MAC0113 - Introdução à Computação para Ciências Humanas

Aula 17 - 1

Sejam bem-vindas, sejam bem-vindos!

Entre no link https://app.sli.do/event/4wqoxqdw ou e responda a primeira pergunta da aula.

R. Hirata Jr.

MAC0113 - Introdução à Computação para Ciências Humanas

Aula 17 - 2

Sejam bem-vindas, sejam bem-vindos!

Entre no link https://app.sli.do/event/iaviyhn4 ou e responda a primeira pergunta da aula.

R. Hirata Jr.

Objetivos de hoje

- Ao final da aula de hoje você deve saber:
  - dataframes
  - escrita de arquivo
  - alguns conceitos e comandos úteis

Pequena recordação da aula passada

ASCii e UTF-8

Existem várias codificações (*enconding*) de texto. As mais comuns são

- *ASCii*: que representa apenas letras do alfabeto latino sem acento, números e pontuações usando no máximo 8 bits

  https://en.wikipedia.org/wiki/ASCII

- *UTF-8*: que representa letras do alfabeto latino e de outros alfabetos, incluindo acentos e diacríticos (p.ex., ç), números, pontuações e outros símbolos (flechas, operadores matemáticos etc), usando uma quantidade de bits variável.

  https://en.wikipedia.org/wiki/UTF-8

O padrão nas versões recentes do R é usar UTF-8.

Codificação de caracteres

Os caracteres no computador são representados por números inteiros positivos. A codificação (encoding) de um arquivo é a lista de códigos usados para representar caracteres. A função utf8ToInt devolve o código de um caractere.

```
V <- c("A","B","C","D","E","F","R","S","T","U","V","X","Y","Z")
i <- 1
tamanhoV <- length(V)
while (i <= tamanhoV) {
  cat(utf8ToInt(V[i])," ")
  i <- i+1
}
cat("\n")
```

Codificação de caracteres

Os caracteres no computador são representados por números inteiros positivos. A codificação (encoding) de um arquivo é a lista de códigos usados para representar caracteres. A função utf8ToInt devolve o código de um caractere.

```
V <- c("a","b","c","d","e","f","r","s","t","u","v","x","y","z")


for (letra in V) {
  cat(utf8ToInt(letra)," ")

}
cat("\n")
```

Codificação de caracteres

Os caracteres no computador são representados por números inteiros positivos. A codificação (encoding) de um arquivo é a lista de códigos usados para representar caracteres. A função utf8ToInt devolve o código de um caractere.

```
V <- c("á","à","ã","â","ä","å")


for (letra in V) {
  cat(utf8ToInt(letra)," ")

}
cat("\n")
```

Matrizes

Uma matriz é um arranjo multidimensional de objetos do mesmo tipo

Por exemplo, uma imagem pode ser representada por uma matriz de valores inteiros:

0	0	1	9
3	1	2	7
1	1	4	3
3	3	3	5
0	2	1	1

Matrizes

Uma outra maneira é usando o operador matrix, com parâmetros ncol e nrow
Por exemplo, para a matriz abaixo, poderíamos fazer:

```
> M <-  matrix(c(0,3,1,3,0,0,1,1,3,2,1,2,4,3,1,9,7,3,5,1),ncol=4,nrow=5)
> M
     [,1] [,2] [,3] [,4]
[1,]    0    0    1    9
[2,]    3    1    2    7
[3,]    1    1    4    3
[4,]    3    3    3    5
[5,]    0    2    1    1
```

0	0	1	9
3	1	2	7
1	1	4	3
3	3	3	5
0	2	1	1

Matrizes

Uma vez criada, podemos acessar, ou acessar e modificar os valores da matriz usando um, ou mais índices. Por exemplo:

```
> M[1,1]
[1] 0
> M[1,2]
[1] 0
> M[2,1]
[1] 3
> M[5,4]
[1] 1
> M[2,2]
[1] 1
> M[4,3]
[1] 3
```

```
> M
     [,1] [,2] [,3] [,4]
[1,]    0    0    1    9
[2,]    3    1    2    7
[3,]    1    1    4    3
[4,]    3    3    3    5
[5,]    0    2    1    1
```

Matrizes

Uma vez criada, podemos acessar, ou acessar e modificar os valores da matriz usando dois, ou mais índices (casos com mais de 2 dimensões). Por exemplo:

```
> M[3,2] <- 50
> M
     [,1] [,2] [,3] [,4]
[1,]    0    0    1    9
[2,]    3    1    2    7
[3,]    1   50    4    3
[4,]    3    3    3    5
[5,]    0    2    1    1
```

Antes

```
> M
     [,1] [,2] [,3] [,4]
[1,]    0    0    1    9
[2,]    3    1    2    7
[3,]    1    1    4    3
[4,]    3    3    3    5
[5,]    0    2    1    1
```

Matrizes

Podemos acessar linhas, ou colunas inteiras, usando o fatiamento. Por exemplo:

```
# linha 3
> W[3,]
[1] 1 50 4 3

# coluna 2
> W[,2]
[1]  0  1 50  3  2
```

```
> W
     [,1] [,2] [,3] [,4]
[1,]    0    0    1    9
[2,]    3    1    2    7
[3,]    1   50    4    3
[4,]    3    3    3    5
[5,]    0    2    1    1
```

Matrizes

Podemos acessar mais de uma linha, ou coluna inteiras, usando o fatiamento e o operador c, também. Por exemplo:

```
# linhas 1, 3 e 5
> W[c(1,3,5),]
    [,1] [,2] [,3] [,4]
[1,]    0    0    1    9
[2,]    1   50    4    3
[3,]    0    2    1    1
```

```
> W
     [,1] [,2] [,3] [,4]
[1,]    0    0    1    9
[2,]    3    1    2    7
[3,]    1   50    4    3
[4,]    3    3    3    5
[5,]    0    2    1    1
```

Matrizes

Podemos acessar mais de uma linha, ou coluna inteiras, usando o fatiamento e o operador c, também. Por exemplo:

```
# colunas 2 e 4
> W[,c(2,4)]
     [,1] [,2]
[1,]    0    9
[2,]    1    7
[3,]   50    3
[4,]    3    5
[5,]    2    1
```

```
> W
     [,1] [,2] [,3] [,4]
[1,]    0    0    1    9
[2,]    3    1    2    7
[3,]    1   50    4    3
[4,]    3    3    3    5
[5,]    0    2    1    1
```

Leitura de arquivo

No entanto, algumas vezes é importante usar arquivos e, neste caso, há duas informações muito importantes que você precisa saber sobre eles:

- Nome, com o caminho completo
- Formato: csv (comma separated values)
  tsv (tab separated values)
  etc

Para carregar um arquivo no R, você pode usar as várias versões do read (read.csv, read.csv2, read.table). O que muda nesses comandos são os parâmetros default. Para ler uma tabela com o separador “;”, você pode fazer:

<nome do objeto> = read.csv(<nome do arquivo>,sep=”;”)

Se o separador for um tab, o separador será o “\t”.

Leitura de arquivo

Os comandos read carregam o arquivo e criam um dataframe (uma lista de listas), que é parecido com uma matriz.

Uma vez carregado e atribuído para um objeto do tipo dataframe, você pode começar a usá-lo. Por exemplo:

```
> dados <- read.csv("/home/hirata/MEGA/Disciplinas/MAC0113/EPs/MAC0113-EP1.csv",sep=";")
```

Usando o comando str, você pode ter uma ideia do objeto:

```
> str(dados)
'data.frame':    600 obs. of  3 variables:
$ Nome.do.objeto: chr  "ring light" "garrafa termica" "Lata de Nescau" "vaso" ...
$ circunferência: num  83 21.4 22.9 32.5 18.5 ...
$ diâmetro      : num  26 6.4 7.3 10.2 5.2 8.5 8.3 18 5.4 6 ...
```

INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
IME 50
UNIVERSIDADE DE SÃO PAULO

Pode acordar, agora é para valer!

Sistema de arquivos

O sistema operacional usa um sistema de arquivos, que pode ser acessado pelo R.

getwd( ) - apresenta o diretório atual

```
> getwd()
[1] "/home/hirata/MEGA/Disciplinas/MAC0113/Lab"
```

setwd( ) - muda de diretório; equivale ao cd

```
> setwd("/usr/local/bin")
> getwd()
[1] "/usr/local/bin"
```

list.files( ) - apresenta os arquivos do diretório

```
> list.files()
 [1] "apt"          "gnome-help"   "highlight-mint" "java"             "mint-sha256sum"
 [6] "R"            "Rscript"      "search"         "warsaw"           "yelp"
```

Data-frames

Um data-frame é um arranjo multidimensional de objetos que podem ser de tipos diferentes, mas o número de elementos em cada dimensão tem que ser igual.

Em outras palavras, é como se fosse uma matriz, com objetos de tipos diferentes.

Os objetos podem ser acessados e modificados da mesma forma que uma matriz.

Quando lemos uma planilha csv, ou Excel, o resultado da leitura é um data-frame.

Existem outras estruturas de dados no R, mas as que já vimos são as mais importantes para esta disciplina.

Data-frames

```
# Vamos criar alguns vetores de mesmo tamanho
Frutas <- c("banana","laranja","uva","ameixa","abacaxi")
Quantidade <- c(12,24,2,1,1)
DeveComprar <- c(TRUE,FALSE,TRUE,FALSE,TRUE)
dados <- data.frame(Frutas,Quantidade,DeveComprar)

> str(dados)
'data.frame':    5 obs. of  3 variables:
 $ Frutas       : chr  "banana" "laranja" "uva" "ameixa" ...
 $ Quantidade : num  12 24 2 1 1
 $ DeveComprar: logi  TRUE FALSE TRUE FALSE TRUE

> dim(dados)
[1] 5 3
```

Data-frames

Como vimos, o data-frame é criado com os vetores concatenados por colunas (cbind)

```
> dim(dados)
[1] 5 3
```

Como antes, podemos acessar elementos dos data-frames, extrair partes dos data-frames, ou modificar valores dos data-frames.

```
> dados[4,2]
[1] 1
> dados[4,2] <- 3
> dados[4,2]
[1] 3
```

Data-frames

Para quem quer treinar com data-frames, há vários prontos no R (nem todos são): https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/00Index.html

Um exemplo é mtcars

```
> str(mtcars)
'data.frame':    32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

Data-frames

Outro exemplo:

```
> str(swiss)
'data.frame':    47 obs. of  6 variables:
 $ Fertility      : num  80.2 83.1 92.5 85.8 76.9 76.1 83.8 92.4 82.4 82.9 ...
 $ Agriculture    : num  17 45.1 39.7 36.5 43.5 35.3 70.2 67.8 53.3 45.2 ...
 $ Examination    : int  15 6 5 12 17 9 16 14 12 16 ...
 $ Education      : int  12 9 5 7 15 7 7 8 7 13 ...
 $ Catholic       : num  9.96 84.84 93.4 33.77 5.16 ...
 $ Infant.Mortality: num  22.2 22.2 20.2 20.3 20.6 26.6 23.6 24.9 21 24.4 ...
```

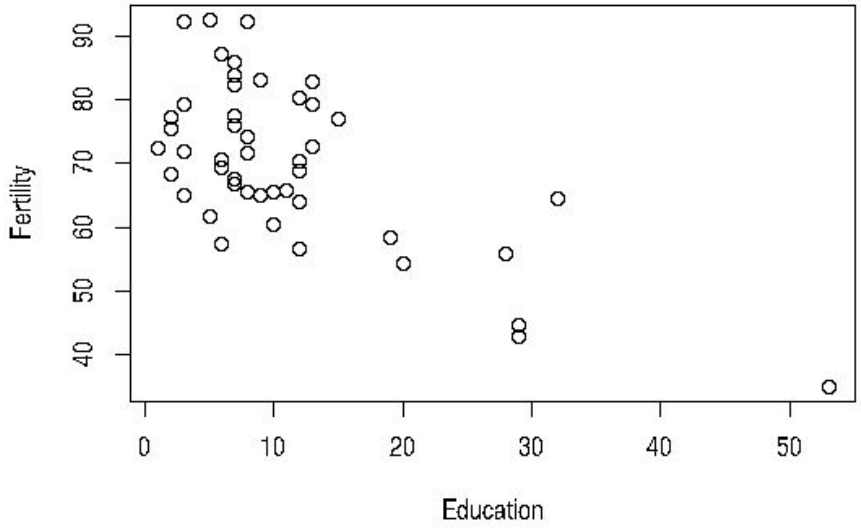
Note que as colunas dos data-frame têm nomes: “Fertility”, “Agriculture”, “Examination”, “Education”, “Catholic” e “Infant.Mortality”

A vantagem disso é que você pode usar os nomes das colunas

Data-frames

Vamos usar os nomes da coluna “Education” e “Fertility” para fazer um gráfico.

```
> plot(swiss[,c("Education","Fertility")])
```



Criar e escrever um arquivo

Normalmente, alguns resultados intermediários, ou finais, merecem serem gravados em arquivos.

No RStudio, nem precisaríamos usar comandos explícitos de escrita pois os objetos podem ser gravados com acionamentos do mouse.

Porém, isso pode dar muito trabalho se o número de arquivos a serem gravados é grande.

Para gravar, em geral o comando é o write, ou o comando write.csv, ou o comando write.table etc.

Criar e escrever um arquivo

```
boats = as.matrix(read.csv("boats.txt",sep="\t",header=FALSE))

# Apresenta a imagem rotacionada de 90 graus à esquerda (coluna 1 na
# parte de baixo da imagem, como uma linha)

image(boats,col  = gray((0:255)/255),asp=1)

# Vamos colocar a imagem de cabeça para baixo.
boatsInv = c()
for (i in 576:1) {
  boatsInv = rbind(boatsInv,boats[i,])
}

image(boatsInv,col  = gray((0:255)/255),asp=1)

write.table(boatsInv,file="boatsInvertido.txt", row.names = FALSE, col.names =
FALSE, sep = "\t")
```

Para a aula de quinta-feira!

Traga _oito_ cartas de baralho numéricas
ou _oito_ pedaços de papéis retangulares,
do tamanho de uma carta numerados


INSTITUTO DE
MATEMÁTICA
E ESTATÍSTICA
IME
50
UNIVERSIDADE
DE SÃO PAULO

Obrigado!