Fundamentos Estatísticos para Ciência dos Dados
R e Dados

Renato Martins Assunção

DCC, UFMG - 2017

Dados, dados, dados..

- Coletamos regularmente dados dos mais variados tipos: numéricos, strings, imagens, sons, vídeos.

Dados, dados, dados..

- Coletamos regularmente dados dos mais variados tipos: numéricos, strings, imagens, sons, vídeos.
- Estes dados podem estar estruturados de forma complexa.

Dados, dados, dados..

- Coletamos regularmente dados dos mais variados tipos: numéricos, strings, imagens, sons, vídeos.
- Estes dados podem estar estruturados de forma complexa.
- Por exemplo, atributos individuais de usuários do Facebook organizados na forma de uma grafo de amizade.

Dados, dados, dados..

- Coletamos regularmente dados dos mais variados tipos: numéricos, strings, imagens, sons, vídeos.
- Estes dados podem estar estruturados de forma complexa.
- Por exemplo, atributos individuais de usuários do Facebook organizados na forma de uma grafo de amizade.
- Ou dados individuais organizados como árvores genealógicas em estudos de DNA.

Dados, dados, dados..

- Coletamos regularmente dados dos mais variados tipos: numéricos, strings, imagens, sons, vídeos.
- Estes dados podem estar estruturados de forma complexa.
- Por exemplo, atributos individuais de usuários do Facebook organizados na forma de uma grafo de amizade.
- Ou dados individuais organizados como árvores genealógicas em estudos de DNA.
- Todos estes dados podem (e são) analisados estatisticamente.

Dados, dados, dados..

- Coletamos regularmente dados dos mais variados tipos: numéricos, strings, imagens, sons, vídeos.
- Estes dados podem estar estruturados de forma complexa.
- Por exemplo, atributos individuais de usuários do Facebook organizados na forma de uma grafo de amizade.
- Ou dados individuais organizados como árvores genealógicas em estudos de DNA.
- Todos estes dados podem (e são) analisados estatisticamente.
- NESTE CURSO, vamos nos concentrar num único tipo de dado: aqueles organizados de forma tabular.

Dados tabulares

	spam	num_char	line_breaks	format	number
1	no	21,705	551	html	small
2	no	7,011	183	html	big
3	yes	631	28	text	none
⋮	⋮	⋮	⋮	⋮	⋮
50	no	15,829	242	html	small

Tabela: Quatro primeiras linhas da tabela `spam`. Fonte: OpenIntro Statistics Project, `https://www.openintro.org/stat/textbook.php`.

Dados tabulares

spam	num_char	line_breaks	format	number	ratioti	%obs
no	21,705	551	html	small	2.45	52.30
no	7,011	183	html	big	0.00	12.30
yes	631	28	text	none	0.00	0.00
⋮	⋮	⋮	⋮	⋮	⋮	
no	15,829	242	html	small	34.71	0.03

variable	description
spam	Specifies whether the message was spam
num char	The number (#) of characters in the email
line breaks	# line breaks in the email (not including text wrapping)
format	Indicates if the email contained special formatting, such as bolding, tables, or links, which would indicate the message is in HTML format
number	Indicates whether the email contained no number, a small number ($<$ 1 million), or a large number
ratioti	ratio of image area to text: a message using images instead of words in order to sidestep text-based filtering.
%obs	% HTML with obfuscated text, such as unnecessary hex-encoding of ASCII characters in an attempt to avoid text-based filters

Dados tabulares

- Cada linha da tabela corresponde a um *caso*.

Dados tabulares

- Cada linha da tabela corresponde a um *caso*.
- Casos também são chamados de *observações*, *instâncias*, ou *exemplos*.

Dados tabulares

- Cada linha da tabela corresponde a um *caso*.
- Casos também são chamados de *observações*, *instâncias*, ou *exemplos*.
- Cada coluna corresponde a uma variável.

Dados tabulares

- Cada linha da tabela corresponde a um *caso*.
- Casos também são chamados de *observações*, *instâncias*, ou *exemplos*.
- Cada coluna corresponde a uma variável.
- Uma variável também é chamada de *atributo*, ou *característica* (feature, em inglês).

Dados tabulares

- Cada linha da tabela corresponde a um *caso*.
- Casos também são chamados de *observações*, *instâncias*, ou *exemplos*.
- Cada coluna corresponde a uma variável.
- Uma variável também é chamada de *atributo*, ou *característica* (feature, em inglês).
- A tabela de dados coletados é chamada de *amostra* (sample, em inglês).

Tipos de variáveis

Tipos de variáveis

spam	num_char	line_breaks	format	number	ratioti	%obs
no	21,705	551	html	small	2.45	52.30
no	7,011	183	html	big	0.00	12.30
yes	631	28	text	none	0.00	0.00
⋮	⋮	⋮	⋮	⋮	⋮	
no	15,829	242	html	small	34.71	0.03

- Numérica $\Rightarrow$ discreta ou contínua.

Tipos de variáveis

spam	num_char	line_breaks	format	number	ratioti	%obs
no	21,705	551	html	small	2.45	52.30
no	7,011	183	html	big	0.00	12.30
yes	631	28	text	none	0.00	0.00
⋮	⋮	⋮	⋮	⋮	⋮	
no	15,829	242	html	small	34.71	0.03

- Numérica $\Rightarrow$ discreta ou contínua.
- Uma variável *numérica*: faz sentido somar, subtrair ou tomar médias destes valores.

Tipos de variáveis

spam	num_char	line_breaks	format	number	ratioti	%obs
no	21,705	551	html	small	2.45	52.30
no	7,011	183	html	big	0.00	12.30
yes	631	28	text	none	0.00	0.00
⋮	⋮	⋮	⋮	⋮	⋮	
no	15,829	242	html	small	34.71	0.03

- Numérica $\Rightarrow$ discreta ou contínua.
- Uma variável *numérica*: faz sentido somar, subtrair ou tomar médias destes valores.
- Exemplos: `num char`, `line breaks`, `ratioti` e `%obs`.

Tipos de variáveis

spam	num_char	line_breaks	format	number	ratioti	%obs
no	21,705	551	html	small	2.45	52.30
no	7,011	183	html	big	0.00	12.30
yes	631	28	text	none	0.00	0.00
⋮	⋮	⋮	⋮	⋮	⋮	
no	15,829	242	html	small	34.71	0.03

- Numérica $\Rightarrow$ discreta ou contínua.
- Uma variável *numérica*: faz sentido somar, subtrair ou tomar médias destes valores.
- Exemplos: `num char`, `line breaks`, `ratioti` e `%obs`.
- `num char` e `line breaks` são variáveis *discretas*: assumem apenas alguns valores com saltos entre eles (inteiros, neste caso).

Tipos de variáveis

spam	num_char	line_breaks	format	number	ratioti	%obs
no	21,705	551	html	small	2.45	52.30
no	7,011	183	html	big	0.00	12.30
yes	631	28	text	none	0.00	0.00
⋮	⋮	⋮	⋮	⋮	⋮	
no	15,829	242	html	small	34.71	0.03

- Numérica $\Rightarrow$ discreta ou contínua.
- Uma variável *numérica*: faz sentido somar, subtrair ou tomar médias destes valores.
- Exemplos: `num char`, `line breaks`, `ratioti` e `%obs`.
- `num char` e `line breaks` são variáveis *discretas*: assumem apenas alguns valores com saltos entre eles (inteiros, neste caso).
- `ratioti` e `%obs` são contínuas: em princípio pode ser qualquer valor num intervalo da reta real.

Tipos de variáveis

spam	num_char	line_breaks	format	number	ratioti	%obs
no	21,705	551	html	small	2.45	52.30
no	7,011	183	html	big	0.00	12.30
yes	631	28	text	none	0.00	0.00
⋮	⋮	⋮	⋮	⋮	⋮	
no	15,829	242	html	small	34.71	0.03

- Variáveis categóricas $\Rightarrow$ nominal ou ordinal.

Tipos de variáveis

spam	num_char	line_breaks	format	number	ratioti	%obs
no	21,705	551	html	small	2.45	52.30
no	7,011	183	html	big	0.00	12.30
yes	631	28	text	none	0.00	0.00
⋮	⋮	⋮	⋮	⋮	⋮	
no	15,829	242	html	small	34.71	0.03

- Variáveis categóricas $\Rightarrow$ nominal ou ordinal.
- Categórica Ordinal: valor é um rótulo para uma categoria dentre k possíveis e as categorias podem ser ordenadas.

Tipos de variáveis

spam	num_char	line_breaks	format	number	ratioti	%obs
no	21,705	551	html	small	2.45	52.30
no	7,011	183	html	big	0.00	12.30
yes	631	28	text	none	0.00	0.00
⋮	⋮	⋮	⋮	⋮	⋮	
no	15,829	242	html	small	34.71	0.03

- Variáveis categóricas $\Rightarrow$ nominal ou ordinal.
- Categórica Ordinal: valor é um rótulo para uma categoria dentre k possíveis e as categorias podem ser ordenadas.
- `number`, por exemplo. Existe uma ordem natural nos valores possíveis: none $<$ small $<$ big.

Tipos de variáveis

spam	num_char	line_breaks	format	number	ratioti	%obs
no	21,705	551	html	small	2.45	52.30
no	7,011	183	html	big	0.00	12.30
yes	631	28	text	none	0.00	0.00
⋮	⋮	⋮	⋮	⋮	⋮	
no	15,829	242	html	small	34.71	0.03

- Variáveis categóricas $\Rightarrow$ nominal ou ordinal.
- Categórica Ordinal: valor é um rótulo para uma categoria dentre k possíveis e as categorias podem ser ordenadas.
- `number`, por exemplo. Existe uma ordem natural nos valores possíveis: none $<$ small $<$ big.
- Numa pesquisa, a resposta (*pouco, médio, muito*) para uma pergunta.

Tipos de variáveis

spam	num_char	line_breaks	format	number	ratioti	%obs
no	21,705	551	html	small	2.45	52.30
no	7,011	183	html	big	0.00	12.30
yes	631	28	text	none	0.00	0.00
⋮	⋮	⋮	⋮	⋮	⋮	
no	15,829	242	html	small	34.71	0.03

- Variáveis categóricas $\Rightarrow$ nominal ou ordinal.
- Categórica Ordinal: valor é um rótulo para uma categoria dentre k possíveis e as categorias podem ser ordenadas.
- `number`, por exemplo. Existe uma ordem natural nos valores possíveis: none $<$ small $<$ big.
- Numa pesquisa, a resposta (*pouco, médio, muito*) para uma pergunta.
- Categórica Nominal: apenas rótulos para categorias, sem uma ordenação.
- `spam` e `format`: nominal

R

- R é uma linguagem de script interpretada, open-source.
- Voltada para:
  - manipulação de dados,
  - análise estatística
  - visualização de dados
- Inspirada na linguagem S desenvolvida na AT & T no anos 80.
- R foi escrita por Ross Ihaka e Robert Gentleman, no Depto de Estatística da Univ de Auckland, NZ.

Lendo dados em R

- Dados tabulares usualmente são organizados em data.frames: matrizes em que as variáveis (ou colunas) podem ser de tipos diferentes.

Lendo dados em R

- Dados tabulares usualmente são organizados em data.frames: matrizes em que as variáveis (ou colunas) podem ser de tipos diferentes.
- Alguns comandos para ler dados em dataframes: `read.table`, `read.csv`, e `read.delim`

Lendo dados em R

- Dados tabulares usualmente são organizados em data.frames: matrizes em que as variáveis (ou colunas) podem ser de tipos diferentes.
- Alguns comandos para ler dados em dataframes: `read.table`, `read.csv`, e `read.delim`

```
> pressao = read.csv("T1.dat", header = T, row.names = NULL)
> dim(pressao)
[1] 500 501
> pressao = pressao[, 1:18] # selec. 1as. 18 colunas
> colnames(pressao)
 [1] "sbp"      "gender"   "married"  "smoke"    "exercise" "age"
 [7] "weight"   "height"   "overwt"   "race"     "alcohol"  "trt"
[13] "bmi"      "stress"   "salt"     "chldbear" "income"   "educatn"
```

Lendo dados em R

- Dataframe `pressao` contem 500 pacientes (linhas) e 501 variáveis ou atributos (colunas).
- Dos 500 pacientes, metade tinha pressão arterial baixa e metade, elevada (hipertensão).
- As 501 variáveis (ou colunas) consistem de:
  - pressão sistólica,
  - 17 variáveis clínicas potencialmente preditoras de hipertensão,
  - 483 marcadores genéticos (que foram eliminados na seleção das colunas)
- Dados obtidos em: http://www.math.yorku.ca/Who/Faculty/Ng/ssc2003/BPMain.htm
- Pesquisa conduzida pela empresa farmacêutica GlaxoSmithKline em Toronto, Canadá.

Variáveis

variable	description
sbp	Systolic Blood Pressure, Continuous Variable
gender	Binary Nominal Variable: M = Male, F = Female
married	Binary Nominal Variable: Y = Married, N = Not Married
smoke	Smoking Status, Binary variable: Y = Smoker, N = Non-Smoker
exercise	Exercise level, Categorical variable: 1 = Low, 2 = Medium, 3 = High
age	Continuous variable (years)
weight	Weight, Continuous variable (lbs)
height	Height, Continuous variable (inches)
overwt	Overweight, Categorical variable: 1 = Normal, 2 = Overweight, 3 = Obese.
race	Race, Categorical variable taking values 1, 2, 3, or 4.
alcohol	Alcohol Use, Categorical variable: 1 = Low, 2 = Medium, 3 = High
trt	Treatment for hypertension, Binary Variable: Y = Treated, N = Untreated
bmi	Body Mass Index (BMI), Continuous variable: Weight / Height2 *703
stress	Stress Level: 1 = Low, 2 = Medium, 3 = High
salt	Salt (NaCl) Intake Level: 1 = Low, 2 = Medium, 3 = High
chldbear	Childbearing Potential: 1 = Male, 2 = Able Female, 3 = Unable Female
income	Income Level, Categorical Variable: 1 = Low, 2 = Medium, 3 = High
educatn	Education Level, Categorical Variable: 1 = Low, 2 = Medium, 3 = High

Visualizando os dados numéricos

- A maneira de visualizar os dados depende do tipo de variável.

Visualizando os dados numéricos

- A maneira de visualizar os dados depende do tipo de variável.
- Para variáveis numéricas, o histograma é uma excelente opção: `hist(pressao[,"sbp"])`

Visualizando os dados numéricos

- Temos N exemplos ou instâncias.

Visualizando os dados numéricos

- Temos N exemplos ou instâncias.
- Forme uma grade quebrando o eixo horizontal em pequenos intervalos de comprimento Δ.

Visualizando os dados numéricos

- Temos N exemplos ou instâncias.
- Forme uma grade quebrando o eixo horizontal em pequenos intervalos de comprimento Δ.
- Conte o número de exemplos em cada um dos intervalos: $n_1, n_2, \ldots$ de forma que $N = \sum_i n_i$.

Visualizando os dados numéricos

- Temos N exemplos ou instâncias.
- Forme uma grade quebrando o eixo horizontal em pequenos intervalos de comprimento Δ.
- Conte o número de exemplos em cada um dos intervalos: $n_1, n_2, \ldots$ de forma que $N = \sum_i n_i$.
- Faça um retângulo usando o intervalo da grade como base.

Visualizando os dados numéricos

- Temos N exemplos ou instâncias.
- Forme uma grade quebrando o eixo horizontal em pequenos intervalos de comprimento Δ.
- Conte o número de exemplos em cada um dos intervalos: $n_1, n_2, \ldots$ de forma que $N = \sum_i n_i$.
- Faça um retângulo usando o intervalo da grade como base.
- A altura do i-ésimo retângulo é igual à:
  - contagem n_i de dados que caem no intervalo i
  - OU igual à proporção n_i/N que cai no intervalo i dividida por Δ. Isto é, altura é $n_i/(N\Delta)$.

Visualizando os dados numéricos

- Temos N exemplos ou instâncias.
- Forme uma grade quebrando o eixo horizontal em pequenos intervalos de comprimento Δ.
- Conte o número de exemplos em cada um dos intervalos: $n_1, n_2, \ldots$ de forma que $N = \sum_i n_i$.
- Faça um retângulo usando o intervalo da grade como base.
- A altura do i-ésimo retângulo é igual à:
  - contagem n_i de dados que caem no intervalo i
  - OU igual à proporção n_i/N que cai no intervalo i dividida por Δ. Isto é, altura é $n_i/(N\Delta)$.
- No segundo caso, a soma das áreas dos retângulos é igual a 1 (como no caso de uma densidade de probabilidade).

Visualizando os dados numéricos

- Temos N exemplos ou instâncias.
- Forme uma grade quebrando o eixo horizontal em pequenos intervalos de comprimento Δ.
- Conte o número de exemplos em cada um dos intervalos: $n_1, n_2, \ldots$ de forma que $N = \sum_i n_i$.
- Faça um retângulo usando o intervalo da grade como base.
- A altura do i-ésimo retângulo é igual à:
  - contagem n_i de dados que caem no intervalo i
  - OU igual à proporção n_i/N que cai no intervalo i dividida por Δ. Isto é, altura é $n_i/(N\Delta)$.
- No segundo caso, a soma das áreas dos retângulos é igual a 1 (como no caso de uma densidade de probabilidade).
- O comando no R é `hist`.

Histograma - algumas opções

```
attach(pressao)          # pode se referir diretamente apenas ‘as colunas
par(mfrow=c(2,2))
hist(sbp)
hist(sbp, probability=T)     # IMPORTANTE, MUITO IMPORTANTE
hist(sbp, breaks = c(50, 130, 160, 200, max(sbp)), prob=T, main="PS")
hist(sbp, breaks = 13, prob=T, xlab="pressao", main="Pressao sistolica")
rug(sbp)                     # tapete com os dados originais
```

Histograma - pirâmides etárias

- Distribuição por idade, separado por sexo.

Histograma - necessidade de transformar

- População dos municípios brasileiros

Histograma - necessidade de transformar

- LOGARITMO da População dos municípios brasileiros

Visualizando os dados numéricos

- Se a quantidade de dados, é pequena, o ramo-e-folhas é bem útil

Visualizando os dados numéricos

- Se a quantidade de dados, é pequena, o ramo-e-folhas é bem útil
- O ramo-e-folhas pode ser feito à mão rapidamente e permite visualizar toda a distribuição dos na sua faixa de variação.

```
> bras = read.csv("CampeonatoBrasileiro2014.txt", header=T, row.names=NULL)
> head(bras)
             Time Pts Jogos Vit Emp Der Gols GolsSofr SaldoGols Aprov
1         Cruzeiro  80    38  24   8   6   67       38        29    70
2        Sao Paulo  70    38  20  10   8   59       40        19    61
3    Internacional  69    38  21   6  11   53       41        12    60
4      Corinthians  69    38  19  12   7   49       31        18    60
5 Atletico Mineiro  62    38  17  11  10   51       38        13    54
6       Fluminense  61    38  17  10  11   61       42        19    53
```

Ramo-e-folhas

```
> bras[, "Gols"]
 [1] 67 59 53 49 51 61 36 43 42 46 36 38 37 42 39 34 37 31 31 28

> stem(bras[,"Gols"])
  The decimal point is 1 digit(s) to the right of the |
  2 | 8
  3 | 114667789
  4 | 22369
  5 | 139
  6 | 17

> sort(bras[,"SaldoGols"])
 [1] -28 -25 -17 -17 -12 -10 -10  -5  -3  -2  -1   1   ...  19  19  29

> stem(bras[,"SaldoGols"])
  -2 | 85
  -0 | 772005321
   0 | 17223899
   2 | 9
```

Ramo-e-folhas

Se quiser quebrar cada categoria-dígito em grupos de 5, use "scale"

```
> stem(bras[,"Gols"], scale=2)

  The decimal point is 1 digit(s) to the right of the |

  2 | 8
  3 | 114
  3 | 667789
  4 | 223
  4 | 69
  5 | 13
  5 | 9
  6 | 1
  6 | 7
```

Boxplot

- É um resumo gráfico dos dados com alta compressão: usa 5 números apenas.

Boxplot

- É um resumo gráfico dos dados com alta compressão: usa 5 números apenas.

- Mostra rapidamente se os dados são simétricos, onde estão concentrados e se existem outliers (valores extremos).

Boxplot

- A caixa (box) central tem extremidades laterais essencialmente em Q1 (first quartile, deixa 25% dos dados abaixo) e Q3 (deixa 25% dos dados acima).

Boxplot

- A caixa (box) central tem extremidades laterais essencialmente em Q1 (first quartile, deixa 25% dos dados abaixo) e Q3 (deixa 25% dos dados acima).
- Em inglês, lower hinge (Q1) and upper hinge (Q3).

Boxplot

- A caixa (box) central tem extremidades laterais essencialmente em Q1 (first quartile, deixa 25% dos dados abaixo) e Q3 (deixa 25% dos dados acima).
- Em inglês, lower hinge (Q1) and upper hinge (Q3).
- A linha que divide a caixa central fica na altura de Q2: a mediana, que deixa 50% dos dados abaixo e 50% acima.

Boxplot

- Duas linhas estendem-se a partir da caixa, chamadas de bigodes de gato (whiskers).

Boxplot

- Duas linhas estendem-se a partir da caixa, chamadas de bigodes de gato (whiskers).
- A linha superior tem comprimento U =1.5*(comprimento da caixa)=1.5*(interquartile range)

Boxplot

- Duas linhas estendem-se a partir da caixa, chamadas de bigodes de gato (whiskers).
- A linha superior tem comprimento U =1.5*(comprimento da caixa)=1.5*(interquartile range) ... OU ... vai apenas até o máximo dos dados se ele for menor que o primeiro limite.

Boxplot

- Duas linhas estendem-se a partir da caixa, chamadas de bigodes de gato (whiskers).
- A linha superior tem comprimento U =1.5*(comprimento da caixa)=1.5*(interquartile range) ... OU ... vai apenas até o máximo dos dados se ele for menor que o primeiro limite.
- Idem para a linha inferior.

Boxplot

- Duas linhas estendem-se a partir da caixa, chamadas de bigodes de gato (whiskers).
- A linha superior tem comprimento U =1.5*(comprimento da caixa)=1.5*(interquartile range) ... OU ... vai apenas até o máximo dos dados se ele for menor que o primeiro limite.
- Idem para a linha inferior.
- Os dados além dos bigodes são mostrados individualmente como pontos: **potencialmente** outliers (valores extremos).

Boxplot

Visualizamos facilmente o formato de diferentes distribuições

Boxplot

Útil para comparar distribuições versus o valor de OUTRA variável categórica: pressao aumenta com peso.

Boxplot

Útil para comparar distribuições versus o valor de OUTRA variável categórica: pressao aumenta com peso.

```
> par(mfrow=c(1,1))
> boxplot(sbp ~ overwt, ylab="pressao", xlab="1 = Normal, 2 = Overweight, 3 = Obese
> title("Pressao por Sobrepeso")
```

Boxplot

Exemplo onde a distribuição não muda: pressão versus renda.

Boxplot

Exemplo onde a distribuição não muda: pressão versus renda.

```
> par(mfrow=c(1,1))
> boxplot(sbp ~ income, ylab="pressao", xlab="1=Low, 2=Middle, 3=High")
> title("Pressao por renda")
```

Boxplot

Usando duas variáveis categóricas ao mesmo tempo: pressão versus sobrepeso e sexo.

Boxplot

Usando duas variáveis categóricas ao mesmo tempo: pressão versus sobrepeso e sexo.

```
par(mfrow=c(1,1))
boxplot(sbp ~ overwt*gender, ylab="pressao", xlab="sobrepeso e sexo")
title("Pressao por sobrepeso e sexo")
```

Boxplot

Fonte e código em: `http://bit.ly/2np7ikU`

Boxplot

Figura: Muitos box plots organizados em grupos de 3

Fonte e código em: `http://bit.ly/2nGyg3G`

Boxplot

Distribuição pode variar de forma complexa:

Boxplot

Distribuição pode variar de forma complexa: grupo tratado tem pressão média um pouco menor mas o mais importante é que quase não variam em torno de sua média. .

Boxplot

Distribuição pode variar de forma complexa: grupo tratado tem pressão média um pouco menor mas o mais importante é que quase não variam em torno de sua média. .

```
par(mfrow=c(1,1))
boxplot(sbp ~ trt, ylab="pressao", xlab="1 = Treated, 0 = Untreated")
title("Pressao por Tratamento para hipertensao")
```

Qual o nome do bebê?

Scatterplot: o campeão dos gráficos

Visualiza a relação entre duas variáveis numéricas com nuvem de pontos.

```
# Simple Scatterplot, codigo do site Quick-R
attach(mtcars)
plot(wt, mpg, main="Scatterplot Example", xlab="Car Weight ",
         ylab="Miles Per Gallon ", pch=19)
```

O campeão dos gráficos

Com mais pontos (mais carros): relação fica mais fácil de ser percebida.

O campeão dos gráficos

Relação pode ser mais complexa, exigindo mais explicação.

Matriz de scatterplots

Muitas nuvens de pontos ao mesmo tempo.

```
# Basic Scatterplot Matrix, codigo do site Quick-R
pairs(~mpg+disp+drat+wt,data=mtcars,
   main="Simple Scatterplot Matrix")
```

Nuvens passageiras

Nuvens às vezes, mal comportadas. As duas primeiras são as "ideais´´, simples de entender: sem relação e relação linear.

Nuvens passageiras

Nuvens passageiras

Particionando a nuvem de acordo com uma terceira variável categórica: pode produzir melhor entendimento da relação entre as duas variáveis do scatterplot.

3D Scatterplot

```
# 3D Scatterplot, codigo do site Quick-R
library(scatterplot3d)
attach(mtcars)
scatterplot3d(wt,disp,mpg, main="3D Scatterplot")
```

Mais um 3D Scatterplot

```
# 3D Scatterplot with Coloring and Vertical Drop Lines, from site Quick-R
library(scatterplot3d) ; attach(mtcars)
scatterplot3d(wt,disp,mpg, pch=16, highlight.3d=TRUE,
  type="h", main="3D Scatterplot")
```

Vetores em R

Para entrar rapidamente com pequenos conjuntos de dados: use a função c, que combina ou concatena elementos num vetor.

```
> # gols marcados no brasileirao de 2014, por time
> x = c(67,59,53,49,51,61,36,43,42,46,36,38,37,42,39,34,37,31,31,28)
> max(x)                                # uma funcao aplicada ao vetor
[1] 67
> mean(x); median(x)        # mais de um comando por linha
[1] 43
[1] 40.5
> summary(x)                        # resumo basico com 5 nos.
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   28.0    36.0    40.5    43.0    49.5    67.0
> x[1]-x[2]           # acessando elementos do vetor
[1] 8
> x > 40 & x < 50              # vetor logico: quem atende 'a condicao
 [1] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE  ...... FALSE
> mean( x[ x > 40 & x < 50] )  # aninhando: media dos x's que sao > 40 e < 50
[1] 44.4
> which(x == max(x))      # quais posicoes do vetor sao T
[1] 1
```

Vetores em R

```
> x[c(3, 5, 8:11)]                # selecionando elementos de x
[1] 53 51 43 42 46 36
> y = log(x/2) - 3                # se alguma vez precisar disso
> y
 [1]  0.51154544  0.38439026  0.27714473  ....
> round(y, 3)
 [1]  0.512  0.384  0.277  .....
 > sum( log(x) + x^2 )          # e' claro que queremos calcular isto com os gols
[1] 39226.67
> sum(x > 50)                      # operacao numerica com vetor logico
[1] 5
> c(x, c(20, 39, 45))              # acrescentando gols de 3 times adicionais
 [1] 67 59 53 49 ....  31 28 20 39 45
> x = c(x, c(20, 39, 45))                      # salvando em x
> x[ (length(x) - 20) : (min(x) - 12) ]          # funcoes dentro de indexadores
[1] 53 49 51 61 36 43
> cumsum(x)                     # soma acumulada de gols, na ordem do vetor x
 [1]  67 126 179 228 279 340 376
 > rev(cumsum(x))            # revertendo a soma acumulada de gols
 [1] 964 919 880 860 832
```

Vetores em R - repetições

```
> 1:9
[1] 1 2 3 4 5 6 7 8 9
> seq(0, 1, by=0.1)
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
> seq(0, 1, length=11)
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

> rep(-1, 5)
[1] -1 -1 -1 -1 -1

> rep(c(-1, 0), 5)
 [1] -1  0 -1  0 -1  0 -1  0 -1  0

> rep(5, c(-1, 0))
Erro em rep(5, c(-1, 0)) : argumento 'times' invalido

> rep(c(-1, 0), c(5, 3))
[1] -1 -1 -1 -1 -1  0  0  0

> rep(-1:2, rep(3, 4))
 [1] -1 -1 -1  0  0  0  1  1  1  2  2  2
```

Dados categóricos em R

```
> y = c("M", "F", "M", "M", "M", "F","M","F")       # vetor de caracteres
> sum(y)                # caracteres nao podem ser somados
Erro em sum(y) : 'type' invalido (character) do argumento
> table(y)               # eles podem ser tabelados
y
F M
3 5
> # valor de retorno de table() e' um vetor numerico de dimensao = numero de string
> yc = table(y)
> yc[2]    # o vetor tem nomes (os strings distintos) para as suas entradas
M
5
> yc/length(y)   # podemos operar numericamente
y
    F     M
0.375 0.625
```

Visualizando dados categóricos em R

```
> y = c("M", "F", "M", "M", "M", "F","M","F")    # vetor de caracteres
> par(mfrow=c(1,3))  # janela grafica dividida em 1 x 3 celulas
> barplot(table(y), main="frequencias")     # grafico das contagens da tabela
> barplot(table(y)/length(y), main="proporcoes")    # plotando as proporcoes
> x = c(2, 4, 1, 0, 9, 2, 2, 1)
> barplot(x, main="barplot de vetor")
```

Dados categóricos em R

- A questão não é apenas ser um vetor de caracteres.
- Vetores com números podem representar categorias.

Dados categóricos em R

- A questão não é apenas ser um vetor de caracteres.
- Vetores com números podem representar categorias.
- Por exemplo: $0 = $ "Masculino", $1 = $ "Feminino"

Dados categóricos em R

- A questão não é apenas ser um vetor de caracteres.
- Vetores com números podem representar categorias.
- Por exemplo: $0 = $ "Masculino", $1 = $ "Feminino"
- R tem uma classe de objetos para trabalhar com variáveis categóricas: `factor`.

Dados categóricos em R

- A questão não é apenas ser um vetor de caracteres.
- Vetores com números podem representar categorias.
- Por exemplo: $0 = $ "Masculino", $1 = $ "Feminino"
- R tem uma classe de objetos para trabalhar com variáveis categóricas: `factor`.
- R adapta-se automaticamente a resposta dos comandos quando o objeto é um factor.

Dados categóricos em R

- A questão não é apenas ser um vetor de caracteres.
- Vetores com números podem representar categorias.
- Por exemplo: $0 = $ "Masculino", $1 = $ "Feminino"
- R tem uma classe de objetos para trabalhar com variáveis categóricas: `factor`.
- R adapta-se automaticamente a resposta dos comandos quando o objeto é um factor.
- Para criar um factor, use o commando `factor` ou `as.factor`.

Dados categóricos em R

```
> y = c("M", "F", "M", "M", "M", "F","M","F")
> y
[1] "M" "F" "M" "M" "M" "F" "M" "F"
> plot(y)
Erro em plot.window(...) : valores finitos sao necessarios para 'ylim'
Alem disso: Mensagens de aviso perdidas:
1: In xy.coords(x, y, xlabel, ylabel, log) : NAs introduzidos por coercao
2: In min(x) : nenhum argumento nao faltante para min; retornando Inf
3: In max(x) : nenhum argumento nao faltante para max; retornando -Inf
> y = factor(y)
> y
[1] M F M M M F M F
Levels: F M
> # armazena y como 3 1's e 5 2's e associa
> # 1="F" e 2="M" internamente (alfabeticamente)
> # y agora e' uma variavel nominal
```

Dados categóricos em R

```
> plot(y, main="levels default")
> levels(y) = c("M" = "Masculino", "F" = "Feminino")
> y
[1] Feminino  Masculino Feminino  Feminino  Feminino  Masculino Feminino  Masculino
Levels: Masculino Feminino
> plot(y, main = "Escolhendo levels")
> summary(y)
Masculino  Feminino
        3         5
```

Números podem ser categorias

- Não é só a questão de ser caracter para que um vetor seja uma variável categórica.
- Números podem representar categorias.
- Transformando em fator: escolha os níveis do fator.

```
> dor = c(0, 3, 5, 5, 1, 1, 0, 1, 0, 0, 1, 3, 3, 0)   # niveis de dor
> fdor = factor(dor, levels=c(0, 1, 3, 5, 1000))    # transformando num objeto tipo
> levels(fdor) = c("nenhuma", "pouca", "media", "alta", "insuportavel")  # expressa
> fdor    # veja que nao existem caso de dor insuportavel
 [1] nenhuma media   alta    alta    pouca   pouca   nenhuma pouca   nenhuma ...
Levels: nenhuma pouca media alta insuportavel
# armazena fdor de forma que 0 --> 1, 1 --> 2, 3 --> 3, 5 --> 4, 1000 --> 5
# e associa 1="nenhuma", 2="pouca", 3="media", 4="alta" , 5 = "insuportavel"
# fdor agora e' um fator com estes niveis
```

plot e fatores

```
> par(mfrow=c(1,2))
> barplot(fdor)
Erro em barplot.default(fdor) : 'height' deve ser um vetor ou uma matriz
> barplot(table(fdor), main="barplot(table(fdor))")
> plot(fdor, main="plot(fdor)")   # comando generico "plot" responde com "barplot" c
```

Tipos de objetos em R

- Tipos de dados: vetores, matrizes, data-frames, listas, funções.

Tipos de objetos em R

- Tipos de dados: vetores, matrizes, data-frames, listas, funções.
- scalars

```
> x = -1.3
> x
[1] -1.3
> x = 2
> x
[1] 2
> x= pi
> x
[1] 3.141593
> x = "Pedro Paulo"
> x
[1] "Pedro Paulo"
```

Tipos de objetos em R: vetores

- Vetores podem ser: numerical, logical, character

Tipos de objetos em R: vetores

- Vetores podem ser: numerical, logical, character
- numerical

```
> x = c(1, 4, -1, 4)
> x
[1]  1  4 -1  4
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
```

Tipos de objetos em R: vetores

- Vetores podem ser: numerical, logical, character
- numerical

```
> x = c(1, 4, -1, 4)
> x
[1]  1  4 -1  4
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
```

- logical

```
> x = c(T, T, F, T)
> x
[1]  TRUE  TRUE FALSE  TRUE
> 1:6 > 2
[1] FALSE FALSE  TRUE  TRUE  TRUE  TRUE
> sum(1:6 > 2)    # transforma em numerico: T=1 e F=0
[1] 4
```

Tipos de objetos em R: vetores

- O último tipo de dado num vetor é aquele composto de caracteres

```
> x = c("Pedro Paulo", "Pedro", "P")
> x
[1] "Pedro Paulo" "Pedro"       "P"
> letters[1:6]
[1] "a" "b" "c" "d" "e" "f"
```

Tipos de objetos em R: vetores

- O último tipo de dado num vetor é aquele composto de caracteres

```
> x = c("Pedro Paulo", "Pedro", "P")
> x
[1] "Pedro Paulo" "Pedro"       "P"
> letters[1:6]
[1] "a" "b" "c" "d" "e" "f"
```

- Uma função muito útil ao lidar com caracteres é paste:

```
> x = c("Pedro", "Paulo", "Pedro", "Manoel")
> paste(x, 1:4)
[1] "Pedro 1"  "Paulo 2"  "Pedro 3"  "Manoel 4"
> x = c("Pedro", "Paulo", "Pedro", "Manoel")
> paste(x, 1:4)
[1] "Pedro 1"  "Paulo 2"  "Pedro 3"  "Manoel 4"
> paste(x, 1:4, sep = "")
[1] "Pedro1"  "Paulo2"  "Pedro3"  "Manoel4"
> rep(paste("T", 1:3, sep = ""), c(4, 4, 3))
 [1] "T1" "T1" "T1" "T1" "T2" "T2" "T2" "T2" "T3" "T3" "T3"
```

Matrizes

- matrizes: são dados tabulares de um único tipo em toda a matriz: ou numérico, ou lógico, ou caracter.

Matrizes

- matrizes: são dados tabulares de um único tipo em toda a matriz: ou numérico, ou lógico, ou caracter.

```
> x = matrix(1:6, ncol=3, byrow=T)
> x
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
> matrix(letters[1:6], ncol=3, byrow=T)
     [,1] [,2] [,3]
[1,] "a"  "b"  "c"
[2,] "d"  "e"  "f"
> cbind(1:3, 10:12)
     [,1] [,2]
[1,]    1   10
[2,]    2   11
[3,]    3   12
> cbind(1:3, c("a", "b", "c"))
     [,1] [,2]
[1,] "1"  "a"
[2,] "2"  "b"
[3,] "3"  "c"
```

Matrizes: operador [..]

```
> x = matrix(1:12, ncol =4, byrow=T)
> x
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
> x[2, 4]   # elemento (2,4)
[1] 8
> x[ , 1:2]  # selecionado as duas 1as colunas
     [,1] [,2]
[1,]    1    2
[2,]    5    6
[3,]    9   10
> x[2:3 , 3:4]  # sub-matriz bloco
     [,1] [,2]
[1,]    7    8
[2,]   11   12
> x[, c(1,3)]  # selecionado colunas 1 e 3
     [,1] [,2]
[1,]    1    3
[2,]    5    7
[3,]    9   11
```

Operações matriciais

```
> x = matrix(1:4, ncol=2, byrow=T)
> x + t(x) # x + sua transposta
     [,1] [,2]
[1,]    2    5
[2,]    5    8
> x/x  # operacao elemento a elemento
     [,1] [,2]
[1,]    1    1
[2,]    1    1
> x %* % t(x) # multiplicacao matricial
     [,1] [,2]
[1,]    5   11
[2,]   11   25
> solve(x) # inversa de x
     [,1] [,2]
[1,] -2.0  1.0
[2,]  1.5 -0.5
```

Decomposições matriciais

```
> eigen(x)        # autovalores e autovetores, saida e' lista com 2 elementos
$values           # 1o elemento e' um vetor com os dois autovalores
[1]  5.3722813 -0.3722813

$vectors          # 2o. elemento e' uma matriz onde cada coluna e' um autovetor
           [,1]       [,2]
[1,] -0.4159736 -0.8245648
[2,] -0.9093767  0.5657675

> svd(x)        # decomposicao do valor singular, saida e' lista
$d
[1] 5.4649857 0.3659662

$u
           [,1]       [,2]
[1,] -0.4045536 -0.9145143
[2,] -0.9145143  0.4045536

$v
           [,1]       [,2]
[1,] -0.5760484  0.8174156
[2,] -0.8174156 -0.5760484
```

Dataframes

- data frames: são dados tabulares, mas as colunas podem ter tipos diferentes.

Dataframes

- data frames: são dados tabulares, mas as colunas podem ter tipos diferentes.

```
> x = c(2, 3, 5)
> y = c("a", "bb", "ccc")
> z = c(TRUE, FALSE, TRUE)
> df = data.frame(x, y, z)
> df
  x   y     z
1 2   a  TRUE
2 3  bb FALSE
3 5 ccc  TRUE

> df[1:2, 2:3]   # operador [...] como em matriz
   y     z
1  a  TRUE
2 bb FALSE

> df$x   # acessando colunas de data frames
[1] 2 3 5
> df$x[1:2]  # colunas sao como vetores
[1] 2 3
```

Listas

- estruturas genéricas para coletar objetos diversos num único objeto.

Listas

- estruturas genéricas para coletar objetos diversos num único objeto.

```
> x = 1:10
> y = c("a", "b", "c")
> z = matrix(1:4, ncol=2)
> w = list(x, y, z)
> w
[[1]]
 [1]  1  2  3  4  5  6  7  8  9 10

[[2]]
[1] "a" "b" "c"

[[3]]
     [,1] [,2]
[1,]    1    3
[2,]    2    4

> w[[3]]        # acessando 3o elemento da lista
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

Listas

- Diferenciar [...] e [[...]] em listas.

Listas

- Diferenciar [...] e [[...]] em listas.
- [[...]] retorna um ELEMENTO da lista: um vetor, uma matriz etc.

Listas

- Diferenciar [...] e [[...]] em listas.
- [[...]] retorna um ELEMENTO da lista: um vetor, uma matriz etc.
- [...] retorna uma sub-lista da lista original.

Listas

- Diferenciar [...] e [[...]] em listas.
- [[...]] retorna um ELEMENTO da lista: um vetor, uma matriz etc.
- [...] retorna uma sub-lista da lista original.

```
> w[c(1, 3)]          # sub-lista contendo apenas o 1o e 3o elementos de w
[[1]]
 [1]  1  2  3  4  5  6  7  8  9 10

[[2]]
     [,1] [,2]
[1,]    1    3
[2,]    2    4

> is.list(w[c(1, 3)])   # testa se e' uma lista
[1] TRUE

> w[[2]]      # retorna o elemento 2 da lista
[1] "a" "b" "c"

> w[[c(2,3)]]   # retorna o 3o. elemento  do elemento  2 da lista w
[1] "c"
```

Funções

- Linguagem R permite extensões com a criação de funções.

Funções

- Linguagem R permite extensões com a criação de funções.
- Estrutura geral:

Funções

- Linguagem R permite extensões com a criação de funções.
- Estrutura geral:

```
myfun = function(arg1, arg2,...)
{
  ....corpo da funcao...
  return(x)  # x e' qualquer objeto mas, em geral, e' uma lista
}
```

- Possui as estruturas de controle usuais: `for`, `while`, `if`, `if else`.
- Permite também chamar funções em C, C++, FORTRAN, java etc.

Funções

```
myfun <- function(x1, x2) {
  pint = sum(x1 * x2) # produto interno dos vetores
  s1 = sqrt(sum(x1*x1)) # comprimento do vetor 1
  s2 = sqrt(sum(x2*x2)) # comprimento do vetor 2
  z = pint/(s1*s2)
  x  = list(prod.int = pint, coseno = z, dados = cbind(x1, x2))
  return(x)
}
myfun # imprime na tela a definicao da funcao
myfun(x1=c(1,2,3), x2=c(3, 4, 7)) # aplica myfun a c(1,2,3) e c(3, 4, 7)
$prod.int
[1] 32

$coseno
[1] 0.9941916

$dados
     x1 x2
[1,]  1  3
[2,]  2  4
[3,]  3  7
```

Vetorizar sempre

- Vetorizar significa transformar loops em operações vetoriais.
- Código fica muito mais eficiente.

```
> x = runif(100000)  # 100 mil numeros aleatorios em x
> y = runif(100000)  # idem em y
> z = numeric()        # criando objeto numerico z

> start <- Sys.time()
> for(i in 1:100000){ z[i] = x[i] + y[i] }
> end <- Sys.time()
> end - start
Time difference of 23.09923 secs

> start <- Sys.time()
> z = x + y
> end <- Sys.time()
> end - start
Time difference of 1.352752 secs
```

Vetorizar sempre

- Este exemplo vem de `http://www.r-bloggers.com/how-to-use-vectorization-to-streamline-simulations/`
- Tarefa: escreva programa que jogue moeda n vezes.
- A cada 100 lançamentos, imprima a proporção de caras menos $1/2$.
- Imprima também o número de caras menos o número de lançamentos dividido por 2.
- Vamos escrever um programa em R com "sabor c"(cheio de loops, sem vetorizar).

Vetorizar sempre

```
coin_toss1 = function(n){
  result = c()
  for(i in c(1:n)) {
    if(i == 1){
      ## the optional outputs are 0 and 1.  I am assigning 1 to heads
      tosses = sample(c(0,1),1)
    }
    else{
      ## creating a vector that has history of all tosses
      tosses = c(tosses,sample(c(0,1),1))
    }
    ## when we reach a toss number that a multiple of 100 we output the status
    if(i %% 100 == 0){
      ## output the percent of heads away from 50%
      percent = (sum(tosses) / length(tosses)) - 0.5
      ## output the number of heads away from half of all tosses
      number = sum(tosses) - (length(tosses) / 2)
      result = rbind(result, c(percent, number))
    }
  }
  result
}
```

Vetorizar sempre

Outro código, sabor R:

```
coin_toss2 = function(n, step=100) {
  # Record number of heads at each step
  tosses = cumsum(sample(c(0,1), n, replace=TRUE))
  # Define steps for summaries
  steps = seq(step,n, by=step)
  # Compute summaries
  cbind(tosses[steps] / steps - .5, tosses[steps] - steps/2)
}
```

Vetorizar sempre

```
> start <- Sys.time()
> x = coin_toss1(100000)
> end <- Sys.time()
> end - start
Time difference of 24.23292 secs
> start <- Sys.time()
> x = coin_toss2(100000)
> end <- Sys.time()
> end - start
Time difference of 1.098358 secs
}
```

- Ver capítulos 3 e 4 do livro *The R Inferno*, free pdf em `http://www.burns-stat.com/documents/books/the-r-inferno/`

Vetorizar sempre

```
> start <- Sys.time()
> x = coin_toss1(100000)
> end <- Sys.time()
> end - start
Time difference of 24.23292 secs
> start <- Sys.time()
> x = coin_toss2(100000)
> end <- Sys.time()
> end - start
Time difference of 1.098358 secs
}
```

- Ver capítulos 3 e 4 do livro *The R Inferno*, free pdf em `http://www.burns-stat.com/documents/books/the-r-inferno/`
- Abstract: If you are using R and you think you're in hell, this is a map for you.

Vetorizar sempre

```
> start <- Sys.time()
> x = coin_toss1(100000)
> end <- Sys.time()
> end - start
Time difference of 24.23292 secs
> start <- Sys.time()
> x = coin_toss2(100000)
> end <- Sys.time()
> end - start
Time difference of 1.098358 secs
}
```

- Ver capítulos 3 e 4 do livro *The R Inferno*, free pdf em `http://www.burns-stat.com/documents/books/the-r-inferno/`
- Abstract: If you are using R and you think you're in hell, this is a map for you.
- Even if it doesn't help you with your problem, it might amuse you (and hence distract you from your sorrow).

Functional Programming: The Reduce function

- Nem sempre conseguimos vetorizar.

Functional Programming: The Reduce function

- Nem sempre conseguimos vetorizar.
- Por exemplo, quando a iteração corrente depende de resultados da iteração anterior.

Functional Programming: The Reduce function

- Nem sempre conseguimos vetorizar.
- Por exemplo, quando a iteração corrente depende de resultados da iteração anterior.
- Alguns casos podem ser resolvidos com `filter` e `Reduce`.

Functional Programming: The Reduce function

- Nem sempre conseguimos vetorizar.
- Por exemplo, quando a iteração corrente depende de resultados da iteração anterior.
- Alguns casos podem ser resolvidos com `filter` e `Reduce`.
- `Reduce(f, x)` aplica `f` sucessivamente ao resultado anterior.

Functional Programming: The Reduce function

- Nem sempre conseguimos vetorizar.
- Por exemplo, quando a iteração corrente depende de resultados da iteração anterior.
- Alguns casos podem ser resolvidos com `filter` e `Reduce`.
- `Reduce(f, x)` aplica `f` sucessivamente ao resultado anterior.
- Suponha que $x = (x_1, x_2, x_3, \ldots)$

Functional Programming: The Reduce function

- Nem sempre conseguimos vetorizar.
- Por exemplo, quando a iteração corrente depende de resultados da iteração anterior.
- Alguns casos podem ser resolvidos com `filter` e `Reduce`.
- `Reduce(f, x)` aplica `f` sucessivamente ao resultado anterior.
- Suponha que $x = (x_1, x_2, x_3, \ldots)$
- e $f(a, b)$ uma função de dois argumentos.

Functional Programming: The Reduce function

- Nem sempre conseguimos vetorizar.
- Por exemplo, quando a iteração corrente depende de resultados da iteração anterior.
- Alguns casos podem ser resolvidos com `filter` e `Reduce`.
- `Reduce(f, x)` aplica `f` sucessivamente ao resultado anterior.
- Suponha que $x = (x_1, x_2, x_3, \ldots)$
- e $f(a, b)$ uma função de dois argumentos.
- `Reduce(f, x)` aplica `f` sucessivamente assim:

Functional Programming: The Reduce function

- Nem sempre conseguimos vetorizar.
- Por exemplo, quando a iteração corrente depende de resultados da iteração anterior.
- Alguns casos podem ser resolvidos com `filter` e `Reduce`.
- `Reduce(f, x)` aplica `f` sucessivamente ao resultado anterior.
- Suponha que $x = (x_1, x_2, x_3, \ldots)$
- e $f(a, b)$ uma função de dois argumentos.
- `Reduce(f, x)` aplica `f` sucessivamente assim:
  - $f(x_1, x_2), x_3, x_4, x_5, \ldots$

Functional Programming: The Reduce function

- Nem sempre conseguimos vetorizar.
- Por exemplo, quando a iteração corrente depende de resultados da iteração anterior.
- Alguns casos podem ser resolvidos com `filter` e `Reduce`.
- `Reduce(f, x)` aplica `f` sucessivamente ao resultado anterior.
- Suponha que $x = (x_1, x_2, x_3, \ldots)$
- e $f(a, b)$ uma função de dois argumentos.
- `Reduce(f, x)` aplica `f` sucessivamente assim:
  - $f(x_1, x_2), x_3, x_4, x_5, \ldots$
  - $f(f(x_1, x_2), x_3), x_4, x_5, \ldots$

Functional Programming: The Reduce function

- Nem sempre conseguimos vetorizar.
- Por exemplo, quando a iteração corrente depende de resultados da iteração anterior.
- Alguns casos podem ser resolvidos com `filter` e `Reduce`.
- `Reduce(f, x)` aplica `f` sucessivamente ao resultado anterior.
- Suponha que $x = (x_1, x_2, x_3, \ldots)$
- e $f(a, b)$ uma função de dois argumentos.
- `Reduce(f, x)` aplica `f` sucessivamente assim:
  - $f(x_1, x_2), x_3, x_4, x_5, \ldots$
  - $f(f(x_1, x_2), x_3), x_4, x_5, \ldots$
  - $f(f(f(x_1, x_2), x_3), x_4), x_5 \ldots$

Functional Programming: The Reduce function

- Nem sempre conseguimos vetorizar.
- Por exemplo, quando a iteração corrente depende de resultados da iteração anterior.
- Alguns casos podem ser resolvidos com `filter` e `Reduce`.
- `Reduce(f, x)` aplica `f` sucessivamente ao resultado anterior.
- Suponha que $x = (x_1, x_2, x_3, \ldots)$
- e $f(a, b)$ uma função de dois argumentos.
- `Reduce(f, x)` aplica `f` sucessivamente assim:
  - $f(x_1, x_2), x_3, x_4, x_5, \ldots$
  - $f(f(x_1, x_2), x_3), x_4, x_5, \ldots$
  - $f(f(f(x_1, x_2), x_3), x_4), x_5 \ldots$

Functional Programming: The Reduce function

- Por default, `Reduce(f, x)` retorna o valor da última avaliação de f.

Functional Programming: The Reduce function

- Por default, `Reduce(f, x)` retorna o valor da última avaliação de f.
- Usando `Reduce(f,x,accumulate=T)` retornamos todos os valores intermediários calculados.

Functional Programming: The Reduce function

- Por default, `Reduce(f, x)` retorna o valor da última avaliação de f.
- Usando `Reduce(f,x,accumulate=T)` retornamos todos os valores intermediários calculados.
- útil para cálculos iterativos que não são facilmente vetorizados em R.

Functional Programming: The Reduce function

- Por default, `Reduce(f, x)` retorna o valor da última avaliação de f.
- Usando `Reduce(f,x,accumulate=T)` retornamos todos os valores intermediários calculados.
- útil para cálculos iterativos que não são facilmente vetorizados em R.
- Toy example: soma obtida via loop:

```
s = x[1] + x[2]
for(i in 1:length(s) ) s = s + x[i]
for(iÊinÊ3:length(s))ÊsÊ<?ÊsÊ+Êx[i]
```

- Com `Reduce`:

```
f = function(a,b) a+ b
 s = Reduce(f, x)
```

- Produto acumulado:

```
f = function(a,b) a * b
 s = Reduce(f, x, accumulate=TRUE)
```

Exemplo mais interessante com Reduce

- Exponentially smoothed moving average: *ewma*

Exemplo mais interessante com Reduce

- Exponentially smoothed moving average: *ewma*
- y_t: série temporal $y_1, y_2, \ldots$ num vetor `y`

Exemplo mais interessante com Reduce

- Exponentially smoothed moving average: *ewma*
- y_t: série temporal $y_1, y_2, \ldots$ num vetor `y`
- Objetivo: obter uma nova série s_t mais suavizada.

Exemplo mais interessante com Reduce

- Exponentially smoothed moving average: *ewma*
- y_t: série temporal $y_1, y_2, \ldots$ num vetor y
- Objetivo: obter uma nova série s_t mais suavizada.
- Tome $\lambda \in (0, 1)$.

Exemplo mais interessante com Reduce

- Exponentially smoothed moving average: *ewma*
- y_t: série temporal $y_1, y_2, \ldots$ num vetor y
- Objetivo: obter uma nova série s_t mais suavizada.
- Tome $\lambda \in (0, 1)$.
- $s_t = \lambda y_t + (1 - \lambda) s_{t-1}$

Exemplo mais interessante com Reduce

- Exponentially smoothed moving average: *ewma*
- y_t: série temporal $y_1, y_2, \ldots$ num vetor y
- Objetivo: obter uma nova série s_t mais suavizada.
- Tome $\lambda \in (0, 1)$.
- $s_t = \lambda y_t + (1 - \lambda) s_{t-1}$
- Condição inicial: $s_1 = y_1$.
- $\lambda \approx 1$ praticamente reproduz a sërie original.
- $\lambda \approx 0.20$ gera série suavizada.

Reduce

```
x = seq(0, 2*pi, length=200); y = sin(x) + rnorm(200, 0, 0.4); plot(x, y, type="l")

lambda = 0.15
myfun = function(prv,nxt){ lambda*nxt + (1-lambda)*prv }
ewma = Reduce(myfun, y, accumulate=T)

lines(x, ewma, lwd=2, col="red")
```

Se precisa usar loops

- Se precisar fazer um loop, tente alguns truques:

Se precisa usar loops

- Se precisar fazer um loop, tente alguns truques:
- Ponha o máximo de cálculos fora dos loops
- Minimize o número de iterações.
- Exemplo: obter a soma de cada coluna de uma matriz: loop sobre as colunas apenas

```
sumxcol = numeric(ncol(x))
for(i in 1:ncol(x) ) sumxcol[i] = sum(x[,i])
```

- Operar repetidamente sobre as colunas ou linhas de uma matriz é uma operação tão comum que tem um comando especial em R: `apply`

Se precisa usar loops

- commando `apply(mat, index, FUN)`

Se precisa usar loops

- commando `apply(mat, index, FUN)`
- Aplica a função `FUN` na matriz `mat` ao longo das suas linhas ou colunas:

Se precisa usar loops

- commando `apply(mat, index, FUN)`
- Aplica a função `FUN` na matriz `mat` ao longo das suas linhas ou colunas:
- se `index=1`, aplica `FUN` em cada linha da matriz `mat`

Se precisa usar loops

- commando `apply(mat, index, FUN)`
- Aplica a função `FUN` na matriz `mat` ao longo das suas linhas ou colunas:
- se `index=1`, aplica `FUN` em cada linha da matriz `mat`
- colunas usa `index=2`. Exemplo:

Se precisa usar loops

- commando `apply(mat, index, FUN)`
- Aplica a função `FUN` na matriz `mat` ao longo das suas linhas ou colunas:
- se `index=1`, aplica `FUN` em cada linha da matriz `mat`
- colunas usa `index=2`. Exemplo:

```
> mat = matrix(1:12, ncol =4)
> mat
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> apply(mat, 2, sum)  # valor de retorno e' um vetor de dimensao = no de cols
[1]  6 15 24 33
```

- `lapply` opera em listas ao invés de matrizes.
- Ver também `tapply`, `mapply`, `rapply`, `eapply`.

Material inicial sobre R

- Comece lendo na wikipedia:
  `http://en.wikipedia.org/wiki/R_(programming_language)`

Material inicial sobre R

- Comece lendo na wikipedia:
  `http://en.wikipedia.org/wiki/R_(programming_language)`
- Download R para Linux, Mac, Windows em
  `http://cran.r-project.org/`

Material inicial sobre R

- Comece lendo na wikipedia: `http://en.wikipedia.org/wiki/R_(programming_language)`
- Download R para Linux, Mac, Windows em `http://cran.r-project.org/`
- Rstudio, free IDE para R: `http://www.rstudio.com/`

Material inicial sobre R

- Comece lendo na wikipedia: `http://en.wikipedia.org/wiki/R_(programming_language)`
- Download R para Linux, Mac, Windows em `http://cran.r-project.org/`
- Rstudio, free IDE para R: `http://www.rstudio.com/`
- Muitos tutoriais disponíveis no CRAN e na web. Escolha o seu sabor...

Material inicial sobre R

- Comece lendo na wikipedia: `http://en.wikipedia.org/wiki/R_(programming_language)`
- Download R para Linux, Mac, Windows em `http://cran.r-project.org/`
- Rstudio, free IDE para R: `http://www.rstudio.com/`
- Muitos tutoriais disponíveis no CRAN e na web. Escolha o seu sabor...
- R-tutorial: excelente - `http://www.r-tutor.com/`

Material inicial sobre R

- Comece lendo na wikipedia: http://en.wikipedia.org/wiki/R_(programming_language)
- Download R para Linux, Mac, Windows em http://cran.r-project.org/
- Rstudio, free IDE para R: http://www.rstudio.com/
- Muitos tutoriais disponíveis no CRAN e na web. Escolha o seu sabor...
- R-tutorial: excelente - http://www.r-tutor.com/
- Outro: http://mran.revolutionanalytics.com/documents/getting-started/
- Em português: http://www.leg.ufpr.br/~paulojus/embrapa/Rembrapa/
- Quick-R: http://www.statmethods.net/
- Curso: http://www.pitt.edu/~njc23/ (apenas os slides, excelente)
- Lista brasileira de discussão do R: http://www.leg.ufpr.br/doku.php/software:rbr

Material mais avançado

- An Introduction to R: `http://cran.r-project.org/doc/manuals/r-release/R-intro.html`

Material mais avançado

- An Introduction to R: `http://cran.r-project.org/doc/manuals/r-release/R-intro.html`
- Livro The Art of R Programming, de Norman Matloff

Material mais avançado

- An Introduction to R: `http://cran.r-project.org/doc/manuals/r-release/R-intro.html`
- Livro The Art of R Programming, de Norman Matloff
- Disponível no site `https://www.safaribooksonline.com/library/view/the-art-of/9781593273842/`

Material mais avançado

- An Introduction to R: `http://cran.r-project.org/doc/manuals/r-release/R-intro.html`
- Livro The Art of R Programming, de Norman Matloff
- Disponível no site `https://www.safaribooksonline.com/library/view/the-art-of/9781593273842/`

- Livro Advanced R, de Hadley Wickham

Material mais avançado

- An Introduction to R: `http://cran.r-project.org/doc/manuals/r-release/R-intro.html`
- Livro The Art of R Programming, de Norman Matloff
- Disponível no site `https://www.safaribooksonline.com/library/view/the-art-of/9781593273842/`

- Livro Advanced R, de Hadley Wickham
- R-bloggers: `http://www.r-bloggers.com/`

Material mais avançado

- Livro *The R Inferno*, free pdf: `http://cran.r-project.org/doc/manuals/r-release/R-intro.html`

Material mais avançado

- Livro *The R Inferno*, free pdf: `http://cran.r-project.org/doc/manuals/r-release/R-intro.html`
- Free pdf em `http://www.burns-stat.com/documents/books/the-r-inferno/`
- Tem também o tutorial *Impatient R* em `http://www.burns-stat.com/documents/tutorials/impatient-r/`

Material mais avançado

- Livro *The R Inferno*, free pdf: `http://cran.r-project.org/doc/manuals/r-release/R-intro.html`
- Free pdf em `http://www.burns-stat.com/documents/books/the-r-inferno/`
- Tem também o tutorial *Impatient R* em `http://www.burns-stat.com/documents/tutorials/impatient-r/`

Cursos online gratuitos

- Lista de cursos online de estatística:
  `https://www.class-central.com/search?q=statistics`

Cursos online gratuitos

- Lista de cursos online de estatística: `https://www.class-central.com/search?q=statistics`
- Coursera: Data Analysis and Statistical Inference, começando em Março, 2015

Cursos online gratuitos

- Lista de cursos online de estatística: `https://www.class-central.com/search?q=statistics`
- Coursera: Data Analysis and Statistical Inference, começando em Março, 2015
- `https://www.coursera.org/course/statistics`

Cursos online gratuitos

- Lista de cursos online de estatística: `https://www.class-central.com/search?q=statistics`
- Coursera: Data Analysis and Statistical Inference, começando em Março, 2015
- `https://www.coursera.org/course/statistics`
- *Specialization in Data Science* em Coursera: nove cursos. FREE but tricky to get

Cursos online gratuitos

- Lista de cursos online de estatística: `https://www.class-central.com/search?q=statistics`
- Coursera: Data Analysis and Statistical Inference, começando em Março, 2015
- `https://www.coursera.org/course/statistics`
- *Specialization in Data Science* em Coursera: nove cursos. FREE but tricky to get
- Um dos nove cursos: R Programming `https://www.coursera.org/course/rprog`

Cursos online gratuitos

- Lista de cursos online de estatística: `https://www.class-central.com/search?q=statistics`
- Coursera: Data Analysis and Statistical Inference, começando em Março, 2015
- `https://www.coursera.org/course/statistics`
- *Specialization in Data Science* em Coursera: nove cursos. FREE but tricky to get
- Um dos nove cursos: R Programming `https://www.coursera.org/course/rprog`
- O campeão dos MOOCs: *Machine Learning* , com Andrew Ng,
- Na plataforma coursera: `https://www.coursera.org/course/ml`
- *Statistical Learning* : Stanford professors Trevor Hastie and Rob Tibshirani

  `https://www.youtube.com/channel/UC4OWDcPB1peiBXDfCSZ3h-w/feed`