

# Programação em R

Copyright: Carlos Cinelli

Julho, 2016

# Introdução e Motivação

# O que é o R?

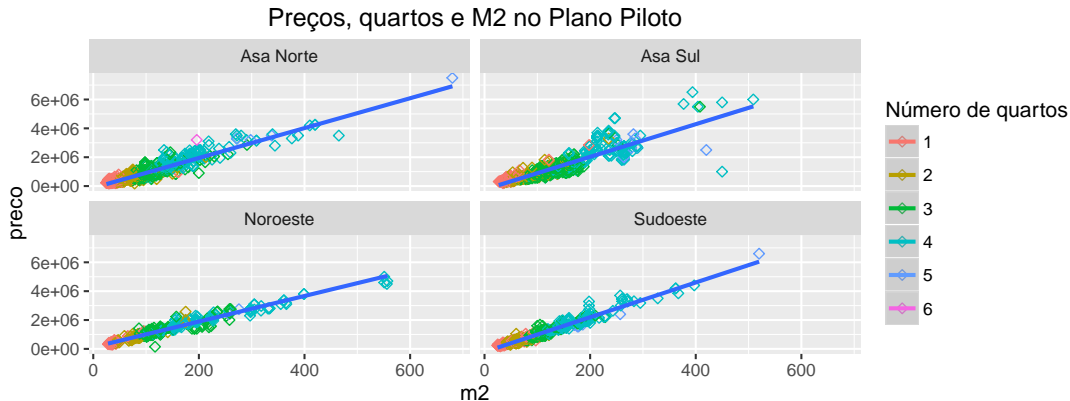
- O R é uma linguagem de programação com foco em análise de dados;
- Criado na Nova Zelândia por dois estatísticos: Ross Ihaka e Robert Gentleman;
  - Baseado na linguagem S, desenvolvida por John Chambers (e colegas) na Bell Laboratories;
- É uma linguagem de programação interpretada, voltada à interação dinâmica com os dados e modelos.

# Por que o R?

- O R é gratuito e de código aberto;
- Compatível com todas as plataformas (Windows, Mac, Linux);
- É mais do que um software estatístico:
  - Ambiente que permite explorar dados interativamente; mas, à medida que a análise evolui, é uma linguagem de programação completa para desenvolver e automatizar soluções, desenvolver software (pacotes);
  - Ferramenta poderosa para manipulação, processamento, visualização e análise de dados, bem como simulações e modelagem estatísticas.
- Comunidade grande que contribui ativamente, com pessoas tanto do mercado (Google, AT&T, empresas de investimento e finanças) quanto da academia (professores de diversas universidades):
  - Mais de 8000 pacotes gratuitos e abertos para download.

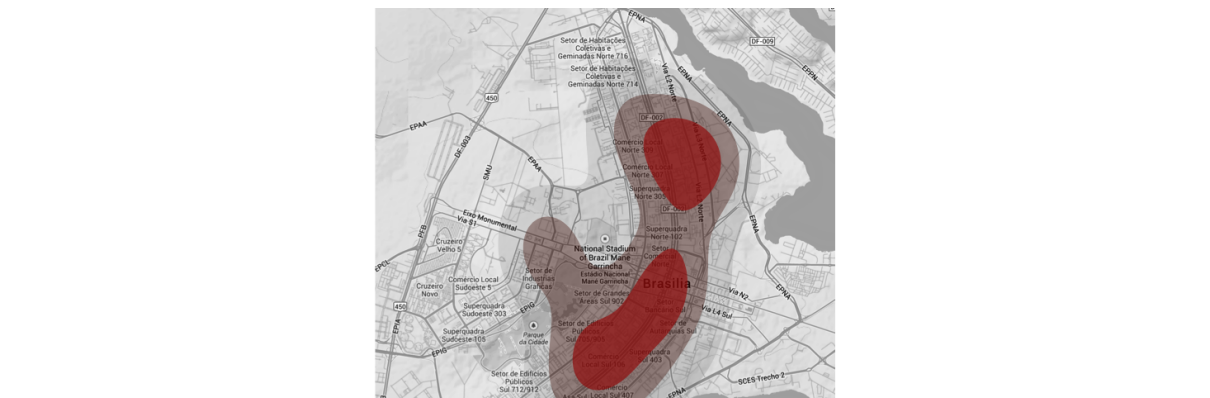
## Alguns exemplos

**Gráficos e estatística:** gráfico de dispersão com preço contra metro quadrado por bairro, cor dos pontos de acordo com número de quartos e linha de regressão.



## Alguns exemplos

**Mapas e estatística:** Webscraping de dados de roubos e download do mapa de Brasília no Google Maps; manipulação dos dados e construção de gráficos de calor geolocalizados.



## Alguns exemplos

**Apresentações e documentos de suas análises:** Integração com *markdown*,  $\text{\LaTeX}$ , HTML + JavaScript e Word, o que permite a elaboração de *papers*, relatórios, apresentações de maneira rápida e conjugada à análise.

Esta e as demais apresentações deste curso, bem como as listas de exercícios, foram todas feitas no R.

## Sobre os slides

Os slides deste curso apresentam códigos em R com o resultado esperado. Você deve **acompanhar ativamente**, digitando (ou, em último caso, copiando e colando) os códigos. Programação não se aprende somente olhando para a apresentação! Também serão realizados exercícios para consolidação do conteúdo.

```
x <- 10; y <- 20;  
x == y
```

CÓDIGO EM R.

```
# [1] FALSE
```

RESULTADO ESPERADO DO CÓDIGO ACIMA

```
x <- c(10, 20, 30); y <- c(10, 10, 30)  
x == y
```

```
# [1] TRUE FALSE TRUE
```



## Sobre os exercícios

- Durante a apresentação de cada tópico faremos exercícios entre os slides. Os exercícios serão feitos com tempo controlado e todas soluções serão apresentadas. Não se preocupe se não conseguir fazer todos durante o tempo controlado.
- Também teremos listas de exercícios. Como a turma é heterogênea, os exercícios da lista variam em nível de dificuldade e você não precisa conseguir fazer todos - a idéia é justamente ter exercícios tanto para consolidar conhecimentos quanto para desafiar um pouco quem for pegando o conteúdo mais rapidamente.
- Vocês receberão a resolução de todos os exercícios da lista também, entretanto, para resolução “ao vivo” em sala de aula, priorizaremos aqueles mais importantes, a depender do andamento das aulas.

# R e Rstudio

# Instalação

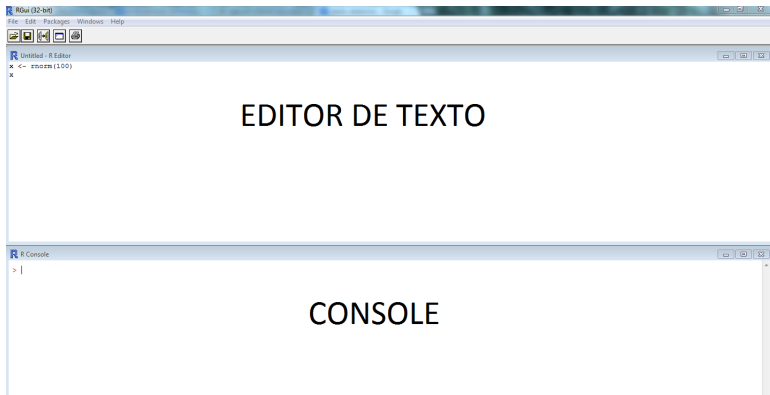
O R e RStudio já vêm com distribuições compiladas para Windows, Mac e Linux. A instalação é bastante fácil e em geral você apenas tem que seguir as instruções da tela. Para ter um ambiente completo de desenvolvimento no R, certifique-se de ter em sua máquina:

- a versão mais recente do R;
- a versão mais recente do RStudio (IDE que vamos usar);
- MikTeX (Win) ou MacTeX (Mac) para relatórios em  $\text{\LaTeX}$ ;
- RTools (Win) ou Xcode com command line tools (Mac), para criar pacotes, usar C++ etc.

Os slides a seguir tomarão como base, em geral, as teclas de atalho do Windows.

# R Gui

Ao instalar o R, automaticamente também é instalada uma interface gráfica chamada R Gui. Abra o R no seu computador. A primeira tela que você verá é a do console. Abra também uma tela de editor de texto em “file” -> “new script”.



# R Gui

Escreva os seguintes comandos no editor de texto e aperte “ctrl+R”.

```
1+1
```

O comando será enviado para o console. Agora faça o seguinte gráfico:

```
plot(1:10)
```

Uma nova tela será aberta com o gráfico. Saia do R Gui escrevendo q() no console ou apertando “alt+f4”. Ele irá perguntar se você deseja salvar o trabalho e o script. Clique em “não” para os dois casos. Estamos saindo do R Gui pois iremos trabalhar no RStudio.

# RStudio

Apesar de o R vir com uma interface gráfica bem interessante, existe um IDE (Integrated Development Environment - Ambiente de Desenvolvimento Integrado) chamado RStudio, com várias funcionalidades e gratuito. No decorrer de nossas aulas iremos utilizar somente o RStudio.

O RStudio tem algumas vantagens em relação ao R Gui:

- Highlight do código;
- Autocomplete;
- Match automático de parenteses e chaves;
- Interface intuitiva para objetos, gráficos, script;
- Projetos (com controle de versão);
- Interação com HTML, entre outras.

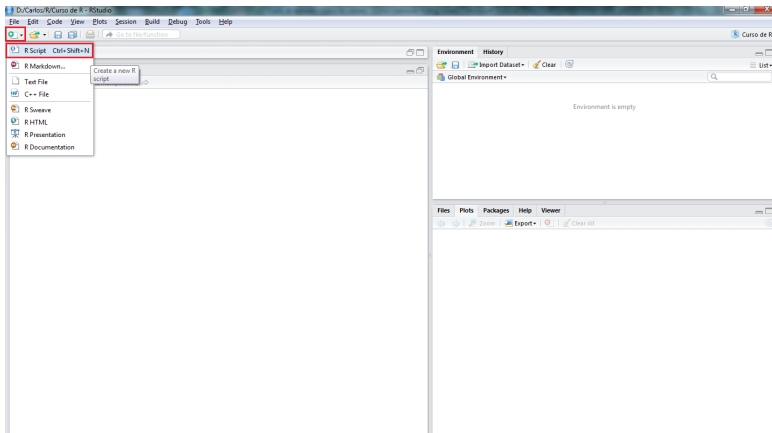
## Projetos no RStudio

Vamos criar um projeto do RStudio para o nosso curso de R. Esta é uma maneira simples e fácil de gerenciar os scripts, dados e demais documentos relacionados a um projeto de R em que você esteja trabalhando. Paremos agora para:

- Criar projeto de RStudio em uma pasta nova;
- Criar uma pasta de “Dados” para guardar algumas bases de dados utilizados em aula;
- Criar uma pasta de slides para guardar os slides (inclusive este) das aulas.

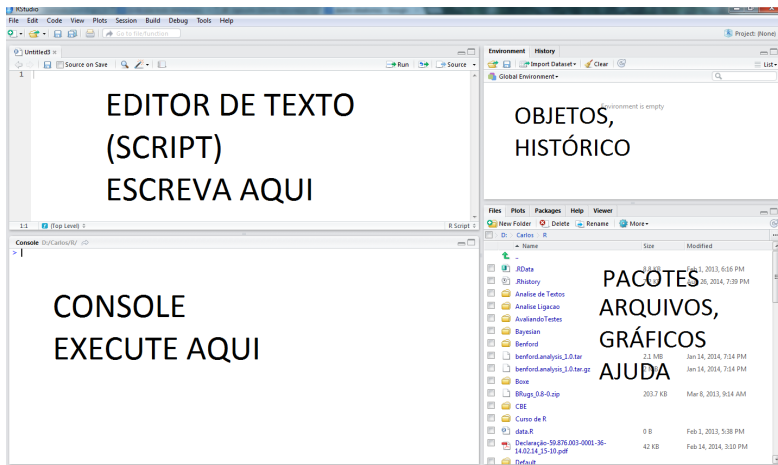
# RStudio

Inicie um novo Script em “File” -> “New File” -> “New RScript”. Você também pode fazer isso com **“CTRL + SHIFT + N”** ou acessando o botão abaixo.





# RStudio



# RStudio

- **Script:** A tela superior esquerda do RStudio é o editor de texto onde você vai escrever seus Scripts. Ele possui code highlighting entre outras funcionalidades.
- **Console:** No canto inferior esquerdo fica o console. O console nada mais é do que uma seção aberta de R, em que os comando são executados.
- **Área de trabalho e histórico:** Ficam no canto superior direito. Os objetos criados na seção e o histórico dos comandos podem ser acessados ali.
- **Arquivos, Gráficos, Pacotes, Ajuda:** Ficam no canto inferior direito. Você pode explorar pastas e arquivos diretamente do RStudio na aba “Files”; os gráficos que forem feitos apareceram na aba “Plots”. Os pacotes instalados em sua máquina estão listados em “Packages”. As ajudas das funções aparecem em “Help”. E o “Viewer” serve para visualização de imagens/páginas em HTML e JavaScript.

## Comandos pelo Script

Acostume-se a escrever o código no Script ao invés de ficar escrevendo diretamente no console.

Escreva o código abaixo no Script:

```
1+1
```

E aperte “ctrl” + “enter”. Isso envia o comando para o console e o resultado é exibido logo abaixo.

```
1+1  
## [1] 2
```

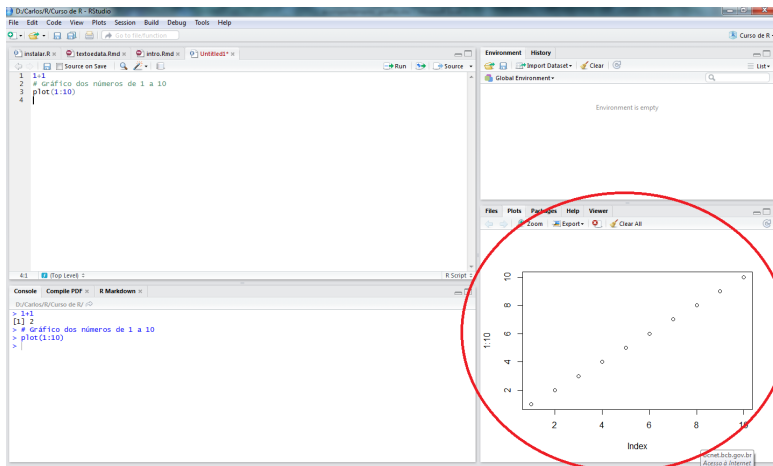
## Comandos pelo Script

Agora escreva o seguinte código no Script.

```
# Gráfico dos números de 1 a 10  
plot(1:10)
```

O primeiro comando “*# Gráfico dos números de 1 a 10*” é, na verdade, um comentário. Comentários em Scripts do R são seguidos do símbolo #, e tudo que estiver após # não será executado. É uma boa prática comentar seu código, para que ele seja de fácil manutenção, tanto para você mesmo quanto para outros colegas. O segundo comando é de um gráfico. Aperte “ctrl” + “enter” nas duas linhas. O gráfico aparecerá no canto inferior direito do RStudio.

# Comandos pelo Script

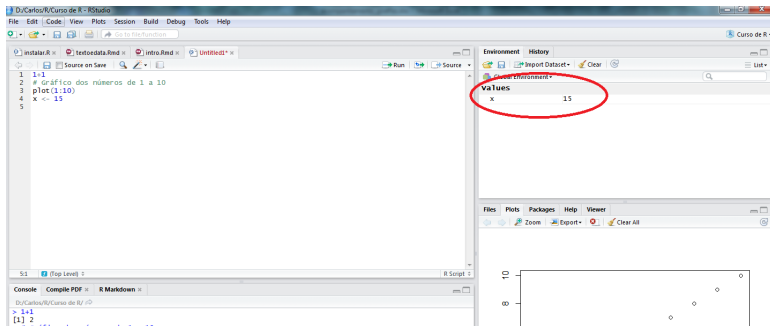


## Comandos pelo Script

Agora digite o seguinte comando no editor e aperte “ctrl” + “enter”:

```
x <- 15
```

Atribuímos o valor 15 à variável x. Note que isso aparece no canto superior direito do RStudio.



# Comandos pelo Script

O RStudio tem autocomplete. Escreve apenas plo e aperte **Tab**.

The screenshot shows the RStudio interface. In the Script editor, the following code is entered:

```
1 1:1  
2 # Gráfico dos números de 1 a 10  
3 plot(1:10)  
4 x <- 15  
5 plo
```

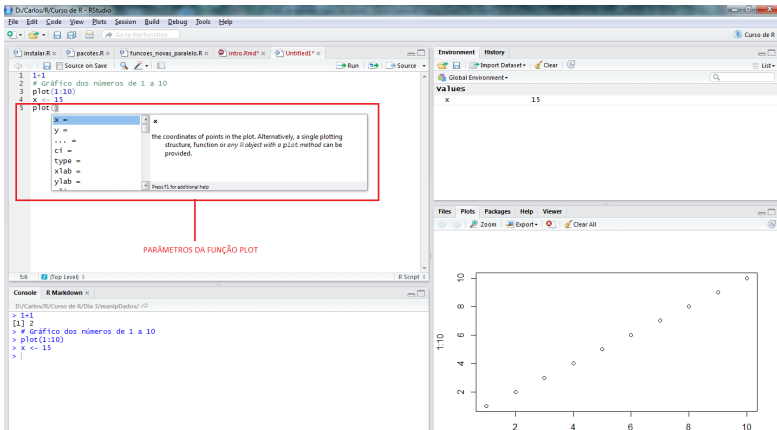
A red box highlights the autocomplete menu that appears after typing 'plo'. The menu lists several functions, with 'plot' selected. A red arrow points from the text 'Tab -> AUTOCOMPLETE' to the 'plot' option in the menu.

The Environment pane on the right shows the 'Global Environment' with a variable 'x' having the value 15.

The Plots pane at the bottom right shows a scatter plot of the data generated by the code. The x-axis is labeled 'Index' and ranges from 1 to 10. The y-axis is labeled '1:10' and ranges from 1 to 10. The plot shows a series of points forming a straight line.

## Comandos pelo Script

Isso também funciona dentro da função, para vermos os parâmetros disponíveis. Escreva `plot()` e aperte **Tab**.



The screenshot shows the RStudio interface. In the Script editor, the following code is entered:

```
1 i=1
2 # Gráfico dos números de 1 a 10
3 plot(1:10)
4 x <- 15
5 plot()
```

A red box highlights the `plot()` function call on line 5. A tooltip menu is displayed, listing the parameters of the `plot` function: `x`, `y`, `...`, `c1`, `type`, `xlab`, and `ylab`. A red arrow points from the text "PARÂMETROS DA FUNÇÃO PLOT" to this menu.

The Environment pane on the right shows the Global Environment with a variable `x` having the value 15.

The Console pane at the bottom shows the execution of the code:

```
> i=1
[1] 2
> # Gráfico dos números de 1 a 10
> plot(1:10)
> x <- 15
>
```

The Viewer pane at the bottom right displays a scatter plot of the numbers 1 to 10. The x-axis is labeled "1:10" and the y-axis is labeled "10". The plot shows a series of points forming a curve.



## Comandos pelo Script

Outras teclas de atalho:

- **CTRL + 1**: Passa o cursor para o Script;
- **CTRL + 2**: Passa o cursor para o console;
- **SETA PARA CIMA (no console)**: acessa o histórico de comandos anteriores.

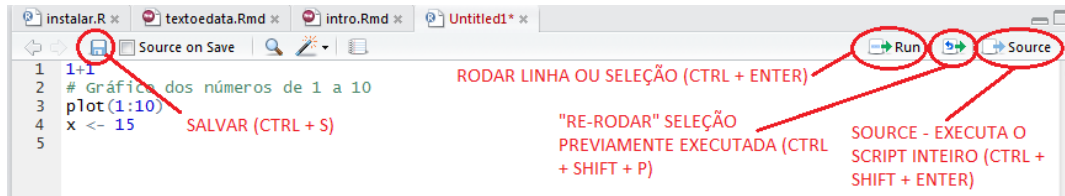
Abra um novo Script:

- **CTRL + ALT + SETA PARA ESQUERDA OU DIREITA**: Navega entre as abas de script abertas.

## Comandos pelo Script

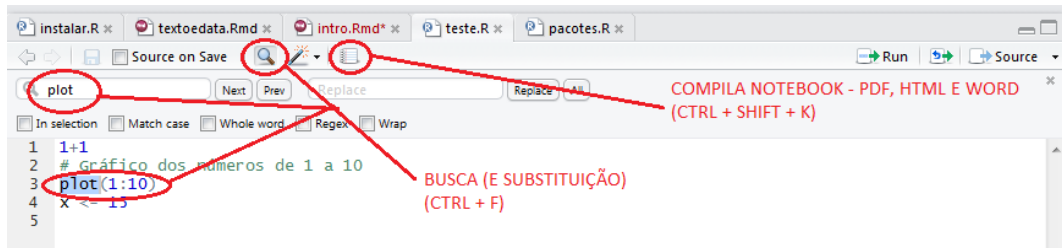
Outras teclas de atalho:

- **CTRL + SHIFT + P**: “Previous command”, roda o último comando executado;
- **CTRL + SHIFT + ENTER**: “Source”. Executa o Script inteiro;
- **CTRL + S**: Salva o Script;



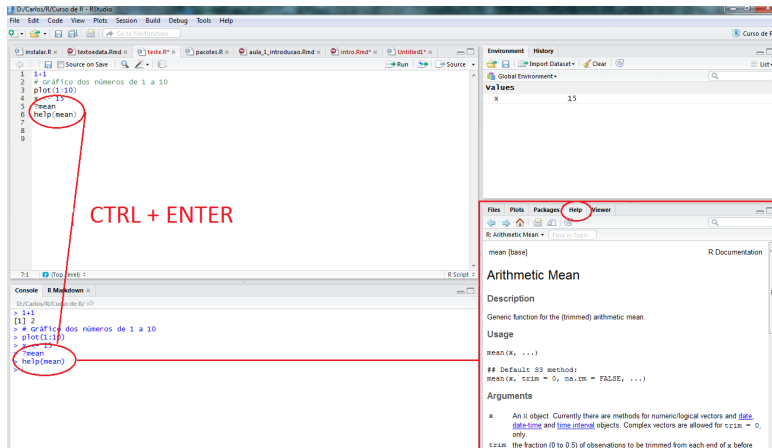
## Comandos pelo Script

- **CTRL + L**: Limpa o console;
- **CTRL + F**: Busca (e substituição). Aceita REGEX;
- **CTRL + SHIFT + K**: Compila “Notebook” em PDF, HTML ou Word;
- **ALT + SHIFT + K**: Veja os outros atalhos.
- Veja mais dicas no Cheat Sheet do RStudio (<http://www.ibpad.com.br/wp-content/uploads/2016/04/rstudio-IDE-cheatsheet.pdf>).



# Ajuda - Função Específica

?mean; help(mean)



# Ajuda - Função Específica

mean (base)

R Documentation

## Arithmetic Mean

### Description

Generic function for the (trimmed) arithmetic mean.

### Usage

```
mean(x, ...)
```

```
## Default S3 method:
```

```
mean(x, trim = 0, na.rm = FALSE, ...)
```

### Arguments

**x** An R object. Currently there are methods for numeric/logical vectors and [date](#), [date-time](#) and [time interval](#) objects. Complex vectors are allowed for `trim = 0`, only.

**trim** the fraction (0 to 0.5) of observations to be trimmed from each end of `x` before the mean is computed. Values of `trim` outside that range are taken as the nearest endpoint.

**na.rm** a logical value indicating whether `NA` values should be stripped before the computation proceeds.

**...** further arguments passed to or from other methods.

### Value

If `trim` is zero (the default), the arithmetic mean of the values in `x` is computed, as a numeric or complex vector of length one. If `x` is not logical (coerced to numeric), numeric (including integer) or complex, `NA_real_` is returned, with a warning.

If `trim` is non-zero, a symmetrically trimmed mean is computed with a fraction of `trim` observations deleted from each end before the mean is computed.

### References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

### See Also

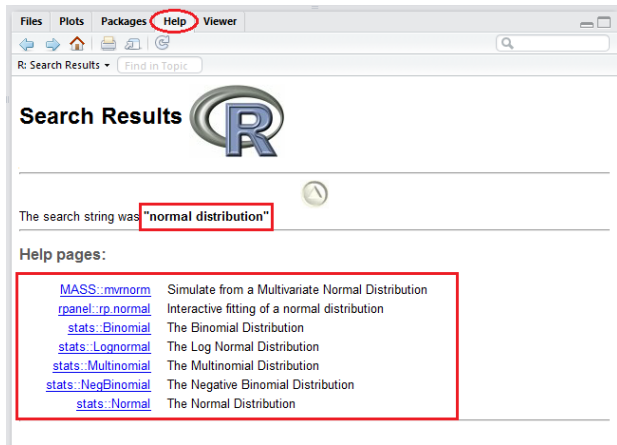
[weighted.mean](#), [mean.POSIXct](#), [colMeans](#) for row and column means.

### Examples

```
x <- c(0:10, 50)
xm <- mean(x)
c(xm, mean(x, trim = 0.10))
```

## Ajuda - Busca por String

```
??"normal distribution"; help.search("normal distribution")
```



# Ajuda - Internet

- StackOverflow (em Inglês e Português)

The screenshot shows the StackOverflow website interface. At the top, there's a navigation bar with the StackOverflow logo and tabs for Questions, Tags, Users, Badges, and Unanswered. A sidebar on the right highlights '66,638 questions tagged r'. The main content area, titled 'Tagged Questions', lists several questions. The first question is 'How to plot the output of a multivariate regression using GGPlot', which has -2 votes and 0 answers. The second question is 'Where are the values of this variable coming from? Doesn't seemed to be defined anywhere', with 0 votes and 0 answers. The third question is 'Want individual components as subsets in a stacked bar chart', also with 0 votes and 0 answers. Each question entry includes a brief description, tags, and the user who asked it.

## Ajuda - Internet

- **Grupos de e-mail:** [www.r-project.org/mail.html](http://www.r-project.org/mail.html)
  - **R Help:** lista de e-mails principal, para dúvidas gerais.
  - **R Announce:** Lista de anúncios sobre desenvolvimento do R.
  - **R Packages:** Lista de anúncios sobre pacotes do R.
  - **R Devel:** Lista de discussão de desenvolvedores no R.
- **Blogs**
  - **R Bloggers:** consolidador de blogs sobre R (em inglês).
  - **Em Português:**
    - **Análise Real** (<http://analisereal.com>) com livro em elaboração (<http://analisereal.com/introducao-a-analise-de-dados-com-r/>) ;
    - **Sociais e Métodos** (<https://sociaisemetodos.wordpress.com>);
    - **Dados Aleatórios** (<http://www.dadosaleatorios.com.br/>) entre outros.



# Pacotes

A principal forma de distribuição de códigos no R é por meio de pacotes. Um pacote pode ser entendido como um conjunto de códigos auto-contido que adiciona funcionalidades ao R.

Para carregar um pacote, use a função `library()`.

Ao carregar um pacote, você está adicionando suas funções ao `search` da seção, permitindo que você chame estas funções diretamente. Por exemplo, a função `mvrnorm`, que gera números aleatórios de uma normal multivariada, está no pacote `MASS`.

```
Sigma <- matrix(c(10,3,3,2), nrow = 2, ncol = 2) # Matriz de Var-Covar
mu <- c(1, 10) # Médias
x <- mvrnorm(n = 100, mu, Sigma) # Tenta gerar 100 obs, mas dá erro
## Error in eval(expr, envir, enclos): não foi possível encontrar a função "mvrnorm"

library(MASS) # Carrega pacote
x <- mvrnorm(n = 100, mu, Sigma) # Agora funciona
```

# Pacotes

Para ver o que está no search do R, utilize a função `search()`. Note que o pacote MASS agora esta lá.

```
search()
## [1] ".GlobalEnv"          "package:MASS"          "package:ggplot2"
## [4] "package:stats"        "package:graphics"      "package:grDevices"
## [7] "package:utils"        "package:datasets"      "package:methods"
## [10] "Autoloads"           "package:base"
```

Para descarregar um pacote, utilize a função `detach()`.

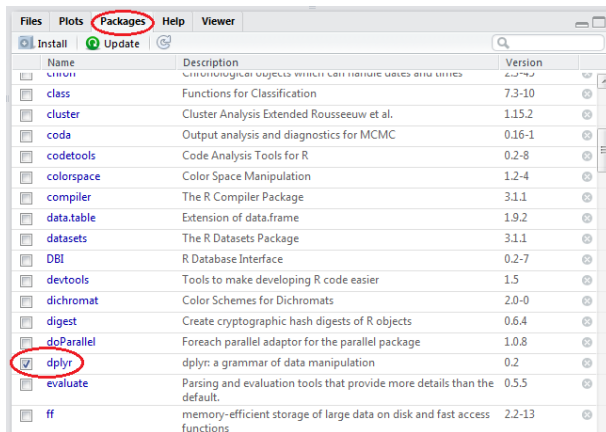
```
detach(package:MASS)
```

Às vezes pacotes tem o mesmo nome de funções. Neste caso, se ambos forem carregados, a função que prevalece é a do pacote que foi carregado por último. Uma outra forma de resolver isso é usar o nome do pacote e o operador `::` antes de chamar a função. Neste caso não há ambiguidade.

```
x <- MASS::mvrnorm(n = 100, mu, Sigma)
```

# Pacotes

Você também pode carregar ou descarregar pacotes pelo menu do canto inferior direito do RStudio, clicando na caixa ao lado do nome do pacote.



# Pacotes

Para instalar um pacote, use a função `install.packages()`. Por exemplo, o pacote `dplyr` é muito utilizado para manipulação de dados. Para instalá-lo e desinstalá-lo, temos os seguintes comandos.

```
install.packages("dplyr")  
  
##### Não rode #####  
remove.packages(dplyr)
```

## Pacotes, CRAN e Sites

Grande parte dos pacotes do R estão centralizados em um repositório chamado CRAN (The Comprehensive R Archive Network), com diversos espelhos ao redor do mundo. Agora, vamos explorar um pouco o site oficial do R, **[www.r-project.org](http://www.r-project.org)**, e do CRAN, **[cran.r-project.org](http://cran.r-project.org)**.

- Manuais, livros, documentação;
- Listas de e-mails;
- The R Journal;
- Lista de pacotes;
- Task Views.

# Exercícios

Sua vez.

- Além do dplyr, vamos usar outros pacotes como ggplot2, ggthemes, tidyr, reshape2 e stringr. Instale esses pacotes.
- Faça seu cadastro no StackOverflow e StackOverflow em português.
- Veja os temas do Task View do R. Há algum tema que te interessa?

# Soluções

Você pode instalar todos os pacotes de uma só vez com `install.packages()`:

```
install.packages(c("ggplot2", "ggthemes",  
                  "tidyr", "reshape2", "stringr"))
```

## Um breve exemplo



## Imóveis do Plano Piloto

Para ilustrar uma análise no R, veremos um exemplo simples com ofertas online de apartamentos à venda no Plano Piloto, do site *Wimoveis*. Os dados foram obtidos por meio de webscraping utilizando o R e já passaram por um processo de “limpeza” prévio. Não se preocupe em entender os comandos agora, iremos trabalhar isso no decorrer do curso.

## Imóveis do Plano Piloto

Baixe os dados em:

<https://dl.dropboxusercontent.com/u/44201187/dados/wi.venda.rds>

Salve na pasta “Dados” do seu projeto.

```
# Carregando os pacotes que vamos usar
```

```
library(dplyr)
```

```
library(ggplot2)
```

```
# Carregando os dados
```

```
dados <- readRDS("Dados/wi.venda.rds")
```

## Imóveis do Plano Piloto

```
# Vendo a estrutura dos dados
```

```
str(dados, vec.len = 1)
```

```
## 'data.frame':    3786 obs. of  9 variables:
```

```
## $ bairro      : chr  "Noroeste" ...
```

```
## $ location    : chr  "SQNW 310 " ...
```

```
## $ preco       : num  150000 175000 ...
```

```
## $ quartos     : num  3 1 ...
```

```
## $ m2          : num  117 30 ...
```

```
## $ corretora   : chr  "12220" ...
```

```
## $ data        : Date, format: "2014-01-23" ...
```

```
## $ link        : chr  "/imovel/venda-apartamento-brasilia-df-3-quartos-sqn"
```

```
## $ pm2         : num  1282 ...
```

# Imóveis do Plano Piloto

```
# Calculando medianas por bairro
mediana_bairro <- dados %>%
  group_by(bairro) %>%
  summarise(Median_Precio = median(preco),
            Median_Quarto = median(quartos),
            Median_pm2 = median(pm2),
            N = n())

mediana_bairro
## Source: local data frame [4 x 5]
##
##      bairro Median_Precio Median_Quarto Median_pm2      N
##      (chr)      (dbl)      (dbl)      (dbl) (int)
## 1 Asa Norte      830000          3        8929  1331
## 2 Asa Sul        1030000          3        8800  1036
## 3 Noroeste       1070000          3        9934   635
## 4 Sudoeste       899500          3        9706   784
```

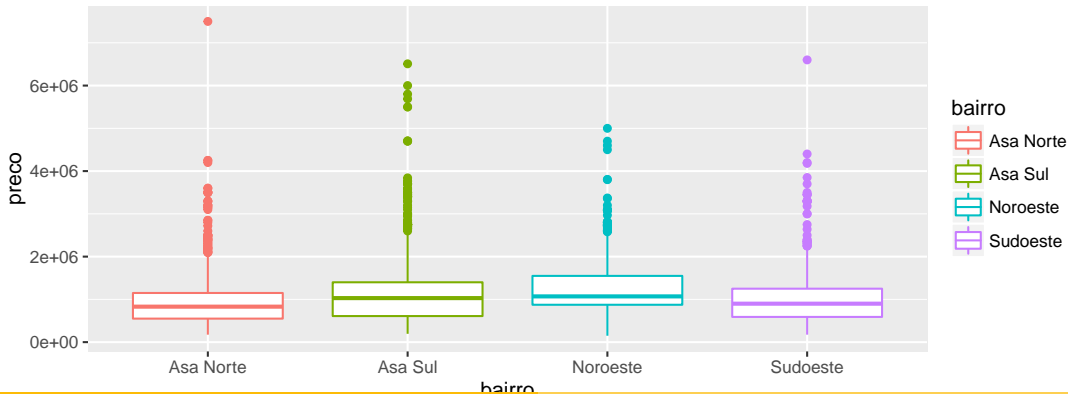
# Imóveis do Plano Piloto

```
# Rodando um modelo linear
modelo <- lm(preco ~ m2 * bairro + quartos , data = dados)

# resultados do modelo
# não cabe tudo na tela do slide
summary(modelo)
##
## Call:
## lm(formula = preco ~ m2 * bairro + quartos, data = dados)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3889078 -105771  -10453   90518  2263876
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -97799     17814   -5.49  4.3e-08 ***
```

# Imóveis do Plano Piloto

```
# Box-Plot dos preços por bairro  
ggplot(dados, aes(bairro, preco, color = bairro)) + geom_boxplot()
```



# Imóveis do Plano Piloto

```
# Gráfico de dispersão com regressão  
ggplot(dados, aes(m2, preco)) +  
  geom_point(shape = 1, aes(color = factor(quartos))) +  
  geom_smooth(method = "lm") +  
  facet_wrap(~bairro)
```

