as seguintes funções: is.na(), is.nan(), is.infinite() e is.null().

Para testar se o elemento possui algum valor especial, utilizamos as seguintes funções: is.na(), is.nan(), is.infinite() e is.null().

Códigos

```
x = c(1,2,3,NA,NULL,NaN,4,Inf)
b=NULL

is.na(x)
is.nan(x)
is.infinite(x)
is.null(b)
```

Listas

A lista é um vetor especial que pode armazenar valores de classes diferentes. É um dos objetos mais importantes para armazenar dados, então é interessante saber manusear bem.

Para criar uma lista utilizamos o comando *list()* na qual aceita qualquer tipo de objeto dentro da lista, é importante destacar que é possível incluir uma lista dentro de uma lista, a seguir temos um exemplo de lista onde seus objetos são numéricos, caracteres, lógicos e vetor.

Listas

A lista é um vetor especial que pode armazenar valores de classes diferentes. É um dos objetos mais importantes para armazenar dados, então é interessante saber manusear bem.

Para criar uma lista utilizamos o comando *list()* na qual aceita qualquer tipo de objeto dentro da lista, é importante destacar que é possível incluir uma lista dentro de uma lista, a seguir temos um exemplo de lista onde seus objetos são numéricos, caracteres, lógicos e vetor.

Códigos

```
x = list(1:5, "Z", TRUE, c("a", "b"))
x[[5]] = list(c(1:10), c("PET", "EJE", "SEst"))
```

Criando funções

Em algumas tarefas, faz-se necessário criarmos funções mais específicas, e temos a vantagem de personalizá-las da melhor maneira para a realização da tarefa. Vejamos um exemplo de uma função que o usuário utiliza como *input* um número e a saída dela é a soma desse número com 10:

Criando funções

Em algumas tarefas, faz-se necessário criarmos funções mais específicas, e temos a vantagem de personalizá-las da melhor maneira para a realização da tarefa. Vejamos um exemplo de uma função que o usuário utiliza como *input* um número e a saída dela é a soma desse número com 10:

Códigos

```
SomaUm = function(x) {
    y = x+1
    return(y)
}
```

Códigos

```
media = function(x) {
  soma = sum(x)
  n_obs = length(x)
  media = soma/n_obs

  return(media)
}
```

No exemplo, vamos salvar a função *SomaUm* com o nome *funcoes.r* em uma pasta no desktop. Feito isso, basta abrir um novo código e digitar o seguinte comando:

Códigos

```
source("C:\\Users\\PET_01\\Documents\\Membros do PET
       \\Clézio Lopes\\funcoes.r")
```

Muito Obrigado!

Tabela de Contingência

As tabelas de contingências são usadas para registrar observações independentes de duas ou mais variáveis aleatórias, normalmente qualitativas.

Códigos

```
fumante<-c("Sim","Não","Sim","Sim","Sim")
sexo <- c("M","F","M","M","F")

table(fumante,sexo)
```

Média Aritmética

A média aritmética de um conjunto de valores é calculado pela função *mean()*.

Códigos

```
x=c(22,32,46,57,10,12,2)
mean(x)
```

Mediana

A mediana ou a observação central de um conjunto de dados é obtida através do comando *median()*.

Códigos

```
x=c(22,32,46,57,10,12,2)
median(x)

x=sort(x)
median(x)
```

Mediana

A mediana ou a observação central de um conjunto de dados é obtida através do comando *median()*.

Códigos

```
x=c(22,32,46,57,10,12,2)
median(x)

x=sort(x)
median(x)
```

Obs.: O comando *sort()* ordena o vetor em ordem crescente. Não é necessário ordenar o vetor x para utilizar o comando *median()*.

Variância

A variância é uma medida de dispersão dos dados em torno da média e é obtida pelo comando *var()*.

Códigos

```
x=1:10
var(x)

y=rep(1,10)
var(y)
```

Variância

A variância é uma medida de dispersão dos dados em torno da média e é obtida pelo comando *var()*.

Códigos

```
x=1:10
var(x)

y=rep(1,10)
var(y)
```

Desvio Padrão

O desvio padrão é a raiz quadrada da variância e pode ser calculada tanto manualmente quanto pelo comando *sd()*.

Códigos

```
x=1:10
var(x)

sqrt(var(x))
sd(x)
```

Resumo de dados

Para obter um resumo geral das medidas descritivas de um conjunto de dados, podemos utilizar o comando *summary()*.

Códigos

```
x=c(4,10,3,0.1,19,55,
    6,4,21,12,23,39)
summary(x)
```

Resumo de dados

Para obter um resumo geral das medidas descritivas de um conjunto de dados, podemos utilizar o comando *summary()*.

Códigos

```
x=c(4,10,3,0.1,19,55,
    6,4,21,12,23,39)
summary(x)
```

Para obtermos os valores dos quantis e os valores de máximo e mínimo, podemos utilizar os comandos *quantile()*, *max()*, e *min()*.

Códigos

```
x=c(4,10,3,0.1,19,55,
    6,4,21,12,23,39)
quantile(x)

max(x)
min(x)
```

Gráficos

Os gráficos do R normalmente são mais interessantes do ponto de vista gráfico do que de outros pacotes estatísticos, devido sua grande versatilidade em recursos. O R possui uma enorme capacidade para gerar diversos tipos de gráficos de alta qualidade totalmente configuráveis, desde cores e tipos de linhas, até legendas e textos adicionais. Uma desvantagem é que para construir gráficos complexos demanda tempo e requer muitos detalhes de programação, porém, adquire-se isso com a prática.

Neste tópico será exibido tópicos os gráficos de barra, pizza, histograma, *boxplot*, e gráfico de pontos (ou gráfico de dispersão).

Argumentos gráficos

Para construção de gráficos no R ou o uso de qualquer função, é necessário conhecer os argumentos das mesmas. Estes permitem a construção de gráficos de forma mais completa, usando títulos, cores, etc.

Argumentos gráficos

Para construção de gráficos no R ou o uso de qualquer função, é necessário conhecer os argumentos das mesmas. Estes permitem a construção de gráficos de forma mais completa, usando títulos, cores, etc.

Nome	Descrição
xlab	Nome do eixo x
ylab	Nome do eixo y
main	Título do gráfico
xlim	(início, fim) vetor contendo os limites do eixo x
ylim	(início, fim) vetor contendo os limites do eixo y
col	Cor de preenchimento do gráfico, pode ser um vetor

Tabela 2: Argumentos gráficos

Gráfico de barras

São gráficos em que visualiza a frequência através de retângulos, sendo uma das suas dimensões proporcional à frequência. Para fazer gráficos de barras no R e função é *barplot()*.

Gráfico de barras

São gráficos em que visualiza a frequência através de retângulos, sendo uma das suas dimensões proporcional à frequência. Para fazer gráficos de barras no R e função é *barplot()*.

Códigos

```
x = c(56,44)
names(x) = c("Casado","Solteiro")
barplot(x,col=c("blue","red"), ylim=c(0,60), space=1,
        main="Número de filhos por estado civil",
        xlab="Estado Civil", ylab="Número de Filhos")
text(locator(n=2),c("56%","44%"))
```

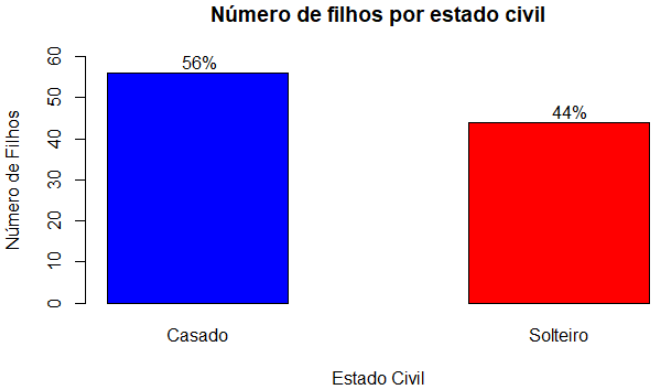


Figura 7: Gráfico de barras

Códigos

```
barplot(c(4500,3000,1200), col=c("blue","red","brown"),
        names.arg=c("Estatístico", "Engenheiro", "Matemático"),
        xlab= "Profissão", ylab="Salário", ylim=c(0,5000))
title(main="Salário da Empresa Xport")
```

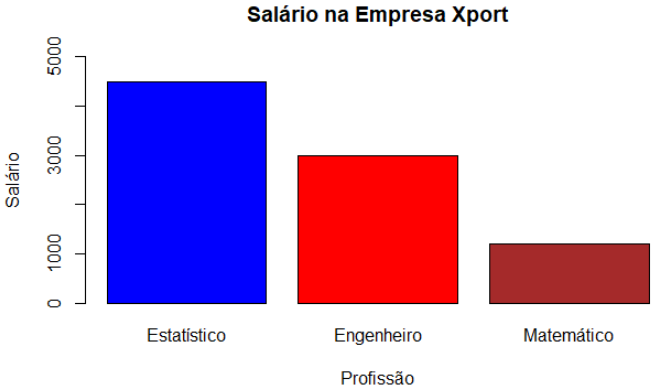


Figura 8: Gráfico de barras

Gráfico de Pizza

Os gráficos de setores ou de pizza, como é mais comumente chamado, também mostram a frequência ou proporção através de fatias de uma circunferência. Para fazer gráficos de pizza utiliza-se a função *pie()*.

Gráfico de Pizza

Os gráficos de setores ou de pizza, como é mais comumente chamado, também mostram a frequência ou proporção através de fatias de uma circunferência. Para fazer gráficos de pizza utiliza-se a função *pie()*.

Códigos

```
venda = c(0.12, 0.3, 0.26, 0.16, 0.04, 0.12)
names(venda) = c("Morango","Holandesa","Mousse Maracujá",
                 "Paulista","outras","Chocolate")
pie(venda, col= c("red", "purple", "green3", "cornsilk", "cyan",
                  "brown4"))
title(main="Tortas mais vendidas")
```

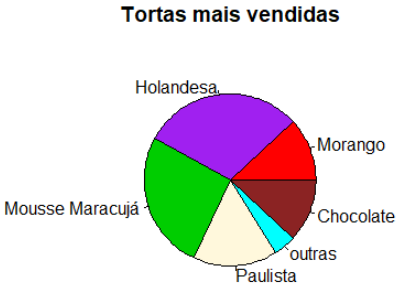


Figura 9: Gráfico de pizza

Códigos

```
educa = c(0.55,0.26,0.19)
pie(educa,labels=c("Ensino Médio","Ensino Fundamental",
                   "Ensino Superior"),
    col=c("DeepSkyBlue","Maroon1","Plum1"), radius=1.1,
    main="Grau de instrução")
text(locator(n=3),c("55%","26%","19%"))
```

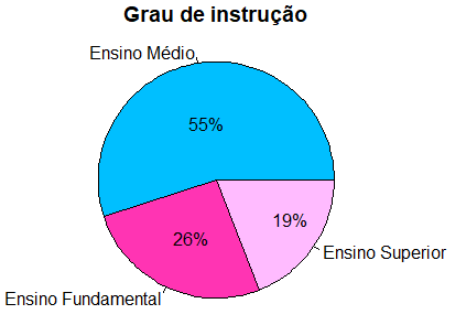


Figura 10: Gráfico de pizza

Gráfico Histograma

O histograma é uma representação gráfica da distribuição de frequência de uma determinada variável, normalmente um gráfico de barras verticais. O histograma é utilizado quando a variável em estudo é "contínua".

Gráfico Histograma

O histograma é uma representação gráfica da distribuição de frequência de uma determinada variável, normalmente um gráfico de barras verticais. O histograma é utilizado quando a variável em estudo é "contínua".

Suponha que um vetor de dados ou um *dataset* com n variáveis, e deseja-se ter uma noção da distribuição de uma determinada variável. No R o comando utilizado para construir um histograma é a função *hist()*.

Gráfico Histograma

O histograma é uma representação gráfica da distribuição de frequência de uma determinada variável, normalmente um gráfico de barras verticais. O histograma é utilizado quando a variável em estudo é "contínua".

Suponha que um vetor de dados ou um *dataset* com n variáveis, e deseja-se ter uma noção da distribuição de uma determinada variável. No R o comando utilizado para construir um histograma é a função *hist()*.

Códigos

```
x=rnorm(1000,0,1)
hist(x, main="Histograma da variável x", ylab="Frequência")
```

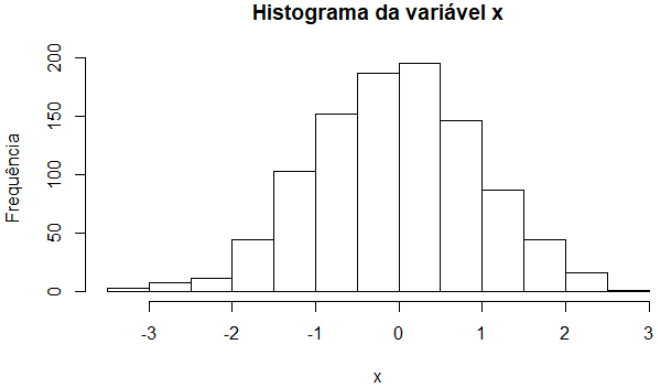


Figura 11: Gráfico histograma

Códigos

```
y=rnorm(5000,20,1)
hist(y, main="Histograma da variavel Idade", prob=T, xlab = "Idade",
     ylab="Densidade", xlim=c(15,25), col="Darkred", breaks = 25)
```

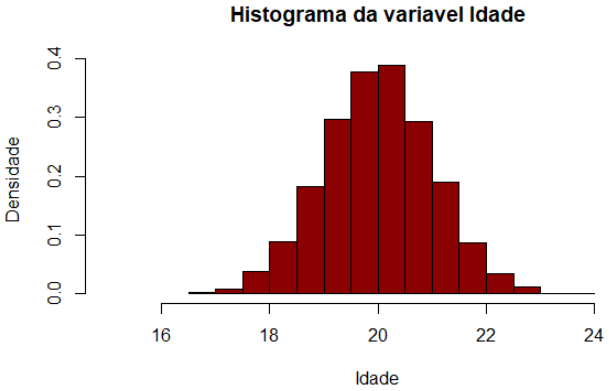
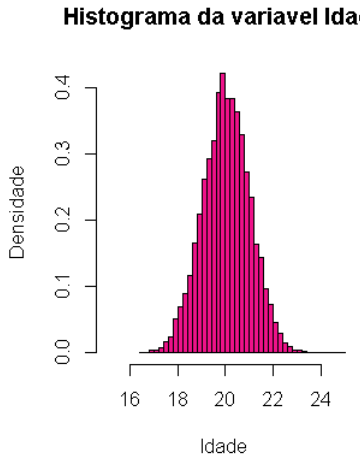


Figura 12: Gráfico histograma

Códigos

```
r=rnorm(5000,20,1)
hist(r, main="Histograma da variavel Idade", prob=T, xlab = "Idade",
ylab="Densidade", xlim=c(15,25), col="DeepPink2", breaks = 65)

w=rnorm(500,4,1)
hist(w, main="Histograma da variavel W", prob=T,
     xlab = "Valores de w,
ylab="Densidade", xlim=c(0,8), col="Green3", breaks = 10)
```


Histograma da variavel Idade
Densidade
0.0
0.1
0.2
0.3
0.4
16
18
20
22
24
Idade

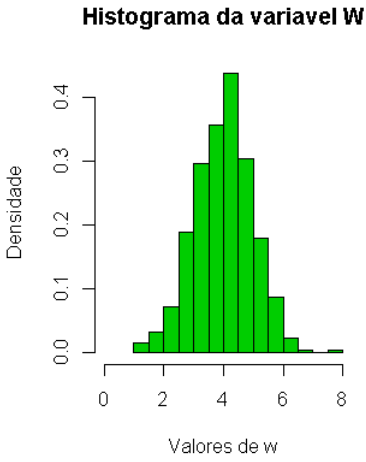

Histograma da variavel W
Densidade
0.0
0.1
0.2
0.3
0.4
0
2
4
6
8
Valores de w

Gráfico Boxplot

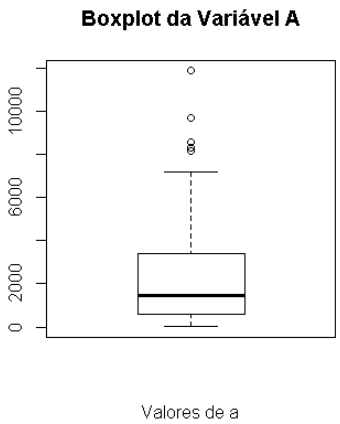
O boxplot é um gráfico que possibilita representar a distribuição de um conjunto de dados com base em alguns de seus parâmetros descritivos, conhecidos como quantis, que são a mediana(q2), o quartil inferior(q1) e o quartil superior(q3).

Para construir um *boxplot* no R basta utilizar a função *boxplot()*. Exemplo:

Códigos

```
a=rexp(100,1/2500)
boxplot(a, main="Boxplot da Variável A",xlab="Valores de a")

boxplot(a, main="Boxplot da Variável A", cex.main=2, cex.lab=1.5,
        xlab="Valores de a", pch=16, col="LightYellow4")
```



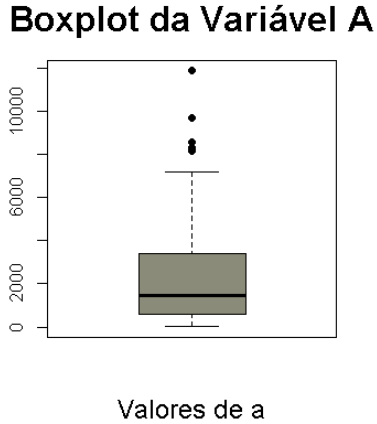


Figura 14: Boxplots

Em muitos casos não estamos interessados somente em observar o comportamento de uma variável em relação a apenas uma categoria, para esses casos não é nada mais complicado, utilizamos um "$\sim$" entre as duas variáveis na hora de passar o argumento do boxplot.

Em muitos casos não estamos interessados somente em observar o comportamento de uma variável em relação a apenas uma categoria, para esses casos não é nada mais complicado, utilizamos um "∼" entre as duas variáveis na hora de passar o argumento do boxplot.

Códigos

```
notas = sample(0:10,30,T)
turma = c(rep("A",10),rep("B",10),rep("C",10))
boxplot(notas ~ turma,main="Boxplot de notas por turma de Cálculo I",
        xlab="Turmas", ylab="Notas", cex.main=1.5, cex.lab=1.1,
        col=c("SpringGreen1","IndianRed2","Purple2"))
```

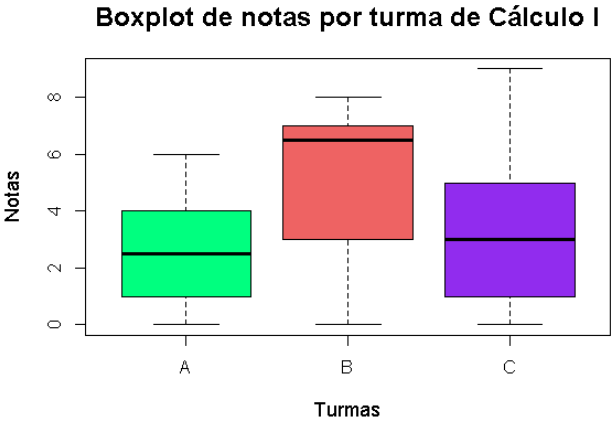


Figura 15: Boxplots

Gráfico de Dispersão

Como o nome já diz, são gráficos de pontos no eixo Y contra o eixo X. É extremamente simples fazer um gráfico de pontos y entre x no R. A função utilizada para a elaboração deste gráfico é *plot()*, e necessita apenas de dois argumentos: o primeiro é o nome da variável do eixo X e o segundo da variável do eixo Y.

Gráfico de Dispersão

Como o nome já diz, são gráficos de pontos no eixo Y contra o eixo X. É extremamente simples fazer um gráfico de pontos y entre x no R. A função utilizada para a elaboração deste gráfico é *plot()*, e necessita apenas de dois argumentos: o primeiro é o nome da variável do eixo X e o segundo da variável do eixo Y.

Códigos

```
s = sample(35:100, 20, T)
r = 1:20
plot(r,s)
```

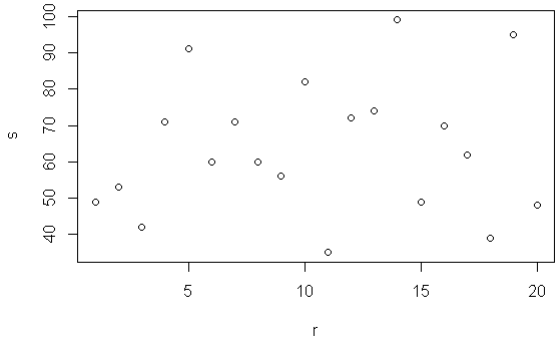


Figura 16: Gráfico de dispersão 1

Em muitos casos, precisamos detalhar melhor o que estamos fazendo, para isso produzimos gráficos cada vez mais elaborados e de fácil compreensão. Portanto a seguir podemos observar o que pode ser feito para elaborar melhores gráficos.

Em muitos casos, precisamos detalhar melhor o que estamos fazendo, para isso produzimos gráficos cada vez mais elaborados e de fácil compreensão. Portanto a seguir podemos observar o que pode ser feito para elaborar melhores gráficos.

Códigos

```
x = 1:100
y = 5 + 2 * x + rnorm(100, sd = 30)

plot(x, y, main = "Gráfico de pontos",
    sub = "Ordem dos pontos representados", type = "b",
    xlab = "Valores de r", ylab="Valores de s", pch=16, col="Blue")
```

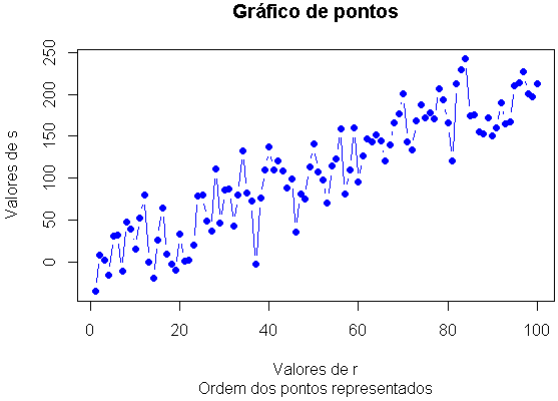


Figura 17: Gráfico de dispersão 2

Códigos

```
plot(x, y,
    main = "Gráfico de pontos",
    sub = "Ordem dos pontos
    representados",type = "l",
    xlab="Valores de r",
    ylab="Valores de s",
    col="red", lwd=2)
```

Códigos

```
plot(x, y,
     main = "Gráfico de pontos",
     sub = "Ordem dos pontos
     representados",type = "l",
     xlab="Valores de r",
     ylab="Valores de s",
     col="red", lwd=2)
```

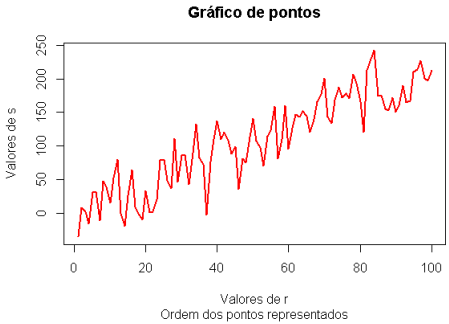


Figura 18: Gráfico de dispersão 3

Códigos

```
plot(x, y,
     main = "Gráfico de pontos",
     sub = "Ordem dos pontos
     representados",type = "s",
     xlab = "Valores de r",
     ylab="Valores de s",
     col = "green", lwd=2)
```

Códigos

```
plot(x, y,
     main = "Gráfico de pontos",
     sub = "Ordem dos pontos
     representados",type = "s",
     xlab = "Valores de r",
     ylab="Valores de s",
     col = "green", lwd=2)
```

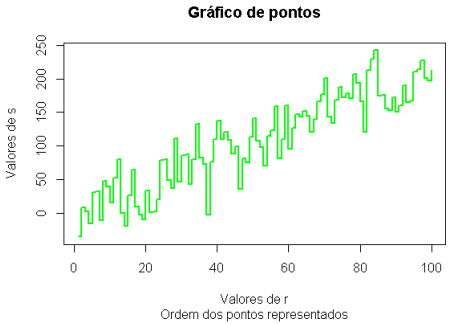


Figura 19: Gráfico de dispersão 4

Dicas!

Alguns pacotes

- tidyverse
- ggplot2
- Rmarkdonw

Dicas!

Alguns pacotes

- tidyverse
- ggplot2
- Rmarkdonw

Alguns fóruns

- Stackoverflow
- Stackoverflow em português


EM BREVE!!!
Processo
Seletivo
PET
Estatística

Contatos

Facebook: PET Estatística UFSCar

Instagram: petestat.ufscar

Site: https://petestatisticaufscar.wordpress.com