Ejemplo taller datos Netflix

Ricardo

05 diciembre, 2020

Table of Contents

# Datos netflix

## Carga de datos

Enlace a estos datos de [Netflix](https://www.kaggle.com/netflix-inc/netflix-prize-data) Generad un proyecto nuevo. Bajad lo datos de netflix a un carpeta/directorio que se llame netflix y dentro de netflix crear una carpeta/directorio que se llame model\_netflix

Sabemos que en combined\_data\_1.txt hay 2342 películas y tiene 12095343. Cada película está separada por un entero por ejemplo 1: es decir un entero seguido de :.

Si queremos leer una cuántas películas tenemos que leer sólo algunas lineas . Por ejemplo para leer las 100 primeras películas tenemos que leer las lineas hasta en encontrar la película 101 es decir 352872 lineas.

|  |  |  |
| --- | --- | --- |
| Película Núm. | ID\_película | fila |
| 1 | 1: | 1 |
| 101 | 101: | 352872 |
| 201 | 201: | 934086 |
| 301 | 301: | 1454270 |
| 501 | 501: | 2799205 |
| 1001 | 1001: | 5011200 |
| 2001 | 2001: | 10319270 |

#Cargamos la librería tidyverse... mejor cargarlo oculto en el setup  
library(tidyverse)  
#Cargamos los datos de combined\_data\_1.txt netflix  
n\_max=352872-1 # leo las primeras 100 películas   
#n\_max=5011200-1 # leo las primeras 1000 películas   
#n\_max=Inf# leo todas  
#Con este límite cargamos hasta la película que hace 1000 del   
##combined\_data\_1.txt, para cargar todas poned n\_max\_1=Inf  
netflix=read\_tsv("data/combined\_data\_1.txt",n\_max = n\_max,col\_names = FALSE)

##   
## -- Column specification --------------------------------------------------------  
## cols(  
## X1 = col\_character()  
## )

dim(netflix)

## [1] 352871 1

# si n\_max=Inf hemos cargado 24058263 lineas unos 24 millones  
# los cuatro ficheros combined tienen en total unos 100 millones de líneas  
head(netflix)

## # A tibble: 6 x 1  
## X1   
## <chr>   
## 1 1:   
## 2 1488844,3,2005-09-06  
## 3 822109,5,2005-05-13   
## 4 885013,4,2005-10-19   
## 5 30878,4,2005-12-26   
## 6 823519,3,2004-05-03

|  |  |
| --- | --- |
| files | number of rows |
| 1 | 24058263 |
| 2 | 26982302 |
| 3 | 22605786 |
| 4 | 26851926 |
| —— | ———— |
| Total | 100498277 |

Arreglamos los datos….

netflix=netflix%>% mutate(fila=row\_number())  
filas=grep(":",netflix$X1)  
#save(filas,file="data/filas\_1.Robj")  
filas\_ID= netflix %>%  
 filter( fila %in% filas ) %>%  
 mutate(ID=as.integer(gsub(":","",X1)))  
#IDs=unique(filas\_ID$X1)  
reps=diff(c(filas\_ID$fila,max(netflix$fila)+1))

netflix=netflix %>%  
 mutate(ID1=rep(filas\_ID$X1,times=reps)) %>%  
 filter(!(fila %in% filas)) %>%  
 select(-fila) %>%  
 separate(X1,into=c("ID\_user","Score","data"),sep=",") %>%  
 mutate(Score=as.integer(Score)) %>%  
 separate(col = ID1,into=c("ID\_film","borrar")) %>%  
 select(-borrar) %>% mutate(ID\_film=as.numeric(ID\_film))

Recapitulamos. Hemos leído los perfiles de 100 películas

glimpse(netflix)

## Rows: 352,771  
## Columns: 4  
## $ ID\_user <chr> "1488844", "822109", "885013", "30878", "823519", "893988",...  
## $ Score <int> 3, 5, 4, 4, 3, 3, 4, 3, 4, 3, 4, 5, 3, 3, 4, 4, 4, 3, 4, 5,...  
## $ data <chr> "2005-09-06", "2005-05-13", "2005-10-19", "2005-12-26", "20...  
## $ ID\_film <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...

class(netflix)

## [1] "tbl\_df" "tbl" "data.frame"

ncol(netflix)

## [1] 4

nrow(netflix)

## [1] 352771

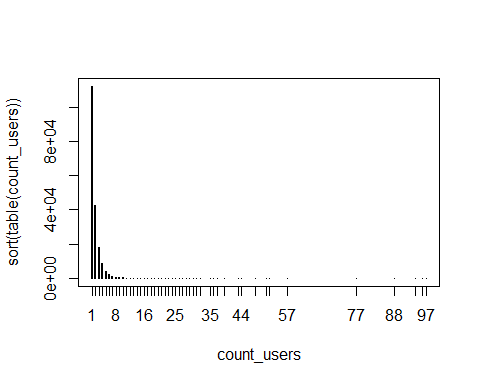
Respecto a los usuarios que han visto alguna de las películas tenemos

length(unique(netflix$ID\_user))

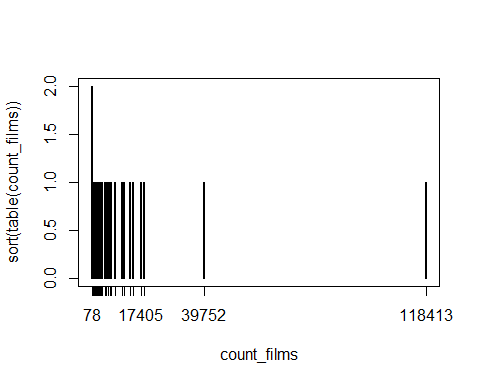
## [1] 191668

table(netflix$ID\_user)-> count\_users  
table(netflix$ID\_film) -> count\_films  
#knitr::kable(sort(count\_users))  
#knitr::kable(sort( count\_films))

plot(sort(table(count\_users)))



plot(sort(table(count\_films)))



#Tabla demasiado larga para mostrar  
#knitr::kable(sort(table(table(netflix$ID\_user))))   
#knitr::kable(sort(table(table(netflix$ID\_film))))

## Similaridades entre películas

Veremos el cálculo de similaridades entre cada par de películas según los sus vectores de score. Lo haremos de dos maneras de forma secuencial y de forma paralela.

### Similaridad coseno entre películas

La similaridad coseno de dos vectores de no nulos y

sim\_cos\_netflix=function(xy,data=netflix){   
 x=xy[1]  
 y=xy[2]  
 x1=filter(data,ID\_film==x)  
 y1=filter(data,ID\_film==y)  
 xy=inner\_join(x1,y1,by="ID\_user")  
 sim= sum(xy$Score.x\*xy$Score.y)/sqrt(sum(x1$Score^2)\*sum(y1$Score^2))  
 sim  
}

sim\_cos\_netflix(c(1,2),netflix)

## [1] 0.01493888

sim\_cos\_netflix(c(2,1),netflix)

## [1] 0.01493888

sim\_cos\_netflix(c(1,1),netflix)

## [1] 1

aux=t(combn(unique(netflix$ID\_film),m=2))  
sim=tibble(x=aux[,1],y=aux[,2])  
  
time\_sim <- system.time(sim$sim <- as.numeric(apply(sim,1,sim\_cos\_netflix)))  
time\_sim

## user system elapsed   
## 55.00 3.92 58.97

#install.packages("reshape2")  
library(reshape2)# libreria que reformatea datos

##   
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':  
##   
## smiths

# añado la diagonal a la similitud todas 1   
diag\_sim=tibble(x=1:100,y=1:100,sim=1)  
# construyo la matriz solo la parte triangular superior  
acast(rbind(sim,diag\_sim), x~y, value.var = "sim")-> Sim\_cos\_matrix1  
# pongo los NA de la parte triangular inferior y diagonal a cero  
Sim\_cos\_matrix1[is.na(Sim\_cos\_matrix1)]=0  
Sim\_cos\_matrix1=Sim\_cos\_matrix1+t(Sim\_cos\_matrix1) # completo la matriz de similitud  
diag(Sim\_cos\_matrix1)=1

### Similitud en paralelo

Sim\_cos\_matrix2=matrix(0,ncol=100,nrow=100)

#install.packages(“parallel”) # libreria de pararelización  
library(parallel)  
# Detectamos el número de cores disponibles y creamos el cluster  
no\_cores <- parallel::makeCluster(detectCores())  
no\_cores

## socket cluster with 8 nodes on host 'localhost'

length(no\_cores)

## [1] 8

#str(no\_cores)  
cl <- makeCluster(length(no\_cores)-1)   
  
  
pares=tibble(xy=t(combn(unique(netflix$ID\_film),m=2))) %>% transmute(x=as.numeric(xy[,1]),y=as.numeric(xy[,2]))  
  
clusterCall(cl, function() library(tidyverse))

## [[1]]  
## [1] "forcats" "stringr" "dplyr" "purrr" "readr" "tidyr"   
## [7] "tibble" "ggplot2" "tidyverse" "stats" "graphics" "grDevices"  
## [13] "utils" "datasets" "methods" "base"   
##   
## [[2]]  
## [1] "forcats" "stringr" "dplyr" "purrr" "readr" "tidyr"   
## [7] "tibble" "ggplot2" "tidyverse" "stats" "graphics" "grDevices"  
## [13] "utils" "datasets" "methods" "base"   
##   
## [[3]]  
## [1] "forcats" "stringr" "dplyr" "purrr" "readr" "tidyr"   
## [7] "tibble" "ggplot2" "tidyverse" "stats" "graphics" "grDevices"  
## [13] "utils" "datasets" "methods" "base"   
##   
## [[4]]  
## [1] "forcats" "stringr" "dplyr" "purrr" "readr" "tidyr"   
## [7] "tibble" "ggplot2" "tidyverse" "stats" "graphics" "grDevices"  
## [13] "utils" "datasets" "methods" "base"   
##   
## [[5]]  
## [1] "forcats" "stringr" "dplyr" "purrr" "readr" "tidyr"   
## [7] "tibble" "ggplot2" "tidyverse" "stats" "graphics" "grDevices"  
## [13] "utils" "datasets" "methods" "base"   
##   
## [[6]]  
## [1] "forcats" "stringr" "dplyr" "purrr" "readr" "tidyr"   
## [7] "tibble" "ggplot2" "tidyverse" "stats" "graphics" "grDevices"  
## [13] "utils" "datasets" "methods" "base"   
##   
## [[7]]  
## [1] "forcats" "stringr" "dplyr" "purrr" "readr" "tidyr"   
## [7] "tibble" "ggplot2" "tidyverse" "stats" "graphics" "grDevices"  
## [13] "utils" "datasets" "methods" "base"

clusterExport(cl,list("sim\_cos\_netflix","Sim\_cos\_matrix2","netflix","pares"))  
  
  
# Lanzo la computación en paralelo  
t1=Sys.time()  
time\_sim\_parallel <- system.time(  
results<-parApply(cl,  
 pares,1,  
 FUN=function(x) {c(as.integer(x[1]),as.integer(x[2]),  
 sim\_cos\_netflix(x,data=netflix))})  
)  
  
#apply(results,2,FUN=function(x) {x[3]->>Sim\_cos\_matrix2[x[1],x[2]]})  
# o, en este caso, con un for como hacemos a continuación  
for(i in 1:dim(results)[2]){  
 x=results[,i]  
 x[3]->Sim\_cos\_matrix2[x[1],x[2]]  
}  
# arreglo la triangular inferior   
Sim\_cos\_matrix2=Sim\_cos\_matrix2+t(Sim\_cos\_matrix2)  
diag(Sim\_cos\_matrix2)=1#arreglo la diagonal  
all(Sim\_cos\_matrix2==Sim\_cos\_matrix1)

## [1] TRUE

t2=Sys.time()  
t2-t1

## Time difference of 36.33856 secs

time\_sim\_parallel

## user system elapsed   
## 0.0 0.0 36.2

# para el cluster  
parallel::stopCluster(cl)  
time\_sim\_parallel

## user system elapsed   
## 0.0 0.0 36.2

time\_sim

## user system elapsed   
## 55.00 3.92 58.97

# Algunos ejemplos con esta similitud: clusterin jerrarquico. mds y kmeans.

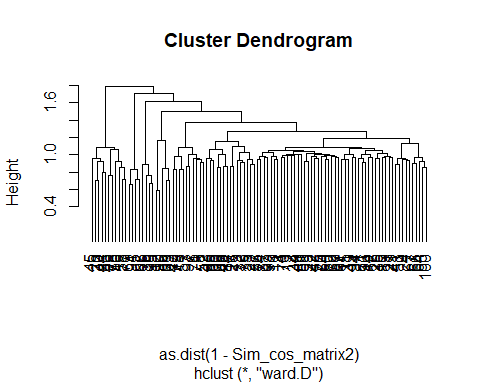
Veremos sólo el código de algunos algoritmos básicos para clasificar las películas a partir de la similitud coseno. Tenemos que transformar la similitud a disimilitud o distancia , haciendo por ejemplo (hay otras maneras)

## Clustering jerarquico

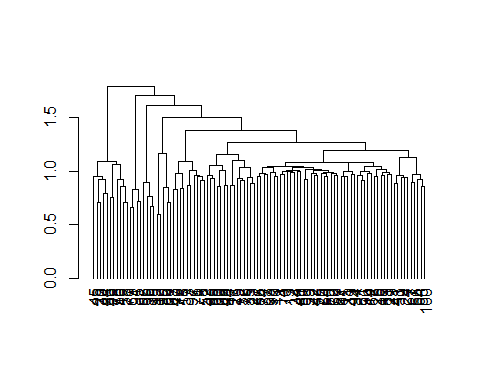
hclust(as.dist(1-Sim\_cos\_matrix2),method = "ward")-> h

## The "ward" method has been renamed to "ward.D"; note new "ward.D2"

plot(h,hang = -1)



h\_d=as.dendrogram(h)  
plot(h\_d)



cluster\_3=cutree(h,k = 3)  
cluster\_3

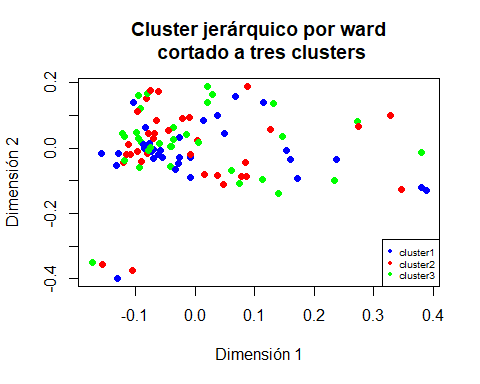
## [1] 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 3 1 3 1 1 1 1 1 1 1  
## [38] 1 1 1 1 1 1 3 3 3 1 1 1 1 1 1 1 1 3 1 1 3 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1  
## [75] 1 1 3 3 1 1 1 1 3 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1

table(cluster\_3)

## cluster\_3  
## 1 2 3   
## 85 4 11

## MDS

sol\_mds=cmdscale(as.dist(1-Sim\_cos\_matrix2), k=2)  
  
plot(sol\_mds[,1:2],col=c("blue","red","green"),  
 xlab="Dimensión 1",ylab="Dimensión 2",  
 main="Cluster jerárquico por ward\n cortado a tres clusters",  
 pch=19)  
legend("bottomright",legend=c(paste0("cluster",1:3)),  
 col=c("blue","red","green"),cex=0.6,pch=19)



## Clasificación por kmeans a partir de corrdenadas del MDS

sol\_kmeans=kmeans(as.dist(1-Sim\_cos\_matrix2),3)  
sol\_kmeans$cluster

## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20   
## 3 3 3 3 2 3 3 1 3 3 3 3 3 3 3 1 3 1 3 3   
## 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40   
## 3 3 3 3 3 3 3 1 3 1 3 3 3 3 3 3 3 3 3 3   
## 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60   
## 3 3 3 1 1 1 3 1 3 3 3 3 3 3 1 3 1 1 3 3   
## 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80   
## 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 1 1 1 3 3   
## 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100   
## 3 3 1 1 3 3 3 3 3 3 2 2 3 3 3 3 1 3 3 3

# numero de películas por cluster  
table(sol\_kmeans$cluster)

##   
## 1 2 3   
## 18 4 78

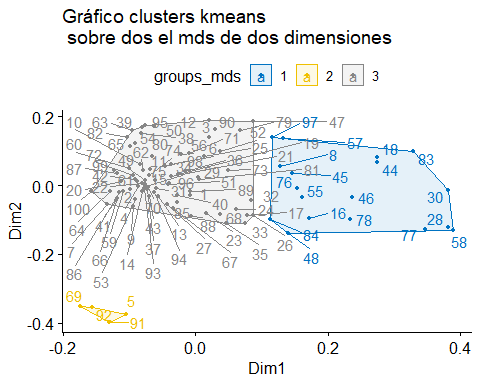
# matriz de confusión cluster jerárquico versus cluster de kmeans  
table(cluster\_3,sol\_kmeans$cluster)

##   
## cluster\_3 1 2 3  
## 1 7 0 78  
## 2 0 4 0  
## 3 11 0 0

clust <- sol\_kmeans$cluster %>%   
 as.factor()  
sol\_mds<- as\_tibble(sol\_mds) %>%  
 mutate(groups\_mds = clust,  
 groups\_h=as.factor(cluster\_3)) %>%  
 dplyr::rename(Dim1=V1,Dim2=V2)

## Warning: The `x` argument of `as\_tibble.matrix()` must have unique column names if `.name\_repair` is omitted as of tibble 2.0.0.  
## Using compatibility `.name\_repair`.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_warnings()` to see where this warning was generated.

library(ggpubr)  
  
# Dibujo y coloreo por grupos KMEANS  
ggscatter(sol\_mds, x = "Dim1", y = "Dim2",   
 label = 1:nrow(sol\_mds),  
 color = "groups\_mds",  
 palette = "jco",  
 size = 1,   
 ellipse = TRUE,  
 ellipse.type = "convex",  
 repel = TRUE,  
 title="Gráfico clusters kmeans \n sobre dos el mds de dos dimensiones"  
 )



# Dibujo y coloreo por grupos clustering jerárquico ward  
ggscatter(sol\_mds, x = "Dim1", y = "Dim2",   
 label = 1:nrow(sol\_mds),  
 color = "groups\_h",  
 palette = "jco",  
 size = 1,   
 ellipse = TRUE,  
 ellipse.type = "convex",  
 repel = TRUE,  
 title="Gráfico clusters jerárquico ward\n sobre el mds de dos dimensiones"  
 )

