Week 1

tafuenza

17-11-2019

# Chequear y crear directorios desde R

Si quiero crear o chequear si existe una carpeta dentro de mi “work directory (wd)” debo usar:

file.exists("DirectoryName") # Chequea si el directorio existe  
dir.create("DirectoryName") # Crea el directorio si no existe

# Descargar archivos desde internet

Para descargar un archivo directamente desde internet, debemos utilizar el siguiente codigo:

download.file(fileURL, destfile, method) # URL, carpeta de destino y nombre de archivo (data/mydata.csv), method (Curl)

# Abrir Archivos

El clasico archivo que se va a leer es un archivo .csv

data <- read.csv(file,header,sep,row.names,row) # Archivo, si tiene o no titulo las variables, como estan separadas las columnas, si las filas tienen nombres, cuantas filas quiero leer  
data <- read.csv(quotes, na.strings, skip) # Si existen valores entre comillas, como se representan los NA, saltar algunas lineas desde el comienzo

# Xlsx

library("xlsx"")  
data <- read.xlsx("data.xls", sheetIndex = 1, HEADER = TRUE, colIndex = 2:3, row.Index = 1:4)

# Paginas Web

R permite leer directamente archivos de paginas web como XML

library('XML')  
url <- "http://www.w3schools.com/xml/simple.xml"  
download.file(url, destfile = "foo.xml")  
doc <- xmlTreeParse("foo.xml", useInternal=TRUE)  
rootNode <- xmlRoot(doc)  
xmlName(rootNode)

## [1] "breakfast\_menu"

rootNode[[1]]

## <food>  
## <name>Belgian Waffles</name>  
## <price>$5.95</price>  
## <description>Two of our famous Belgian Waffles with plenty of real maple syrup</description>  
## <calories>650</calories>  
## </food>

rootNode[[1]][[1]]

## <name>Belgian Waffles</name>

xmlSApply(rootNode,xmlValue)

## food   
## "Belgian Waffles$5.95Two of our famous Belgian Waffles with plenty of real maple syrup650"   
## food   
## "Strawberry Belgian Waffles$7.95Light Belgian waffles covered with strawberries and whipped cream900"   
## food   
## "Berry-Berry Belgian Waffles$8.95Light Belgian waffles covered with an assortment of fresh berries and whipped cream900"   
## food   
## "French Toast$4.50Thick slices made from our homemade sourdough bread600"   
## food   
## "Homestyle Breakfast$6.95Two eggs, bacon or sausage, toast, and our ever-popular hash browns950"

#Para encontrar otros datos especificos se puede usar  
#/node  
#//node  
#node[@attr-name]  
#node[@attr-name = 'bob']  
xpathSApply(rootNode,'//name',xmlValue)

## [1] "Belgian Waffles" "Strawberry Belgian Waffles"   
## [3] "Berry-Berry Belgian Waffles" "French Toast"   
## [5] "Homestyle Breakfast"

xpathSApply(rootNode,'//price',xmlValue)

## [1] "$5.95" "$7.95" "$8.95" "$4.50" "$6.95"

# Abrir Json

library('jsonlite')

## Warning: package 'jsonlite' was built under R version 3.6.1

URL <- "https://api.github.com/users/jtleek/repos"  
data <- fromJSON(URL)  
names(data)

## [1] "id" "node\_id" "name"   
## [4] "full\_name" "private" "owner"   
## [7] "html\_url" "description" "fork"   
## [10] "url" "forks\_url" "keys\_url"   
## [13] "collaborators\_url" "teams\_url" "hooks\_url"   
## [16] "issue\_events\_url" "events\_url" "assignees\_url"   
## [19] "branches\_url" "tags\_url" "blobs\_url"   
## [22] "git\_tags\_url" "git\_refs\_url" "trees\_url"   
## [25] "statuses\_url" "languages\_url" "stargazers\_url"   
## [28] "contributors\_url" "subscribers\_url" "subscription\_url"   
## [31] "commits\_url" "git\_commits\_url" "comments\_url"   
## [34] "issue\_comment\_url" "contents\_url" "compare\_url"   
## [37] "merges\_url" "archive\_url" "downloads\_url"   
## [40] "issues\_url" "pulls\_url" "milestones\_url"   
## [43] "notifications\_url" "labels\_url" "releases\_url"   
## [46] "deployments\_url" "created\_at" "updated\_at"   
## [49] "pushed\_at" "git\_url" "ssh\_url"   
## [52] "clone\_url" "svn\_url" "homepage"   
## [55] "size" "stargazers\_count" "watchers\_count"   
## [58] "language" "has\_issues" "has\_projects"   
## [61] "has\_downloads" "has\_wiki" "has\_pages"   
## [64] "forks\_count" "mirror\_url" "archived"   
## [67] "disabled" "open\_issues\_count" "license"   
## [70] "forks" "open\_issues" "watchers"   
## [73] "default\_branch"

names(data$owner) # Los data frame te permite tener dataframes unos dentro de otros

## [1] "login" "id" "node\_id"   
## [4] "avatar\_url" "gravatar\_id" "url"   
## [7] "html\_url" "followers\_url" "following\_url"   
## [10] "gists\_url" "starred\_url" "subscriptions\_url"   
## [13] "organizations\_url" "repos\_url" "events\_url"   
## [16] "received\_events\_url" "type" "site\_admin"

# Tambien puedo transformar un DataFrame a JSON con toJSON()

# Data table

Data.table() es capaz de hacer lo mismo que data.frame() pero tiene algunas opciones interesantes que pueden ser utiles

library('data.table')

## Warning: package 'data.table' was built under R version 3.6.1

set.seed(1)  
DT <- data.table( x = rnorm(10), y = rbinom(10,1,0.5), z = rpois(10,1))  
tables() ## Permite conocer algunas caracteristicas de la tabla de datos

## NAME NROW NCOL MB COLS KEY  
## 1: DT 10 3 0 x,y,z   
## Total: 0MB

DT[2,]

## x y z  
## 1: 0.1836433 0 1

DT[c(2,3)] # Cuando le entrego un indice, siempre lo comprende como las filas

## x y z  
## 1: 0.1836433 0 1  
## 2: -0.8356286 1 1

DT[,c(2,3)] # Si lo tratas de hacer por columnas no funciona

## y z  
## 1: 1 1  
## 2: 0 1  
## 3: 1 1  
## 4: 0 0  
## 5: 0 2  
## 6: 0 1  
## 7: 0 2  
## 8: 0 0  
## 9: 1 1  
## 10: 0 1

DT[,list(sum(x),mean(z))] # Le puedes entregar una lista de funciones que hacer por variable

## V1 V2  
## 1: 1.322028 1

DT[,w := x^2] # Es mas eficiente al agregar nuevas variables  
DT[,m:= {tmp <-(x+z); log2(tmp+5)}] # Puedo crear funciones para agregar nuevas variables  
DT[,a:= x>0] # Comprobar logicas y agregar nueva variable  
DT[,b:= mean(x+w), by = a] # Calcula el promedio de todos los a = TRUE y lo de A = FALSE separados  
DT

## x y z w m a b  
## 1: -0.6264538 1 1 0.39244438 2.425874 FALSE -0.1826973  
## 2: 0.1836433 0 1 0.03372487 2.628457 TRUE 1.2852371  
## 3: -0.8356286 1 1 0.69827518 2.368593 FALSE -0.1826973  
## 4: 1.5952808 0 0 2.54492084 2.721434 TRUE 1.2852371  
## 5: 0.3295078 0 2 0.10857537 2.873716 TRUE 1.2852371  
## 6: -0.8204684 0 1 0.67316837 2.372822 FALSE -0.1826973  
## 7: 0.4874291 0 2 0.23758708 2.904470 TRUE 1.2852371  
## 8: 0.7383247 0 0 0.54512337 2.520630 TRUE 1.2852371  
## 9: 0.5757814 1 1 0.33152416 2.717162 TRUE 1.2852371  
## 10: -0.3053884 0 1 0.09326207 2.509597 FALSE -0.1826973

set.seed(2)  
DATA <- data.table(x=sample(letters[1:3], 1E5,TRUE))  
DATA[,.N, by = x] # Cuenta la cantidad de veces que se repite cada uno de los elementos de x

## x N  
## 1: a 33098  
## 2: c 33566  
## 3: b 33336

DT <- data.table(x=rep(c("a","b","c"), each = 100), y= rnorm(300))  
setkey(DT,x)  
A <- DT['a'] ## Al asignar a x como la llave, al entregarle directamente algun cambio simplemente usa la llave  
A[1:10]

## x y  
## 1: a 1.66474613  
## 2: a -0.82298104  
## 3: a -0.31633398  
## 4: a -1.24322526  
## 5: a -1.18849475  
## 6: a -2.09007244  
## 7: a -1.03697558  
## 8: a 0.09620607  
## 9: a -1.19671910  
## 10: a -1.51553259