Aula 2.1 - abrindo e salvando dados

Elder Sodre

Monday, February 23, 2015

#### Pasta de trabalho/Working directory

Existe um conceito importante chamado pasta de trabalho (working directory, em inglês) Todos os arquivos que você quer importar para o R deverão estar na pasta de trabalho OU o caminho do arquivo deve estar especificado. Da mesma forma, quando você for exportar um conjunto de dados para fora do R, ele será salvo na pasta de trabalho a não ser que o caminho esteja especificado.

Não sabe qual é a sua pasta de trabalho? Use a função getwd().

getwd() #Esta função não tem argumentos.

## [1] "C:/Users/Elder/AppData/Local/Temp/RtmpWGr2we"

Normalmente, a pasta de trabalho do R é a pasta meus documentos do seu computador.

Você pode alterar o seu working directory com a função setwd().

setwd("C:\\Users\\Elder\\Pictures") #Alterei para a pasta minhas imagens.  
getwd()

## [1] "C:/Users/Elder/Pictures"

Perceba que o "endereço" da pasta de trabalho deve estar entre aspas.

setwd("C:\\Users\\Elder\\Desktop\\Minicurso Elementos de Linguagem R")  
getwd()

## [1] "C:/Users/Elder/Desktop/Minicurso Elementos de Linguagem R"

IMPORTANTE: É necessário usar as barras invertidas duplas para especificar o caminho da pasta. O R não entende quando apenas uma barra é utilizada.

Dica: Se você não sabe o caminho da pasta desejada, abra o windows explorer, copie a barra de endereços e cole no R.

Dica 2: Tenha um working directory diferente para cada projeto. Você pode cirar uma pasta apenas para este curso!

#### Escolhendo o arquivo

Algumas funções úteis:

As funções dir() e list.files() retornam uma lista de todos os arquivos contidos na sua pasta de trabalho.

dir()

## [1] "arquivo1.txt" "arquivo2.txt"   
## [3] "arquivo3.txt" "arquivo4.csv"   
## [5] "arquivo5.csv" "arquivox.txt"   
## [7] "ecolagoas2008.csv" "ecolagoas2008R.csv"   
## [9] "EXERCÍCIOS I.pdf" "EXERCÍCIOS II.pdf"   
## [11] "EXERCÍCIOS III.pdf" "grupos funcionais.png"   
## [13] "índices.png" "Preview-1564287c7e57.knit.md"  
## [15] "script.R" "zoobatata1998.csv"

list.files()

## [1] "arquivo1.txt" "arquivo2.txt"   
## [3] "arquivo3.txt" "arquivo4.csv"   
## [5] "arquivo5.csv" "arquivox.txt"   
## [7] "ecolagoas2008.csv" "ecolagoas2008R.csv"   
## [9] "EXERCÍCIOS I.pdf" "EXERCÍCIOS II.pdf"   
## [11] "EXERCÍCIOS III.pdf" "grupos funcionais.png"   
## [13] "índices.png" "Preview-1564287c7e57.knit.md"  
## [15] "script.R" "zoobatata1998.csv"

A função list.dirs() é bastante similar, mas em vez do nome dos arquivos ela retorna o nome das pastas dentro da sua pasta de trabalho.

list.dirs()

## [1] "."

Como foi comentado, não é necessário especificar o caminho de um arquivo a ser lido se ele estiver na sua pasta de trabalho. Mas se ele não estiver, as funções file.choose() e choose.files() são bastante úteis.

Estas funções abrem uma janela que permite escolher um arquivo. Como resultado, o R retorna o endereço deste arquivo, pronto para copiar e colar.

file.choose()

## [1] "C:\\Users\\Elder\\Desktop\\Minicurso Elementos de Linguagem R\\arquivo1.txt"

choose.files()

## [1] "C:\\Users\\Elder\\Desktop\\Minicurso Elementos de Linguagem R\\arquivo1.txt"  
## [2] "C:\\Users\\Elder\\Desktop\\Minicurso Elementos de Linguagem R\\arquivo2.txt"

A única diferença entre estas funções é file.choose() permite escolher apenas um arquivo por vez, enquanto choose.files() permite escolher vários.

#### Função source()

A função source() permite que o R leia um script inteiramente, do início ao fim. Vamos abrir o arquivo script.R, que está nos materiais desta aula.

source("script.R") #Lembre se de escrever o nome do arquivo entre aspas e de colocar a extensão do arquivo (.R,neste caso).

Se o arquivo não estiver na sua pasta de trabalho, lembre de escrever o endereço completo do arquivo. Por exemplo:

source("C:\\Users\\Elder\\SkyDrive\\Documentos\\Backups\\Minicurso Elementos de Linguagem R\\Materiais para aula 2\\script.R")

A função source() é útil para scripts que criem objetos ou funções importantes para o seu trabalho. Mas para trabalhar no seu projeto, é muito mais útil abrir o script no R da forma tradicional.

#### read.table()

Para importar seus dados ao R, usa-se a função read.table(). Vamos abrir o "arquivo1.txt" que está nos materiais da nossa aula.

read.table("arquivo1.txt")

## V1 V2 V3 V4  
## 1 A 45 0.32 S  
## 2 B 34 0.22 N  
## 3 C 51 0.36 S  
## 4 D 29 0.18 N  
## 5 E 44 0.30 S  
## 6 F 28 0.23 N  
## 7 G 47 0.37 S  
## 8 H 35 0.24 N

Se tudo deu certo, você importou um pequeno data frame para o R. Note que a função read.table() apenas leu o arquivo1.txt . Se você quiser gravar os dados, lembre-se de criar um objeto.

dados <- read.table("arquivo1.txt")  
dados

## V1 V2 V3 V4  
## 1 A 45 0.32 S  
## 2 B 34 0.22 N  
## 3 C 51 0.36 S  
## 4 D 29 0.18 N  
## 5 E 44 0.30 S  
## 6 F 28 0.23 N  
## 7 G 47 0.37 S  
## 8 H 35 0.24 N

Agora, vamos abrir o arquivo2.txt.

read.table("arquivo2.txt")

## V1 V2 V3 V4  
## 1 Ind Altura Crescimento Tratamento  
## 2 A 45 0.32 S  
## 3 B 34 0.22 N  
## 4 C 51 0.36 S  
## 5 D 29 0.18 N  
## 6 E 44 0.30 S  
## 7 F 28 0.23 N  
## 8 G 47 0.37 S  
## 9 H 35 0.24 N

No arquivo 2, a primeira linha corresponde ao nome das colunas, mas a função read.table() não reconhece isto automaticamente. Não se preocupe! Há um argumento chamado header. Por default, header=FALSE, porém podemos alterar para header=TRUE, o que significa que a primeira linha será entendida como os nomes das colunas.

read.table("arquivo2.txt",header=TRUE)

## Ind Altura Crescimento Tratamento  
## 1 A 45 0.32 S  
## 2 B 34 0.22 N  
## 3 C 51 0.36 S  
## 4 D 29 0.18 N  
## 5 E 44 0.30 S  
## 6 F 28 0.23 N  
## 7 G 47 0.37 S  
## 8 H 35 0.24 N

#Notou a diferença?

Agora, vamos abrir o arquivo 3:

read.table("arquivo3.txt")

## V1  
## 1 A/45/0;32/S  
## 2 B/34/0;22/N  
## 3 C/51/0;36/S  
## 4 D/29/0;18/N  
## 5 E/44/0;30/S  
## 6 F/28/0;23/N  
## 7 G/47/0;37/S  
## 8 H/35/0;24/N

Notou que o R reconheceu apenas uma coluna? Antes de prosseguirmos, abra os arquivos 1 e 3 no bloco de notas. -

Já abriu? Notou alguma diferença?

No arquivo 1, as colunas foram separadas por espaço (com a tecla tab).

No arquivo 3, as colunas foram separadas por uma barra.

Um dos argumentos da função read.table() é sep. Este argumento especifica que caracter foi utilizado para separar as colunas. Por default, sep=" ".

Vamos alterar para sep="/" e ver o que acontece.

read.table("arquivo3.txt",sep="/")

## V1 V2 V3 V4  
## 1 A 45 0;32 S  
## 2 B 34 0;22 N  
## 3 C 51 0;36 S  
## 4 D 29 0;18 N  
## 5 E 44 0;30 S  
## 6 F 28 0;23 N  
## 7 G 47 0;37 S  
## 8 H 35 0;24 N

Agora as colunas estão corretas. Mas ainda há alguma coisa estranha na coluna 3. Ao criar o arquivo 3, eu utilizei ponto e vírgula para separar as decimais. Podemos resolver isso com o argumento dec, cujo default é dec="."

read.table("arquivo3.txt",sep="/",dec=";")

## V1 V2 V3 V4  
## 1 A 45 0.32 S  
## 2 B 34 0.22 N  
## 3 C 51 0.36 S  
## 4 D 29 0.18 N  
## 5 E 44 0.30 S  
## 6 F 28 0.23 N  
## 7 G 47 0.37 S  
## 8 H 35 0.24 N

#Pronto!

OBS: Eu sei que ponto e vírgula é uma escolha não convencional para separar os decimais, mas utilizei apenas para demonstrar que qualquer caracter pode ser usado para separar os decimais, desde que você diga ao R.

#### Abrindo arquivos do excel

A função read.table() também abre arquivos do excel no formato csv (comma separated values).

ATENÇÂO! A função read.table() lê arquivos do excel em formato csv, mas não lê arquivos xls ou xlsx.

Vamos começar abrindo o arquivo 4.

setwd("C:\\Users\\Elder\\Desktop\\Minicurso Elementos de Linguagem R")  
read.table("arquivo4.csv")

## V1  
## 1 Ind;Altura;Crescimento;Tratamento  
## 2 A;45;0,32;S  
## 3 B;34;0,22;N  
## 4 C;51;0,36;S  
## 5 D;29;0,18;N  
## 6 E;44;0,3;S  
## 7 F;28;0,23;N  
## 8 G;47;0,37;S  
## 9 H;35;0,24;N

Em computadores configurados na língua inglesa, o formato csv usa vírgulas para separas as colunas, e pontos para separar os decimais.

Em computadores configurados em português, o formato csv usa ponto e vírgula para separas as colunas, e vírgulas para separar os decimais.

Agora, vamos abrir o arquivo 4 novamente, desta vez com os argumentos corretos:

read.table("arquivo4.csv",header=T,sep=";",dec=",")

## Ind Altura Crescimento Tratamento  
## 1 A 45 0.32 S  
## 2 B 34 0.22 N  
## 3 C 51 0.36 S  
## 4 D 29 0.18 N  
## 5 E 44 0.30 S  
## 6 F 28 0.23 N  
## 7 G 47 0.37 S  
## 8 H 35 0.24 N

Não é necessário especificar os argumentos corretos toda vez que formos abrir arquivos em formato csv. Para isso, existem as funções read.csv() e read.csv2().

read.csv2("arquivo4.csv")

## Ind Altura Crescimento Tratamento  
## 1 A 45 0.32 S  
## 2 B 34 0.22 N  
## 3 C 51 0.36 S  
## 4 D 29 0.18 N  
## 5 E 44 0.30 S  
## 6 F 28 0.23 N  
## 7 G 47 0.37 S  
## 8 H 35 0.24 N

read.csv() e read.csv2() fazem exatamente a mesma coisa que read.table(), mas os defaults de alguns argumentos são diferentes.

Vamos olhar as diferenças, nos atendo apenas aos argumentos header, sep e dec.

* read.table(file,header=FALSE,sep=" ",dec=".")
* read.csv(file,header=TRUE,sep=",",dec=".")
* read.csv2(file,header=TRUE,sep=";",dec=",")

Em primeiro lugar, por default, read.csv() e read.csv2() automaticamente reconhecem a primeira linha como o cabeçalho, ou seja, o nome das colunas.

read.csv() utiliza sep="," e dec=".", ou seja, ESTÁ OTIMIZADA PARA ARQUIVOS CSV EM LÍNGUA INGLESA

read.csv2() utiliza sep=";" e dec=",", ou seja, ESTÁ OTIMIZADA PARA ARQUIVOS CSV CRIADOS EM PORTUGUÊS (ou qualquer língua que use vírgula para separar os decimais)

#### Dados faltantes

O argumento na.strings serve para especificar que tipo caracter foi utilizado para representar os dados faltantes (NAs).

Abramos o arquivo 5:

read.csv2("arquivo5.csv")

## Ind Altura Crescimento Tratamento  
## 1 A - 0,32 S  
## 2 B 34 0,22 N  
## 3 C 51 0,36 S  
## 4 D 29 - N  
## 5 E 44 - S  
## 6 F 28 0,23 N  
## 7 G 47 0,37 S  
## 8 H 35 0,24 N

Neste caso, um hífen (-) foi utilizado para representar dados faltantes.

read.csv2("arquivo5.csv",na.strings="-") #Notou a diferença?

## Ind Altura Crescimento Tratamento  
## 1 A NA 0.32 S  
## 2 B 34 0.22 N  
## 3 C 51 0.36 S  
## 4 D 29 NA N  
## 5 E 44 NA S  
## 6 F 28 0.23 N  
## 7 G 47 0.37 S  
## 8 H 35 0.24 N

DICA: Quando tabelar seus dados, represente dados faltantes sempre da mesma forma. Escolha "-", "?", "NA", ou o que você achar melhor, mas use sempre o mesmo símbolo! O R só reconhece um na.string por vez, então se você misturar os símbolos de NA na mesma planilha será necessário consertá-los antes de por os dados no R.

DICA2: Não represente dados faltantes por espaços vazios. Desta forma, você pode confundir NAs com dados que existem, mas ainda não foram digitalizados.

Por fim, a função read.table() possui ainda vários outros argumentos que podem ser úteis; não se esqueça de ler a documentação disponível em help(read.table).

#### Exportando dados

Primeiro, vamos criar uma matriz contendo os dados que queremos salvar.

tabela <- cbind(1:10,N=rnorm(10))  
tabela

## N  
## [1,] 1 -1.41768  
## [2,] 2 0.52696  
## [3,] 3 -0.02651  
## [4,] 4 0.49792  
## [5,] 5 1.60031  
## [6,] 6 0.58512  
## [7,] 7 0.42589  
## [8,] 8 1.12430  
## [9,] 9 0.66793  
## [10,] 10 -0.98019

Para criar um arquivo txt, utilizamos a função write.table(). O primeiro argumento desta função é uma matriz ou data frame do R. O segundo argumento é o nome do arquivo que iremos criar.

write.table(tabela,file="arquivox.txt")

Agora olhe em sua pasta de trabalho e veja o novo arquivo criado!

Lembre que, para salvar o arquivo em outra pasta, é necessário especificar o caminho. Por exemplo:

write.table(tabela,file="C:\\Users\\Elder\\Desktop\\arquivox.txt")

Assim como read.csv() e read.csv2(), também há funções otimizadas para criar arquivos csv. Estas são write.csv() e write.csv2().

Use write.csv2() para criar um arquivo que o seu excel em português pode abrir.

write.csv2(tabela,file="arquivox2.csv")

Se o seu excel estiver em inglês, use write.csv()

write.csv(tabela,file="arquivox2.csv")

Não se esqueça de conferir maiores informações em help(write.table).