

Métodos Computacionais Aplicados (CEC0031)

Curso Introatório de R

Prof.: Dr. José Weligton Félix Gomes

Universidade Federal do Ceará - UFC
Campus Avançado de Sobral

11 de setembro de 2023

Introdução ao R

- R é uma linguagem e um ambiente de desenvolvimento integrado para cálculos estatísticos e gráficos.
- Foi criada originalmente por Ross Ihaka e por Robert Gentleman no departamento de Estatística da universidade de Auckland, Nova Zelândia, e foi desenvolvido por um esforço colaborativo de pessoas em vários locais do mundo.
- R é também altamente expansível com o uso dos pacotes, que são bibliotecas para funções específicas ou áreas de estudo específicas.
- Um conjunto de pacotes é incluído com a instalação de R, com muito outros disponíveis na rede de distribuição do R (em inglês CRAN).

Introdução ao R

- R é uma linguagem e um ambiente de desenvolvimento integrado para cálculos estatísticos e gráficos.
- Foi criada originalmente por Ross Ihaka e por Robert Gentleman no departamento de Estatística da universidade de Auckland, Nova Zelândia, e foi desenvolvido por um esforço colaborativo de pessoas em vários locais do mundo.
- R é também altamente expansível com o uso dos pacotes, que são bibliotecas para funções específicas ou áreas de estudo específicas.
- Um conjunto de pacotes é incluído com a instalação de R, com muito outros disponíveis na rede de distribuição do R (em inglês CRAN).

Introdução ao R

- R é uma linguagem e um ambiente de desenvolvimento integrado para cálculos estatísticos e gráficos.
- Foi criada originalmente por Ross Ihaka e por Robert Gentleman no departamento de Estatística da universidade de Auckland, Nova Zelândia, e foi desenvolvido por um esforço colaborativo de pessoas em vários locais do mundo.
- R é também altamente expansível com o uso dos pacotes, que são bibliotecas para funções específicas ou áreas de estudo específicas.
- Um conjunto de pacotes é incluído com a instalação de R, com muito outros disponíveis na rede de distribuição do R (em inglês CRAN).

Introdução ao R

- R é uma linguagem e um ambiente de desenvolvimento integrado para cálculos estatísticos e gráficos.
- Foi criada originalmente por Ross Ihaka e por Robert Gentleman no departamento de Estatística da universidade de Auckland, Nova Zelândia, e foi desenvolvido por um esforço colaborativo de pessoas em vários locais do mundo.
- R é também altamente expansível com o uso dos pacotes, que são bibliotecas para funções específicas ou áreas de estudo específicas.
- Um conjunto de pacotes é incluído com a instalação de R, com muito outros disponíveis na rede de distribuição do R (em inglês CRAN).

Introdução ao R

- O uso de softwares e pacotes estatísticos para a análise de dados é de grande importância, desde o desenvolvimento e aplicação de métodos até a análise e interpretação de resultados (CHAMBERS (2008) *apud* MELLO e PETERNELLI, 2013).
- Os autores salientam que o R não é apenas um programa estatístico, mas também uma poderosa ferramenta que permite operações matemáticas, manipulação de vetores e matrizes, confecção de gráficos, e manipulação de bancos de dados, entre outros.
- Diferentemente dos *softwares* pagos ou licenciados que contam com o suporte técnico oferecido pela empresa que mantém o *software*, o R conta com a colaboração de dezenas (talvez centenas) de milhares de usuários ao redor do mundo.
- Esse suporte se dá principalmente via lista de e-mail contendo dúvidas e respostas sobre diversos temas relacionados ao R.

Introdução ao R

- O uso de softwares e pacotes estatísticos para a análise de dados é de grande importância, desde o desenvolvimento e aplicação de métodos até a análise e interpretação de resultados (CHAMBERS (2008) *apud* MELLO e PETERNELLI, 2013).
- Os autores salientam que o R não é apenas um programa estatístico, mas também uma poderosa ferramenta que permite operações matemáticas, manipulação de vetores e matrizes, confecção de gráficos, e manipulação de bancos de dados, entre outros.
- Diferentemente dos *softwares* pagos ou licenciados que contam com o suporte técnico oferecido pela empresa que mantém o *software*, o R conta com a colaboração de dezenas (talvez centenas) de milhares de usuários ao redor do mundo.
- Esse suporte se dá principalmente via lista de e-mail contendo dúvidas e respostas sobre diversos temas relacionados ao R.

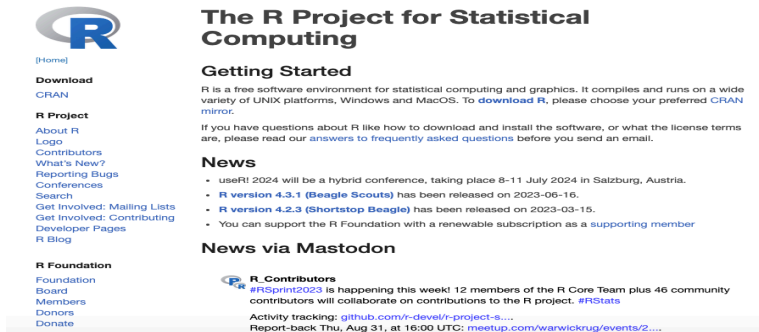
Introdução ao R

- O uso de softwares e pacotes estatísticos para a análise de dados é de grande importância, desde o desenvolvimento e aplicação de métodos até a análise e interpretação de resultados (CHAMBERS (2008) *apud* MELLO e PETERNELLI, 2013).
- Os autores salientam que o R não é apenas um programa estatístico, mas também uma poderosa ferramenta que permite operações matemáticas, manipulação de vetores e matrizes, confecção de gráficos, e manipulação de bancos de dados, entre outros.
- Diferentemente dos *softwares* pagos ou licenciados que contam com o suporte técnico oferecido pela empresa que mantém o *software*, o R conta com a colaboração de dezenas (talvez centenas) de milhares de usuários ao redor do mundo.
- Esse suporte se dá principalmente via lista de e-mail contendo dúvidas e respostas sobre diversos temas relacionados ao R.

Introdução ao R

- O uso de softwares e pacotes estatísticos para a análise de dados é de grande importância, desde o desenvolvimento e aplicação de métodos até a análise e interpretação de resultados (CHAMBERS (2008) *apud* MELLO e PETERNELLI, 2013).
- Os autores salientam que o R não é apenas um programa estatístico, mas também uma poderosa ferramenta que permite operações matemáticas, manipulação de vetores e matrizes, confecção de gráficos, e manipulação de bancos de dados, entre outros.
- Diferentemente dos *softwares* pagos ou licenciados que contam com o suporte técnico oferecido pela empresa que mantém o *software*, o R conta com a colaboração de dezenas (talvez centenas) de milhares de usuários ao redor do mundo.
- Esse suporte se dá principalmente via lista de e-mail contendo dúvidas e respostas sobre diversos temas relacionados ao R.

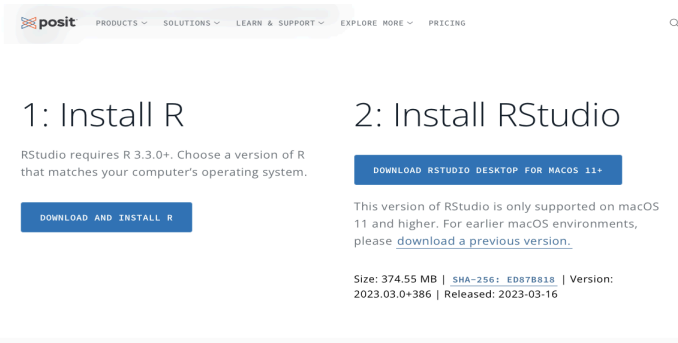
Instalando o R Project



The screenshot shows the official R Project website. At the top left is the R logo, a stylized 'R' inside a blue circle. Below it is a navigation menu with links: [Home], Download, CRAN, R Project, About R, Logo, Contributors, What's New?, Reporting Bugs, Conferences, Search, Get Involved: Mailing Lists, Get Involved: Contributing, Developer Pages, R Blog, R Foundation, Foundation, Board, Members, Donors, and Donate. The main heading is 'The R Project for Statistical Computing'. Below this is a 'Getting Started' section with a paragraph about R being a free software environment and a link to download R. It also includes a link to frequently asked questions. The 'News' section lists recent events like useR! 2024 and R version 4.3.1 (Beagle Scouts). At the bottom, there's a 'News via Mastodon' section with a link to the R Contributors Mastodon account.

- O programa R pode ser instalado a partir do website oficial <http://www.r-project.org/>: CRAN → Mirror (por exemplo, Brazil-University of Sao Paulo, Sao Paulo) → Download and Install R for Windows.

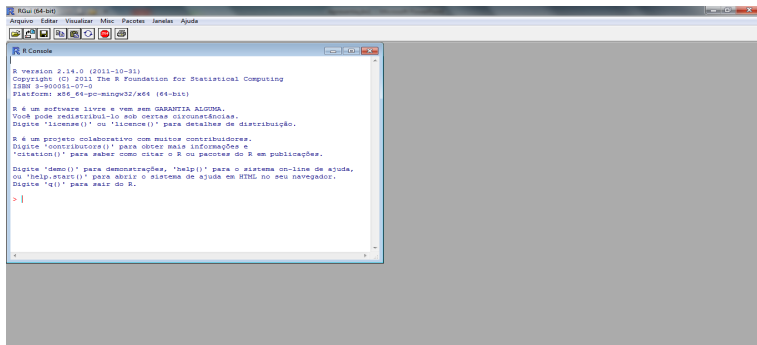
Instalando o R Project



The screenshot shows the Posit website with a navigation bar at the top containing links for PRODUCTS, SOLUTIONS, LEARN & SUPPORT, EXPLORE MORE, and PRICING. A search icon is on the right. The main content is divided into two columns. The left column is titled '1: Install R' and states 'RStudio requires R 3.3.0+. Choose a version of R that matches your computer's operating system.' Below this is a blue button labeled 'DOWNLOAD AND INSTALL R'. The right column is titled '2: Install RStudio' and features a blue button labeled 'DOWNLOAD RSTUDIO DESKTOP FOR MACOS 11+'. Below the button, it says 'This version of RStudio is only supported on macOS 11 and higher. For earlier macOS environments, please [download a previous version.](#)' At the bottom of the right column, it provides details: 'Size: 374.55 MB | [SHA-256: ED87B818](#) | Version: 2023.03.0+386 | Released: 2023-03-16'.

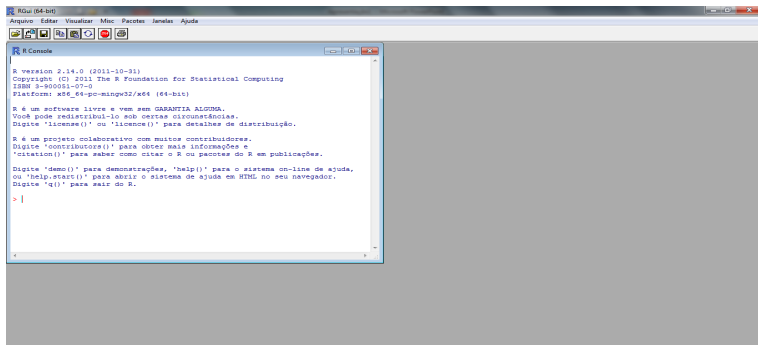
- O RStudio é uma ferramenta integrada que ajuda a aumentar a produtividade com o R. Possui console, editor de sintaxe, ferramentas de plotagem, histórico, depuração e gerenciamento de espaço de trabalho. Para baixá-lo, basta acessar o site <https://posit.co>.

Tela Principal do R



- Como podemos observar, a tela principal do R é bastante simples, e sua utilização é feita através de comandos.
- No R, os valores numéricos ou não (caracteres) podem ser dispostos em um vetor, matriz ou data frame.

Tela Principal do R



```
RGui (64-bit)
Arquivo  Editar  Visualizar  Misc  Pacotes  Janelas  Ajuda

R Console

R version 2.14.0 (2011-10-31)
Copyright (C) 2011 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-pc-mingw32/x64 (64-bit)

R é um software livre e vem sem GARANTIA ALGUMA.
Você pode redistribuí-lo sob certas circunstâncias.
Digite 'license()' ou 'licence()' para detalhes de distribuição.

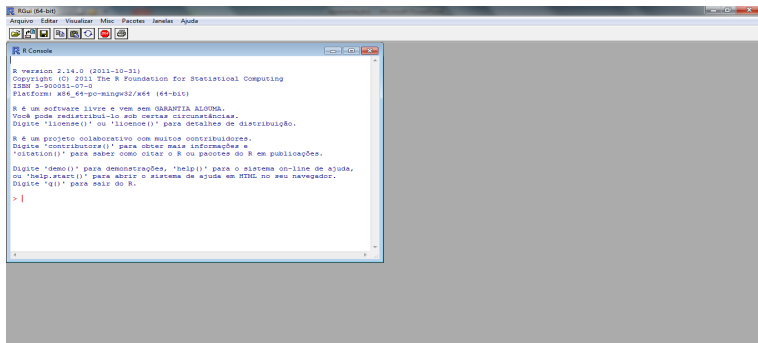
R é um projeto colaborativo com muitos contribuidores.
Digite 'contributors()' para obter mais informações e
'citation()' para saber como citar o R em publicações.

Digite 'demo()' para demonstrações, 'help()' para o sistema on-line de ajuda,
ou 'help.start()' para abrir o sistema de ajuda em HTML no seu navegador.
Digite 'q()' para sair do R.

> |
```

- Como podemos observar, a tela principal do R é bastante simples, e sua utilização é feita através de comandos.
- No R, os valores numéricos ou não (caracteres) podem ser dispostos em um vetor, matriz ou data frame.

Tela Principal do R



- No console, janela onde os comandos são digitados, existe o menu, que é um conjunto de ícones que se encontram na parte superior, e o **prompt** de comandos, que é um sinal indicador de que o programa está pronto para receber o comando.

Uma primeira sessão com o R

- Vamos começar "experimentando o R" para ter uma ideia de seus recursos e a forma de trabalhar. Para isto vamos rodar e estudar os comandos abaixo e seus resultados para nos familiarizar com o programa. Nas sessões seguintes iremos ver com mais detalhes o uso do programa R. Siga os seguintes passos:
- Inicie o R em seu computador.
- Você verá uma janela de comandos com o símbolo `>`. Este é o **prompt** do R indicando que o programa está pronto para receber comandos.
- A seguir digite (ou "recorte e cole") os comandos mostrados abaixo.

Uma primeira sessão com o R

- Vamos começar "experimentando o R" para ter uma ideia de seus recursos e a forma de trabalhar. Para isto vamos rodar e estudar os comandos abaixo e seus resultados para nos familiarizar com o programa. Nas sessões seguintes iremos ver com mais detalhes o uso do programa R. Siga os seguintes passos:
- Inicie o R em seu computador.
- Você verá uma janela de comandos com o símbolo `>`. Este é o **prompt** do R indicando que o programa está pronto para receber comandos.
- A seguir digite (ou "recorte e cole") os comandos mostrados abaixo.

Uma primeira sessão com o R

- Vamos começar "experimentando o R" para ter uma ideia de seus recursos e a forma de trabalhar. Para isto vamos rodar e estudar os comandos abaixo e seus resultados para nos familiarizar com o programa. Nas sessões seguintes iremos ver com mais detalhes o uso do programa R. Siga os seguintes passos:
- Inicie o R em seu computador.
- Você verá uma janela de comandos com o símbolo $>$. Este é o **prompt** do R indicando que o programa está pronto para receber comandos.
- A seguir digite (ou "recorte e cole") os comandos mostrados abaixo.

Uma primeira sessão com o R

- Vamos começar "experimentando o R" para ter uma ideia de seus recursos e a forma de trabalhar. Para isto vamos rodar e estudar os comandos abaixo e seus resultados para nos familiarizar com o programa. Nas sessões seguintes iremos ver com mais detalhes o uso do programa R. Siga os seguintes passos:
- Inicie o R em seu computador.
- Você verá uma janela de comandos com o símbolo $>$. Este é o **prompt** do R indicando que o programa está pronto para receber comandos.
- A seguir digite (ou "recorte e cole") os comandos mostrados abaixo.

Tela Principal

```
Console ~/R/ ↵
R version 3.1.2 (2014-10-31) -- "Pumpkin Helmet"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[workspace loaded from ~/R/.RData]
> help(lm)
> |
```

- Como pedir ajuda?
- As funções do R têm documentação *online*.
- `help(lm)` ou `?lm` - ajuda da função `lm()`.

Tela Principal

```
Console ~/R/ ↵
R version 3.1.2 (2014-10-31) -- "Pumpkin Helmet"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[workspace loaded from ~/R/.RData]
> help(lm)
> |
```

- Como pedir ajuda?
- As funções do R têm documentação *online*.
- `help(lm)` ou `?lm` - ajuda da função `lm()`.

Tela Principal

```
Console ~/R/ ↵
R version 3.1.2 (2014-10-31) -- "Pumpkin Helmet"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[workspace loaded from ~/R/.RData]
> help(lm)
> |
```

- Como pedir ajuda?
- As funções do R têm documentação *online*.
- **help(lm)** ou **?lm** - ajuda da função **lm()**.

Tela Principal

```
Console -/R/ ↻
R version 3.1.2 (2014-10-31) -- "Pumpkin Helmet"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[workspace loaded from ~/R/.RData]
> help(1m)
> |
```

- **help.search("linear model")** ou ?? **"linear model"** - busca em todo sistema de ajuda a função ou objeto que possui em sua descrição a expressão *linear model*.

Instalar e Carregar Pacotes

- 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*) 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.

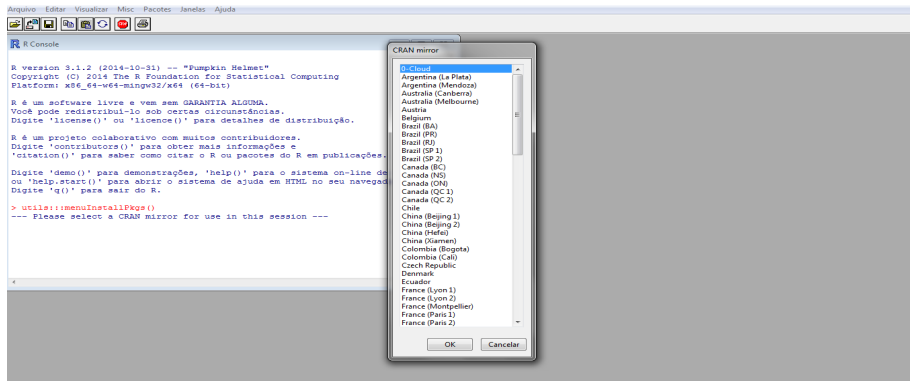
Instalar e Carregar Pacotes

- 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*) 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.

Instalar e Carregar Pacotes

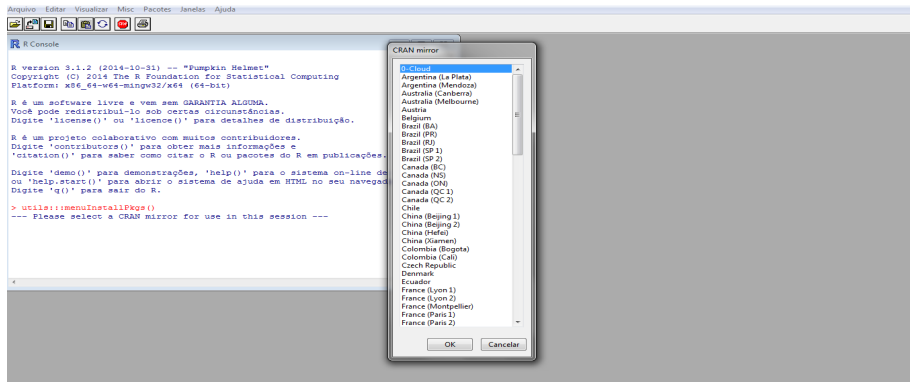
- O programa R é composto de 3 partes básicas:
 - ① o **R-base**, o "coração" do R que contém as funções principais disponíveis quando iniciamos o programa.
 - ② os **pacotes recomendados** (*recommended packages*) que são instalados junto com o **R-base** mas não são carregados quando iniciamos o programa. 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*) 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.

Instalar e Carregar Pacotes



- Os pacotes podem ser instalados de servidor local (i.e., local *mirror*) a partir do próprio programa em "instalar pacotes" no menu Pacotes ou utilizando a função ***install.packages()*** no Console.

Instalar e Carregar Pacotes



- **require(epicalc)** - carrega pacote previamente instalado no programa.
- DICA:** carregar sempre o pacote no início de cada sessão de trabalho.

Rcmdr

- O programa Rcmdr possibilita aos analistas acessar uma seleção de comandos, frequentemente utilizados no R, usando uma interface simples que seria muito mais familiar para a maioria dos usuários de computador.
- Ele também serve de papel importante para ajudar os usuários a implementar comandos no R e desenvolver seus conhecimentos e expertise no uso da linha de comandos.
- Existem atualmente 29 plugins que fornecem suporte para análises específicas, gráficos, livros e ensino.
- Após a instalação do pacote "**Rcmdr**".
 - > **library(Rcmdr)**
 - > **require(Rcmdr)**

Rcmdr

- O programa Rcmdr possibilita aos analistas acessar uma seleção de comandos, frequentemente utilizados no R, usando uma interface simples que seria muito mais familiar para a maioria dos usuários de computador.
- Ele também serve de papel importante para ajudar os usuários a implementar comandos no R e desenvolver seus conhecimentos e expertise no uso da linha de comandos.
- Existem atualmente 29 plugins que fornecem suporte para análises específicas, gráficos, livros e ensino.
- Após a instalação do pacote "Rcmdr".
 - > `library(Rcmdr)`
 - > `require(Rcmdr)`

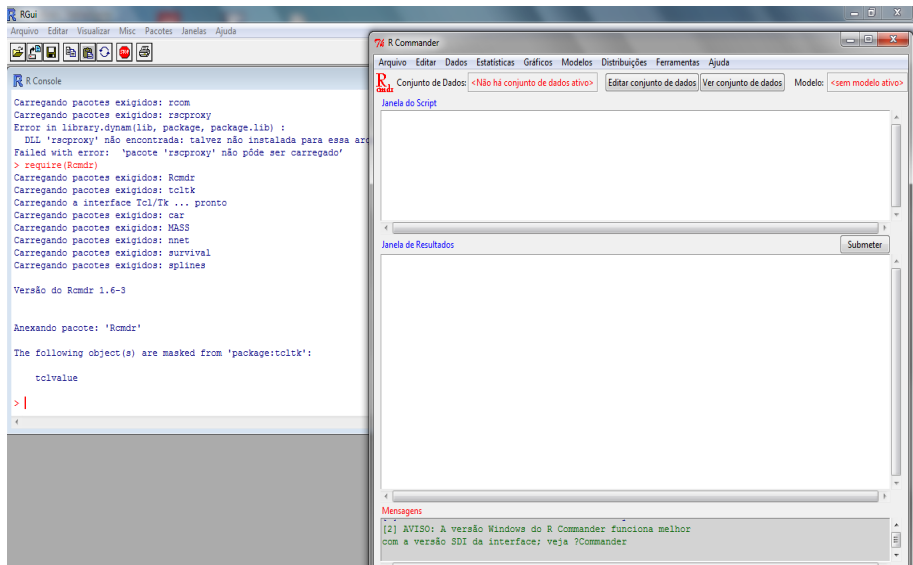
Rcmdr

- O programa Rcmdr possibilita aos analistas acessar uma seleção de comandos, frequentemente utilizados no R, usando uma interface simples que seria muito mais familiar para a maioria dos usuários de computador.
- Ele também serve de papel importante para ajudar os usuários a implementar comandos no R e desenvolver seus conhecimentos e expertise no uso da linha de comandos.
- Existem atualmente 29 plugins que fornecem suporte para análises específicas, gráficos, livros e ensino.
- Após a instalação do pacote "Rcmdr".
 > `library(Rcmdr)`
 > `require(Rcmdr)`

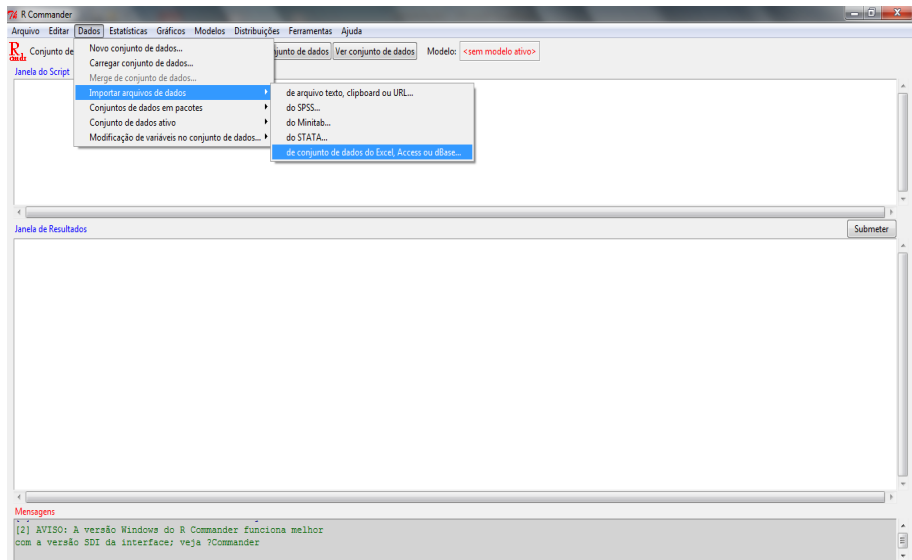
Rcmdr

- O programa Rcmdr possibilita aos analistas acessar uma seleção de comandos, frequentemente utilizados no R, usando uma interface simples que seria muito mais familiar para a maioria dos usuários de computador.
- Ele também serve de papel importante para ajudar os usuários a implementar comandos no R e desenvolver seus conhecimentos e expertise no uso da linha de comandos.
- Existem atualmente 29 plugins que fornecem suporte para análises específicas, gráficos, livros e ensino.
- Após a instalação do pacote "**Rcmdr**".
 - > **library(Rcmdr)**
 - > **require(Rcmdr)**

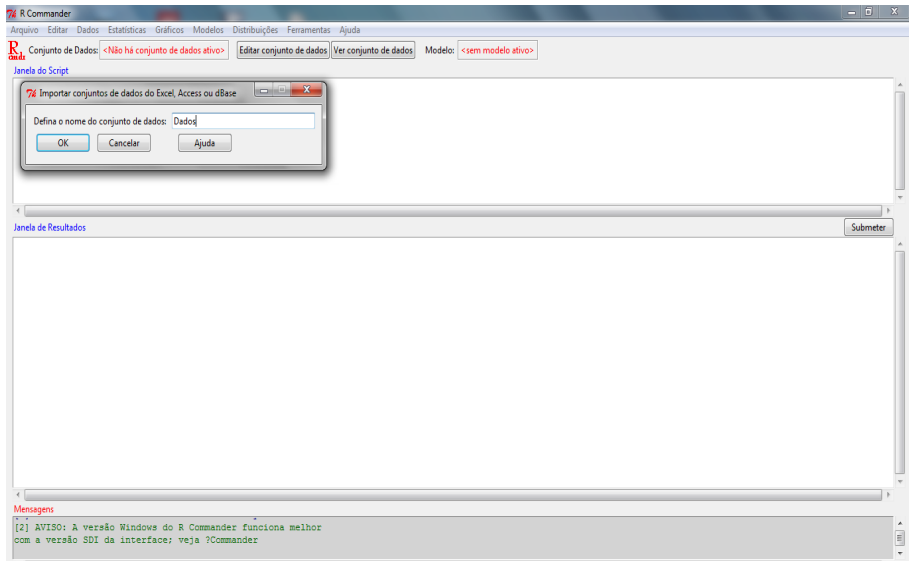
Rcmdr



Rcmdr



Rcmdr



Rcmdr

The screenshot displays the R Commander application window. The menu bar includes Arquivo, Editar, Dados, Estatísticas, Gráficos, Modelos, Distribuições, Ferramentas, and Ajuda. The 'Dados' menu is open, showing options: Conjunto de Dados: Dados, Editar conjunto de dados, Ver conjunto de dados, and Modelo: <sem modelo ativo>.

The 'Janela do Script' (Script Window) contains the following R code:

```
Dados <- sqlQuery(channel = 1, select * from [401K$])
library(relimp, pos=4)
showData(Dados, placement='-20+200', font=getRcmdr('logFont'), maxwidth=80,
  maxheight=30)
```

The 'Janela de Resultados' (Results Window) shows the execution of the code:

```
> Dados <- sqlQuery(channel = 1, select * from [401K$])
> library(relimp, pos=4)
> showData(Dados, placement='-20+200', font=getRcmdr('logFont'), maxwidth=80,
+ maxheight=30)
```

The 'Dados' window displays a table with 30 rows and 8 columns:

	taxac	taxcont	totpart	totqual	idade	totemp	único	ltotemp
1	26.1	0.21	1653	6322	8	8709	0	9.072112
2	100.0	1.42	262	262	6	315	1	5.752573
3	97.6	0.91	166	170	10	275	1	5.616771
4	100.0	0.42	257	257	7	500	0	6.214608
5	82.5	0.53	591	716	28	933	1	6.838405
6	100.0	1.82	92	92	7	143	1	4.962845
7	100.0	0.53	399	399	31	628	1	6.442540
8	92.5	0.34	210	227	13	823	0	6.712956
9	100.0	0.22	234	234	21	279	1	5.631212
10	96.8	0.60	120	124	10	141	1	4.948760
11	83.0	0.50	156	188	5	252	0	5.529429
12	100.0	1.11	187	187	20	237	1	5.468060
13	65.5	0.43	350	534	5	681	1	6.523562
14	84.9	0.51	1519	1790	10	2557	0	7.846590
15	100.0	0.19	150	150	7	187	1	5.231109
16	100.0	0.41	160	160	7	188	1	5.236442
17	100.0	0.84	3448	3448	13	4120	0	8.323608
18	77.6	0.46	225	290	15	341	0	5.831882
19	100.0	0.45	519	519	15	598	0	6.393591
20	100.0	1.13	176	176	8	273	1	5.609472
21	66.9	0.48	79	118	5	224	1	5.411646
22	100.0	0.39	270	270	7	415	0	6.028278
23	70.3	0.21	201	286	7	361	0	5.888878
24	100.0	0.95	381	381	23	422	0	6.045005
25	51.3	0.34	284	554	9	797	1	6.680855
26	61.3	0.36	171	279	15	2102	1	7.650645
27	88.3	0.96	3858	4371	28	4371	0	8.382747
28	100.0	0.87	143	143	8	143	1	4.962845
29	82.9	4.20	107	129	7	131	1	4.875197
30	77.8	0.08	770	990	6	1042	0	6.948897

The 'Mensagens' (Messages) window at the bottom shows the following output:

```
com a versão SDI da interface; veja ?Commander
[3] NOTA: Os dados Dados tem 1534 linhas e 8 colunas.
```

Introdução ao RStudio

- O **RStudio** permite-nos usar o **R** em um ambiente mais amigável. Esta disponível para download no site:
<https://posit.co>
<https://posit.cloud>
- Para alguns tutoriais e/ou recursos relacionados ao **R** veja os links:
<https://www.princeton.edu/~otorres/>
https://www.lampada.uerj.br/arquivosdb/_book2/introducaoR.html

Introdução ao RStudio

- O **RStudio** permite-nos usar o **R** em um ambiente mais amigável. Esta disponível para download no site:
<https://posit.co>
<https://posit.cloud>
- Para alguns tutoriais e/ou recursos relacionados ao **R** veja os links:
<https://www.princeton.edu/~otorres/>
https://www.lampada.uerj.br/arquivosdb/_book2/introducaoR.html

Introdução ao RStudio

Tela Inicial do RStudio

RStudio screen

The screenshot displays the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Project, Build, Tools, and Help. Below the menu is a toolbar with icons for file operations and running code. The main interface is divided into four panes:

- Console:** Shows the R version (3.0.0), copyright information, and a series of commands and their outputs. The commands include `getwd()`, `setwd("C:/Users/sof/Desktop/MyData/Files/01")`, `matrix()`, and `matrix()` with `byrow=TRUE`.
- Workspace:** Displays the active objects in the current session. It shows two objects, 'A' and 'B', both of type '4x2 double matrix'.
- History:** Shows a list of commands that have been executed in the session.
- Files:** Displays the file explorer for the current project, showing the directory structure and file names.

The **console** is where you can type commands and see output

The **workspace** tab shows all the active objects (see next slide). The **history** tab shows a list of commands used so far.

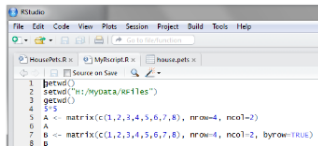
The **files** tab shows all the files and folders in your default workspace as if you were on a PC/Mac window. The **plots** tab will show all your graphs. The **packages** tab will list a series of packages or add-ons needed to run certain processes. For additional info see the **help** tab

Introdução ao RStudio

Workspace

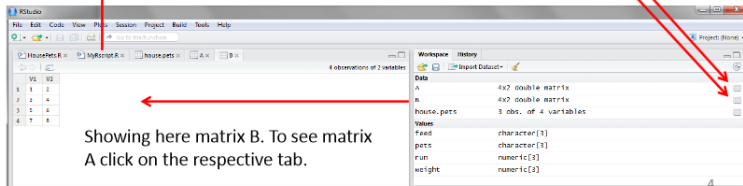
Workspace tab (1)

The workspace tab stores any object, value, function or anything you create during your R session. In the example below, if you click on the dotted squares you can see the data on a screen to the left.



```

1 | petw()
2 | setwd("~/MyData/Rfiles")
3 | getwd()
4 | 5-5
5 | A <- matrix(c(1,2,3,4,5,6,7,8), nrow=4, ncol=2)
6 | A
7 | B <- matrix(c(1,2,3,4,5,6,7,8), nrow=4, ncol=2, byrow=TRUE)
8 | B
  
```



Showing here matrix B. To see matrix A click on the respective tab.

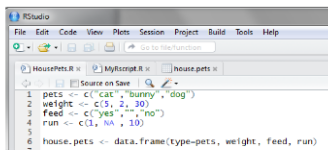
Workspace	History
Data	
A	4x2 double matrix
B	4x2 double matrix
house.pets	3 obs. of 4 variables
Values	
feed	character[1]
pets	character[3]
run	numeric[3]
weight	numeric[2]

Introdução ao RStudio

Workspace

Workspace tab (2)

Here is another example on how the workspace looks like when more objects are added. Notice that the data frame `house.pets` is formed from different individual values or vectors.

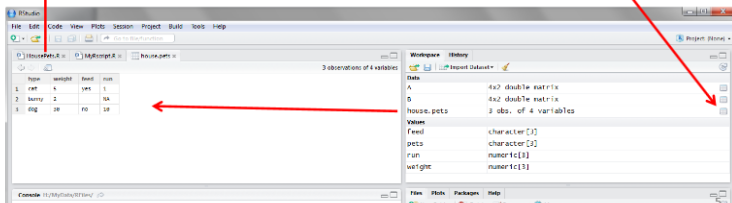


```

1 pets <- c("cat", "bunny", "dog")
2 weight <- c(5, 2, 30)
3 feed <- c("yes", "no")
4 run <- c(1, NA, 10)
5
6 house.pets <- data.frame(type=pets, weight, feed, run)
7

```

Click on the dotted square to look at the dataset in a spreadsheet form.



The screenshot shows the RStudio interface with the 'house.pets' data frame loaded in the Environment pane. The data is displayed as a table with 3 observations and 4 variables:

type	weight	feed	run
cat	5	yes	1
bunny	2	NA	
dog	30	no	10

Red arrows indicate the workflow: one arrow points from the script editor to the Environment pane, and another points from the 'house.pets' entry in the Environment pane to the data table.

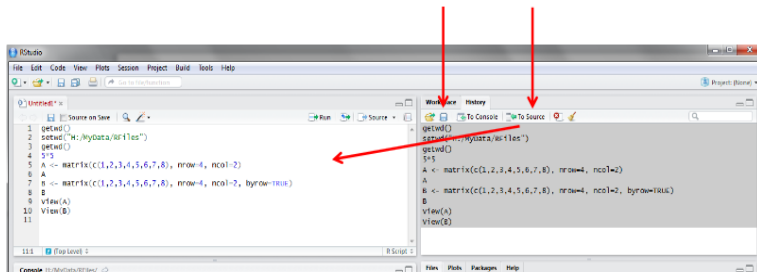
Introdução ao RStudio

History

History tab

The history tab keeps a record of all previous commands. It helps when testing and running processes. Here you can either **save** the whole list or you can **select** the commands you want and send them to an R script to keep track of your work.

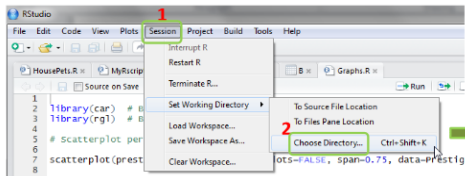
In this example, we select all and click on the “To Source” icon, a window on the left will open with the list of commands. Make sure to save the ‘untitled1’ file as an *.R script.



Introdução ao RStudio

Diretório

Changing the working directory



If you have different projects you can change the working directory for that session, see above. Or you can type:

```
# Shows the working directory (wd)
```

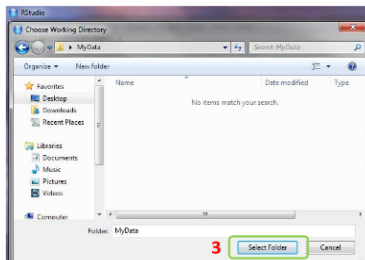
```
getwd()
```

```
# Changes the wd
```

```
setwd("C:/myfolder/data")
```

More info see the following document:

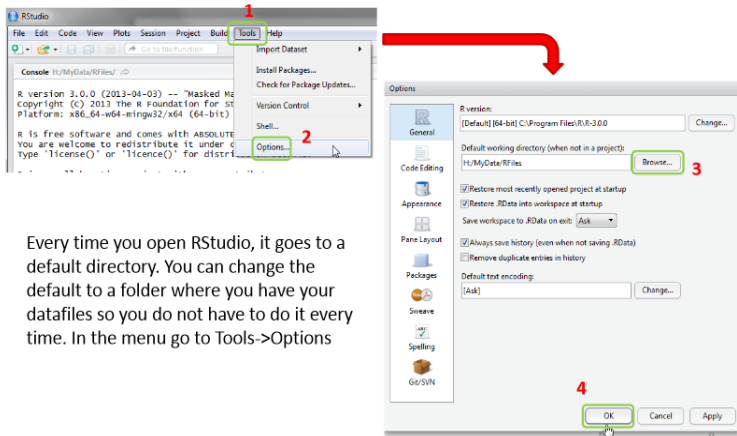
<http://dss.princeton.edu/training/RStata.pdf>



Introdução ao RStudio

Diretório

Setting a default working directory



Every time you open RStudio, it goes to a default directory. You can change the default to a folder where you have your datafiles so you do not have to do it every time. In the menu go to Tools->Options

Introdução ao RStudio

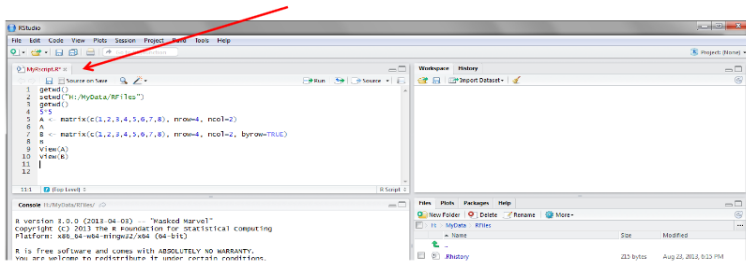
R Script

R script (1)

The usual Rstudio screen has four windows:

1. Console.
2. Workspace and history.
3. Files, plots, packages and help.
4. The R script(s) and data view.

The R script is where you keep a record of your work. For Stata users this would be like the do-file, for SPSS users is like the syntax and for SAS users the SAS program.

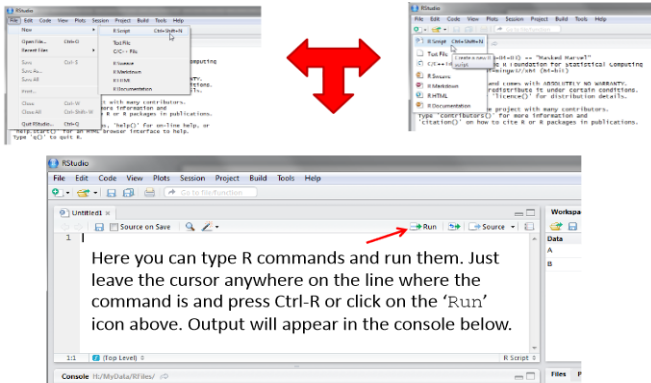


Introdução ao RStudio

R Script

R script (2)

To create a new R script you can either go to File -> New -> R Script, or click on the icon with the “+” sign and select “R Script”, or simply press Ctrl+Shift+N. Make sure to save the script.

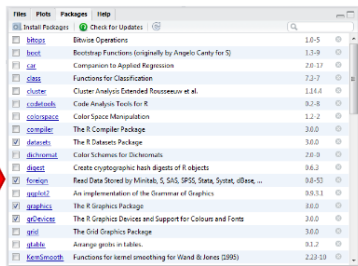
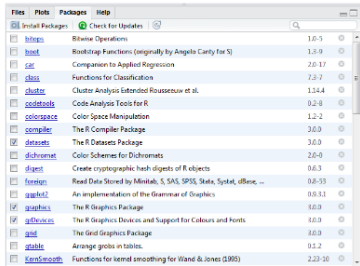


Introdução ao RStudio

Pacotes

Packages tab

The package tab shows the list of add-ons included in the installation of RStudio. If checked, the package is loaded into R, if not, any command related to that package won't work, you will need select it. You can also install other add-ons by clicking on the 'Install Packages' icon. Another way to activate a package is by typing, for example, `library(foreign)`. This will automatically check the `--foreign` package (it helps bring data from proprietary formats like Stata, SAS or SPSS).



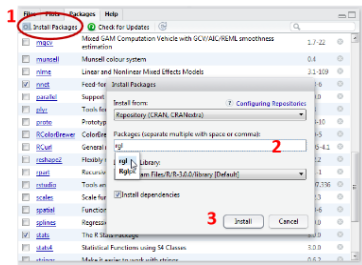
Introdução ao RStudio

Pacotes

Installing a package

<input type="checkbox"/> RCurl	General network (HTTP/FTP/...) client interface for R	1.95-4.1
<input type="checkbox"/> reshape2	Flexibly reshape data: a reboot of the reshape package.	1.2.2
<input type="checkbox"/> rpart	Recursive Partitioning	4.1-1

Before



We are going to install the package – `rgl` (useful to plot 3D images). It does not come with the original R install.

Click on “Install Packages”, write the name in the pop-up window and click on “Install”.

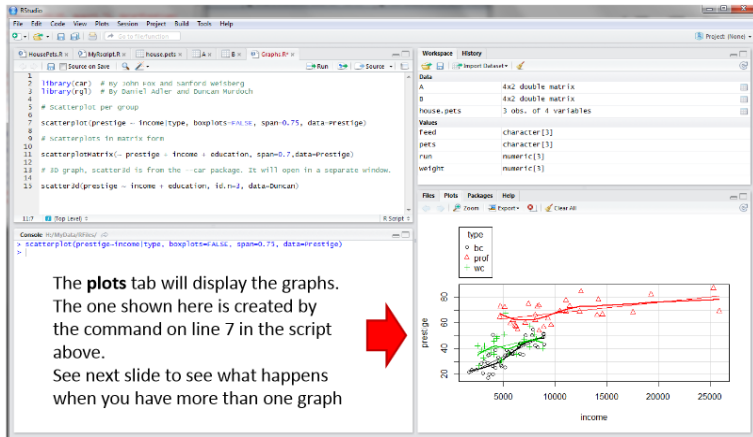
After

<input type="checkbox"/> RCurl	General network (HTTP/FTP/...) client interface for R	1.95-4.1
<input type="checkbox"/> reshape2	Flexibly reshape data: a reboot of the reshape package.	1.2.2
<input type="checkbox"/> rgl	3D visualization device system (OpenGL)	0.93.952
<input type="checkbox"/> rpart	Recursive Partitioning	4.1-1 12

Introdução ao RStudio

Gráficos

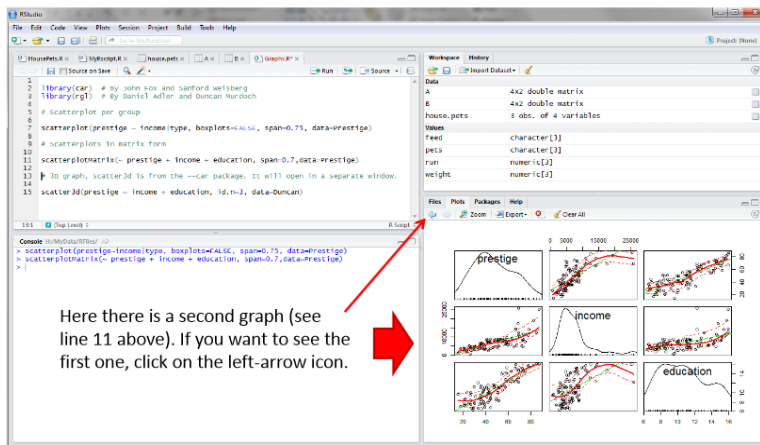
Plots tab (1)



Introdução ao RStudio

Gráficos

Plots tab (2)

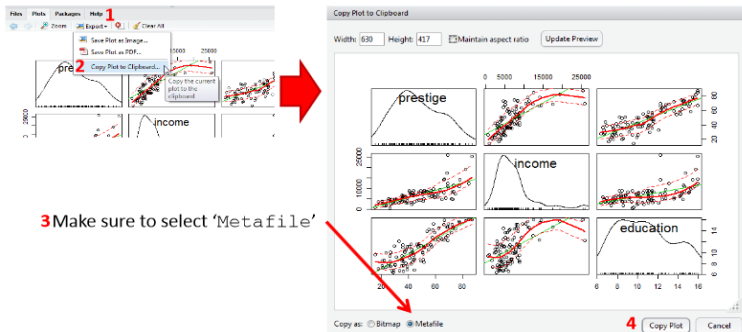


Introdução ao RStudio

Gráficos

Plots tab (3) – Graphs export

To extract the graph, click on “Export” where you can save the file as an image (PNG, JPG, etc.) or as PDF, these options are useful when you only want to share the graph or use it in a LaTeX document. Probably, the easiest way to export a graph is by copying it to the clipboard and then paste it directly into your Word document.



5 Paste it into your Word document

Script do R

- Para poupar tempo no uso do R é importante trabalharmos com um **Script**. Ou seja, um arquivo onde você irá digitar todos os comandos e análises (comentários) e deixará salvo no seu computador. Neste caso, todos os comandos que você está utilizando não serão mais digitado na linha de comando do R e sim em um editor de texto.

Uma primeira sessão com o R

- Gerando dois vetores de coordenadas x e y de números pseudo-aleatórios e inspecionando os valores gerados.

```
> x <- rnorm(5)
```

```
> x
```

```
> y <- rnorm(x)
```

```
> y
```

- Colocando os pontos em um gráfico.

```
> plot(x, y)
```

- Verificando os objetos existentes na área de trabalho.

```
> ls()
```

- Removendo objetos que não são mais necessários.

```
> rm(x, y)
```

Uma primeira sessão com o R

- Gerando dois vetores de coordenadas x e y de números pseudo-aleatórios e inspecionando os valores gerados.

```
> x <- rnorm(5)
```

```
> x
```

```
> y <- rnorm(x)
```

```
> y
```

- Colocando os pontos em um gráfico.

```
> plot(x, y)
```

- Verificando os objetos existentes na área de trabalho.

```
> ls()
```

- Removendo objetos que não são mais necessários.

```
> rm(x, y)
```

Uma primeira sessão com o R

- Gerando dois vetores de coordenadas x e y de números pseudo-aleatórios e inspecionando os valores gerados.

```
> x <- rnorm(5)
```

```
> x
```

```
> y <- rnorm(x)
```

```
> y
```

- Colocando os pontos em um gráfico.

```
> plot(x, y)
```

- Verificando os objetos existentes na área de trabalho.

```
> ls()
```

- Removendo objetos que não são mais necessários.

```
> rm(x, y)
```

Uma primeira sessão com o R

- Gerando dois vetores de coordenadas x e y de números pseudo-aleatórios e inspecionando os valores gerados.

```
> x <- rnorm(5)
```

```
> x
```

```
> y <- rnorm(x)
```

```
> y
```

- Colocando os pontos em um gráfico.

```
> plot(x, y)
```

- Verificando os objetos existentes na área de trabalho.

```
> ls()
```

- Removendo objetos que não são mais necessários.

```
> rm(x, y)
```

Uma primeira sessão com o R

- Criando um vetor com uma sequência de números de 1 a 20.
`> x <- 1 : 20`
- Um vetor de pesos com os desvios padrões de cada observação.
`> w <- sqrt(x)/2`
- Montando um "data-frame" de 2 colunas, x e y, e inspecionando o objeto.
`> dummy <- data.frame(x = x, y = x + rnorm(x) * w)`
`> dummy`

Uma primeira sessão com o R

- Criando um vetor com uma sequência de números de 1 a 20.
`> x <- 1 : 20`
- Um vetor de pesos com os desvios padrões de cada observação.
`> w <- sqrt(x)/2`
- Montando um "data-frame" de 2 colunas, x e y, e inspecionando o objeto.
`> dummy <- data.frame(x = x, y = x + rnorm(x) * w)`
`> dummy`

Uma primeira sessão com o R

- Criando um vetor com uma sequência de números de 1 a 20.
`> x <- 1 : 20`
- Um vetor de pesos com os desvios padrões de cada observação.
`> w <- sqrt(x)/2`
- Montando um "data-frame" de 2 colunas, x e y, e inspecionando o objeto.
`> dummy <- data.frame(x = x, y = x + rnorm(x) * w)`
`> dummy`

Manipulando Objetos

- Uma das tarefas mais simples no R é executar cálculos aritméticos e receber os resultados. Por exemplo:

```
> 2 + 2
```

```
[1] 4
```

```
> exp(-2)
```

```
[1] 0.1353353
```

Naturalmente o programa executa outros tipos de operações.

Manipulando Objetos

- Criando objeto: Um objeto pode ser criado com a operação de “atribuição”, o qual se denota como uma flecha, com o sinal de menos e o símbolo $>$ ou $<$, dependendo da direção em que se atribui o objeto. Ou com um único sinal de igual. É importante dizer que o nome de um objeto deve começar com uma letra qualquer, maiúscula ou minúscula, que pode ser seguida de outra letra, número, ou caracteres especiais como o ponto.

- Exemplo:

```
x<-10 #o objeto x receberá o valor 10
```

```
15->y #o objeto y receberá o valor 15
```

```
X<-6 #o objeto X receberá o valor 6
```

```
Y=15 #o objeto Y receberá o valor 15
```

Manipulando Objetos

- Criando objeto: Um objeto pode ser criado com a operação de “atribuição”, o qual se denota como uma flecha, com o sinal de menos e o símbolo $>$ ou $<$, dependendo da direção em que se atribui o objeto. Ou com um único sinal de igual. É importante dizer que o nome de um objeto deve começar com uma letra qualquer, maiúscula ou minúscula, que pode ser seguida de outra letra, número, ou caracteres especiais como o ponto.
- Exemplo:

```
x<-10 #o objeto x receberá o valor 10  
15->y #o objeto y receberá o valor 15  
X<-6 #o objeto X receberá o valor 6  
Y=15 #o objeto Y receberá o valor 15
```

Comandos

```

•
> x<-c(10.4,5.6,3.1,6.4,21.7)
> x
[1] 10.4 5.6 3.1 6.4 21.7

```

```

•
> 1/x
[1] 0.09615385 0.17857143 0.32258065 0.15625000 0.04608295

```

- Criando um vetor y com 11 entradas consistindo de duas cópias de x com zero entre eles.

```

y<-c(x,0,x)
> y
[1] 10.4 5.6 3.1 6.4 21.7 0.0 10.4 5.6 3.1 6.4 21.7

```

Comandos

```
•
> x<-c(10.4,5.6,3.1,6.4,21.7)
> x
[1] 10.4 5.6 3.1 6.4 21.7
```

```
•
> 1/x
[1] 0.09615385 0.17857143 0.32258065 0.15625000 0.04608295
```

- Criando um vetor y com 11 entradas consistindo de duas cópias de x com zero entre eles.

```
y<-c(x,0,x)
> y
[1] 10.4 5.6 3.1 6.4 21.7 0.0 10.4 5.6 3.1 6.4 21.7
```

Comandos

```
•  
> x<-c(10.4,5.6,3.1,6.4,21.7)  
> x  
[1] 10.4 5.6 3.1 6.4 21.7
```

```
•  
> 1/x  
[1] 0.09615385 0.17857143 0.32258065 0.15625000 0.04608295
```

- Criando um vetor y com 11 entradas consistindo de duas cópias de x com zero entre eles.

```
y<-c(x,0,x)  
> y  
[1] 10.4 5.6 3.1 6.4 21.7 0.0 10.4 5.6 3.1 6.4 21.7
```


Comandos

- Os vetores pode ser usados em expressões aritméticas. Exemplo:

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



```
> y<-c(2,3,4,5,6)
```

```
> y
```

```
[1] 2 3 4 5 6
```



```
> v<-2*x + y
```

```
> v
```

```
[1] 22.8 14.2 10.2 17.8 49.4
```

Comandos

- Os vetores pode ser usados em expressões aritméticas. Exemplo:

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



```
> y<-c(2,3,4,5,6)
```

```
> y
```

```
[1] 2 3 4 5 6
```



```
> v<-2*x + y
```

```
> v
```

```
[1] 22.8 14.2 10.2 17.8 49.4
```

Alguns Comandos

•

```
> t<-5*sqrt(x) + sqrt(y)
> t
[1] 17.53873 13.56421 10.80341 14.88518 25.74112
```

•

```
> k<-sqrt(256)
> k
[1] 16
```

•

```
> z<-exp(k)
> z
[1] 8886111
```

Alguns Comandos

```
> t<-5*sqrt(x) + sqrt(y)
> t
[1] 17.53873 13.56421 10.80341 14.88518 25.74112
```

```
> k<-sqrt(256)
> k
[1] 16
```

```
> z<-exp(k)
> z
[1] 8886111
```

Alguns Comandos

```
> t<-5*sqrt(x) + sqrt(y)
> t
[1] 17.53873 13.56421 10.80341 14.88518 25.74112
```

```
> k<-sqrt(256)
> k
[1] 16
```

```
> z<-exp(k)
> z
[1] 8886111
```

Alguns Comandos



```
> v1 <- c(4, 6, 8, 24)
> 2*v1
[1] 8 12 16 48
```



```
> v2 <- c(10, 2, 4, 8)
> v2
[1] 10 2 4 8
```



```
> 2*v1 - v2
[1] -2 10 12 40
```

Alguns Comandos



```
> v1 <- c(4, 6, 8, 24)
> 2*v1
[1] 8 12 16 48
```



```
> v2 <- c(10, 2, 4, 8)
> v2
[1] 10 2 4 8
```



```
> 2*v1 - v2
[1] -2 10 12 40
```

Alguns Comandos



```
> v1 <- c(4, 6, 8, 24)
> 2*v1
[1] 8 12 16 48
```



```
> v2 <- c(10, 2, 4, 8)
> v2
[1] 10 2 4 8
```



```
> 2*v1 - v2
[1] -2 10 12 40
```


Principais Distribuições

Tabela: Principais Distribuições

Distribuição	Nome no R	Distribuição	Nome no R	Distribuição	Nome no R
Normal	norm	t	t	Qui-Quadrado	chisq
Exponencial	exp	F	f	Uniforme	unif
Log-Normal	lnorm	Beta	beta	Gamma	gamma
Logística	logis	Weibull	weibull	Cauchy	cauchy
Geométrica	geom	Binomial	binom	Hipergeométrica	hyper
Poisson	pois	Binomial Negativa	nbinom		

Principais Distribuições

- Por exemplo o número aleatório (**r**) de uma distribuição normal (**norm**) pode ser extraído utilizando a função *rnorm*. Vejamos:
 - *rnorm*(10) Extrai 10 números aleatórios de uma distribuição padrão.
 - *rnorm*(10, 5, 2) Extrai 10 números aleatórios de uma distribuição $N(\mu = 5, \sigma = 2)$.
 - *dnorm*(*x*) Retorna o valor da função densidade da normal padrão em *x*.
 - *pnorm*(0) Retorna o valor de uma CDF normal padrão em $x = 0$.
 - *qnorm*(0.5) retorna o quantil 50% de uma distribuição normal padrão.

Principais Distribuições

- Por exemplo o número aleatório (**r**) de uma distribuição normal (**norm**) pode ser extraído utilizando a função *rnorm*. Vejamos:
 - *rnorm*(10) Extrai 10 números aleatórios de uma distribuição padrão.
 - *rnorm*(10, 5, 2) Extrai 10 números aleatórios de uma distribuição $N(\mu = 5, \sigma = 2)$.
 - *dnorm*(x) Retorna o valor da função densidade da normal padrão em x.
 - *pnorm*(0) Retorna o valor de uma CDF normal padrão em $x = 0$.
 - *qnorm*(0.5) retorna o quantil 50% de uma distribuição normal padrão.

Principais Distribuições

- Por exemplo o número aleatório (**r**) de uma distribuição normal (**norm**) pode ser extraído utilizando a função *rnorm*. Vejamos:
 - *rnorm*(10) Extrai 10 números aleatórios de uma distribuição padrão.
 - *rnorm*(10, 5, 2) Extrai 10 números aleatórios de uma distribuição $N(\mu = 5, \sigma = 2)$.
 - *dnorm*(x) Retorna o valor da função densidade da normal padrão em x .
 - *pnorm*(0) Retorna o valor de uma CDF normal padrão em $x = 0$.
 - *qnorm*(0.5) retorna o quantil 50% de uma distribuição normal padrão.

Principais Distribuições

- Por exemplo o número aleatório (**r**) de uma distribuição normal (**norm**) pode ser extraído utilizando a função *rnorm*. Vejamos:
 - *rnorm*(10) Extrai 10 números aleatórios de uma distribuição padrão.
 - *rnorm*(10, 5, 2) Extrai 10 números aleatórios de uma distribuição $N(\mu = 5, \sigma = 2)$.
 - *dnorm*(*x*) Retorna o valor da função densidade da normal padrão em *x*.
 - *pnorm*(0) Retorna o valor de uma CDF normal padrão em $x = 0$.
 - *qnorm*(0.5) retorna o quantil 50% de uma distribuição normal padrão.

Principais Distribuições

- Por exemplo o número aleatório (**r**) de uma distribuição normal (**norm**) pode ser extraído utilizando a função *rnorm*. Vejamos:
 - *rnorm*(10) Extrai 10 números aleatórios de uma distribuição padrão.
 - *rnorm*(10, 5, 2) Extrai 10 números aleatórios de uma distribuição $N(\mu = 5, \sigma = 2)$.
 - *dnorm*(*x*) Retorna o valor da função densidade da normal padrão em *x*.
 - *pnorm*(0) Retorna o valor de uma CDF normal padrão em $x = 0$.
 - *qnorm*(0.5) retorna o quantil 50% de uma distribuição normal padrão.

Principais Distribuições

- Por exemplo o número aleatório (**r**) de uma distribuição normal (**norm**) pode ser extraído utilizando a função *rnorm*. Vejamos:
 - *rnorm*(10) Extrai 10 números aleatórios de uma distribuição padrão.
 - *rnorm*(10, 5, 2) Extrai 10 números aleatórios de uma distribuição $N(\mu = 5, \sigma = 2)$.
 - *dnorm*(*x*) Retorna o valor da função densidade da normal padrão em *x*.
 - *pnorm*(0) Retorna o valor de uma CDF normal padrão em $x = 0$.
 - *qnorm*(0.5) retorna o quantil 50% de uma distribuição normal padrão.

Gerar repetições (rep)

- A função **rep** é utilizada para repetir algo n vezes.
- `rep(5, 10)` repete o valor 5 10 vezes.
- A função **rep** também funciona da seguinte forma:
- `rep(x, times = y)` `rep(repitax, yvezes)` onde x é o valor ou conjunto de valores que deve ser repetido, e times é o número de vezes).
- `rep(1 : 4, 2)` repete a sequência de 1 a 4 duas vezes.
- `rep(1 : 4, each = 2)` note a diferença ao usar o comando `each = 2`.

Gerar repetições (rep)

- A função **rep** é utilizada para repetir algo n vezes.
- `rep(5, 10)` repete o valor 5 10 vezes.
- A função **rep** também funciona da seguinte forma:
- `rep(x, times = y)` `rep(repitax, yvezes)` onde x é o valor ou conjunto de valores que deve ser repetido, e times é o número de vezes).
- `rep(1 : 4, 2)` repete a sequência de 1 a 4 duas vezes.
- `rep(1 : 4, each = 2)` note a diferença ao usar o comando `each = 2`.

Gerar repetições (rep)

- A função **rep** é utilizada para repetir algo n vezes.
- `rep(5, 10)` repete o valor 5 10 vezes.
- A função **rep** também funciona da seguinte forma:
 - `rep(x, times = y)` `rep(repita x, y vezes)` onde x é o valor ou conjunto de valores que deve ser repetido, e times é o número de vezes).
 - `rep(1 : 4, 2)` repete a sequência de 1 a 4 duas vezes.
 - `rep(1 : 4, each = 2)` note a diferença ao usar o comando `each = 2`.

Gerar repetições (rep)

- A função **rep** é utilizada para repetir algo n vezes.
- `rep(5, 10)` repete o valor 5 10 vezes.
- A função **rep** também funciona da seguinte forma:
- `rep(x, times = y)` `rep(repita x, y vezes)` onde x é o valor ou conjunto de valores que deve ser repetido, e times é o número de vezes).
- `rep(1 : 4, 2)` repete a sequência de 1 a 4 duas vezes.
- `rep(1 : 4, each = 2)` note a diferença ao usar o comando `each = 2`.

Gerar repetições (rep)

- A função **rep** é utilizada para repetir algo n vezes.
- `rep(5, 10)` repete o valor 5 10 vezes.
- A função **rep** também funciona da seguinte forma:
- `rep(x, times = y)` `rep(repita x, y vezes)` onde x é o valor ou conjunto de valores que deve ser repetido, e times é o número de vezes).
- `rep(1 : 4, 2)` repete a sequência de 1 a 4 duas vezes.
- `rep(1 : 4, each = 2)` note a diferença ao usar o comando `each = 2`.

Gerar repetições (rep)

- A função **rep** é utilizada para repetir algo n vezes.
- `rep(5, 10)` repete o valor 5 10 vezes.
- A função **rep** também funciona da seguinte forma:
- `rep(x, times = y)` `rep(repita x, y vezes)` onde x é o valor ou conjunto de valores que deve ser repetido, e times é o número de vezes).
- `rep(1 : 4, 2)` repete a sequência de 1 a 4 duas vezes.
- `rep(1 : 4, each = 2)` note a diferença ao usar o comando `each = 2`.

Gerar dados aleatórios

- **runif** (Gerar dados aleatórios com distribuição uniforme)
- *runif(n , $min = 0$, $max = 1$)* gera uma distribuição uniforme com n valores, começando em min e terminando em max .
- *runif(200, 80, 100)* gera 200 valores que vão de um mínimo de 80 até um máximo de 100.
- **rnorm** (Gerar dados aleatórios com distribuição normal).
- *rnorm(n , $mean = 0$, $sd = 1$)* gera n valores com distribuição uniforme, com média 0 e desvio padrão 1.
- *temp2 < -rnorm(200, 8, 10)* 200 valores com média 8 e desvio padrão 10.
- *hist(temp2)* Faz um histograma de frequências dos valores.

Gerar dados aleatórios

- **runif** (Gerar dados aleatórios com distribuição uniforme)
- *runif*(*n*, *min* = 0, *max* = 1) gera uma distribuição uniforme com *n* valores, começando em *min* e terminando em *max*.
- *runif*(200, 80, 100) gera 200 valores que vão de um mínimo de 80 até um máximo de 100.
- **rnorm** (Gerar dados aleatórios com distribuição normal).
- *rnorm*(*n*, *mean* = 0, *sd* = 1) gera *n* valores com distribuição uniforme, com média 0 e desvio padrão 1.
- *temp2* < -*rnorm*(200, 8, 10) 200 valores com média 8 e desvio padrão 10.
- *hist*(*temp2*) Faz um histograma de frequências dos valores.

Gerar dados aleatórios

- **runif** (Gerar dados aleatórios com distribuição uniforme)
- `runif(n, min = 0, max = 1)` gera uma distribuição uniforme com n valores, começando em `min` e terminando em `max`.
- `runif(200, 80, 100)` gera 200 valores que vão de um mínimo de 80 até um máximo de 100.
- **rnorm** (Gerar dados aleatórios com distribuição normal).
- `rnorm(n, mean = 0, sd = 1)` gera n valores com distribuição uniforme, com média 0 e desvio padrão 1.
- `temp2 <- -rnorm(200, 8, 10)` 200 valores com média 8 e desvio padrão 10.
- `hist(temp2)` Faz um histograma de frequências dos valores.

Gerar dados aleatórios

- **runif** (Gerar dados aleatórios com distribuição uniforme)
- `runif(n, min = 0, max = 1)` gera uma distribuição uniforme com n valores, começando em `min` e terminando em `max`.
- `runif(200, 80, 100)` gera 200 valores que vão de um mínimo de 80 até um máximo de 100.
- **rnorm** (Gerar dados aleatórios com distribuição normal).
- `rnorm(n, mean = 0, sd = 1)` gera n valores com distribuição uniforme, com média 0 e desvio padrão 1.
- `temp2 <- -rnorm(200, 8, 10)` 200 valores com média 8 e desvio padrão 10.
- `hist(temp2)` Faz um histograma de frequências dos valores.

Gerar dados aleatórios

- **runif** (Gerar dados aleatórios com distribuição uniforme)
- `runif(n, min = 0, max = 1)` gera uma distribuição uniforme com n valores, começando em `min` e terminando em `max`.
- `runif(200, 80, 100)` gera 200 valores que vão de um mínimo de 80 até um máximo de 100.
- **rnorm** (Gerar dados aleatórios com distribuição normal).
- `rnorm(n, mean = 0, sd = 1)` gera n valores com distribuição uniforme, com média 0 e desvio padrão 1.
- `temp2 <- -rnorm(200, 8, 10)` 200 valores com média 8 e desvio padrão 10.
- `hist(temp2)` Faz um histograma de frequências dos valores.

Gerar dados aleatórios

- **runif** (Gerar dados aleatórios com distribuição uniforme)
- `runif(n, min = 0, max = 1)` gera uma distribuição uniforme com n valores, começando em `min` e terminando em `max`.
- `runif(200, 80, 100)` gera 200 valores que vão de um mínimo de 80 até um máximo de 100.
- **rnorm** (Gerar dados aleatórios com distribuição normal).
- `rnorm(n, mean = 0, sd = 1)` gera n valores com distribuição uniforme, com média 0 e desvio padrão 1.
- `temp2 <- -rnorm(200, 8, 10)` 200 valores com média 8 e desvio padrão 10.
- `hist(temp2)` Faz um histograma de frequências dos valores.

Gerar dados aleatórios

- **runif** (Gerar dados aleatórios com distribuição uniforme)
- `runif(n, min = 0, max = 1)` gera uma distribuição uniforme com n valores, começando em `min` e terminando em `max`.
- `runif(200, 80, 100)` gera 200 valores que vão de um mínimo de 80 até um máximo de 100.
- **rnorm** (Gerar dados aleatórios com distribuição normal).
- `rnorm(n, mean = 0, sd = 1)` gera n valores com distribuição uniforme, com média 0 e desvio padrão 1.
- `temp2 < -rnorm(200, 8, 10)` 200 valores com média 8 e desvio padrão 10.
- `hist(temp2)` Faz um histograma de frequências dos valores.

Amostras aleatórias com o *Sample*

A função *sample*

- A função ***sample*** é utilizada para realizar amostras aleatórias e funciona da seguinte forma:
- *sample(x, size = 1, replace = FALSE)* onde:
 - *x* é o conjunto de dados do qual as amostras serão retiradas.
 - *size* é o número de amostras; e
 - *replace* é onde você indica se a amostra deve ser feita com reposição (*TRUE*) ou sem reposição (*FALSE*).
- *sample(1 : 10, 5)* tira 5 amostras com valores entre 1 e 10.
- **OBS.:** Como não especificamos o argumento ***replace*** o padrão é considerar que a amostra é sem reposição (= *FALSE*).
- *sample(1 : 10, 15, replace = TRUE)*

Amostras aleatórias com o *Sample*

A função *sample*

- A função ***sample*** é utilizada para realizar amostras aleatórias e funciona da seguinte forma:
- *sample*(*x*, *size* = 1, *replace* = *FALSE*) onde:
 - *x* é o conjunto de dados do qual as amostras serão retiradas.
 - *size* é o número de amostras; e
 - *replace* é onde você indica se a amostra deve ser feita com reposição (*TRUE*) ou sem reposição (*FALSE*).
- *sample*(1 : 10, 5) tira 5 amostras com valores entre 1 e 10.
- **OBS.:** Como não especificamos o argumento ***replace*** o padrão é considerar que a amostra é sem reposição (= *FALSE*).
- *sample*(1 : 10, 15, *replace* = *TRUE*)

Amostras aleatórias com o *Sample*

A função *sample*

- A função ***sample*** é utilizada para realizar amostras aleatórias e funciona da seguinte forma:
- *sample*(*x*, *size* = 1, *replace* = *FALSE*) onde:
 - *x* é o conjunto de dados do qual as amostras serão retiradas.
 - *size* é o número de amostras; e
 - *replace* é onde você indica se a amostra deve ser feita com reposição (*TRUE*) ou sem reposição (*FALSE*).
- *sample*(1 : 10, 5) tira 5 amostras com valores entre 1 e 10.
- **OBS.:** Como não especificamos o argumento ***replace*** o padrão é considerar que a amostra é sem reposição (= *FALSE*).
- *sample*(1 : 10, 15, *replace* = *TRUE*)

Amostras aleatórias com o *Sample*

A função *sample*

- A função ***sample*** é utilizada para realizar amostras aleatórias e funciona da seguinte forma:
- *sample*(*x*, *size* = 1, *replace* = *FALSE*) onde:
 - ***x*** é o conjunto de dados do qual as amostras serão retiradas.
 - ***size*** é o número de amostras; e
 - *replace* é onde você indica se a amostra deve ser feita com reposição (*TRUE*) ou sem reposição (*FALSE*).
- *sample*(1 : 10, 5) tira 5 amostras com valores entre 1 e 10.
- **OBS.:** Como não especificamos o argumento ***replace*** o padrão é considerar que a amostra é sem reposição (= *FALSE*).
- *sample*(1 : 10, 15, *replace* = *TRUE*)

Amostras aleatórias com o *Sample*

A função *sample*

- A função ***sample*** é utilizada para realizar amostras aleatórias e funciona da seguinte forma:
- *sample*(*x*, *size* = 1, *replace* = *FALSE*) onde:
 - ***x*** é o conjunto de dados do qual as amostras serão retiradas.
 - ***size*** é o número de amostras; e
 - ***replace*** é onde você indica se a amostra deve ser feita com reposição (*TRUE*) ou sem reposição (*FALSE*).
- *sample*(1 : 10, 5) tira 5 amostras com valores entre 1 e 10.
- **OBS.:** Como não especificamos o argumento ***replace*** o padrão é considerar que a amostra é sem reposição (= *FALSE*).
- *sample*(1 : 10, 15, *replace* = *TRUE*)

Amostras aleatórias com o *Sample*

A função *sample*

- A função ***sample*** é utilizada para realizar amostras aleatórias e funciona da seguinte forma:
- *sample*(*x*, *size* = 1, *replace* = *FALSE*) onde:
 - ***x*** é o conjunto de dados do qual as amostras serão retiradas.
 - ***size*** é o número de amostras; e
 - *replace* é onde você indica se a amostra deve ser feita com reposição (*TRUE*) ou sem reposição (*FALSE*).
- *sample*(1 : 10, 5) tira 5 amostras com valores entre 1 e 10.
- **OBS.:** Como não especificamos o argumento ***replace*** o padrão é considerar que a amostra é sem reposição (= *FALSE*).
- *sample*(1 : 10, 15, *replace* = *TRUE*)

Amostras aleatórias com o *Sample*

A função *sample*

- A função ***sample*** é utilizada para realizar amostras aleatórias e funciona da seguinte forma:
- *sample*(*x*, *size* = 1, *replace* = *FALSE*) onde:
 - ***x*** é o conjunto de dados do qual as amostras serão retiradas.
 - ***size*** é o número de amostras; e
 - ***replace*** é onde você indica se a amostra deve ser feita com reposição (*TRUE*) ou sem reposição (*FALSE*).
- *sample*(1 : 10, 5) tira 5 amostras com valores entre 1 e 10.
- **OBS.:** Como não especificamos o argumento ***replace*** o padrão é considerar que a amostra é sem reposição (= *FALSE*).
- *sample*(1 : 10, 15, *replace* = *TRUE*)

Amostras aleatórias com o *Sample*

A função *sample*

- A função ***sample*** é utilizada para realizar amostras aleatórias e funciona da seguinte forma:
- *sample*(*x*, *size* = 1, *replace* = *FALSE*) onde:
 - ***x*** é o conjunto de dados do qual as amostras serão retiradas.
 - ***size*** é o número de amostras; e
 - *replace* é onde você indica se a amostra deve ser feita com reposição (*TRUE*) ou sem reposição (*FALSE*).
- *sample*(1 : 10, 5) tira 5 amostras com valores entre 1 e 10.
- **OBS.:** Como não especificamos o argumento ***replace*** o padrão é considerar que a amostra é sem reposição (= *FALSE*).
- *sample*(1 : 10, 15, *replace* = *TRUE*)

Criando Matrices

```
•  
> x<-1:12  
> x  
[1] 1 2 3 4 5 6 7 8 9 10 11 12  
> xmat<-matrix(x,ncol=3)  
> xmat  
[,1] [,2] [,3]  
[1,] 1 5 9  
[2,] 2 6 10  
[3,] 3 7 11  
[4,] 4 8 12
```

Criando Matrizes

- Para inverter este padrão deve-se adicionar o argumento `byrow=TRUE`, (que em inglês significa "por linhas").

```
> xmat<-matrix(x,ncol=3,byrow=TRUE)
```

```
> xmat
```

```
[,1] [,2] [,3]
```

```
[1,] 1 2 3
```

```
[2,] 4 5 6
```

```
[3,] 7 8 9
```

```
[4,] 10 11 12
```

Criando Matrices

- ```
coluna1<-c(2,1,0) #Cria vetor coluna
coluna2<-c(1,3,1)
coluna3<-c(1,1,2)
```

- ```
A<-cbind(coluna1,coluna2,coluna3)  
#Cria uma matriz de nome A.  
A
```

```
coluna1 coluna2 coluna3  
[1,] 2 1 1  
[2,] 1 3 1  
[3,] 0 1 2
```

Criando Matrices



```
coluna1<-c(2,1,0) #Cria vetor coluna  
coluna2<-c(1,3,1)  
coluna3<-c(1,1,2)
```



```
A<-cbind(coluna1,coluna2,coluna3)  
#Cria uma matriz de nome A.  
A
```

```
coluna1 coluna2 coluna3  
[1,] 2 1 1  
[2,] 1 3 1  
[3,] 0 1 2
```


Criando Matrizes

t(A) #Obtém a transposta de A

[,1] [,2] [,3]

coluna1 2 1 0

coluna2 1 3 1

coluna3 1 1 2

solve(A) #inversa da matriz A

[,1] [,2] [,3]

coluna1 0.5555556 -0.1111111 -0.2222222

coluna2 -0.2222222 0.4444444 -0.1111111

coluna3 0.1111111 -0.2222222 0.5555556

Criando Matrizes



`t(A)` #Obtém a transposta de A

`[,1] [,2] [,3]`

coluna1 2 1 0

coluna2 1 3 1

coluna3 1 1 2



`solve(A)` #inversa da matriz A

`[,1] [,2] [,3]`

coluna1 0.5555556 -0.1111111 -0.2222222

coluna2 -0.2222222 0.4444444 -0.1111111

coluna3 0.1111111 -0.2222222 0.5555556

Criando Matrizes

- Finalmente esta mesma função pode ser usada para resolver sistemas de equações lineares. Imaginemos o seguinte sistema de equações,

$$-4x + 0.3y = 12.3$$

$$54.3x - 4y = 45$$

```
> coefs<-matrix(c(-4,0.3,54.3,-4),2,2,byrow=T)
```

```
> ys <- c(12.3, 45)
```

```
> solve(coefs, ys)
```

```
[1] 216.2069 2923.7586
```

Resolvendo com o R

- Considere as variáveis X e Y que representam, respectivamente, as notas de duas disciplinas, para um grupo de 6 alunos.

$X = \{90, 95, 97, 98, 100, 60\}$

$Y = \{60, 70, 80, 60, 90, 75\}$

$X \leftarrow c(90, 95, 97, 98, 100, 60)$ #criando o vetor X

$\text{sum}(X)$ #calculando o somatório

[1] 540

$Y \leftarrow c(60, 70, 80, 60, 90, 75)$ #criando o vetor Y

$\text{sum}(Y^2) - Y[1]^2 - Y[5]^2$ #somatório subtraindo da exceção

[1] 20525

Resolvendo com o R

- Considere as variáveis X e Y que representam, respectivamente, as notas de duas disciplinas, para um grupo de 6 alunos.

$X = \{90, 95, 97, 98, 100, 60\}$

$Y = \{60, 70, 80, 60, 90, 75\}$



`X<-c(90,95,97,98,100,60) #criando o vetor X`

`sum(X) #calculando o somatório`

`[1] 540`



`Y<-c(60,70,80,60,90,75) #criando o vetor Y`

`sum(Y^2)-Y[1]^2-Y[5]^2 #somatório subtraindo da exceção`

`[1] 20525`

Resolvendo com o R

- Considere as variáveis X e Y que representam, respectivamente, as notas de duas disciplinas, para um grupo de 6 alunos.

$X = \{90, 95, 97, 98, 100, 60\}$

$Y = \{60, 70, 80, 60, 90, 75\}$



`X<-c(90,95,97,98,100,60) #criando o vetor X`

`sum(X) #calculando o somatório`

`[1] 540`



`Y<-c(60,70,80,60,90,75) #criando o vetor Y`

`sum(Y^2)-Y[1]^2-Y[5]^2 #somatório subtraindo da exceção`

`[1] 20525`

workspace

- O **workspace** é a área de trabalho do R, quando salvamos um **workspace** em uma pasta e o abrimos a partir dela o R imediatamente reconhecerá o endereço da pasta como sendo seu diretório de trabalho.
- Para conferir o diretório de trabalho use o comando abaixo:
- `getwd()`

workspace

- O **workspace** é a área de trabalho do R, quando salvamos um **workspace** em uma pasta e o abrimos a partir dela o R imediatamente reconhecerá o endereço da pasta como sendo seu diretório de trabalho.
- Para conferir o diretório de trabalho use o comando abaixo:
 - `getwd()`

workspace

- O **workspace** é a área de trabalho do R, quando salvamos um **workspace** em uma pasta e o abrirmos a partir dela o R imediatamente reconhecerá o endereço da pasta como sendo seu diretório de trabalho.
- Para conferir o diretório de trabalho use o comando abaixo:
- `getwd()`

Importando Dados no R

- Importar dados no R é bastante simples. Para arquivos no formato do Stata use o pacote **foreign**. Para dados no SPSS recomenda-se o pacote **Hmisc** para facilitar.
- Suponha que você tenha um arquivo no formato .txt, ou seja, "data.txt", da seguinte forma:

```
1997,3.1,4
1998,7.2,19
1999,1.7,2
2000,1.1,13
```

- Importando Dados

```
A <- read.table("x.data.txt", header=F, sep=",",
col.names=c("ano", "x", "y"))
View(A)
```

Importando Dados no R

- Importar dados no R é bastante simples. Para arquivos no formato do Stata use o pacote **foreign**. Para dados no SPSS recomenda-se o pacote **Hmisc** para facilitar.
- Suponha que você tenha um arquivo no formato .txt, ou seja, "data.txt", da seguinte forma:

```
1997,3.1,4
1998,7.2,19
1999,1.7,2
2000,1.1,13
```

- Importando Dados

```
A <- read.table("x.data.txt", header=F, sep=",",
col.names=c("ano", "x", "y"))
View(A)
```

Importando Dados no R

Importando do Excel, SPSS e Stata

- **Excel:**

Uma das melhores formas para ler um arquivo em Excel é exportá-lo para **.CSV** e importar para o **R** utilizando o seguinte comando:

- `# read in the first worksheet from the workbook myexcel.xlsx.
first row contains variable names
library(xlsx)
mydata <- read.xlsx("c:/myexcel.xlsx", 1)`
- `# read in the worksheet named mysheet
mydata <- read.xlsx("c:/myexcel.xlsx", sheetName = "mysheet")`

Importando Dados no R

Importando do Excel, SPSS e Stata

- **Excel:**

Uma das melhores formas para ler um arquivo em Excel é exportá-lo para **.CSV** e importar para o **R** utilizando o seguinte comando:

- ```
read in the first worksheet from the workbook myexcel.xlsx.
first row contains variable names
library(xlsx)
mydata <- read.xlsx("c:/myexcel.xlsx", 1)
```
- ```
# read in the worksheet named mysheet  
mydata <- read.xlsx("c:/myexcel.xlsx", sheetName = "mysheet")
```

Importando Dados no R

Importando do Excel, SPSS e Stata

- **Excel:**

Uma das melhores formas para ler um arquivo em Excel é exportá-lo para **.CSV** e importar para o **R** utilizando o seguinte comando:

- ```
read in the first worksheet from the workbook myexcel.xlsx.
first row contains variable names
library(xlsx)
mydata <- read.xlsx("c:/myexcel.xlsx", 1)
```
- ```
# read in the worksheet named mysheet  
mydata <- read.xlsx("c:/myexcel.xlsx", sheetName = "mysheet")
```

Importando Dados no R

Importando do Excel, SPSS e Stata

- **SPSS:**

save SPSS dataset in transport format get file="c:/mydata.sav".
Export outfile="c:/mydata.por".

- # No R:

```
library(Hmisc)
```

```
mydata <- spss.get("c:/mydata.por", use.value.labels=TRUE)
```

Importando Dados no R

Importando do Excel, SPSS e Stata

- **SPSS:**

save SPSS dataset in transport format get file="c:/mydata.sav".
Export outfile="c:/mydata.por".

- # No R:

```
library(Hmisc)
```

```
mydata <- spss.get("c:/mydata.por", use.value.labels=TRUE)
```


Importando Dados no R

Importando do Excel, SPSS e Stata

- **STATA:**

- # input Stata file.

- # No R:

- library(foreign)

- mydata <- read.systat("c:/mydata.dta")

Importando Dados no R

Importando do Excel, SPSS e Stata

- **STATA:**

- # input Stata file.

- # No R:

- library(foreign)

- mydata <- read.systat("c:/mydata.dta")

Importando Dados no R

Importar e Exportar Dados

- É possível importar no R arquivos SAS, Stata, SPSS, Minitab, DBF e Epilnfo utilizando as funções do pacote **foreign**. A seguir estão as funções de importação e exportação:
- `read.table(arquivo, header = TRUE, sep = " ")` - importa arquivo em formato de planilha e cria data.frame com o mesmo. O nome do arquivo tem de estar entre aspas e conter a extensão (e.g., .txt). Em geral, utiliza-se o argumento **header = TRUE** para utilizar a primeira linha da tabela como cabeçalho (i.e., linha com o nome das colunas). Finalmente, utiliza-se o argumento **sep = " "** que indica espaço em branco como separador de colunas para importar corretamente a estrutura da planilha.

Importando Dados no R

Importar e Exportar Dados

- É possível importar no R arquivos SAS, Stata, SPSS, Minitab, DBF e Epilnfo utilizando as funções do pacote **foreign**. A seguir estão as funções de importação e exportação:
- **read.table(arquivo, header = TRUE, sep = " ")** - importa arquivo em formato de planilha e cria data.frame com o mesmo. O nome do arquivo tem de estar entre aspas e conter a extensão (e.g., .txt). Em geral, utiliza-se o argumento **header = TRUE** para utilizar a primeira linha da tabela como cabeçalho (i.e., linha com o nome das colunas). Finalmente, utiliza-se o argumento **sep = " "** que indica espaço em branco como separador de colunas para importar corretamente a estrutura da planilha.

Importando Dados no R

Importar e Exportar Dados

- **read.csv(arquivo, header=TRUE, sep =",")** - importa arquivo .csv (i.e., arquivo com vírgula como separador de colunas). Esta é a forma de importação mais utilizada nesse curso porque esse tipo de arquivo pode ser elaborado na planilha de dados mais utilizada do planeta (i.e., Excel). As mesmas regras de **read.table()** podem ser aplicadas com essa função, mas é importante enfatizar que o tipo de separador é vírgula, sendo então utilizado o argumento **sep = ","**.
- Escolhendo o arquivo do Computador:
`Data <- read.table(file.choose(),header=TRUE, sep=",")`

Importando Dados no R

Importar e Exportar Dados

- **`read.csv(arquivo, header=TRUE, sep =",")`** - importa arquivo .csv (i.e., arquivo com vírgula como separador de colunas). Esta é a forma de importação mais utilizada nesse curso porque esse tipo de arquivo pode ser elaborado na planilha de dados mais utilizada do planeta (i.e., Excel). As mesmas regras de **`read.table()`** podem ser aplicadas com essa função, mas é importante enfatizar que o tipo de separador é vírgula, sendo então utilizado o argumento **`sep = ","`**.
- **Escolhendo o arquivo do Computador:**
`Data <- read.table(file.choose(),header=TRUE, sep=",")`

Importando Dados no R

Importar e Exportar Dados

- **Exportando para Planilha de Excel:**

- `library(xlsx)`
- `write.xlsx(mydata, "c:/mydata.xlsx")`

- **Exportando para SPSS:**

- `library(foreign)`
- `write.foreign(mydata, "c:/mydata.txt", "c:/mydata.sps",
package="SPSS")`

- **Exportando para Stata:**

- `library(foreign)`
- `write.dta(mydata, "c:/mydata.dta")`

Importando Dados no R

Importar e Exportar Dados

- **Exportando para Planilha de Excel:**

- `library(xlsx)`
- `write.xlsx(mydata, "c:/mydata.xlsx")`

- **Exportando para SPSS:**

- `library(foreign)`
- `write.foreign(mydata, "c:/mydata.txt", "c:/mydata.sps",
package="SPSS")`

- **Exportando para Stata:**

- `library(foreign)`
- `write.dta(mydata, "c:/mydata.dta")`

Importando Dados no R

Importar e Exportar Dados

- **Exportando para Planilha de Excel:**

- `library(xlsx)`
- `write.xlsx(mydata, "c:/mydata.xlsx")`

- **Exportando para SPSS:**

- `library(foreign)`
- `write.foreign(mydata, "c:/mydata.txt", "c:/mydata.sps",
package="SPSS")`

- **Exportando para Stata:**

- `library(foreign)`
- `write.dta(mydata, "c:/mydata.dta")`

Importando Dados no R

Importar e Exportar Dados

- **Exportando para Planilha de Excel:**

- `library(xlsx)`
- `write.xlsx(mydata, "c:/mydata.xlsx")`

- **Exportando para SPSS:**

- `library(foreign)`
- `write.foreign(mydata, "c:/mydata.txt", "c:/mydata.sps",
package="SPSS")`

- **Exportando para Stata:**

- `library(foreign)`
- `write.dta(mydata, "c:/mydata.dta")`

Importando Dados no R

Importar e Exportar Dados

- **Exportando para Planilha de Excel:**

- `library(xlsx)`
- `write.xlsx(mydata, "c:/mydata.xlsx")`

- **Exportando para SPSS:**

- `library(foreign)`
- `write.foreign(mydata, "c:/mydata.txt", "c:/mydata.sps",
package="SPSS")`

- **Exportando para Stata:**

- `library(foreign)`
- `write.dta(mydata, "c:/mydata.dta")`

Importando Dados no R

Importar e Exportar Dados

- **Exportando para Planilha de Excel:**

- `library(xlsx)`
- `write.xlsx(mydata, "c:/mydata.xlsx")`

- **Exportando para SPSS:**

- `library(foreign)`
- `write.foreign(mydata, "c:/mydata.txt", "c:/mydata.sps",
package="SPSS")`

- **Exportando para Stata:**

- `library(foreign)`
- `write.dta(mydata, "c:/mydata.dta")`

Importando Dados no R

Importar e Exportar Dados

- **Exportando para Planilha de Excel:**

- `library(xlsx)`
- `write.xlsx(mydata, "c:/mydata.xlsx")`

- **Exportando para SPSS:**

- `library(foreign)`
- `write.foreign(mydata, "c:/mydata.txt", "c:/mydata.sps", package="SPSS")`

- **Exportando para Stata:**

- `library(foreign)`
- `write.dta(mydata, "c:/mydata.dta")`

Importando Dados no R

Importar e Exportar Dados

- **Exportando para Planilha de Excel:**

- `library(xlsx)`
- `write.xlsx(mydata, "c:/mydata.xlsx")`

- **Exportando para SPSS:**

- `library(foreign)`
- `write.foreign(mydata, "c:/mydata.txt", "c:/mydata.sps",
package="SPSS")`

- **Exportando para Stata:**

- `library(foreign)`
- `write.dta(mydata, "c:/mydata.dta")`

Importando Dados no R

Importar e Exportar Dados

- **Exportando para Planilha de Excel:**

- `library(xlsx)`
- `write.xlsx(mydata, "c:/mydata.xlsx")`

- **Exportando para SPSS:**

- `library(foreign)`
- `write.foreign(mydata, "c:/mydata.txt", "c:/mydata.sps", package="SPSS")`

- **Exportando para Stata:**

- `library(foreign)`
- `write.dta(mydata, "c:/mydata.dta")`

Algumas Funções no R

Plot

- **# Especificando as opções dos eixos com plot()**
 - `plot(x, y, main="title", sub="subtitle", xlab="X-axis label", ylab="y-axis label", xlim=c(xmin, xmax), ylim=c(ymin, ymax))`
- **Exemplo:**
 - `attach(mtcars)`
 - `plot(wt, mpg, main="Milage vs. Car Weight", xlab="Weight", ylab="Mileage", pch=18, col="blue")`
 - `text(wt, mpg, row.names(mtcars), cex=0.6, pos=4, col="red")`

Algumas Funções no R

Plot

- **# Especificando as opções dos eixos com plot()**
 - `plot(x, y, main="title", sub="subtitle", xlab="X-axis label", ylab="y-axis label", xlim=c(xmin, xmax), ylim=c(ymin, ymax))`
- **Exemplo:**
 - `attach(mtcars)`
 - `plot(wt, mpg, main="Milage vs. Car Weight", xlab="Weight", ylab="Mileage", pch=18, col="blue")`
 - `text(wt, mpg, row.names(mtcars), cex=0.6, pos=4, col="red")`

Algumas Funções no R

Plot

- **# Especificando as opções dos eixos com plot()**
 - `plot(x, y, main="title", sub="subtitle", xlab="X-axis label", ylab="y-axis label", xlim=c(xmin, xmax), ylim=c(ymin, ymax))`
- **Exemplo:**
 - `attach(mtcars)`
 - `plot(wt, mpg, main="Milage vs. Car Weight", xlab="Weight", ylab="Mileage", pch=18, col="blue")`
 - `text(wt, mpg, row.names(mtcars), cex=0.6, pos=4, col="red")`

Algumas Funções no R

Plot

- **# Especificando as opções dos eixos com plot()**
 - `plot(x, y, main="title", sub="subtitle", xlab="X-axis label", ylab="y-axis label", xlim=c(xmin, xmax), ylim=c(ymin, ymax))`
- **Exemplo:**
 - `attach(mtcars)`
 - `plot(wt, mpg, main="Milage vs. Car Weight", xlab="Weight", ylab="Mileage", pch=18, col="blue")`
 - `text(wt, mpg, row.names(mtcars), cex=0.6, pos=4, col="red")`

Algumas Funções no R

Plot

- **# Especificando as opções dos eixos com plot()**
 - `plot(x, y, main="title", sub="subtitle", xlab="X-axis label", ylab="y-axis label", xlim=c(xmin, xmax), ylim=c(ymin, ymax))`
- **Exemplo:**
 - `attach(mtcars)`
 - `plot(wt, mpg, main="Milage vs. Car Weight", xlab="Weight", ylab="Mileage", pch=18, col="blue")`
 - `text(wt, mpg, row.names(mtcars), cex=0.6, pos=4, col="red")`

Algumas Funções no R

Plot

- **# Especificando as opções dos eixos com plot()**
 - `plot(x, y, main="title", sub="subtitle", xlab="X-axis label", ylab="y-axis label", xlim=c(xmin, xmax), ylim=c(ymin, ymax))`
- **Exemplo:**
 - `attach(mtcars)`
 - `plot(wt, mpg, main="Milage vs. Car Weight", xlab="Weight", ylab="Mileage", pch=18, col="blue")`
 - `text(wt, mpg, row.names(mtcars), cex=0.6, pos=4, col="red")`

Algumas Funções no R

Funções `par()` ou `layout()`

- Com a função `par()`, você pode incluir as opções `mfrow=c(nrows, ncols)` para criar uma matriz de gráficos de `nrows` x `ncols` que são ajustados pela linha. `mfcow=c(nrows, ncols)` ajusta os gráficos na matriz por coluna.
- Exemplo: `# 4 figuras organizadas em 2 linhas e 2 colunas`
 - `attach(mtcars)`
 - `par(mfrow=c(2,2))`
 - `plot(wt,mpg, main="Scatterplot of wt vs. mpg")`
 - `plot(wt,disp, main="Scatterplot of wt vs disp")`
 - `hist(wt, main="Histogram of wt")`
 - `boxplot(wt, main="Boxplot of wt")`

Algumas Funções no R

Funções **par()** ou **layout()**

- Com a função **par()**, você pode incluir as opções **mfrow=c(nrows, ncols)** para criar uma matriz de gráficos de *nrows* x *ncols* que são ajustados pela linha. **mfcrow=c(nrows, ncols)** ajusta os gráficos na matriz por coluna.
- **Exemplo: # 4 figuras organizadas em 2 linhas e 2 colunas**
 - `attach(mtcars)`
 - `par(mfrow=c(2,2))`
 - `plot(wt,mpg, main="Scatterplot of wt vs. mpg")`
 - `plot(wt,disp, main="Scatterplot of wt vs disp")`
 - `hist(wt, main="Histogram of wt")`
 - `boxplot(wt, main="Boxplot of wt")`

Algumas Funções no R

Funções **par()** ou **layout()**

- Com a função **par()**, você pode incluir as opções **mfrow=c(nrows, ncols)** para criar uma matriz de gráficos de *nrows* x *ncols* que são ajustados pela linha. **mfcrow=c(nrows, ncols)** ajusta os gráficos na matriz por coluna.
- **Exemplo: # 4 figuras organizadas em 2 linhas e 2 colunas**
 - `attach(mtcars)`
 - `par(mfrow=c(2,2))`
 - `plot(wt,mpg, main="Scatterplot of wt vs. mpg")`
 - `plot(wt,disp, main="Scatterplot of wt vs disp")`
 - `hist(wt, main="Histogram of wt")`
 - `boxplot(wt, main="Boxplot of wt")`

Algumas Funções no R

Funções **par()** ou **layout()**

- Com a função **par()**, você pode incluir as opções **mfrow=c(nrows, ncols)** para criar uma matriz de gráficos de *nrows* x *ncols* que são ajustados pela linha. **mfcow=c(nrows, ncols)** ajusta os gráficos na matriz por coluna.
- **Exemplo: # 4 figuras organizadas em 2 linhas e 2 colunas**
 - `attach(mtcars)`
 - `par(mfrow=c(2,2))`
 - `plot(wt,mpg, main="Scatterplot of wt vs. mpg")`
 - `plot(wt,disp, main="Scatterplot of wt vs disp")`
 - `hist(wt, main="Histogram of wt")`
 - `boxplot(wt, main="Boxplot of wt")`

Algumas Funções no R

Funções `par()` ou `layout()`

- Com a função `par()`, você pode incluir as opções `mfrow=c(nrows, ncols)` para criar uma matriz de gráficos de *nrows* x *ncols* que são ajustados pela linha. `mfcow=c(nrows, ncols)` ajusta os gráficos na matriz por coluna.
- Exemplo: # 4 figuras organizadas em 2 linhas e 2 colunas**
 - `attach(mtcars)`
 - `par(mfrow=c(2,2))`
 - `plot(wt,mpg, main="Scatterplot of wt vs. mpg")`
 - `plot(wt,disp, main="Scatterplot of wt vs disp")`
 - `hist(wt, main="Histogram of wt")`
 - `boxplot(wt, main="Boxplot of wt")`

Algumas Funções no R

Funções **par()** ou **layout()**

- Com a função **par()**, você pode incluir as opções **mfrow=c(nrows, ncols)** para criar uma matriz de gráficos de *nrows* x *ncols* que são ajustados pela linha. **mfcrow=c(nrows, ncols)** ajusta os gráficos na matriz por coluna.
- **Exemplo: # 4 figuras organizadas em 2 linhas e 2 colunas**
 - `attach(mtcars)`
 - `par(mfrow=c(2,2))`
 - `plot(wt,mpg, main="Scatterplot of wt vs. mpg")`
 - `plot(wt,disp, main="Scatterplot of wt vs disp")`
 - `hist(wt, main="Histogram of wt")`
 - `boxplot(wt, main="Boxplot of wt")`

Algumas Funções no R

Funções **par()** ou **layout()**

- Com a função **par()**, você pode incluir as opções **mfrow=c(nrows, ncols)** para criar uma matriz de gráficos de *nrows* x *ncols* que são ajustados pela linha. **mfcow=c(nrows, ncols)** ajusta os gráficos na matriz por coluna.
- Exemplo: # 4 figuras organizadas em 2 linhas e 2 colunas**
 - `attach(mtcars)`
 - `par(mfrow=c(2,2))`
 - `plot(wt,mpg, main="Scatterplot of wt vs. mpg")`
 - `plot(wt,disp, main="Scatterplot of wt vs disp")`
 - `hist(wt, main="Histogram of wt")`
 - `boxplot(wt, main="Boxplot of wt")`

Algumas Funções no R

Funções **par()** ou **layout()**

- Com a função **par()**, você pode incluir as opções **mfrow=c(nrows, ncols)** para criar uma matriz de gráficos de *nrows* x *ncols* que são ajustados pela linha. **mfcow=c(nrows, ncols)** ajusta os gráficos na matriz por coluna.
- **Exemplo: # 4 figuras organizadas em 2 linhas e 2 colunas**
 - `attach(mtcars)`
 - `par(mfrow=c(2,2))`
 - `plot(wt,mpg, main="Scatterplot of wt vs. mpg")`
 - `plot(wt,disp, main="Scatterplot of wt vs disp")`
 - `hist(wt, main="Histogram of wt")`
 - `boxplot(wt, main="Boxplot of wt")`

Algumas Funções no R

Funções `par()` ou `layout()`

- **Exemplo: # 3 figuras organizadas em 3 linhas e 1 coluna**

- `attach(mtcars)`
- `par(mfrow=c(3,1))`
- `hist(wt)`
- `hist(mpg)`
- `hist(displ)`

Algumas Funções no R

Funções `par()` ou `layout()`

- **Exemplo: # 3 figuras organizadas em 3 linhas e 1 coluna**
 - `attach(mtcars)`
 - `par(mfrow=c(3,1))`
 - `hist(wt)`
 - `hist(mpg)`
 - `hist(displ)`

Algumas Funções no R

Funções `par()` ou `layout()`

- **Exemplo: # 3 figuras organizadas em 3 linhas e 1 coluna**
 - `attach(mtcars)`
 - `par(mfrow=c(3,1))`
 - `hist(wt)`
 - `hist(mpg)`
 - `hist(displ)`

Algumas Funções no R

Funções `par()` ou `layout()`

- **Exemplo: # 3 figuras organizadas em 3 linhas e 1 coluna**
 - `attach(mtcars)`
 - `par(mfrow=c(3,1))`
 - `hist(wt)`
 - `hist(mpg)`
 - `hist(displ)`

Algumas Funções no R

Funções `par()` ou `layout()`

- **Exemplo: # 3 figuras organizadas em 3 linhas e 1 coluna**
 - `attach(mtcars)`
 - `par(mfrow=c(3,1))`
 - `hist(wt)`
 - `hist(mpg)`
 - `hist(displ)`

Algumas Funções no R

Funções `par()` ou `layout()`

- **Exemplo: # 3 figuras organizadas em 3 linhas e 1 coluna**
 - `attach(mtcars)`
 - `par(mfrow=c(3,1))`
 - `hist(wt)`
 - `hist(mpg)`
 - `hist(displ)`

Algumas Funções no R

Funções **par()** ou **layout()**

- A função **layout()** tem a forma **layout(mat)** onde *mat* é um objeto da matriz que especifica o local das N figuras para plotagem.
- Exemplo: `# Uma figura na linha 1 e duas figuras na linha 2`
 - `attach(mtcars)`
 - `layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))`
 - `hist(wt)`
 - `hist(mpg)`
 - `hist(displ)`

Algumas Funções no R

Funções **par()** ou **layout()**

- A função **layout()** tem a forma **layout(mat)** onde *mat* é um objeto da matriz que especifica o local das N figuras para plotagem.
- **Exemplo: # Uma figura na linha 1 e duas figuras na linha 2**
 - `attach(mtcars)`
 - `layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))`
 - `hist(wt)`
 - `hist(mpg)`
 - `hist(displ)`

Algumas Funções no R

Funções **par()** ou **layout()**

- A função **layout()** tem a forma **layout(mat)** onde *mat* é um objeto da matriz que especifica o local das N figuras para plotagem.
- **Exemplo: # Uma figura na linha 1 e duas figuras na linha 2**
 - `attach(mtcars)`
 - `layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))`
 - `hist(wt)`
 - `hist(mpg)`
 - `hist(displ)`

Algumas Funções no R

Funções **par()** ou **layout()**

- A função **layout()** tem a forma **layout(mat)** onde *mat* é um objeto da matriz que especifica o local das N figuras para plotagem.
- **Exemplo: # Uma figura na linha 1 e duas figuras na linha 2**
 - `attach(mtcars)`
 - `layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))`
 - `hist(wt)`
 - `hist(mpg)`
 - `hist(displ)`

Algumas Funções no R

Funções **par()** ou **layout()**

- A função **layout()** tem a forma **layout(mat)** onde *mat* é um objeto da matriz que especifica o local das N figuras para plotagem.
- **Exemplo: # Uma figura na linha 1 e duas figuras na linha 2**
 - `attach(mtcars)`
 - `layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))`
 - `hist(wt)`
 - `hist(mpg)`
 - `hist(displ)`

Algumas Funções no R

Funções **par()** ou **layout()**

- A função **layout()** tem a forma **layout(mat)** onde *mat* é um objeto da matriz que especifica o local das N figuras para plotagem.
- **Exemplo: # Uma figura na linha 1 e duas figuras na linha 2**
 - `attach(mtcars)`
 - `layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))`
 - `hist(wt)`
 - `hist(mpg)`
 - `hist(displ)`

Algumas Funções no R

Funções **par()** ou **layout()**

- A função **layout()** tem a forma **layout(mat)** onde *mat* é um objeto da matriz que especifica o local das N figuras para plotagem.
- **Exemplo: # Uma figura na linha 1 e duas figuras na linha 2**
 - `attach(mtcars)`
 - `layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))`
 - `hist(wt)`
 - `hist(mpg)`
 - `hist(displ)`

Data-frame

- Um **data-frame** é utilizado para armazenar tabelas de dados. É uma lista de vetores de mesmo comprimento.

- **Por Exemplo:** A variável *df* é um data-frame contendo três vetores: *n*, *s* e *b*.

```
> n = c(2, 3, 5)
```

```
> s = c("aa", "bb", "cc")
```

```
> b = c(TRUE, FALSE, TRUE)
```

```
> df = data.frame(n, s, b) # df is a data frame
```

Data-frame

- Um **data-frame** é utilizado para armazenar tabelas de dados. É uma lista de vetores de mesmo comprimento.
- **Por Exemplo:** A variável *df* é um data-frame contendo três vetores: *n*, *s* e *b*.

```
> n = c(2, 3, 5)
> s = c("aa", "bb", "cc")
> b = c(TRUE, FALSE, TRUE)
> df = data.frame(n, s, b) # df is a data frame
```

Data-frame

- Os **data-frame** criam-se indicando ao R o nome de cada coluna da tabela de dados e respectivo conteúdo.

```
> notas.inform <- data.frame(  
  nros = c(2355, 3456, 2334, 5456),  
  turma = c("tp1", "tp1", "tp2", "tp3"),  
  notas = c(10.3, 9.3, 14.2, 15))
```

- A diferença fundamental entre um **data-frame** e uma matriz é que num **data-frame** os dados não precisam de ser todos do mesmo tipo.

Data-frame

- Os **data-frame** criam-se indicando ao R o nome de cada coluna da tabela de dados e respectivo conteúdo.

```
> notas.inform <- data.frame(  
  nros = c(2355, 3456, 2334, 5456),  
  turma = c("tp1", "tp1", "tp2", "tp3"),  
  notas = c(10.3, 9.3, 14.2, 15))
```

- A diferença fundamental entre um **data-frame** e uma matriz é que num **data-frame** os dados não precisam de ser todos do mesmo tipo.

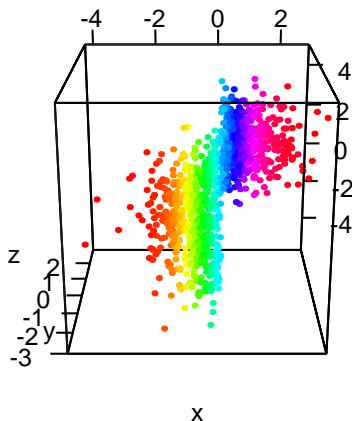
Gráficos em 3D

Package(rgl) - Plot 3D

- `open3d()`
- `x <- -sort(rnorm(1000))`
- `y <- -rnorm(1000)`
- `z <- -rnorm(1000) + atan2(x, y)`
- `plot3d(x, y, z, col = rainbow(1000))`
- `rgl.postscript("plot3d.pdf", "pdf")`

Gráficos em 3D

Package(rgl) - Plot 3D



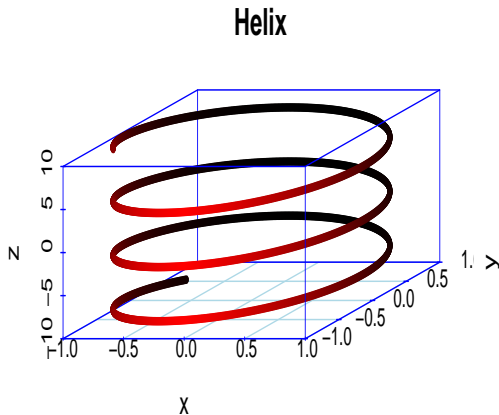
Gráficos em 3D

Exemplo:

- $z < -seq(-10, 10, 0.01)$
- $x < -cos(z)$
- $y < -sin(z)$
- `scatterplot3d(x, y, z, highlight.3d = TRUE, col.axis = "blue", col.grid = "lightblue", main = "Helix", pch = 20)`

Gráficos em 3D

Exemplo:



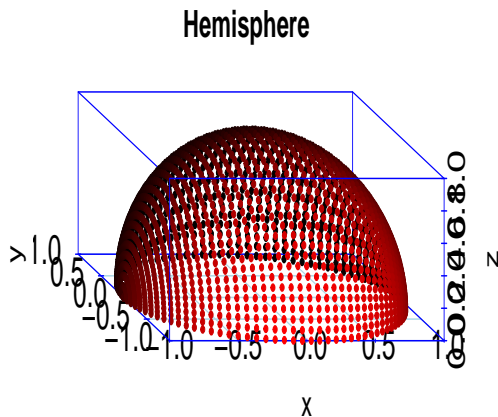
Gráficos em 3D

Exemplo:

- `temp <- seq(-pi, 0, length = 50)`
- `x <- -c(rep(1, 50)% * %t(cos(temp)))`
- `y <- -c(cos(temp)% * %t(sin(temp)))`
- `z <- -c(sin(temp)% * %t(sin(temp)))`
- `scatterplot3d(x, y, z, highlight.3d = TRUE, angle = 120,
col.axis = "blue", col.grid = "lightblue", cex.axis = 1.3,
cex.lab = 1.1, main = "Hemisphere", pch = 20)`

Gráficos em 3D

Exemplo:



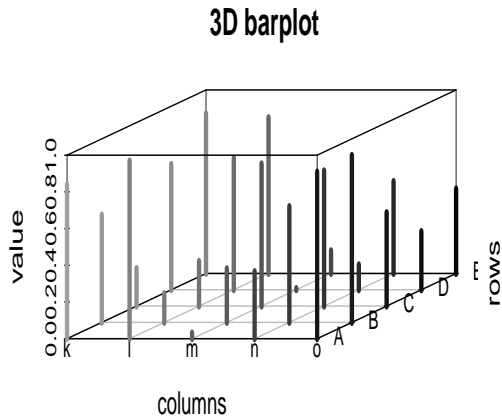
Gráficos em 3D

Exemplo:

- `my.mat <- matrix(runif(25), nrow = 5)`
- `dimnames(my.mat) <- list(LETTERS[1 : 5], letters[11 : 15])`
- `s3d.dat <- data.frame(columns = c(col(my.mat)),
rows = c(row(my.mat)), value = c(my.mat))`
- `scatterplot3d(s3d.dat, type = "h", lwd = 5, pch = , x.ticklabs =
colnames(my.mat),
y.ticklabs = rownames(my.mat),
color = grey(25 : 1/40), main = "3Dbarplot")`

Gráficos em 3D

Exemplo:



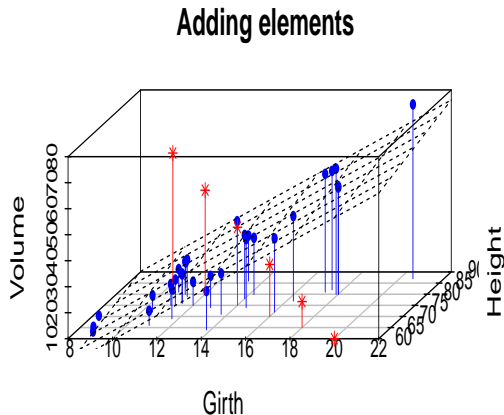
Gráficos em 3D

Exemplo:

- `data(trees)`
- `s3d <- scatterplot3d(trees, type = "h", color = "blue",
angle = 55, scale.y = 0.7, pch = 16, main = "Adding elements")`
- `my.lm <- lm(trees$Volume ~ trees$Girth + trees$Height)`
- `s3d$plane3d(my.lm)`
- `s3d$points3d(seq(10, 20, 2), seq(85, 60, -5), seq(60, 10, -10),
col = "red", type = "h", pch = 8)`

Gráficos em 3D

Exemplo:



Escrevendo Funções no R

- o R é uma linguagem que permite criar novas funções.
- Na verdade, muitas das funções em R são atualmente funções de funções.
- A estrutura de uma função é dada abaixo:

Estrutura de uma Função:

```
> myfunction <- function(arg1, arg2,...){  
> statements  
> return(object)  
> }
```

Escrevendo Funções no R

- o R é uma linguagem que permite criar novas funções.
- Na verdade, muitas das funções em R são atualmente funções de funções.
- A estrutura de uma função é dada abaixo:

Estrutura de uma Função:

```
> myfunction <- function(arg1, arg2,...){  
> statements  
> return(object)  
> }
```

Escrevendo Funções no R

- o R é uma linguagem que permite criar novas funções.
- Na verdade, muitas das funções em R são atualmente funções de funções.
- A estrutura de uma função é dada abaixo:

Estrutura de uma Função:

```
> myfunction <- function(arg1, arg2,...){  
> statements  
> return(object)  
> }
```

Escrevendo Funções no R

Exemplo: Soma de uma Progressão Aritmética.

```
> soma <- function(a1, d, n){  
>   an <- -a1 + (n - 1) * d  
>   ((a1 + an) * n) / 2  
> }
```

Operadores de Comparação:

Operadores de Comparação:

- equal: `==`
- not equal: `!=`
- greater/less than: `>` / `<`
- greater/less than or equal: `>=` / `<=`

Operadores Lógicos:

Operadores Lógicos:

- and: `&`. Exemplo: `x & y`. Lê-se x e y.
- or: `|`. Exemplo: `x | y`. Lê-se x ou y.
- not: `!`. Exemplo: `!x`. Lê-se não x.

Escrevendo Funções no R

Usando o If (se):

Syntax:

```
if (cond1=true) { cmd1 } else { cmd2 }
```

- ```
if(1 == 0){
 print(1)
} else {
 print(2)
}
[1] 2
```

# Escrevendo Funções no R

## Usando o Ifelse (Caso Contrário):

### Syntax:

`ifelse(test, true_value, false_value)`

- `x < -1 : 10` *#Creates sample data*  
`ifelse(x < 5 | x > 8, x, 0)`  
`[1] 1 2 3 4 0 0 0 0 9 10`



# Escrevendo Funções no R

## for:

- As estruturas de Loop mais frequentemente utilizadas no R são: for, while, repeat, break e apply.

- Exemplo:**

```
x <- -1 : 10
```

```
z <- NULL
```

```
for(i in seq(along = x)){
```

```
 if(x[i] < 5) {
```

```
 z <- c(z, x[i] - 1)
```

```
 } else {
```

```
 z <- c(z, x[i]/x[i])
```

```
 }
```

```
}
```

```
z
```

```
[1] 0 1 2 3 1 1 1 1 1 1
```

# Escrevendo Funções no R

## while:

- **Syntax:**  
while(condition) statements

- **Exemplo:**

```
z <- -0 while(z < 5){
```

```
z <- -z + 2
```

```
print(z)
```

```
}
```

```
[1]2
```

```
[1]4
```

```
[1]6
```

# Escrevendo Funções no R

## repeat:

- **Syntax:**

repeat statements

- **Exemplo:**

```
z < -0
```

```
repeat{
```

```
z < -z + 1
```

```
print(z)
```

```
if(z > 100) break()
```

```
}
```

# Manipulando Base de Dados no R

## A Função Merge

- A Função **Merge** (Mesclar, Fundir) adiciona variáveis à Base de Dados.
- Mesclar dois conjuntos de dados requer que ambos tenham pelo menos uma variável em comum (ou string ou numérica). Se string assegure-se que as categorias estão com mesma escrita. Por exemplo: Nomes de Municípios, Estados e Países.
- **ATENÇÃO:** explore cada base de dados separadamente antes de realizar a função **Merge**.

# Manipulando Base de Dados no R

## A Função **Append**

- A Função **Append** (Juntar, Acrescentar) adiciona casos/observações à Base de Dados.
- Para isso, utiliza-se a função **rbind**.
- Acrescentar observações a um conjunto de dados requer que ambos possuam as mesmas variáveis com exatamente os mesmos nomes.
- Se utilizar dados categóricos assegure-se que as categorias sobre ambos conjuntos de dados se referem exatamente a mesma coisa. Por exemplo: (1 "Agree", 2 "Disagree", 3 "DK" em ambos).

# Manipulando Base de Dados no R

## A Função Merge

mydata1

|    | country | year | y     | y_bin | x1    | x2    | x3    |
|----|---------|------|-------|-------|-------|-------|-------|
| 1  | A       | 2000 | 1343  | 1     | 0.28  | -1.11 | 0.28  |
| 2  | A       | 2001 | -1900 | 0     | 0.32  | -0.95 | 0.49  |
| 3  | A       | 2002 | -11   | 0     | 0.36  | -0.79 | 0.7   |
| 4  | A       | 2003 | 2646  | 1     | 0.25  | -0.89 | -0.09 |
| 5  | B       | 2000 | -5935 | 0     | -0.08 | 1.43  | 0.02  |
| 6  | B       | 2001 | -712  | 0     | 0.11  | 1.65  | 0.26  |
| 7  | B       | 2002 | -1933 | 0     | 0.35  | 1.59  | -0.23 |
| 8  | B       | 2003 | 3073  | 1     | 0.73  | 1.69  | 0.26  |
| 9  | C       | 2000 | -1292 | 0     | 1.31  | -1.29 | 0.2   |
| 10 | C       | 2001 | -3416 | 0     | 1.18  | -1.34 | 0.28  |
| 11 | C       | 2002 | -356  | 0     | 1.26  | -1.26 | 0.37  |
| 12 | C       | 2003 | 1225  | 1     | 1.42  | -1.31 | -0.38 |



mydata2

|    | country | year | x4 | x5 | x6 |
|----|---------|------|----|----|----|
| 1  | A       | 2000 | 10 | 1  | 9  |
| 2  | A       | 2001 | 7  | 1  | 9  |
| 3  | A       | 2002 | 7  | 9  | 4  |
| 4  | A       | 2003 | 1  | 2  | 3  |
| 5  | B       | 2000 | 0  | 5  | 6  |
| 6  | B       | 2001 | 5  | 8  | 5  |
| 7  | B       | 2002 | 9  | 4  | 5  |
| 8  | B       | 2003 | 1  | 5  | 1  |
| 9  | C       | 2000 | 4  | 5  | 4  |
| 10 | C       | 2001 | 6  | 9  | 6  |
| 11 | C       | 2002 | 6  | 5  | 3  |
| 12 | C       | 2003 | 7  | 3  | 3  |

```
mydata <- merge(mydata1, mydata2, by=c("country", "year"))
```

```
edit(mydata)
```

|   | country | year | y     | y_bin | x1    | x2    | x3    | x4 | x5 | x6 |
|---|---------|------|-------|-------|-------|-------|-------|----|----|----|
| 1 | A       | 2000 | 1343  | 1     | 0.28  | -1.11 | 0.28  | 10 | 1  | 9  |
| 2 | A       | 2001 | -1900 | 0     | 0.32  | -0.95 | 0.49  | 7  | 1  | 9  |
| 3 | A       | 2002 | -11   | 0     | 0.36  | -0.79 | 0.7   | 7  | 9  | 4  |
| 4 | A       | 2003 | 2646  | 1     | 0.25  | -0.89 | -0.09 | 1  | 2  | 3  |
| 5 | B       | 2000 | -5935 | 0     | -0.08 | 1.43  | 0.02  | 0  | 5  | 6  |
| 6 | B       | 2001 | -712  | 0     | 0.11  | 1.65  | 0.26  | 5  | 8  | 5  |
| 7 | B       | 2002 | -1933 | 0     | 0.35  | 1.59  | -0.23 | 9  | 4  | 5  |
| 8 | B       | 2003 | 3073  | 1     | 0.73  | 1.69  | 0.26  | 1  | 5  | 1  |
| 9 | C       | 2000 | -1292 | 0     | 1.31  | -1.29 | 0.2   | 4  | 5  | 4  |

# Manipulando Base de Dados no R

## A Função Merge

- 1 Importe a Base de Dados **rental1** e **rental2**.
- 2 `rental1 <- read.csv( "~/R/rental1.csv", sep=";", dec=",")`  
`View(rental1)`
- 3 `rental2 <- read.csv( "~/R/rental2.csv", sep=";",")`  
`View(rental2)`
- 4 Juntando as Bases de Dados pelo Comando **Merge**:  
`rental <- merge(rental1, rental2, by=c("city","year"))`  
`View(rental)`

# Manipulando Base de Dados no R

## A Função Merge

- 1 Importe a Base de Dados **rental1** e **rental2**.
- 2 **rental1** < -read.csv( "~/R/rental1.csv", sep=";", dec=",")  
**View(rental1)**
- 3 rental2 < -read.csv( "~/R/rental2.csv", sep=";")  
View(rental2)
- 4 Juntando as Bases de Dados pelo Comando **Merge**:  
rental < -merge(rental1, rental2, by=c("city","year"))  
View(rental)



# Manipulando Base de Dados no R

## A Função Merge

- 1 Importe a Base de Dados **rental1** e **rental2**.
- 2 **rental1** < -read.csv( "~/R/rental1.csv", sep=";", dec=",")  
**View(rental1)**
- 3 **rental2** < -read.csv( "~/R/rental2.csv", sep=";",")  
**View(rental2)**
- 4 Juntando as Bases de Dados pelo Comando **Merge**:  
**rental** < -merge(rental1, rental2, by=c("city","year"))  
**View(rental)**

# Manipulando Base de Dados no R

## A Função Merge

- 1 Importe a Base de Dados **rental1** e **rental2**.
- 2 `rental1 <- read.csv( "~/R/rental1.csv", sep=";", dec=",")`  
`View(rental1)`
- 3 `rental2 <- read.csv( "~/R/rental2.csv", sep=";",")`  
`View(rental2)`
- 4 Juntando as Bases de Dados pelo Comando **Merge**:  
`rental <- merge(rental1, rental2, by=c("city","year"))`  
`View(rental)`

# Manipulando Base de Dados no R

## A Função Append

mydata7

| R Data Editor  |         |      |       |       |       |       |      |
|----------------|---------|------|-------|-------|-------|-------|------|
| File Edit Help |         |      |       |       |       |       |      |
|                | country | year | y     | y_bin | x1    | x2    | x3   |
| 1              | A       | 2000 | 1343  | 1     | 0.28  | -1.11 | 0.28 |
| 2              | A       | 2001 | -1900 | 0     | 0.32  | -0.95 | 0.49 |
| 3              | B       | 2000 | -5935 | 0     | -0.08 | 1.43  | 0.02 |
| 4              | B       | 2001 | -712  | 0     | 0.11  | 1.65  | 0.26 |
| 5              | C       | 2000 | -1292 | 0     | 1.31  | -1.29 | 0.2  |
| 6              | C       | 2001 | -3416 | 0     | 1.18  | -1.34 | 0.28 |



mydata8

| R Data Editor  |         |      |       |       |      |       |       |
|----------------|---------|------|-------|-------|------|-------|-------|
| File Edit Help |         |      |       |       |      |       |       |
|                | country | year | y     | y_bin | x1   | x2    | x3    |
| 1              | A       | 2002 | -11   | 0     | 0.36 | -0.79 | 0.7   |
| 2              | A       | 2003 | 2646  | 1     | 0.25 | -0.89 | -0.09 |
| 3              | B       | 2002 | -1933 | 0     | 0.35 | 1.59  | -0.23 |
| 4              | B       | 2003 | 3073  | 1     | 0.73 | 1.69  | 0.26  |
| 5              | C       | 2002 | -356  | 0     | 1.26 | -1.26 | 0.37  |
| 6              | C       | 2003 | 1225  | 1     | 1.42 | -1.31 | -0.38 |

```
mydata <- rbind(mydata7, mydata8)
```

```
edit(mydata)
```

| R Data Editor  |         |      |       |       |       |       |       |
|----------------|---------|------|-------|-------|-------|-------|-------|
| File Edit Help |         |      |       |       |       |       |       |
|                | country | year | y     | y_bin | x1    | x2    | x3    |
| 1              | A       | 2000 | 1343  | 1     | 0.28  | -1.11 | 0.28  |
| 2              | A       | 2001 | -1900 | 0     | 0.32  | -0.95 | 0.49  |
| 3              | B       | 2000 | -5935 | 0     | -0.08 | 1.43  | 0.02  |
| 4              | B       | 2001 | -712  | 0     | 0.11  | 1.65  | 0.26  |
| 5              | C       | 2000 | -1292 | 0     | 1.31  | -1.29 | 0.2   |
| 6              | C       | 2001 | -3416 | 0     | 1.18  | -1.34 | 0.28  |
| 7              | A       | 2002 | -11   | 0     | 0.36  | -0.79 | 0.7   |
| 8              | A       | 2003 | 2646  | 1     | 0.25  | -0.89 | -0.09 |
| 9              | B       | 2002 | -1933 | 0     | 0.35  | 1.59  | -0.23 |
| 10             | B       | 2003 | 3073  | 1     | 0.73  | 1.69  | 0.26  |
| 11             | C       | 2002 | -356  | 0     | 1.26  | -1.26 | 0.37  |
| 12             | C       | 2003 | 1225  | 1     | 1.42  | -1.31 | -0.38 |

# Manipulando Base de Dados no R

## A Função Append

- 1 Importe a Base de Dados **rental3** e **rental4**.
- 2 `rental3 <- read.csv( "~/R/rental3.csv", sep="";  
View(rental3)`
- 3 `rental4 <- read.csv( "~/R/rental4.csv", sep=";")  
View(rental4)`
- 4 Acrescentando observações à Base de Dados pelo Comando **Append**:  
`rentall <- rbind(rental3,rental4)  
View(rentall)`

# Manipulando Base de Dados no R

## A Função Append

- 1 Importe a Base de Dados **rental3** e **rental4**.
- 2 **rental3** < -read.csv( "~/R/rental3.csv", sep="";  
**View(rental3)**
- 3 rental4 < -read.csv( "~/R/rental4.csv", sep=";")  
**View(rental4)**
- 4 Acrescentando observações à Base de Dados pelo Comando **Append**:  
rentall < -rbind(rental3,rental4)  
**View(rentall)**

# Manipulando Base de Dados no R

## A Função Append

- 1 Importe a Base de Dados **rental3** e **rental4**.
- 2 **rental3** < -read.csv( "~/R/rental3.csv", sep="";  
**View(rental3)**
- 3 **rental4** < -read.csv( "~/R/rental4.csv", sep="";")  
**View(rental4)**
- 4 Acrescentando observações à Base de Dados pelo Comando **Append**:  
**rentall** < -rbind(rental3,rental4)  
**View(rentall)**

# Manipulando Base de Dados no R

## A Função Append

- 1 Importe a Base de Dados **rental3** e **rental4**.
- 2 **rental3** < -read.csv( "~/R/rental3.csv", sep="";  
View(rental3)
- 3 **rental4** < -read.csv( "~/R/rental4.csv", sep=";";  
View(rental4)
- 4 Acrescentando observações à Base de Dados pelo Comando **Append**:  
**rentall** < -rbind(rental3,rental4)  
View(rentall)

# Manipulando Base de Dados no R

- Ordenando os Dados em Ordem Crescente:

```
attach(mydata)
```

```
mydata_sorted <- mydata[order(country, year),]
```

```
detach(mydata)
```

```
edit(mydata_sorted)
```



# Manipulando Base de Dados no R

## Ordenando os Dados



mydata\_sorted

| R Data Editor  |           |         |      |       |       |       |       |       |
|----------------|-----------|---------|------|-------|-------|-------|-------|-------|
| File Edit Help |           |         |      |       |       |       |       |       |
|                | row.names | country | year | y     | y_bin | x1    | x2    | x3    |
| 1              | 1         | A       | 2000 | 1343  | 1     | 0.28  | -1.11 | 0.28  |
| 2              | 2         | A       | 2001 | -1900 | 0     | 0.32  | -0.95 | 0.49  |
| 3              | 7         | A       | 2002 | -11   | 0     | 0.36  | -0.79 | 0.7   |
| 4              | 8         | A       | 2003 | 2646  | 1     | 0.25  | -0.89 | -0.09 |
| 5              | 3         | B       | 2000 | -5935 | 0     | -0.08 | 1.43  | 0.02  |
| 6              | 4         | B       | 2001 | -712  | 0     | 0.11  | 1.65  | 0.26  |
| 7              | 9         | B       | 2002 | -1933 | 0     | 0.35  | 1.59  | -0.23 |
| 8              | 10        | B       | 2003 | 3073  | 1     | 0.73  | 1.69  | 0.26  |
| 9              | 5         | C       | 2000 | -1292 | 0     | 1.31  | -1.29 | 0.2   |
| 10             | 6         | C       | 2001 | -3416 | 0     | 1.18  | -1.34 | 0.28  |
| 11             | 11        | C       | 2002 | -356  | 0     | 1.26  | -1.26 | 0.37  |
| 12             | 12        | C       | 2003 | 1225  | 1     | 1.42  | -1.31 | -0.38 |

# Manipulando Base de Dados no R

## Saída de Dados em Formato de Texto

- **`install.packages("stargazer")`**  
**`library(stargazer)`**  
**`summary(rental$pop)`**
- Saída de Dados com Variáveis na Linha:

```
[rgb]0,0,1stargazer(rental, type = "text", title="Descriptive sta
```

- Saída de Dados com Variáveis na Coluna:

```
[rgb]0,0,1stargazer(rental, type = "text", title="Descriptive sta
```

# Manipulando Base de Dados no R

## Saída de Dados em Formato de Texto

- **`install.packages("stargazer")`**  
**`library(stargazer)`**  
**`summary(rental$pop)`**
- Saída de Dados com Variáveis na Linha:

```
[rgb]0,0,1stargazer(rental, type = "text", title="Descriptive sta
```

- Saída de Dados com Variáveis na Coluna:

```
[rgb]0,0,1stargazer(rental, type = "text", title="Descriptive sta
```

# Manipulando Base de Dados no R

## Saída de Dados em Formato de Texto

- **`install.packages("stargazer")`**  
**`library(stargazer)`**  
**`summary(rental$pop)`**
- Saída de Dados com Variáveis na Linha:

```
[rgb]0,0,1stargazer(rental, type = "text", title="Descriptive sta
```

- Saída de Dados com Variáveis na Coluna:

```
[rgb]0,0,1stargazer(rental, type = "text", title="Descriptive sta
```

# Manipulando Base de Dados no R

## Variáveis na Linha

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code for installing the `stargazer` package and running it to generate descriptive statistics for the `rental` dataset. The code includes comments in Portuguese.
- Environment:** Shows the loaded objects: `rental` (128 obs. of 23 variables), `rental1` (128 obs. of 14 variables), and `rental2` (128 obs. of 11 variables).
- Files:** Shows the installed packages and their versions.
- Console:** Displays the output of the `stargazer` function, showing the package was built under R version 3.1.3 and the resulting descriptive statistics table.

**Descriptive statistics**

| Statistic | N   | Mean     | St. Dev. | Min    | Max     |
|-----------|-----|----------|----------|--------|---------|
| city      | 128 | 32.5     | 18.5     | 1      | 64      |
| year      | 128 | 85.0     | 5.0      | 80     | 90      |
| pop       | 128 | 84,645.4 | 80,322.1 | 25,728 | 632,910 |
| enroll    | 128 | 18,459.4 | 9,914.4  | 5,645  | 74,047  |
| rent      | 128 | 331.4    | 119.6    | 186    | 925     |
| rnthsg    | 128 | 16,698.6 | 16,735.6 | 4,062  | 137,242 |
| tothsg    | 128 | 35,666.8 | 56,644.9 | 7,130  | 560,011 |
| avginc    | 128 | 18,912.0 | 6,966.0  | 9,262  | 56,307  |
| tenroll   | 128 | 9.7      | 0.5      | 8.6    | 11.2    |
| ipop      | 128 | 11.1     | 0.6      | 10.2   | 13.4    |
| lrent     | 128 | 5.7      | 0.3      | 5.2    | 6.8     |
| ltothsg   | 128 | 10.1     | 0.7      | 8.9    | 13.2    |
| lrnthsg   | 128 | 9.5      | 0.7      | 8.3    | 11.8    |
| lavginc   | 128 | 9.8      | 0.3      | 9.1    | 10.9    |
| y90       | 128 | 0.5      | 0.5      | 0      | 1       |

# Manipulando Base de Dados no R

## Variáveis na Coluna

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code for generating descriptive statistics using the `stargazer` function. The code is as follows:
 

```
61 -----
62
63 #Saída de Dados com Variáveis na Coluna#
64
65 stargazer(rental, type = "text", title="Descriptive statistics", digits=1, out="tab
66
67 descriptive statistics
68
69 statistic city year pop enroll rent rnthsg tothsg avginc lenroll lp
70
71
72 [Top Level] :
```
- Console:** Shows the execution of the `stargazer` function and the resulting output:
 

```
> stargazer(rental, type = "text", title="Descriptive statistics", digits=1, out="table2
.txt", flp=TRUE)

Descriptive statistics
=====
Statistic city year pop enroll rent rnthsg tothsg avginc lenroll lp len
t ltothsg lnrthsg lavginc y90

N 128 128 128 128 128 128 128 128 128 128
Mean 32.5 85.0 84,645.4 18,459.4 331.4 16,698.6 35,666.8 18,912.0 9.7 11.1 5.7
10.1 9.5 9.8 0.5
St. Dev. 18.5 5.0 80,322.1 9,914.4 119.6 16,735.6 56,644.9 6,966.0 0.5 0.6 0.3
0.7 0.7 0.3 0.5
Min 1 80 25,728 5,645 186 4,062 7,130 9,262 8.6 10.2 5.2
8.9 8.3 9.1 0
Max 64 90 632,910 74,047 925 137,242 560,011 56,307 11.2 13.4 6.8
13.2 11.8 10.9 1

```
- Environment:** Lists the objects in the global environment:
  - `rental`: 128 obs. of 23 variables
  - `rental1`: 128 obs. of 14 variables
  - `rental2`: 128 obs. of 11 variables
- Files:** Lists installed and available packages, including:
  - `abind`: Combine Multidimensional Arrays (1.4-3)
  - `acepack`: `ace()` and `avas()` for selecting regression transformations (1.3-3)
  - `ADGofTest`: Anderson-Darling Gof Test (0.3)
  - `AER`: Applied Econometrics with R (1.2-3)
  - `agricolae`: Statistical Procedures for Agricultural Research (1.2-1)
  - `AlgDesign`: Algorithmic Experimental Design (1.1-7.3)
  - `assertthat`: Easy pre and post assertions. (0.1)
  - `BCA`: Business and Customer Analytics (0.9-3)
  - `bdsmatrix`: Routines for Block Diagonal Symmetric matrices (1.3-2)
  - `BH`: Boost C++ Header Files (1.55.0-3)
  - `Biobase`: Biobase: Base functions for Bioconductor (2.26.0)
  - `BiocGenerics`: S4 generic functions for Bioconductor (0.121)
  - `BiocInstaller`: Install/Update Bioconductor and CRAN Packages (1.16.4)
  - `bitops`: Bitwise Operations (1.0-6)

# Manipulando Base de Dados no R

## Explorando o Conjunto de Dados:

- Baixando Base de Dados:

```
d <- read.csv("http://www.ats.ucla.edu/stat/data/hsb2.csv")
```

- Usando **dim**, verifica-se o número de observações (linha) e variáveis (coluna) em **d**.

```
dim(d)
```

- Usando **str**, verifica-se a estrutura de **d**, incluindo as classes (tipos) de todas as variáveis.

```
str(d)
```

- **summary** é uma função genérica para resumir (sumarizar) muitos tipos de objetos no **R**, incluindo conjunto de dados.

```
summary(d)
```

# Manipulando Base de Dados no R

## Explorando o Conjunto de Dados:

- Baixando Base de Dados:

```
d <- read.csv("http://www.ats.ucla.edu/stat/data/hsb2.csv")
```

- Usando **dim**, verifica-se o número de observações (linha) e variáveis (coluna) em **d**.

```
dim(d)
```

- Usando **str**, verifica-se a estrutura de **d**, incluindo as classes (tipos) de todas as variáveis.

```
str(d)
```

- summary** é uma função genérica para resumir (sumarizar) muitos tipos de objetos no **R**, incluindo conjunto de dados.

```
summary(d)
```



# Manipulando Base de Dados no R

## Explorando o Conjunto de Dados:

- Baixando Base de Dados:

```
d <- read.csv("http://www.ats.ucla.edu/stat/data/hsb2.csv")
```

- Usando **dim**, verifica-se o número de observações (linha) e variáveis (coluna) em **d**.

```
dim(d)
```

- Usando **str**, verifica-se a estrutura de **d**, incluindo as classes (tipos) de todas as variáveis.

```
str(d)
```

- summary** é uma função genérica para resumir (sumarizar) muitos tipos de objetos no **R**, incluindo conjunto de dados.

```
summary(d)
```

# Manipulando Base de Dados no R

## Explorando o Conjunto de Dados:

- Baixando Base de Dados:  
`d <- read.csv("http://www.ats.ucla.edu/stat/data/hsb2.csv")`
- Usando **dim**, verifica-se o número de observações (linha) e variáveis (coluna) em **d**.  
`dim(d)`
- Usando **str**, verifica-se a estrutura de **d**, incluindo as classes (tipos) de todas as variáveis.  
`str(d)`
- **summary** é uma função genérica para resumir (sumarizar) muitos tipos de objetos no **R**, incluindo conjunto de dados.  
`summary(d)`

# Manipulando Base de Dados no R

## Explorando o Conjunto de Dados:

- Se você quer resumos condicionais, por exemplo, apenas os alunos com alta pontuação em leitura (**read**  $\geq$  **60**) o **R** limita o conjunto de dados e computa os resultados.
- Por Exemplo:  
`summary(subset(d, read  $\geq$  60))`
- Também podemos separar os dados de outras formas, tais como por grupos. Vejamos a média das 5 variáveis para cada tipo de programa, **prog**.  
`by(d[, 7:11], d$prog, colMeans)`

# Manipulando Base de Dados no R

## Explorando o Conjunto de Dados:

- Se você quer resumos condicionais, por exemplo, apenas os alunos com alta pontuação em leitura (**read**  $\geq$  **60**) o **R** limita o conjunto de dados e computa os resultados.
- **Por Exemplo:**  
**`summary(subset(d, read  $\geq$  60))`**
- Também podemos separar os dados de outras formas, tais como por grupos. Vejamos a média das 5 variáveis para cada tipo de programa, **prog**.  
**`by(d[, 7:11], d$prog, colMeans)`**

# Manipulando Base de Dados no R

## Explorando o Conjunto de Dados:

- Se você quer resumos condicionais, por exemplo, apenas os alunos com alta pontuação em leitura (**read**  $\geq$  **60**) o **R** limita o conjunto de dados e computa os resultados.
- **Por Exemplo:**  
`summary(subset(d, read  $\geq$  60))`
- Também podemos separar os dados de outras formas, tais como por grupos. Vejamos a média das 5 variáveis para cada tipo de programa, **prog**.  
`by(d[, 7:11], d$prog, colMeans)`

# Manipulando Base de Dados no R

## Explorando o Conjunto de Dados:

- Podemos verificar a distribuição das variáveis categóricas com tabelas de frequências:

- Por sexo:

```
[rgb]0,0,1xtabs(~ [rgb]0,0,1female, data = d)
```

- Por raça:

```
[rgb]0,0,1xtabs(~ [rgb]0,0,1race, data = d)
```

- Pelo Programa:

```
[rgb]0,0,1xtabs(~ [rgb]0,0,1prog, data = d)
```

# Manipulando Base de Dados no R

## Explorando o Conjunto de Dados:

- Podemos verificar a distribuição das variáveis categóricas com tabelas de frequências:

- Por sexo:

```
[rgb]0,0,1xtabs(~ [rgb]0,0,1female, data = d)
```

- Por raça:

```
[rgb]0,0,1xtabs(~ [rgb]0,0,1race, data = d)
```

- Pelo Programa:

```
[rgb]0,0,1xtabs(~ [rgb]0,0,1prog, data = d)
```

# Manipulando Base de Dados no R

## Explorando o Conjunto de Dados:

- Podemos verificar a distribuição das variáveis categóricas com tabelas de frequências:

- Por sexo:

```
[rgb]0,0,1xtabs(~ [rgb]0,0,1female, data = d)
```

- Por raça:

```
[rgb]0,0,1xtabs(~ [rgb]0,0,1race, data = d)
```

- Pelo Programa:

```
[rgb]0,0,1xtabs(~ [rgb]0,0,1prog, data = d)
```



# Manipulando Base de Dados no R

## Explorando o Conjunto de Dados:

- Podemos verificar a distribuição das variáveis categóricas com tabelas de frequências:

- Por sexo:

```
[rgb]0,0,1xtabs(~ [rgb]0,0,1female, data = d)
```

- Por raça:

```
[rgb]0,0,1xtabs(~ [rgb]0,0,1race, data = d)
```

- Pelo Programa:

```
[rgb]0,0,1xtabs(~ [rgb]0,0,1prog, data = d)
```

# Manipulando Base de Dados no R

## Explorando o Conjunto de Dados:

- Tabela Cruzada com Duas Variáveis:

```
[rgb]0,0,1xtabs(rgb]0,0,1~ [rgb]0,0,1ses + schtyp, data = d)
```

- Tabela Cruzada com Três Variáveis: `[rgb]0,0,1(tab3 i- xtabs([rgb]0,0,1ses + prog + schtyp, data = d))`
- Como um último passo para nossa exploração de dados, gostaríamos de observar alguns aspectos rápidos das relações bivariadas (aos pares) em nosso conjunto de dados. Podemos usar a função `cor`.  
`[rgb]0,0,1cor(d[, 7:11])`  
`[rgb]0,0,1ggpairs(d[, 7:11])` `#(usando a matriz scatter plot)`

# Manipulando Base de Dados no R

## Explorando o Conjunto de Dados:

- Tabela Cruzada com Duas Variáveis:

```
[rgb]0,0,1xtabs(rgb]0,0,1~ [rgb]0,0,1ses + schtyp, data = d)
```

- Tabela Cruzada com Três Variáveis: **[rgb]0,0,1(tab3 i- xtabs( [rgb]0,0,1ses + prog + schtyp, data = d))**

- Como um último passo para nossa exploração de dados, gostaríamos de observar alguns aspectos rápidos das relações bivariadas (aos pares) em nosso conjunto de dados. Podemos usar a função **cor**.

```
[rgb]0,0,1cor(d[, 7:11])
```

```
[rgb]0,0,1ggpairs(d[, 7:11]) #(usando a matriz scatter plot)
```

# Manipulando Base de Dados no R

## Explorando o Conjunto de Dados:

- Tabela Cruzada com Duas Variáveis:

```
[rgb]0,0,1xtabs(rgb]0,0,1~ [rgb]0,0,1ses + schtyp, data = d)
```

- Tabela Cruzada com Três Variáveis: **[rgb]0,0,1(tab3 i- xtabs([rgb]0,0,1ses + prog + schtyp, data = d))**

- Como um último passo para nossa exploração de dados, gostaríamos de observar alguns aspectos rápidos das relações bivariadas (aos pares) em nosso conjunto de dados. Podemos usar a função **cor**.

```
[rgb]0,0,1cor(d[, 7:11])
```

```
[rgb]0,0,1ggpairs(d[, 7:11]) #(usando a matriz scatter plot)
```

# Cálculo Diferencial e Integral no R

- Para resolver uma **Derivada**<sup>a</sup>, utiliza-se a função **D**. Por Exemplo:  
 $D(\text{expression}(\text{sqrt}(1 - x^2)), 'x')$  Primeira Derivada.  
 $D(D(\text{expression}(\text{sqrt}(1 - x^2)), 'x'), 'x')$  Segunda Derivada.
- Para resolver uma **Integral Unidimensional** utilize a função **integrate**. Por exemplo: Encontre a Integral da função:

$$\int_0^{\infty} \frac{1}{(x+1)\sqrt{x}} dx.$$

- Defina a função integrando.  
`integrand <- function(x) 1/((x+1)*sqrt(x))`
- Integre a função de 0 ao  $\infty$ .  
`integrate(integrand, lower = 0, upper = Inf)`

<sup>a</sup>O R não simplifica os resultados da função **D()**.

# Cálculo Diferencial e Integral no R

- Para resolver uma **Derivada**<sup>a</sup>, utiliza-se a função **D**. Por Exemplo:  
 $D(\text{expression}(\text{sqrt}(1 - x^2)), 'x')$  Primeira Derivada.  
 $D(D(\text{expression}(\text{sqrt}(1 - x^2)), 'x'), 'x')$  Segunda Derivada.
- Para resolver uma **Integral Unidimensional** utilize a função **integrate**. Por exemplo: Encontre a Integral da função:

$$\int_0^{\infty} \frac{1}{(x+1)\sqrt{x}} dx.$$

- Defina a função integrando.  
`integrand <- function(x) 1/((x+1)*sqrt(x))`
- Integre a função de 0 ao  $\infty$ .  
`integrate(integrand, lower = 0, upper = Inf)`

<sup>a</sup>O R não simplifica os resultados da função **D()**.

# Cálculo Diferencial e Integral no R

- Para resolver uma **Derivada**<sup>a</sup>, utiliza-se a função **D**. Por Exemplo:  
 $D(\text{expression}(\text{sqrt}(1 - x^2)), 'x')$  Primeira Derivada.  
 $D(D(\text{expression}(\text{sqrt}(1 - x^2)), 'x'), 'x')$  Segunda Derivada.
- Para resolver uma **Integral Unidimensional** utilize a função **integrate**. Por exemplo: Encontre a Integral da função:

$$\int_0^{\infty} \frac{1}{(x+1)\sqrt{x}} dx.$$

- Defina a função integrando.  
`integrand <- -function(x) 1/((x+1)*sqrt(x))`
- Integre a função de 0 ao  $\infty$ .  
`integrate(integrand, lower = 0, upper = Inf)`

<sup>a</sup>O R não simplifica os resultados da função **D()**.

# Cálculo Diferencial e Integral no R

- Para resolver uma **Derivada**<sup>a</sup>, utiliza-se a função **D**. Por Exemplo:  
 $D(\text{expression}(\text{sqrt}(1 - x^2)), 'x')$  Primeira Derivada.  
 $D(D(\text{expression}(\text{sqrt}(1 - x^2)), 'x'), 'x')$  Segunda Derivada.
- Para resolver uma **Integral Unidimensional** utilize a função **integrate**. Por exemplo: Encontre a Integral da função:

$$\int_0^{\infty} \frac{1}{(x+1)\sqrt{x}} dx.$$

- Defina a função integrando.  
`integrand <- function(x) 1/((x+1)*sqrt(x))`
- Integre a função de 0 ao  $\infty$ .  
`integrate(integrand, lower = 0, upper = Inf)`

<sup>a</sup>O R não simplifica os resultados da função **D()**.



# Cálculo Diferencial e Integral no R

- Outro Exemplo:

$$\int_{-1.96}^{1.96} \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} dx. \quad (1)$$

- Descrever a função:

```
f <- function(x) {1/sqrt(2*pi)*exp(-x^2/2)}
```

- Integre a função de  $-1.96$  a  $1.96$ .

```
integrate(f, lower = -1.96, upper = 1.96)
```

# Cálculo Diferencial e Integral no R

- Outro Exemplo:

$$\int_{-1.96}^{1.96} \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} dx. \quad (1)$$

- Descrever a função:

```
f <- function(x) {1/sqrt(2*pi)*exp(-x^2/2)}
```

- Integre a função de  $-1.96$  a  $1.96$ .

```
integrate(f, lower = -1.96, upper = 1.96)
```