Tarea: Importación de ficheros sin formato

Análisis Exploratorio de Datos, Máster en Ciencia de Datos- UV

Daniel Lillo Plaza

2023-07-09

Índice

---

1 Introducción.

El objetivo de esta tarea es realizar la correcta importación de los datos presentes en el fichero de texto *TKD_002_10_C.asc*, los cuales no solo no están en forma *tidy*, sino que además no tienen ningún tipo de formato.

2 Instalación de paquetes

Incluimos todas las librerías necesarias para la ejecucion del código en la siguiente lista:

$$packages = c(\text{“}knitr\text{”}, \text{“}tidyr\text{”}, \text{“}dplyr\text{”})$$

```
[1] ".GlobalEnv"        "package:dplyr"     "package:tidyr"     "package:knitr"
[5] "package:stats"     "package:graphics"  "package:grDevices" "package:utils"
[9] "package:datasets"  "package:methods"   "Autoloads"         "package:base"
```

3 Código base.

Primero vamos a diseñar el código base para la lectura de este tipo de ficheros en particular. Destacamos que este código tan solo sirve para leer los datos del archivo *TKD_002_10_C.asc* y de todos aquellos que tengan la misma estructura interna y extensión.

```
library(dplyr)
library(tidyr)
# Leemos el nombre del fichero que nos interesa y guardamos a qué sujeto
# corresponde.
nombre<-grep(pattern=".asc$", x=dir(), value=TRUE)
fichero<-readLines(nombre)
Sujeto<-unlist(strsplit(nombre, split="_"))[2]

# Obtenemos la variable SamplingFreq
SamplingFreq<-grep(pattern="SamplingFreq=.",fichero, value=TRUE)
SamplingFreq<-unlist(strsplit(SamplingFreq, split="="))
SamplingFreq<-as.numeric(SamplingFreq[2])

# Obtenemos la variable ChannelCount
ChannelCount<-grep(pattern="ChannelCount=.",fichero, value=TRUE)
ChannelCount<-unlist(strsplit(ChannelCount, split="="))
ChannelCount<-as.numeric(ChannelCount[2])

# Obtenemos la variable Units
Units<-grep(pattern="[UNITS]",fichero, fixed=TRUE)
Units<-fichero[Units+1]

# Obtenemos las variables TipoPatada y OrdenComienzo
TipoPatada_index<-grep(pattern="[COMMENT]", fichero, fixed=TRUE)
OrdenComienzo<-unlist(strsplit(gsub(x=fichero[TipoPatada_index+1], replacement="",
pattern="[^d_i]| [^i_d]"), split=""))
OrdenComienzo<-paste0(OrdenComienzo[(length(OrdenComienzo)-2)],
OrdenComienzo[(length(OrdenComienzo)-1)], OrdenComienzo[(length(OrdenComienzo))])
TipoPatada<-regexpr("([0-9]+_[CGL])|([0-9]+[CGL])",text=fichero[TipoPatada_index+1])
TipoPatada<-unlist(strsplit(fichero[TipoPatada_index+1], ""))[TipoPatada:attributes(TipoPatada)$match.l
TipoPatada<-tail(TipoPatada, 1)

# Contruimos una tabla con los datos igual que la del fichero
source_names<-grep(pattern="[SOURCE NAMES]", fichero, fixed=TRUE)
side_info<-grep(pattern="[SIDE INFO]", fichero, fixed=TRUE)
units<-grep(pattern="[UNITS]", fichero, fixed=TRUE)

source_names<-fichero[(source_names+1):(source_names+ChannelCount)]
side_info<-fichero[(side_info+1):(side_info+ChannelCount)]

columnas<-paste(side_info,source_names, sep="_")

datos_index<-grep(pattern="[DATA]", fichero, fixed=TRUE)+1
N<-length(fichero)
datos<-matrix(0, nrow=(N-datos_index+1), ncol=length(columnas))
colnames(datos)<-columnas

for(j in datos_index:N){
```

```
datos[j-datos_index+1,]<-as.numeric(gsub(x=unlist(strsplit(fichero[j], split="\t")),
pattern=",", replacement="\\."))
}

datos<-as.data.frame(datos)

# Transformamos la tabla en un conjunto Tidy
DataTK<-datos%>%
  pivot_longer(names_to="NombreCanal", values_to="Data", cols=everything())%>%
  separate(col=NombreCanal, into=c("NombreCanal","Side"),sep="_")%>%
  transmute("Sujeto"=Sujeto,"TipoPatada"=TipoPatada,"NombreCanal"=NombreCanal,
            "Side"=Side, "OrdenComienzo"=OrdenComienzo, "Data"=Data)

# Guardamos los datos que nos interesan
save(DataTK, SamplingFreq, Units, ChannelCount, file="Ejemplo.RData")
```

4 Funciones auxiliares.

A continuación, vamos a perfeccionar el proceso anterior y generalizarlo para poder aplicarlo a todos los ficheros que tengan la extensión deseada ('.asc' en este caso) y que internamente tengan la misma estructura, pese a que no tengan formato.

Para ello, hemos creado una serie de funciones auxiliares que se detallan en el chunk a continuación:

```
# Creamos la función lectura para poder hacer un apply.
lectura<-function(fichero){
require(dplyr)
require(tidyr)
# Obtenemos la variable SamplingFreq
SamplingFreq<-grep(pattern="SamplingFreq=.",fichero, value=TRUE)
SamplingFreq<-unlist(strsplit(SamplingFreq, split="="))
SamplingFreq<-as.numeric(SamplingFreq[2])

# Obtenemos la variable ChannelCount
ChannelCount<-grep(pattern="ChannelCount=.",fichero, value=TRUE)
ChannelCount<-unlist(strsplit(ChannelCount, split="="))
ChannelCount<-as.numeric(ChannelCount[2])

# Obtenemos la variable Units
Units<-grep(pattern="[UNITS]",fichero, fixed=TRUE)
Units<-fichero[Units+1]

# Obtenemos las variables TipoPatada y OrdenComienzo
TipoPatada_index<-grep(pattern="[COMMENT]", fichero, fixed=TRUE)
OrdenComienzo<-unlist(strsplit(gsub(x=fichero[TipoPatada_index+1], replacement="",
pattern="[^d_i]| [^i_d]"), split=""))
OrdenComienzo<-paste0(OrdenComienzo[(length(OrdenComienzo)-2)],
OrdenComienzo[(length(OrdenComienzo)-1)], OrdenComienzo[(length(OrdenComienzo))])
TipoPatada<-regexpr("([0-9]+_[CGL])|([0-9]+[CGL])",text=fichero[TipoPatada_index+1])
TipoPatada<-unlist(strsplit(fichero[TipoPatada_index+1], ""))[TipoPatada:attributes(TipoPatada)$match.l
TipoPatada<-tail(TipoPatada, 1)
```

```
# Contruimos una tabla con los datos igual que la del fichero
source_names<-grep(pattern="[SOURCE NAMES]", fichero, fixed=TRUE)
side_info<-grep(pattern="[SIDE INFO]", fichero, fixed=TRUE)
units<-grep(pattern="[UNITS]", fichero, fixed=TRUE)

source_names<-fichero[(source_names+1):(source_names+ChannelCount)]
side_info<-fichero[(side_info+1):(side_info+ChannelCount)]

columnas<-paste(side_info,source_names, sep="_")

datos_index<-grep(pattern="[DATA]", fichero, fixed=TRUE)+1
N<-length(fichero)
datos<-matrix(0, nrow=(N-datos_index+1), ncol=length(columnas))
colnames(datos)<-columnas

for(j in datos_index:N){
datos[j-datos_index+1,]<-as.numeric(gsub(x=unlist(strsplit(fichero[j], split="\t")),
pattern=",", replacement="\\."))
}

datos<-as.data.frame(datos)

# Transformamos la tabla en un conjunto Tidy
DataTK<-datos%>%
  pivot_longer(names_to="NombreCanal", values_to="Data", cols=everything())%>%
  separate(col=NombreCanal, into=c("NombreCanal", "Side"), sep="_")%>%
  transmute("TipoPatada"=TipoPatada,"NombreCanal"=NombreCanal,"Side"=Side,
            "OrdenComienzo"=OrdenComienzo, "Data"=Data)

return(list("DataTK"=DataTK, "SamplingFreq"=SamplingFreq, "Units"=Units,
"ChannelCount"=ChannelCount))
}


######################################################################
# Creamos la función no_formato que nos devuelve todos los datos leidos en una
# lista tidy.
no_formato<-function(directorio=dir(), extension=".asc$"){
  if(tail(unlist(strsplit(extension, split="")),1)!="$"){
    stop("'extension' debe de terminar con el símbolo '$'.")
  }
# Leemos el nombre del fichero que nos interesa y guardamos a qué sujeto
# corresponde.
  nombre<-as.matrix(grep(pattern=extension, x=dir("./data", full.names = TRUE),
  value=TRUE))
# Leemos cada uno de los ficheros línea a línea
  fichero<-apply(nombre,1,readLines)
  Sujeto<-apply(nombre, 1,FUN=function(x){unlist(strsplit(x, split="_"))[2]})
# Llamamos a la fucnión lectura, creada abajo, que se encargará de crear el
# data.frame casi final.
  ficheros_leidos<-lapply(fichero, lectura)
  names(ficheros_leidos)<-paste0("fichero_",(1:length(ficheros_leidos)))
  for (f in 1:length(fichero)){
```

```
    ficheros_leidos[[f]][[1]]<-ficheros_leidos[[f]][[1]]%>%mutate("Sujeto"=Sujeto[f])
  }
  return(ficheros_leidos)
}

#####################################################################
# Creamos la función comprobacion para ver si SamplingFreq, Units y ChannelCount
# son iguales en todos los ficheros y devolvemos SamplingFreq, Units y ChannelCount
# (si no son iguales devolveremos un aviso pero continuaremos con la ejecución).
comprobacion<-function(list){
  n<-length(list)
  S<-numeric(n); U<-numeric(n); C<-numeric(n)
  for (j in 1:length(list)){
    S[j]<-list[[j]][["SamplingFreq"]]
    U[j]<-list[[j]][["Units"]]
    C[j]<-list[[j]][["ChannelCount"]]
  }
  S<-unique(S)
  U<-unique(U)
  C<-unique(C)
  if (length(S)!=1){
    warning("'SamplingFreq' se esperaba que fuese un único valor")
  }
  if (length(U)!=1){
    warning("'Units' se esperaba que fuese un único valor")
  }
  if (length(C)!=1){
    warning("'ChannelCount' se esperaba que fuese un único valor")
  }
  return(list("SamplingFreq"=S, "Units"=U, "ChannelCount"=C))
}
```

5 Ejecución y guardado.

Ahora sí, empleando las funciones creadas anteriormente procedemos a la lectura y guardado de todos los ficheros.

```
# Creamos una lista con todos los ficheros leídos y tidy.
datos<-no_formato()

# Juntamos todos los data.frame en DataTKAII.
DataTKAII<-datos[[1]][[1]]
for (i in 2:length(datos)){
  DataTKAII<-rbind(DataTKAII, datos[[i]][[1]])
}

datos_adicionales<-comprobacion(datos)
SamplingFreq<-datos_adicionales[["SamplingFreq"]]
Units<-datos_adicionales[["Units"]]
ChannelCount<-datos_adicionales[["ChannelCount"]]
```

```
# Guardamos los datos que nos interesan
save(DataTKAII, SamplingFreq, Units, ChannelCount, file="ImportaTKAll.Rdata")
```