

# Aula 1 - Introdução ao R

Vitor Rios

10 de outubro de 2017

# Bem vindos

## Cronograma

Dia 1 - 21/11 - Introdução e como usar o R e Rstudio, boas práticas de programação

Dia 2 - 22/11 - Criando e manipulando dados

Dia 3 - 23/11 - Trabalhando com arquivos externos

Dia 4 - 28/12 - Noções de programação

Dia 5 - 05/12 - Gráficos

Dia 6 - 06/12 - Análise exploratória e funções matemáticas

Dia 7 - 07/12 - Análise exploratória e funções matemáticas

Dia 8 - 12/12 - Análise estatística básica

Dia 9 - 13/12 - Versionamento de código: noções de git

Dia 10 - 14/12 - Programação para iniciantes, Função final

Aulas: teoria (1 a 2 h) + prática (wiki do curso) Avaliação: exercícios e construção de uma função (entrega 2 semanas após fim da disciplina)

Enviar exercícios para [cursolinguagemr@gmail.com](mailto:cursolinguagemr@gmail.com)

# O que é o R?

- ▶ R é uma linguagem de programação voltada para análises estatísticas e manipulação de dados
- ▶ RStudio é um ambiente de desenvolvimento integrado (IDE) que estende as capacidades do R e facilita o uso

## Jeito R de ser

- ▶ Praticidade
- ▶ Reprodutibilidade
- ▶ Simplicidade (sim, você leu direito)
- ▶ Cooperação

Ao fazer o tratamento e análise dos dados dentro do R, você não altera os dados originais, e pode reproduzir tudo, bastando ter o script e os dados originais. Não é necessário nem salvar os gráficos!

# Porque Usar R?

- Uma vastidão de pacotes de análises prontas

## CRAN Task Views

<a href="#">Bayesian</a>	Bayesian Inference	<a href="#">Multivariate</a>	Multivariate Statistics
<a href="#">ChemPhys</a>	Chemometrics and Computational Physics	<a href="#">NaturalLanguageProcessing</a>	Natural Language Processing
<a href="#">ClinicalTrials</a>	Clinical Trial Design, Monitoring, and Analysis	<a href="#">NumericalMathematics</a>	Numerical Mathematics
<a href="#">Cluster</a>	Cluster Analysis & Finite Mixture Models	<a href="#">OfficialStatistics</a>	Official Statistics & Survey Methodology
<a href="#">DifferentialEquations</a>	Differential Equations	<a href="#">Optimization</a>	Optimization and Mathematical Programming
<a href="#">Distributions</a>	Probability Distributions	<a href="#">Pharmacokinetics</a>	Analysis of Pharmacokinetic Data
<a href="#">Econometrics</a>	Econometrics	<a href="#">Phylogenetics</a>	Phylogenetics, Especially Comparative Methods
<a href="#">Environmetrics</a>	Analysis of Ecological and Environmental Data	<a href="#">Psychometrics</a>	Psychometric Models and Methods
<a href="#">ExperimentalDesign</a>	Design of Experiments (DoE) & Analysis of Experimental Data	<a href="#">ReproducibleResearch</a>	Reproducible Research
<a href="#">ExtremeValue</a>	Extreme Value Analysis	<a href="#">Robust</a>	Robust Statistical Methods
<a href="#">Finance</a>	Empirical Finance	<a href="#">SocialSciences</a>	Statistics for the Social Sciences
<a href="#">FunctionalData</a>	Functional Data Analysis	<a href="#">Spatial</a>	Analysis of Spatial Data
<a href="#">Genetics</a>	Statistical Genetics	<a href="#">SpatioTemporal</a>	Handling and Analyzing Spatio-Temporal Data
<a href="#">Graphics</a>	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization	<a href="#">Survival</a>	Survival Analysis
<a href="#">HighPerformanceComputing</a>	High-Performance and Parallel Computing with R	<a href="#">TimeSeries</a>	Time Series Analysis
<a href="#">MachineLearning</a>	Machine Learning & Statistical Learning	<a href="#">WebTechnologies</a>	Web Technologies and Services
<a href="#">MedicalImaging</a>	Medical Image Analysis	<a href="#">gR</a>	gRaphical Models in R
<a href="#">MetaAnalysis</a>	Meta-Analysis		

Figure 1: CranTaskView

- ▶ Além disso, o R tem **Interfaces amigáveis que não precisam de programação**
  - ▶ RCommander <- fácil acesso às análises principais
  - ▶ Action <- análises integradas ao Excel
  - ▶ RStudio <- ambiente completo de programação
- ▶ Possibilidade de programar suas próprias análises, de acordo com suas necessidades
- ▶ Integração com outros softwares
  - ▶ GIS
  - ▶ Git <- **Salvação da humanidade**
  - ▶ Excel
  - ▶ C++
  - ▶ Python
  - ▶ Esta aula foi feita em R no RStudio

# Primeiros passos no R

## Abrir o RStudio

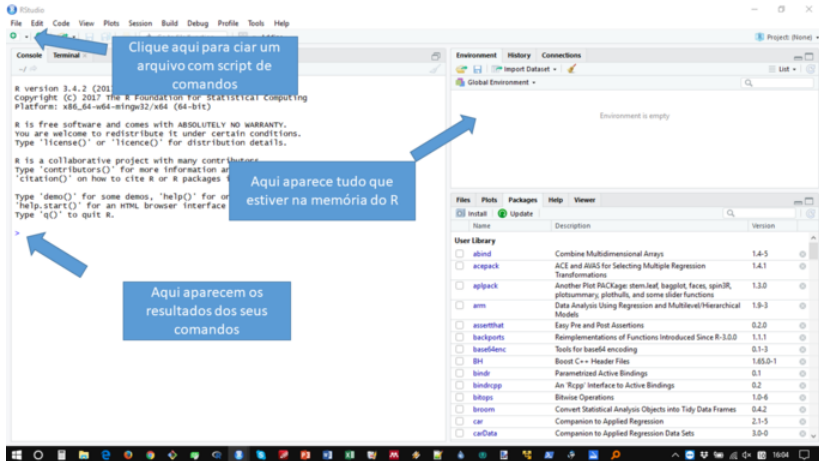


Figure 2:

# No RStudio

## Vantagens do RStudio

Note que o RStudio muda automaticamente as cores do código. Isto é chamado `syntax highlighting`, e serve pra facilitar a leitura. O que cada cor quer dizer?

O Rstudio também insere automaticamente um `)`, `}` ou `]` quando você digita `(`, `{`, ou `[`, o que diminui a maior parte dos erros de digitação

Digite o comando abaixo na aba console, onde está o caractere `>`, e depois tecla **enter**

```
4 + 12
```

```
## [1] 16
```

O resultado aparece logo abaixo, na própria aba, em cor diferente

O R é uma calculadora completa, tente várias combinações.

Qualquer texto após # é ignorado pelo R

```
1 + 1 # soma
```

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

```
3 - 74 ## subtração
```

```
## [1] -71
```

```
4 * 12 ## multiplicação
```

```
## [1] 48
```

```
3 / 7 ## divisão
```

```
## [1] 0.4285714
```



## O R também aceita comandos mais complicados

```
pi ^ 2 # potência
```

```
## [1] 9.869604
```

```
((4 * 7) + 45) / 2
```

```
## [1] 36.5
```

O que está dentro dos parênteses é avaliado de dentro para fora, e o resultado é usado para a operação seguinte, e assim por diante

# Logaritmos, exponenciações, raízes

```
log(10, base=2)
```

```
## [1] 3.321928
```

```
exp(1)
```

```
## [1] 2.718282
```

```
sqrt(2)
```

```
## [1] 1.414214
```

# Funções

Quando temos um nome seguido de parêntese com alguma coisa dentro, temos uma **função**

Função é um conjunto de comandos que faz uma operação em um objeto

`sqrt(x)` calcula a raiz quadrada de x

Funções podem ter opções, chamadas argumentos

`log(10, base=2)`

calcula o logaritmo de 10 na base 2

Quase tudo que você vai usar no R são funções

Muito cuidado com a digitação: para o R, maiúsculas e minúsculas são diferentes, e um espaço no meio da palavra atrapalha tudo

**O R é burro, qualquer erro de digitação faz com que ele não encontre o que você procura**

# Ajuda das funções

you can get help from R at any time

```
?log()
```

```
?exp()
```

```
?plot()
```

```
help(library)
```

Functions are often grouped into specialized sets for some analysis, called *packages*

## Outros lugares para pedir ajuda

Quase todos os problemas que você tem com R, alguém já teve e resolveu

Google <- busque por “R manova” ou “R phylogenetics” ou “R cluster analysis”, etc.

Stack Overflow <- geralmente é o primeiro resultado do Google

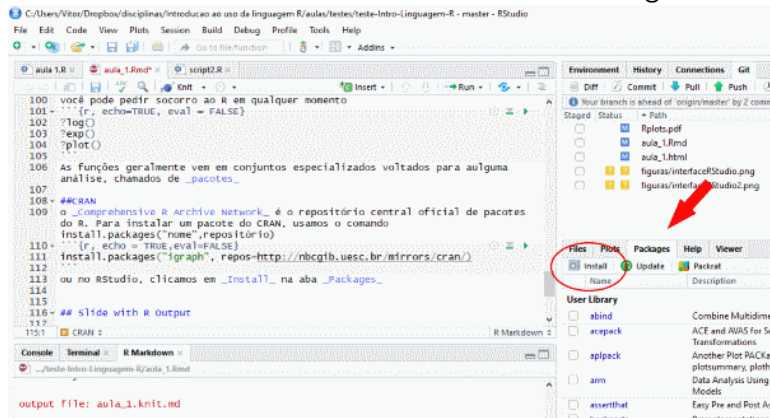
Quando o R der erro, copie a mensagem e cole no Google!

# CRAN

o *Comprehensive R Archive Network* é o repositório central oficial de pacotes do R. Para instalar um pacote do CRAN, usamos o comando `install.packages("nome",repositório)`

`install.packages("igraph", repos=http://nbcgib.uesc.br/mirrors/cran/)`

ou no RStudio, clicamos em **Install** na aba **Packages**



## Para usar um pacote

```
library(igraph)
```

```
##
```

```
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
```

```
##
```

```
##      decompose, spectrum
```

```
## The following object is masked from 'package:base':
```

```
##
```

```
##      union
```

agora as funções do igraph estão disponíveis para serem usadas

# Dados e variáveis

## R é uma linguagem de manipulação de objetos

No R, podemos criar caixas para guardar coisas que iremos usar depois, que chamamos de variáveis, ou objetos

Para isso usamos o sinal = ou <-. Pense na expressão `a = 10` como o objeto `a` recebe o valor 10

<- existe por motivos de compatibilidade com teclados que não possuíam = (lá dos tempos jurássicos da computação). Use =

```
pote.de.sorvete = "feijão"
```

Na linha acima o R interpreta o que está dentro das aspas como texto, e guarda esse valor dentro de um pedaço da memória que recebe o nome `pote.de.sorvete`.

O nome das variáveis não deve conter espaços, acentos, ou caracteres especiais como `ç`, `~`, `!`, `?`, `/`, `|`, `+`, `,`, `^`, `,`, `'`

**\*\*** Ao digitarmos `pote.de.sorvete` no console, o R nos mostra o



## Escolha nomes que façam sentido para suas variáveis

Lembre que você vai ter de ler seus códigos depois, e adivinhar o que você quis dizer com a, aa, x, X, a7589, MyVar e outras bizarrices que pareciam fazer sentido.

Aproveite que o RStudio tem autocompletar!

Para trocar o que está dentro dessa variável, basta usar o = novamente

```
pote.de.sorvete = "sorvete"  
pote.de.sorvete
```

```
## [1] "sorvete"
```

Porque o código baixo não funciona?

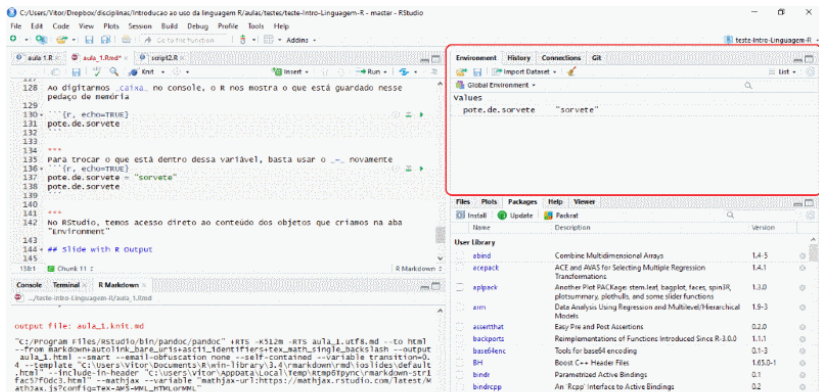
```
pote .de.sorvete = "chuchu"
```

Para ver o que tem na memória do R, use o comando `ls()`

`ls()`

`## [1] "pote.de.sorvete"`

No RStudio, podemos ver o conteúdo dos objetos que criamos na aba "Environment"



## Variáveis também podem guardar conjuntos de dados

Chamamos estas variáveis de vector ou vetor. Criamos elas com a função `combine`, `c()`

```
geladeira = c("sorvete", "cebola", "agua", "miojo", "macarrão", "pacote.de.linha")
geladeira
```

```
## [1] "sorvete"          "cebola"            "agua"
## [4] "miojo"            "macarrão"          "pacote.de.linha"
```

Vetores só guardam objetos iguais: ou texto, ou números, nunca os dois juntos

Veja o que as funções `seq()` e `rep()` fazem!

# Vetor é a unidade básica do R

Quase todas as funções do R são baseadas em manipulação de vetores

```
peso.real = c(100, 45, 77, 60)
peso.no.perfil = c(70, 50, 60, 55)
quilos.omitidos = peso.real - peso.no.perfil
quilos.omitidos
```

```
## [1] 30 -5 17 5
```

Ao somar, multiplicar, subtrair ou dividir vetores, o R pega os valores na mesma posição e faz a operação em questão

**regra da reciclagem** Quando um vetor é menor que o outro, o R repete o vetor menor até completar o maior

```
peso.real = c(100, 45, 77, 60, 50, 77.5)
ganho.apos.almoco = c(2, 5.3)
peso.gordo= peso.real + ganho.apos.almoco
peso.gordo
```

```
## [1] 102.0  50.3  79.0  65.3  52.0  82.8
```

Sempre confira o tamanho dos seus vetores!

## Tamanho e conteúdo dos vetores

Função `length()` retorna o tamanho do vetor

```
peso.real = c(100, 45, 77, 60, 50, 77.5)  
length(peso.real)
```

```
## [1] 6
```

Para acessar o elemento `i` do vetor, use `nome[i]`

```
peso.real = c(100, 45, 77, 60, 50, 77.5)  
peso.real[3]
```

```
## [1] 77
```

# Dataframes

Dataframes são o formato de tabela mais comum do R. No dataframe você pode ter colunas com tipos de dados diferentes, como numérico e texto.

Para criar o dataframe, podemos unir 2 ou mais vetores usando a função `data.frame()`

```
vetor1=LETTERS  
vetor2=1:26  
dados=data.frame(x=vetor1,y=vetor2)  
class(dados)
```

```
## [1] "data.frame"
```

As colunas do dataframe podem ter nomes:

```
vetor3=letters  
vetor4=26:1  
nomesColunas= c("Letras","Numeros")  
dadosNomeados=data.frame(vetor3,y=vetor4)
```

## Acessando dados no dataframe

Existem dois jeitos principais de acessar dados em um dataframe:  
colchetes [] ou cifrão \$

Isto se chama indexação:

dataframe[linha, coluna]

dadosNomeados[3,2]

```
## [1] 24
```

dataframe\$nomeDaColuna

dadosNomeados\$Numeros

```
## [1] 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10
```

```
## [24] 3 2 1
```

você também pode usar ambas notações:

dataframe\$nomeDaColuna[linha]

```
1 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
```



## Indexação

O colchete aceita também vetores como índice, incluindo vetores lógicos (sequencias de TRUE / FALSE)

```
Idade = c(3.0, 4.0, 5.0, 6.0, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0)
```

```
Idade[c(5, 4, 1, 10, 12)]
```

```
## [1] 8 6 3 14 16
```

```
Idade > 6 # retorna um vetor lógico dizendo se cada elemento é maior que 6
```

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

```
## [12] TRUE TRUE
```

```
Idade[ Idade> 6 ] #retorna um vetor com todos os elementos maiores que 6
```

```
## [1] 8 9 10 11 12 14 15 16 17
```

# Matrix

Matrizes são tabelas cujas colunas possuem apenas um tipo de dado, seja numerico ou texto. Podem ser criadas com as funções `matrix`, `cbind`, ou `rbind()`

## Array

Array são estruturas de dados com mais de duas dimensões. pensando no vetor como uma linha, uma dataframe seria uma folha de papel, e o array com 3 dimensões seria o equivalente a um caderno, várias folhas empilhadas. Podemos também ter arrays com 4 ou cinco dimensões, que podemos fazer analogia com uma prateleira de cadernos, uma estante com várias prateleiras, e assim por diante.