



Instituto Tecnológico de Aeronáutica
Divisão de Engenharia Mecânica

Introdução ao Software Estatístico R

Profa. Denise Beatriz Ferrari
www.mec.ita.br/~denise
denise@ita.br

04 / 03 / 2011

Conhecendo o R

“R é uma linguagem computacional que permite que usuário programe algoritmos e utilize ferramentas que foram programadas por outras pessoas ¹”

Com o R podemos...

- ▶ fazer cálculos
- ▶ realizar análises estatísticas
- ▶ gerar gráficos com qualidade de publicação
- ▶ construir funções e programas para necessidades específicas


¹Zuur et. al. (2009) A Beginner's Guide to R. Use R! Springer

Por que aprender R?

- ▶ Custo (\$\$)
- ▶ Disponibilidade para as plataformas UNIX, Windows, MacOS
- ▶ Software Livre
- ▶ Possibilidade de criar e compartilhar pacotes
- ▶ Contém implementações de métodos avançados, que não são facilmente encontrados em outros programas estatísticos
- ▶ Capacidade de produção de gráficos de alta qualidade
- ▶ É amplamente utilizado não apenas na academia, mas em empresas como Google, New York Times, Pfizer, Bank of America, Merck, InterContinental Hotels Group, Shell, etc.

Comprehensive R Archive Network

(www.r-project.org)



About R
[What is R?](#)
[Contributors](#)
[Screenshots](#)
[What's new?](#)

Download, Packages
[CRAN](#)

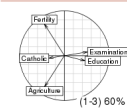
R Project
[Foundation](#)
[Members & Donors](#)
[Mailing Lists](#)
[Bug Tracking](#)
[Developer Page](#)
[Conferences](#)
[Search](#)

Documentation
[Manuals](#)
[FAQs](#)
[The R Journal](#)
[Wiki](#)
[Books](#)
[Certification](#)
[Other](#)

Misc
[Bioconductor](#)
[Related Projects](#)
[User Groups](#)
[Links](#)


The R Project for Statistical Computing

PCA: 5 vars
`prcomp(x = data, cor = cor)`

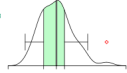


(1-3) 60%

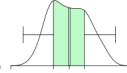
Clustering: 4 groups



Factor 1 [41%]



Factor 3 [19%]



Getting Started:

- R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To [download R](#), please choose your preferred [CRAN mirror](#).
- If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

News:

- [R 2.12.2 prerelease versions](#) will appear starting February 15. Final release is scheduled for February 25, 2011.
- [The R Journal Vol.2/2](#) is available
- R has participated with 5 project in the [Google Summer of Code 2010](#).
- [user! 2010](#), the R user conference, has been held at NIST, Gaithersburg, Maryland, USA, July 21-23, 2010.
- [user! 2011](#), will take place at the University of Warwick, Coventry, UK, August 16-18, 2011.

This server is hosted by the [Institute for Statistics and Mathematics](#) of the [WU Wien](#).

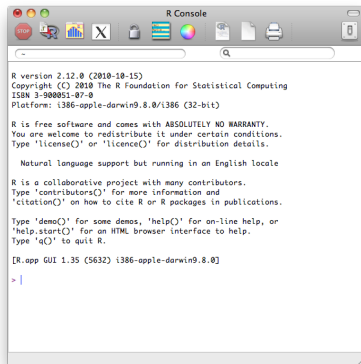
Iniciando uma sessão

Ao clicar o ícone do **R** na área de trabalho, uma janela de comando fica disponível. Cada vez que o **R** é iniciado, cria-se uma nova *sessão*.

O processo de utilização do **R** requer que o usuário digite comandos na janela de comandos e aperte “enter” para que o comando seja executado.

O *prompt* “>” indica que o **R** está pronto para receber um comando.

Para finalizar a sessão, utiliza-se o comando `q()`.

A screenshot of the R Console window on a macOS system. The window has a title bar with the text "R Console" and standard macOS window controls (red, yellow, green buttons). Below the title bar is a toolbar with icons for various functions like opening files, saving, and printing. The main area of the window displays the R startup message, which includes the version number (2.12.0), copyright information (© 2010 The R Foundation for Statistical Computing), and the platform (i386-apple-darwin9.8.0/i386 (32-bit)). It also contains a disclaimer about the software being free and without warranty, followed by instructions on how to cite R or R packages in publications. The prompt ">|" is visible at the bottom of the console.

```
R version 2.12.0 (2010-10-15)
Copyright (C) 2010 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: i386-apple-darwin9.8.0/i386 (32-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.

Natural language support but running in an English locale

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.

[R.app GUI 1.35 (5632) i386-apple-darwin9.8.0]
> |
```

Tela inicial do **R**:
janela de comando ou console.

Operações aritméticas em R

R utiliza os seguintes símbolos para realizar operações aritméticas:

$+$: adição

$-$: subtração

$*$: multiplicação

$/$: divisão

$()$: ordenamento de operações

$%%$: resto de divisão

$\%/\%$: divisão inteira

```
> (1 + 1/100)^100
```

```
[1] 2.704814
```

```
> 5 %% 2
```

```
[1] 1
```

```
> 5 \%/% 2
```

```
[1] 2
```

Em cada caso, uma expressão matemática é inserida, avaliada e o resultado da operação é impresso na tela.

Note que cada resultado é precedido por “[1]”. O prefixo [1] indica que o resultado é o primeiro elemento do vetor de saída.

Operações lógicas em R

R também realiza operações lógicas:

`!x` : NÃO

Operações elemento a elemento:

`x & y` : E

`x | y` : OU

`xor(x,y)` : OU exclusivo

`isTRUE(x)` : (para vetor unitário)

Operações com apenas o primeiro elemento de um vetor:

`x && y` : E

`x || y` : OU

R disponibiliza diversas funções pré-programadas, tais como `sin(x)`, `cos(x)`, `log(x)`, `sqrt(x)`, entre muitas outras.

```
> exp(1)
[1] 2.718282
```

```
> pi
[1] 3.141593
```

```
> sin(pi/6)
[1] 0.5
```

```
> floor(exp(1))
[1] 2
```

```
> ceiling(pi)
[1] 4
```

R calcula valores numéricos com precisão elevada. Porém, está pré-programado para representar apenas 7 dígitos significativos. Esta opção pode ser modificada utilizando a função `options(digits=x)`:

```
> options(digits=16)
> pi
[1] 3.141592653589793
```

As funções `floor(x)` e `ceiling(x)` arredondam, respectivamente, para o menor e maior número inteiro mais próximo.

Constantes pré-definidas

Algumas constantes especiais estão disponíveis.

- ▶ Lógicas: TRUE, FALSE (evite T e F)
- ▶ Valores especiais:
 - NaN “not a number” (0/0)
 - NA valor faltante (desconhecido)
 - NULL valor indefinido (objeto nulo)
 - Inf ou -Inf infinito (1/0, -1/0)
 - pi 3.141593...
- ▶ Outras:
 - LETTERS “A”, “B”, ..., “Z”
 - letters “a”, “b”, ..., “z”
 - month.abb “Jan”, “Feb”, ..., “Dec”
 - month.name “January”, “February”, ..., “December”

Buscando ajuda

Aprender a programar em **R** envolve lembrar funções e saber encontrar ajuda quando necessário.

Para obter detalhes a respeito da função `sqrt(x)`, por exemplo:

```
> ?sqrt
```

```
> help("sqrt")
```

```
> help.search("sqrt")
```

Obs. Por se tratar de um software livre, existe uma grande grande quantidade de informação disponível na internet, no entanto pode ser difícil encontrá-la (“R”, além de ser o nome do software também é uma letra do alfabeto, portanto presente em diversos sites).

– O buscador <http://www.rseek.org/> restringe a busca para os sites que possuem conteúdo relacionado apenas à linguagem **R**.

MathFun {base}

Miscellaneous Mathematical Functions

Description

These functions compute miscellaneous mathematical functions. The naming follows the standard for computer languages such as C or Fortran.

Usage

```
abs(x)
sqrt(x)
```

Arguments

\times a numeric or **complex** vector or array.

Details

These are [internal generic primitive](#) functions: methods can be defined for them individually or via the [Math](#) group generic. For complex arguments (and the default method), $z, \text{abs}(z) == \text{Mod}(z)$ and $\text{sqrt}(z) == z^{0.5}$.

`abs(x)` returns an **integer** vector when `x` is integer or **logical**.

S4 methods

Both are S4 generic and members of the [Math](#) group generic.

References

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

See Also

Arithmetic for simple, log for logarithmic, sin for trigonometric, and Special for special mathematical functions.

'plotmath' for the use of `sqrt` in plot annotation.

Examples

```
require(stats) # for spline
require(graphics)
xx <- -9:9
plot(xx, sqrt(abs(xx)), col = "red")
lines(spline(xx, sqrt(abs(xx)), n=101), col = "pink")
```

[Package *base* version 2.12.0 [Index](#)]

As operações anteriores mostram o resultado da avaliação de comandos sendo impressos em tela. No entanto, um resultado pode ser armazenado através da *atribuição* do valor calculado a uma *variável*.

```
> x <- 5 + 7
```

```
> x
```

```
[1] 12
```

```
> y <- sqrt(4)
```

```
> y
```

```
[1] 2
```

```
> z <- x^y
```

```
> z [1] 144
```

```
> n <- 1
```

```
> (n <- n + 1)
```

```
[1] 2
```

A atribuição de valores se dá através da utilização do operador "<-".

Nomes de variáveis podem incluir letras, números e caracteres "." ou "_", desde que iniciem com uma letra ou ".".

Para visualizar o valor de uma variável, basta digitar o seu nome ou os comandos `print(x)` ou `show(x)` ou, ainda, digitando a expressão de atribuição entre parênteses.

Procure nomear suas variáveis de maneira informativa, afim de melhorar a inteligibilidade de seu código.

Objetos

Toda informação é armazenada em **R** na forma de *objetos*. Variáveis são apenas um tipo de objeto.

Durante uma sessão, todos os objetos são armazenados na área de trabalho, ou *workspace*.

Podemos visualizar o conteúdo da área de trabalho utilizando as funções `objects()` ou `ls()`. Para remover objetos utilizamos as funções `remove()` ou `rm()`.

```
> x <- 5 + 7; y <- sqrt(4)
> z <- x^y
```

Ponto e vírgula (;) separa comandos distintos.

```
> ls()
[1] "x" "y" "z"
```

O comando `rm(list=ls())` remove todos os objetos da área de trabalho:

```
> rm("x")
> objects()
[1] "y" "z"
```

```
> rm(list=ls())
> objects()
character(0)
```

A close-up photograph of several purple iris flowers. The focus is on the lower petals, which are a deep, dark purple with prominent white or light-colored markings that resemble a mask or a series of vertical stripes. The upper petals are a lighter, vibrant purple. The background is a soft, out-of-focus green, suggesting foliage.

Exemplo: Análise da massa dados “Iris”

(Imagem retirada do site: <http://www.ubcbotanicalgarden.org>)

Os dados

A massa de dados “Iris” é provavelmente o exemplo mais famoso na literatura de reconhecimento de padrões.

Foi publicada por Ronald Fisher em 1936 para demonstrar a técnica de Análise Discriminante Linear:

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annual Eugenics*, 7, Part II, 179–188.

Temos 150 observações de cada um dos seguintes tipos de flores:

- ▶ Iris-setosa
- ▶ Iris-versicolor
- ▶ Iris-virginica

Atributos considerados:

- ▶ Sepal Length
- ▶ Sepal Width
- ▶ Petal Length
- ▶ Petal Width

Obtendo os dados na internet

Armazenamos os dados disponíveis no site em uma estrutura de dados chamada “iris.dat”, utilizando a função `read.table()`:

```
> iris.dat <- read.table( file =  
"http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data",  
sep = ",",  
header = FALSE)
```

A função `read.table()` recebe os argumentos:

<code>sep = ","</code>	:	para valores separados por VÍRGULAS
<code>sep = "\t"</code>	:	para valores separados por TABULAÇÃO
<code>sep = " "</code>	:	para valores separados por ESPAÇO

`header = TRUE` : o arquivo contém nomes de variáveis na primeira linha

Estruturas de dados

Como foi dito anteriormente, **R** armazena informação através de objetos. Diferentes estruturas de dados em **R** são representadas através de diferentes *classes* de objetos. Algumas classes comuns são:

<code>vector</code>	:	coleção ordenada de elementos primitivos (objeto básico de R)
<code>matrix</code> e <code>array</code>	:	coleção “retangular” de elementos primitivos de dimensão 2, ou maior.
<code>factor</code>	:	variável categórica
<code>data.frame</code>	:	tabela de dados de dimensão 2.

Os elementos primitivos podem ser de vários tipos:

<code>numeric</code>	:	números reais
<code>complex</code>	:	números complexos
<code>integer</code>	:	números inteiros
<code>character</code>	:	cadeia de caracteres
<code>logical</code>	:	elementos são TRUE ou FALSE
<code>raw</code>	:	elementos são bytes

Manipulando estruturas de dados

Podemos verificar a classe do objeto `iris.dat` utilizando a função `class()`:

```
> class(iris.dat)
[1] "data.frame"
```

Para visualizar partes (inicial ou final) da estrutura de dados, aplicamos as funções `head(x, n)` e `tail(x, n)`:

```
> head(iris.dat, n = 2) # n determina o número de elementos mostrados
```

	V1	V2	V3	V4	V5
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa

```
> tail(iris.dat, n = 3)
```

	V1	V2	V3	V4	V5
148	6.5	3.0	5.2	2.0	Iris-virginica
149	6.2	3.4	5.4	2.3	Iris-virginica
150	5.9	3.0	5.1	1.8	Iris-virginica

Manipulando estruturas de dados

As variáveis (atributos) que compõe a tabela de dados não apresentam nomes ilustrativos. Podemos atribuir nomes para estas variáveis:

```
# Primeiro, criamos um vetor com os novos nomes,
# utilizando a função c() para concatenar elementos primitivos
# (neste caso, cadeias de caracteres):
> nomes <- c("Sepal.L", "Sepal.W", "Petal.L", "Petal.W", "Species")
> class(nomes)
[1] "character"
> is.vector(nomes)
[1] TRUE

# Atribuímos os valores deste vetor aos nomes das colunas de iris.dat:
> colnames(iris.dat) <- nomes; colnames(iris.dat)
[1] "Sepal.L" "Sepal.W" "Petal.L" "Petal.W" "Species"

# Para utilizar o nome das variáveis diretamente para
# acessar objetos na estrutura de dados, utilizamos a função attach():
> attach(iris.dat)
# Podemos fazer agora:
> unique(iris.dat$Species)
[1] Iris-setosa Iris-versicolor Iris-virginica
Levels: Iris-setosa Iris-versicolor Iris-virginica
> is.factor(Species)
[1] TRUE
```

Criação de vetores

(pequeno desvio de curso)

No slide anterior, criamos um vetor de nomes para os atributos da estrutura de dados `iris.dat` utilizando a função `c()`.

Podemos criar vetores nulos, ou que contenham seqüências de números, ou sem valores específicos:

```
# Vetor nulo:
```

```
> v1 <- c(); v1
```

```
NULL
```

```
# Vetores contendo seqüências de números:
```

```
> v2 <- 1:10; v2
```

```
[1] 1 2 3 4 5 6 7 8 9 10
```

```
> v3 <- seq(from = 0, to = 1, by = 0.1); v3
```

```
[1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

```
# Vetor contendo elementos repetidos:
```

```
> v4 <- rep(NA, times = 5); v4
```

```
[1] NA NA NA NA NA
```

```
# Vetor com seqüência de números aleatórios:
```

```
> v5 <- runif(3, min = 0, max = 1); v5
```

```
[1] 0.003155844 0.389096909 0.890503041
```

Explorando os dados

...de volta ao exemplo...

Obter um conjunto resumido de informações estatísticas pode ser bastante útil antes do início da análise dos dados, propriamente dita, já que permite uma visão global da massa de dados em estudo (se existem pontos fora da curva, se os limites para cada variável fazem sentido, etc.).

A função `summary()` fornece um resumo dos dados:

```
> summary(iris.dat)
```

Sepal.L	Sepal.W	Petal.L	Petal.W
Min. :4.300	Min. :2.000	Min. :1.000	Min. :0.100
1st Qu.:5.100	1st Qu.:2.800	1st Qu.:1.600	1st Qu.:0.300
Median :5.800	Median :3.000	Median :4.350	Median :1.300
Mean :5.843	Mean :3.054	Mean :3.759	Mean :1.199
3rd Qu.:6.400	3rd Qu.:3.300	3rd Qu.:5.100	3rd Qu.:1.800
Max. :7.900	Max. :4.400	Max. :6.900	Max. :2.500

Species

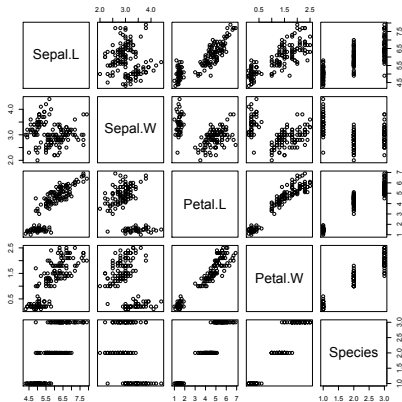
Iris-setosa	:50
Iris-versicolor	:50
Iris-virginica	:50

Como vimos anteriormente, a variável `Species` é um fator. Portanto, `summary()` retorna uma tabela com o número de observações para cada nível.

Explorando os dados

Podemos também explorar a massa de dados utilizando ferramentas gráficas simples:

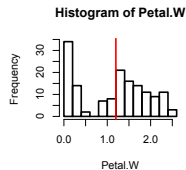
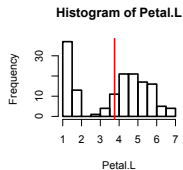
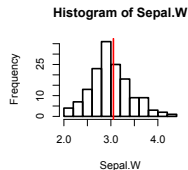
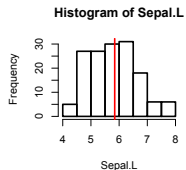
```
> plot(iris.dat)
```



Explorando os dados

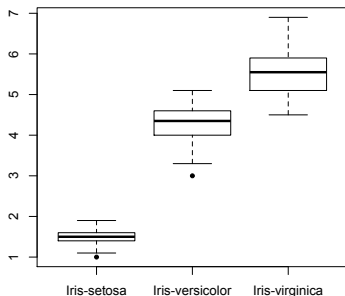
A função `hist()` gera histogramas, que são úteis em representar graficamente a distribuição dos valores nos dados analisados:

```
# Definição de parâmetros gráficos:  
> par(mfrow = c(2,2), lwd = 2)  
  
# Gera histograma para Sepal.L  
> hist(Sepal.L)  
# Adiciona média:  
> abline(v = mean(Sepal.L), col = "red")  
  
> hist(Sepal.W)  
> abline(v = mean(Sepal.W), col = "red")  
  
> hist(Petal.L)  
> abline(v = mean(Petal.L), col = "red")  
  
> hist(Petal.W)  
> abline(v = mean(Petal.W), col = "red")
```



Outra representação gráfica da distribuição de valores é dada pelo *boxplot*:

```
> boxplot(Petal.L ~ Species)
```



A relação $y \sim x$ é chamada “fórmula” em R, em que o símbolo \sim representa uma relação de dependência.

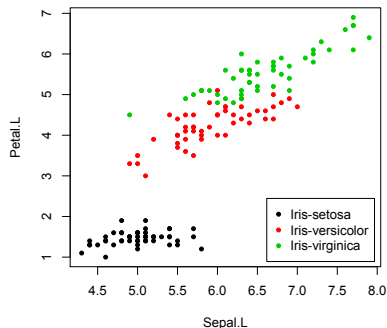
Neste caso, estamos gerando o *boxplot* para os valores da variável “Petal.L” como função da variável “Species”.

Podemos escolher representar graficamente a relação entre um único par de variáveis:

```
> plot(iris.dat[c(1,3)], pch = 20,  
      col = as.numeric(Species))  
> legend("bottomright",  
      legend = unique(Species),  
      col = 1:3,  
      pch = 20,  
      inset = 0.02)
```

Escolhemos apenas a 1a. e 3a. variáveis com o comando `[c(1,3)]`.

A função `as.numeric()` transforma fatores em números (1, 2, 3).



Alguns parâmetros definem elementos gráficos: “pch” determina o caractere usado para representar cada ponto do gráfico, enquanto “col” atribui uma cor.

A função `legend()` acrescenta a legenda no gráfico.

Explorando os dados

Extraindo subconjuntos de dados

Observamos no gráfico do slide anterior que a espécie Iris-setosa se destaca das demais. Podemos excluí-la de nossa análise da seguinte maneira:

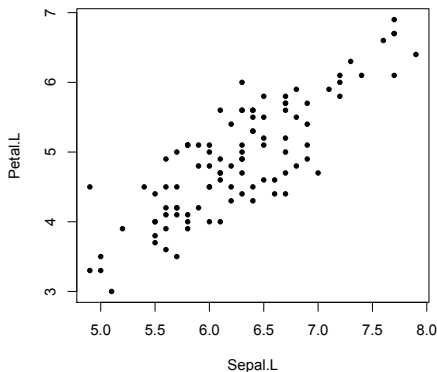
```
# A função which(x) retorna a posição no vetor x
# cujo resultado da avaliação vale TRUE
> setosa <- which(Species == "Iris-setosa")

# Para excluir as observações da espécie Iris-setosa, fazemos:
> iris.datS <- iris.dat[-setosa, ]
# O sinal “-” remove as entradas definidas pela variável “setosa”.
```

Modelo linear

Observamos anteriormente que, removendo-se as observações da espécie Iris-setosa, que existe uma relação aproximadamente linear entre as variáveis “Petal.L” e “Sepal.L” para as observações das outras espécies:

```
> plot(Petal.L ~ Sepal.L, data = iris.dat2)  
# Note que estamos utilizando os dados "iris.dat2"
```



Vamos agora construir um modelo de regressão linear:

$$\text{Petal.L}_i = \alpha + \beta \text{Sepal.L}_i + \epsilon_i, \quad \epsilon_i \text{ i.i.d. } N(0, 1)$$

Queremos estimar os parâmetros α e β .

Em **R**, utilizamos a função `lm()` para gerar um objeto da classe “`lm`” (linear model). O modelo é especificado da seguinte maneira:

```
> reg <- lm(Petal.L ~ Sepal.L, data = iris.dat2)
```

Modelo linear

Podemos utilizar a função `summary()` para obter informação a respeito do ajuste do modelo:

```
> summary(reg)
```

Call:

```
lm(formula = Petal.L ~ Sepal.L, data = iris.dat2)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.96754	-0.32448	-0.03883	0.32768	1.05479

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-1.55571	0.44366	-3.507	0.000687 ***
Sepal.L	1.03189	0.07046	14.645	< 2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4647 on 98 degrees of freedom

Multiple R-squared: 0.6864, Adjusted R-squared: 0.6832

F-statistic: 214.5 on 1 and 98 DF, p-value: < 2.2e-16

Modelo linear

Um objeto da classe `lm` possui muitos atributos.

```
> names(reg)
```

[1]	"coefficients"	"residuals"	"effects"	"rank"
[5]	"fitted.values"	"assign"	"qr"	"df.residual"
[9]	"xlevels"	"call"	"terms"	"model"

```
> names(summary(reg))
```

[1]	"call"	"terms"	"residuals"	"coefficients"
[5]	"aliased"	"sigma"	"df"	"r.squared"
[9]	"adj.r.squared"	"fstatistic"	"cov.unscaled"	

Podemos acessar os valores de um determinado atributo utilizando o símbolo "\$":

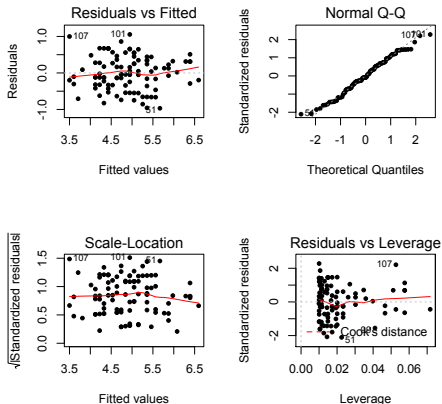
```
> reg$coefficients
```

(Intercept)	Sepal.L
-1.555713	1.031893

Modelo linear

É possível obter alguns gráficos de diagnóstico para o modelo linear ajustado utilizando a função `plot()`:

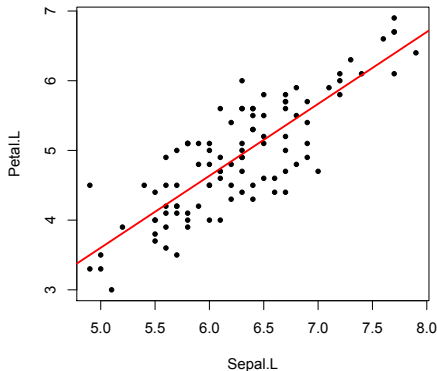
```
> par(mfrow = c(2,2)) # gera gráficos em uma matriz 2 x 2  
> plot(reg)  
> par(mfrow = c(1,1)) # retorna à configuração original
```



Modelo linear

Podemos adicionar a curva de regressão ao gráfico `Petal.L` \times `Sepal.L` utilizando a função `abline()`:

```
> plot(Petal.L ~ Sepal.L, data = iris.dat2)
> abline(coef(reg), col = "red", lwd = 2)
```



Gráficos podem ser salvos em formato PDF para serem utilizados posteriormente, na geração de relatórios em LaTeX, por exemplo.

```
> pdf("iris-Regressao.pdf", width = 5, height = 5, onefile = TRUE)
> plot(Petal.L ~ Sepal.L, data = iris.dat2)
> dev.off()
```

Basta, então verificar no diretório de trabalho o arquivo “iris-Regressao.pdf” contendo o gráfico do slide anterior.

Salvando o trabalho

É possível salvar o conteúdo da área de trabalho, para utilizar posteriormente, através dos comandos `save.image()` e `save()`.

```
# Para salvar todos os objetos na área de trabalho:  
> save.image(file = "iris-Ex.Rdata")  
  
# Para especificar os objetos a serem salvos:  
> save(reg, file = "iris-Reg.Rdata")
```

Cada um destes arquivos será salvo no diretório de trabalho em que **R** foi iniciado.

```
# Para verificar o diretório de trabalho ("working directory"):  
> getwd()  
[1] "/Users/deniseferrari"  
  
# Para modificar o diretório de trabalho:  
> setwd()  
[1] "/Users/deniseferrari/Documents"  
  
# Para carregar um objeto salvo previamente:  
> load(file = "iris-Ex.Rdata")
```

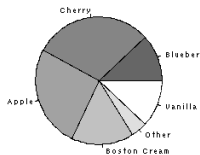
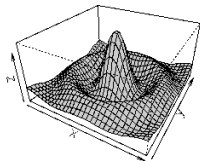
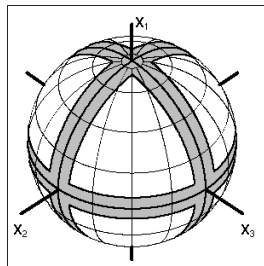
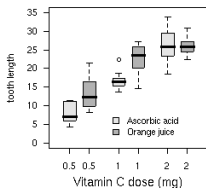
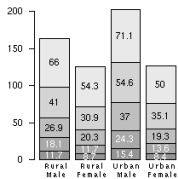
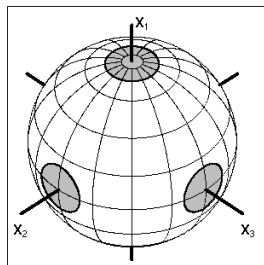
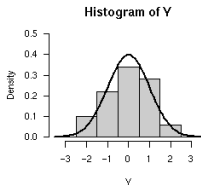
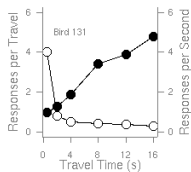
Assim termina nossa primeira sessão em **R**.

Assim termina nossa primeira sessão em **R**.

... E este é apenas o começo!

Propaganda

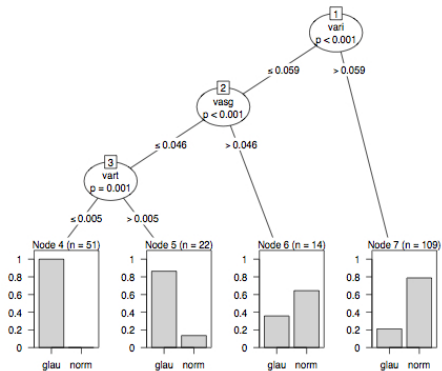
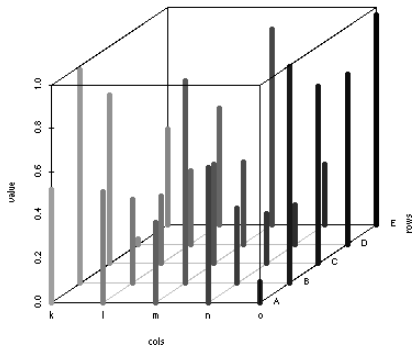
Gráficos Estáticos



fonte:

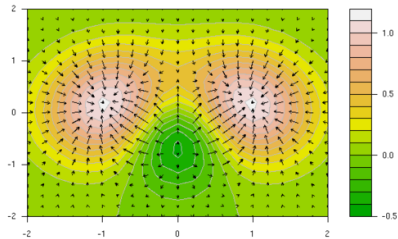
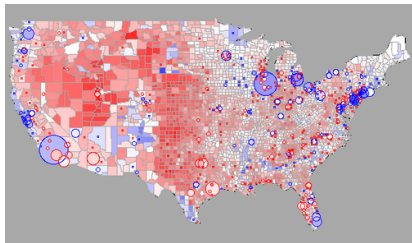
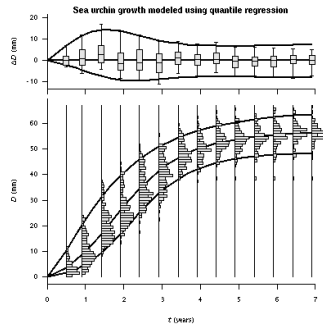
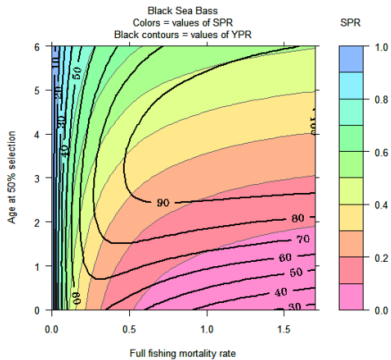
(<http://www.stat.auckland.ac.nz/~paul/RGraphics/rgraphics.html>)

scatterplot3d - 4

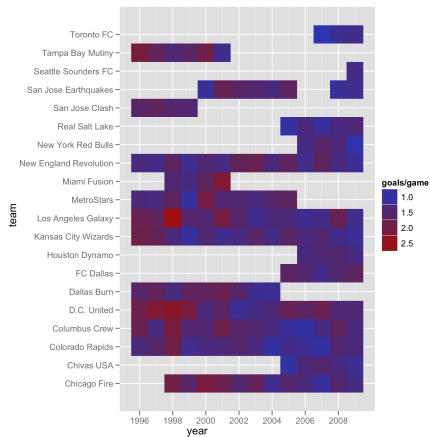
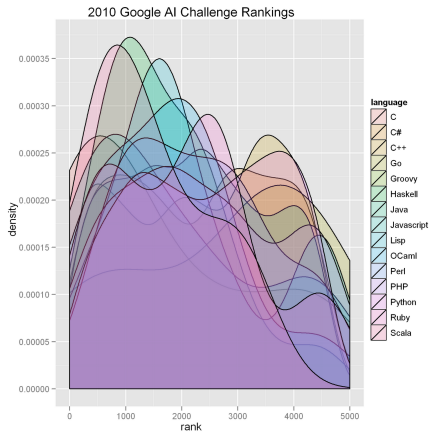


fonte:

(<http://statgraphics.blog.com/>)



fonte:
(<http://addictedtor.free.fr/graphiques>)



fonte:
(<http://www.r-bloggers.com>)

Animações
