Ejemplo taller datos Netflix

Ricardo

05 diciembre, 2020

Contenidos

1 Datos netflix

1.1 Carga de datos

Enlace a estos datos de Netflix 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)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
#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
```

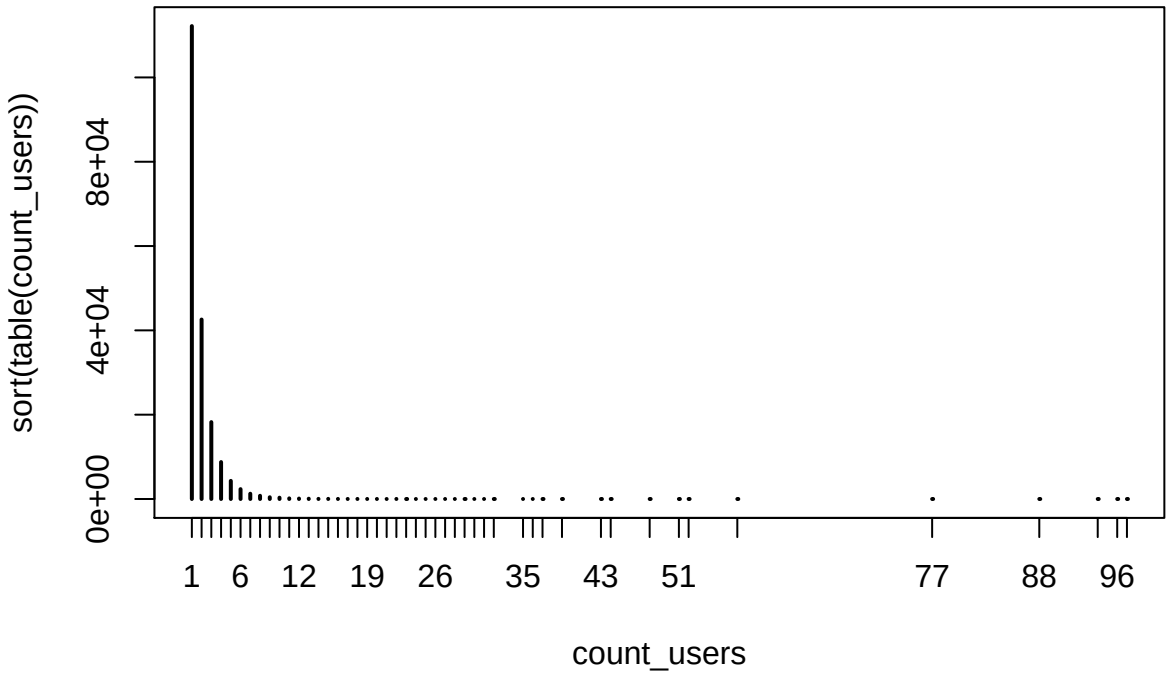
Respecto a los usuarios que han visto alguna de las películas tenemos

```
length(unique(netflix$ID_user))
```

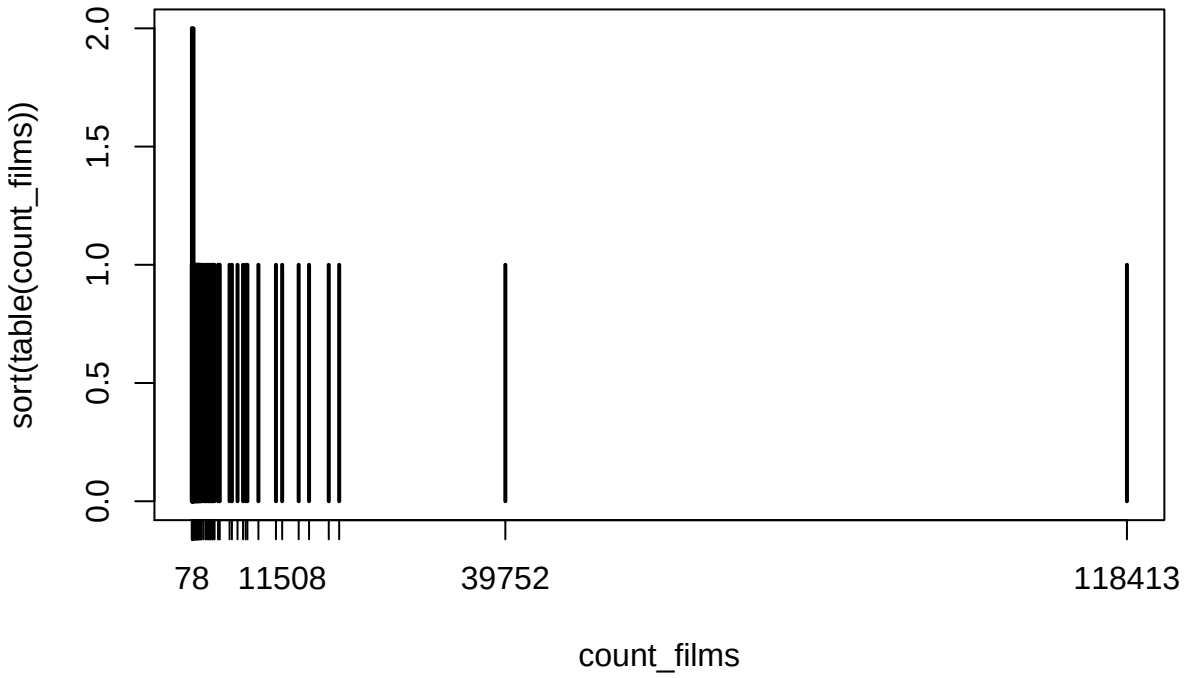
```
## [1] 191668
```

```r
table(netflix$ID_user)-> count_users
table(netflix$ID_film) -> count_films
#knitr::kable(sort(count_users))
#knitr::kable(sort( count_films))
```

```r
plot(sort(table(count_users)))
```




```r
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))))
```

1.2 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.

1.2.1 Similaridad coseno entre películas

La similaridad coseno de dos vectores de $\mathbb{R}^n$ no nulos $A = (A_1, A_2, \ldots, A_n)$ y $B = (B_1, B_2, \ldots, B_n)$

$$sim_{\cos}(A, B) = cos(A, B) = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i^2 \cdot \sum_{i=1}^{n} B_i^2}}$$

```
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
##   52.59    0.00   52.59
```

```
#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
```

1.2.2 Similitud en paralelo

```
Sim_cos_matrix2=matrix(0,ncol=100,nrow=100)
```

```r
#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'
```

```r
length(no_cores)
```

```
## [1] 8
```

```r
#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(

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 36.35848 secs
```

```
time_sim_parallel
```

```
##    user  system elapsed
##    0.01    0.00   36.23
```

```
# para el cluster
parallel::stopCluster(cl)
time_sim_parallel
```

```
##    user  system elapsed
##    0.01    0.00   36.23
```

```
time_sim
```

```
##    user  system elapsed
##   52.59    0.00   52.59
```

2 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)

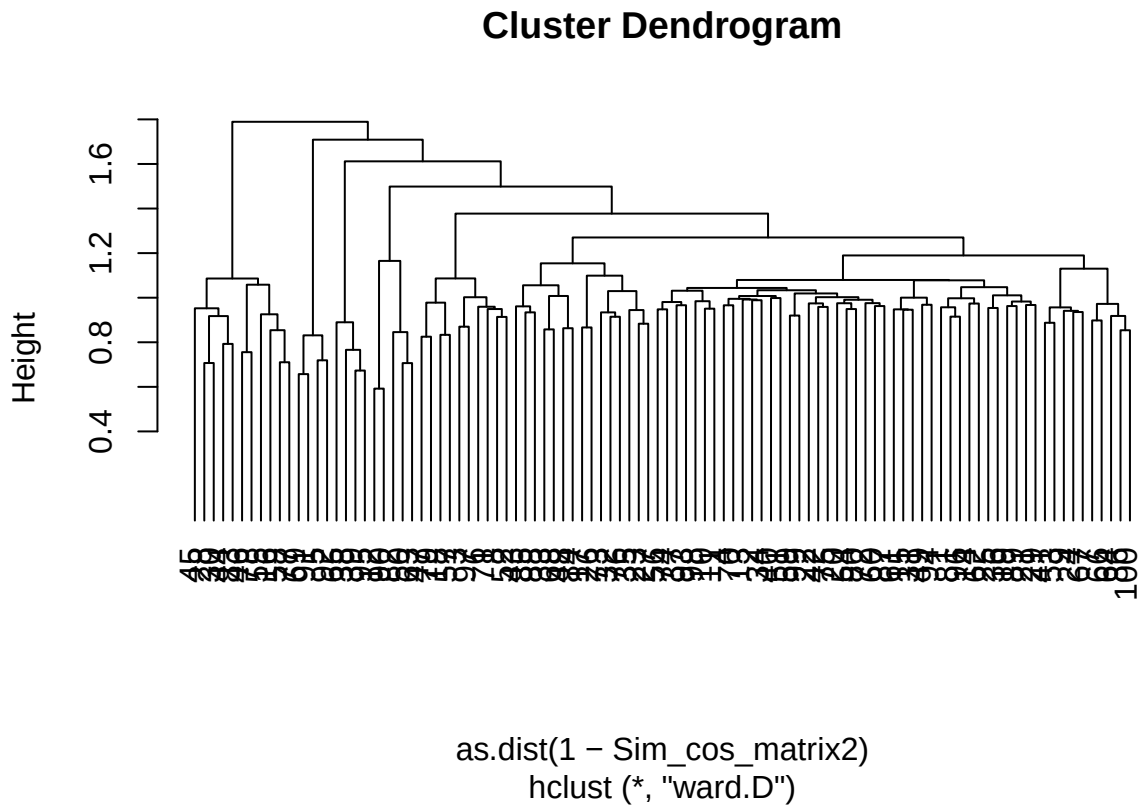
$$dist_{\cos}(A, B) = 1 - sim_{cos}(A, B).$$

2.1 Clustering jerarquico

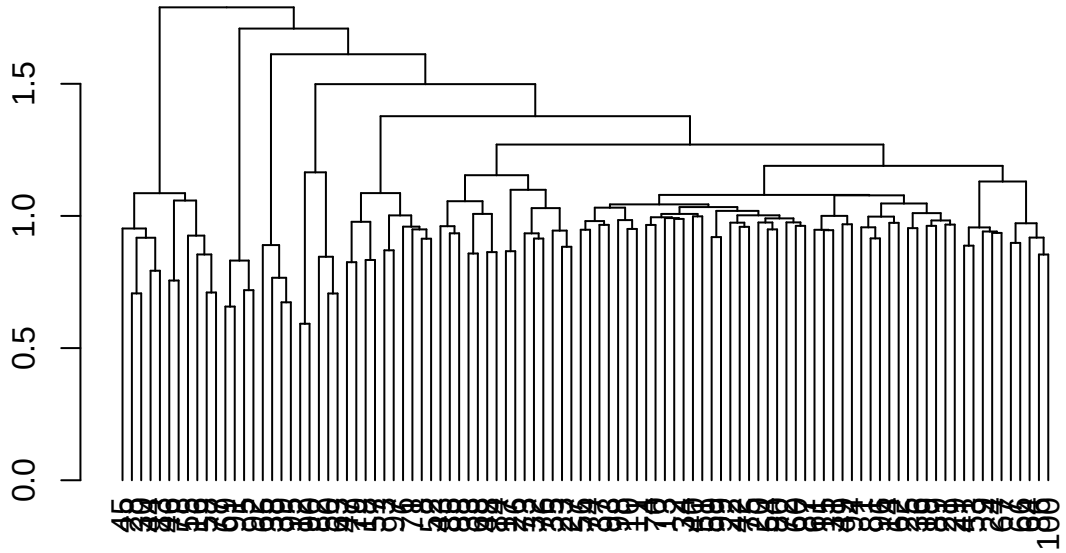
```
hclust(as.dist(1-Sim_cos_matrix2),method = "ward")-> h
```

```
## 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 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 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
```

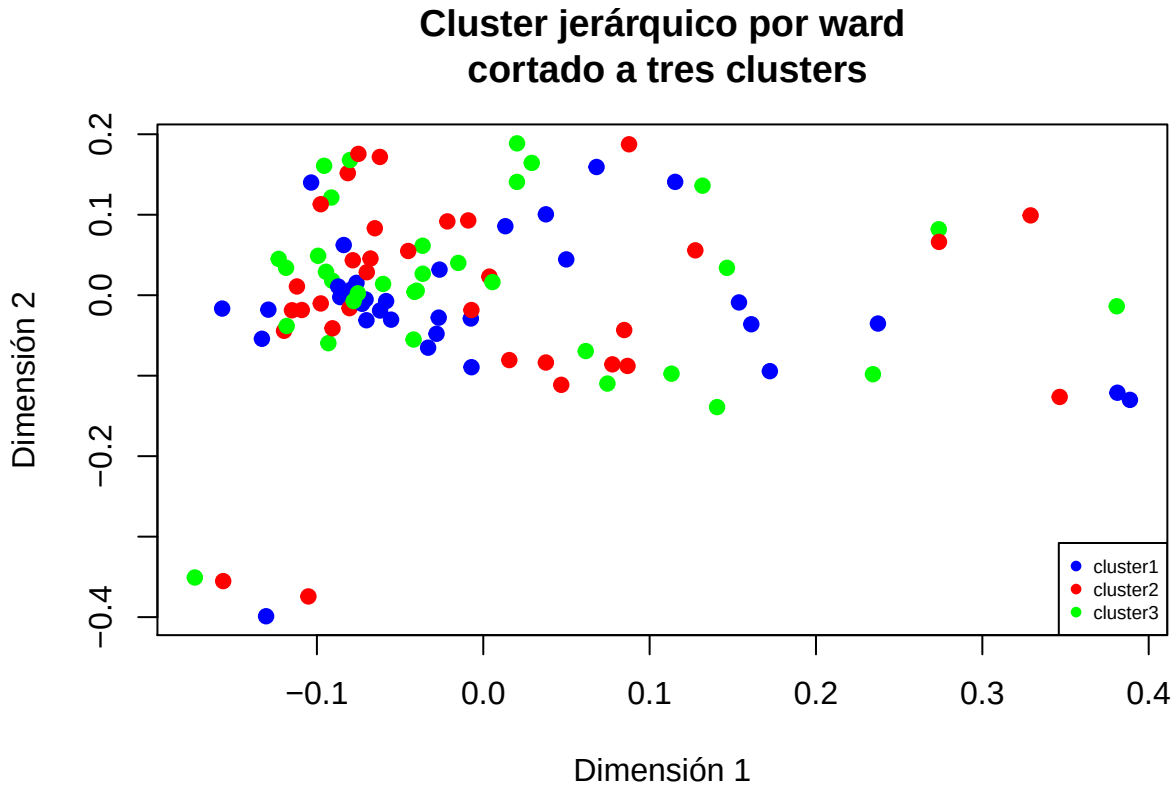
```
table(cluster_3)
```

```
## cluster_3
##  1  2  3
## 85  4 11
```

2.2 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)
```



2.3 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
##   2   2   2   2   2   2   2   1   2   2   2   3   2   2   2   1   2   1   2   2
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   2   2   2   2   2   2   2   1   2   1   2   2   2   2   2   2   2   2   2   2
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   2   2   2   1   1   1   3   1   2   3   2   2   2   3   1   2   1   1   2   2
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   2   2   3   2   2   2   2   2   2   2   2   2   3   2   2   1   1   1   3   2
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   2   3   1   1   2   2   2   2   2   3   2   2   2   2   2   2   1   2   2   2
```

```
# numero de películas por cluster
table(sol_kmeans$cluster)
```

```
##
##  1  2  3
## 18 73  9
```

```r
# matriz de confusión cluster jerárquico versus cluster de kmeans
table(cluster_3,sol_kmeans$cluster)
```

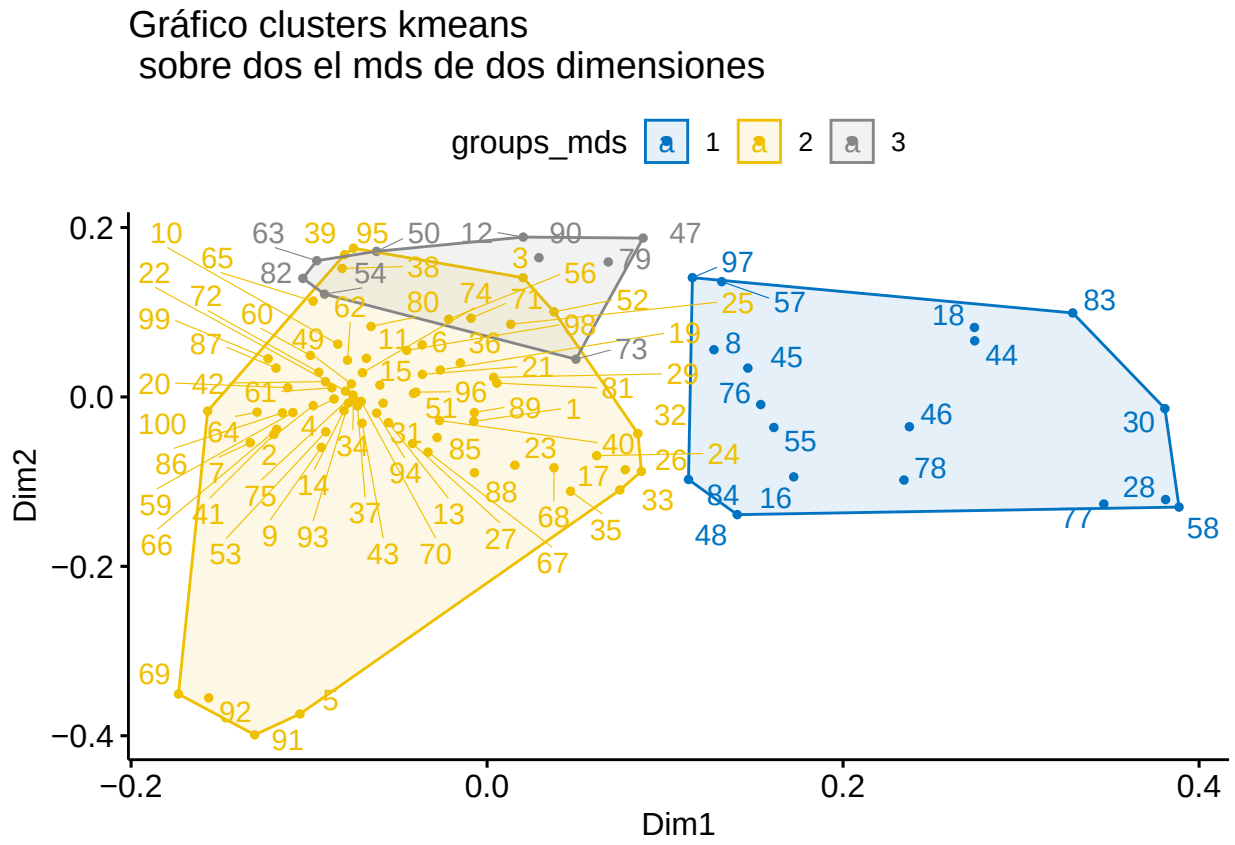
```
##
## cluster_3  1  2  3
##         1  7 69  9
##         2  0  4  0
##         3 11  0  0
```

```r
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
## Using compatibility `.name_repair`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```r
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"
          )
```

Gráfico clusters jerárquico ward
sobre el mds de dos dimensiones