Críticas de Apps en Google Play Store

Ignacio Loayza.
Miguel Huichaman.
Jorge Caullán.

15/04/2019

En esta ocasión se estudiaran las críticas a varias aplicaciones de la Google Play Store. El código para importar los datos a la base de datos, asi como también el esquema de jerarquía con el que se modelaron los datos puede ser encontrado en el repositorio del proyecto.

```
# Paquetes requeridos
list.of.packages <- c("SnowballC", "tm", "mongolite", "wordcloud", "RColorBrewer", "fpc", "cluster")
new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages)

library("SnowballC")
library("NLP")
library("tm")
library("mongolite")
library("wordcloud")
```

```
## Loading required package: RColorBrewer
```

```
library("RColorBrewer")
library("fpc")
library("cluster")



if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install()
```

```
## Bioconductor version 3.8 (BiocManager 1.30.4), R 3.5.1 (2018-07-02)
```

```
## Update old packages: 'assertthat', 'backports', 'BH', 'broom', 'caTools',
##   'class', 'cli', 'cluster', 'codetools', 'colorspace', 'curl', 'dbplyr',
##   'digest', 'dplyr', 'evaluate', 'fansi', 'forcats', 'ggplot2', 'glue',
##   'gtable', 'haven', 'highr', 'htmlwidgets', 'httpuv', 'httr',
##   'IRdisplay', 'IRkernel', 'jsonlite', 'knitr', 'later', 'lattice',
##   'lazyeval', 'markdown', 'MASS', 'Matrix', 'mgcv', 'mime', 'mongolite',
##   'nlme', 'NLP', 'odbc', 'packrat', 'pillar', 'pkgconfig', 'polyclip',
##   'purrr', 'R6', 'RCurl', 'readr', 'readxl', 'repr', 'rJava', 'RJSONIO',
##   'rlang', 'rmarkdown', 'rpart', 'rsconnect', 'rstudioapi', 'shiny',
##   'slam', 'SnowballC', 'sparklyr', 'stringi', 'stringr', 'survival',
##   'tibble', 'tidyr', 'tidyselect', 'tinytex', 'tm', 'xfun', 'xtable'
```

```
BiocManager::install("graph", version = "3.8")
```

```
## Bioconductor version 3.8 (BiocManager 1.30.4), R 3.5.1 (2018-07-02)
```

```
## Installing package(s) 'graph'
```

```
## Updating HTML index of packages in '.Library'
```

```
## Making 'packages.html' ... done
```

```
## Update old packages: 'assertthat', 'backports', 'BH', 'broom', 'caTools',
##   'class', 'cli', 'cluster', 'codetools', 'colorspace', 'curl', 'dbplyr',
##   'digest', 'dplyr', 'evaluate', 'fansi', 'forcats', 'ggplot2', 'glue',
##   'gtable', 'haven', 'highr', 'htmlwidgets', 'httpuv', 'httr',
##   'IRdisplay', 'IRkernel', 'jsonlite', 'knitr', 'later', 'lattice',
##   'lazyeval', 'markdown', 'MASS', 'Matrix', 'mgcv', 'mime', 'mongolite',
##   'nlme', 'NLP', 'odbc', 'packrat', 'pillar', 'pkgconfig', 'polyclip',
##   'purrr', 'R6', 'RCurl', 'readr', 'readxl', 'repr', 'rJava', 'RJSONIO',
##   'rlang', 'rmarkdown', 'rpart', 'rsconnect', 'rstudioapi', 'shiny',
##   'slam', 'SnowballC', 'sparklyr', 'stringi', 'stringr', 'survival',
##   'tibble', 'tidyr', 'tidyselect', 'tinytex', 'tm', 'xfun', 'xtable'
```

```
BiocManager::install("Rgraphviz", version = "3.8")
```

```
## Bioconductor version 3.8 (BiocManager 1.30.4), R 3.5.1 (2018-07-02)
## Installing package(s) 'Rgraphviz'
```

```
## Warning in install.packages(pkgs = doing, lib = lib, repos = repos, ...):
## installation of package 'Rgraphviz' had non-zero exit status
```

```
## Updating HTML index of packages in '.Library'
## Making 'packages.html' ... done
## Update old packages: 'assertthat', 'backports', 'BH', 'broom', 'caTools',
##   'class', 'cli', 'cluster', 'codetools', 'colorspace', 'curl', 'dbplyr',
##   'digest', 'dplyr', 'evaluate', 'fansi', 'forcats', 'ggplot2', 'glue',
##   'gtable', 'haven', 'highr', 'htmlwidgets', 'httpuv', 'httr',
##   'IRdisplay', 'IRkernel', 'jsonlite', 'knitr', 'later', 'lattice',
##   'lazyeval', 'markdown', 'MASS', 'Matrix', 'mgcv', 'mime', 'mongolite',
##   'nlme', 'NLP', 'odbc', 'packrat', 'pillar', 'pkgconfig', 'polyclip',
##   'purrr', 'R6', 'RCurl', 'readr', 'readxl', 'repr', 'rJava', 'RJSONIO',
##   'rlang', 'rmarkdown', 'rpart', 'rsconnect', 'rstudioapi', 'shiny',
##   'slam', 'SnowballC', 'sparklyr', 'stringi', 'stringr', 'survival',
##   'tibble', 'tidyr', 'tidyselect', 'tinytex', 'tm', 'xfun', 'xtable'
```

```
library("Rgraphviz")
```

```
## Loading required package: graph
## Loading required package: BiocGenerics
## Loading required package: parallel
##
## Attaching package: 'BiocGenerics'
##
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
##
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
##
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind,
##     colMeans, colnames, colSums, dirname, do.call, duplicated,
```

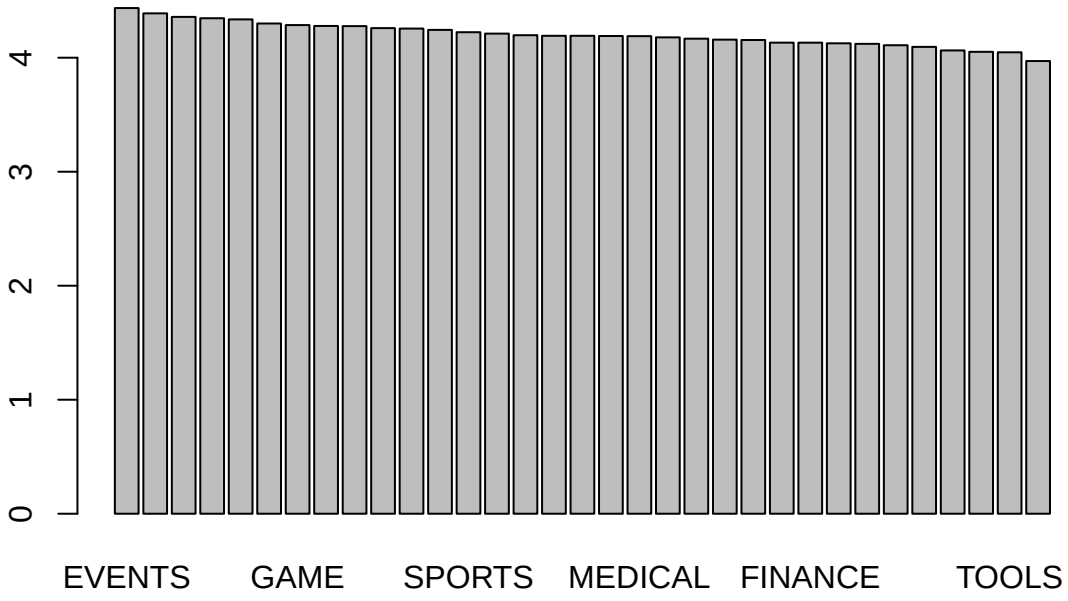
```
##     eval, evalq, Filter, Find, get, grep, grepl, intersect,
##     is.unsorted, lapply, lengths, Map, mapply, match, mget, order,
##     paste, pmax, pmax.int, pmin, pmin.int, Position, rank, rbind,
##     Reduce, rowMeans, rownames, rowSums, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which, which.max,
##     which.min
##
## Loading required package: grid
```

```
categories_mongo <- mongo(collection = "categories", db = "tarea1BDA", url = 'mongodb://127.0.0.1:27017
categories <- categories_mongo$find('{}','{"category":1, "_id":0}') # Con esto tendremos una lista con
```

1.- ¿Qué categorias tienen mayor ranking promedio?

Nos interesa conocer el puntaje promedio de cada categoría y ordernarlas para encontrars aquellas que poseen mayor puntaje promedio.

```
df_ratings <- data.frame(categ = as.character(character()), mean_rating = as.character(character()), str

for (cat in categories$category){
  category_name <- cat
  query_cat <- sprintf('{"category":"%s"}',category_name)
  cat_ratings <- categories_mongo$find(query_cat,'{"apps.Rating":1, "_id":0}')
  category_mean_rating <- mean(as.numeric(as.character(unlist(cat_ratings[[1]]))))
  #temp_list <- list(categ=as.character(category_name), mean_rating=category_mean_rating)
  temp_df <- data.frame(categ = as.character(category_name), mean_rating=category_mean_rating, stringsAs
  df_ratings <- rbind(df_ratings,temp_df)
}

newdata <- df_ratings[order(df_ratings$mean_rating, decreasing = TRUE),]
barplot(newdata[,2], names.arg = newdata[,1])  # Barplot horrible
```




```
print(newdata[0:3,])
```

```
##            categ mean_rating
```

```
## 11          EVENTS    4.435556
## 9        EDUCATION    4.389032
## 1  ART_AND_DESIGN    4.358065
```

Las categorías de apps con mayor puntaje promedio son *"Eventos"*, *"Educación"* y *"Arte y diseño"*, hay que notar, sin embargo, que la media del puntaje de las categorias es 4.202 y con una desviación estándar de 0.106 y una media mínima de 3.971, lo que hace poco distinguibles las categorías en cuanto a puntaje, además de que en general todas las categorías tienen un puntaje medio cercano al máximo.

```
summary(newdata$mean_rating)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.971   4.132   4.192   4.202   4.277   4.436
```

```
sd(newdata$mean_rating)
```

```
## [1] 0.1056063
```

2.- Términos descriptivos a los que más se hace alusión

Ahora, se estudiará la repetición de términos considerando todas las reviews disponibles.

Primero almacenaremos los textos de todas las reviews en formato .txt, separadas inmediatamente por sentimiento, para poder conformar de manera formal el corpus:

```
positive <- categories_mongo$find('{}','{"cumulated_reviews.positives":1, "_id":0}')
negative <- categories_mongo$find('{}','{"cumulated_reviews.negatives":1, "_id":0}')
neutral <- categories_mongo$find('{}','{"cumulated_reviews.neutrals":1, "_id":0}')

#all reviews
rev_all <- c(positive[,1],negative[,1],neutral[,1])
vector_reviews <- VectorSource(rev_all)
```

Cargamos el corpus leyendo los textos de todas las reviews.

```
docs <- VCorpus(vector_reviews)

# Transformador de espacios
toSpace <- content_transformer(function(x, pattern) {return (gsub(pattern, " ", x))})

# Limpieza de puntuacion
# Usar el transformador anterior para eliminar comas, dos puntos y otros...
docs <- tm_map(docs, toSpace, "-")
docs <- tm_map(docs, toSpace, ":")
docs <- tm_map(docs, removePunctuation)
docs <- tm_map(docs, toSpace, "’")
docs <- tm_map(docs, toSpace, "‘")
docs <- tm_map(docs, toSpace, " -")

# There are some parasite words that need cleaning
docs <- tm_map(docs, toSpace, "data frame")
docs <- tm_map(docs, toSpace, "Translated_Review")
docs <- tm_map(docs, toSpace, "datafram")

# Transformar todo a minusculas
docs <- tm_map(docs,content_transformer(tolower))
```

```r
# Eliminar digitos
docs <- tm_map(docs, removeNumbers)

# Remover stopwords usando la lista estándar de tm
docs <- tm_map(docs, removeWords, stopwords("english"))

# Borrar todos los espacios en blanco extraños
docs <- tm_map(docs, stripWhitespace)

# Stemming
docs <- tm_map(docs,stemDocument)
#writeLines(as.character(docs[[1]]))

# Lemmatization (toma en cuenta el contexto) ... podrían hacerse varias más
docs <- tm_map(docs, content_transformer(gsub), pattern = "organiz", replacement = "organ")
docs <- tm_map(docs, content_transformer(gsub), pattern = "organis", replacement = "organ")
docs <- tm_map(docs, content_transformer(gsub), pattern = "andgovern", replacement = "govern")
docs <- tm_map(docs, content_transformer(gsub), pattern = "inenterpris", replacement = "enterpris")
docs <- tm_map(docs, content_transformer(gsub), pattern = "team-", replacement = "team")

#writeLines(as.character(docs[[1]]))

# Matriz de documentos - términos (MDT)
dtm <- DocumentTermMatrix(docs, control=list(wordLengths=c(4, 20)))
# Matriz de 30 x 4200, en la cual un 89% de filas son cero
dtm
```

```
## <<DocumentTermMatrix (documents: 3, terms: 14215)>>
## Non-/sparse entries: 21089/21556
## Sparsity           : 51%
## Maximal term length: 20
## Weighting          : term frequency (tf)
```

```r
freq <- colSums(as.matrix(dtm))
ord <- order(freq,decreasing=TRUE)
print("Términos más frecuentes: ")
```

```
## [1] "Términos más frecuentes: "
```

```r
print(freq[head(ord)])
```

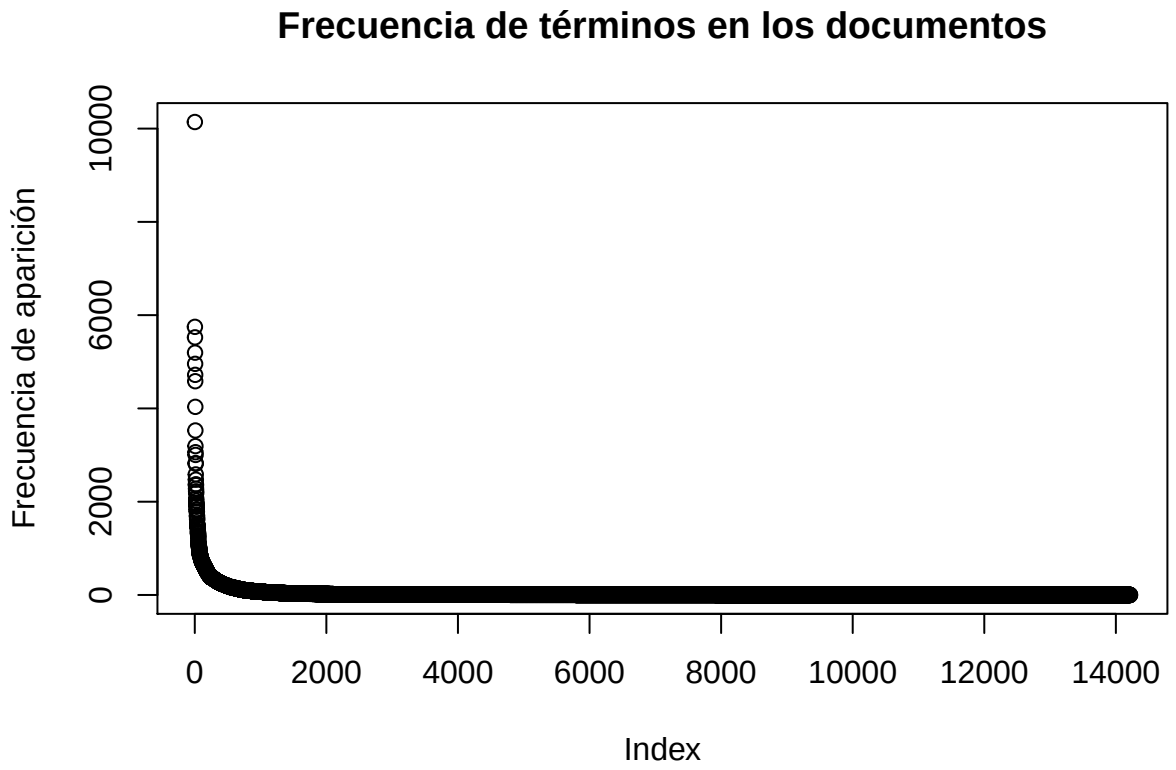
```
##  game  like  time  love  good great
## 10141  5749  5524  5196  4956  4720
```

Se puede observar que los términos más frecuentes en todas las reviews suelen ser positivos, esto puede explicar los altos puntajes que tienen las medias de los puntajes de todas las categorías.

También se puede notar que de los seis términos mostrados, los cuatro que se relacionan con un sentiemiento positivo son palabras que expresan este sentiemiento de forma marcada, siendo de hecho *"love"* y *"great"* los segundo y terceros términos que más se repiten.

El hecho de que la palabra *"game"* sea la que más se repite no aporta mayor información pues el contexto esta implícito.

```r
plot(freq[ord], main = "Frecuencia de términos en los documentos", ylab="Frecuencia de aparición")
```

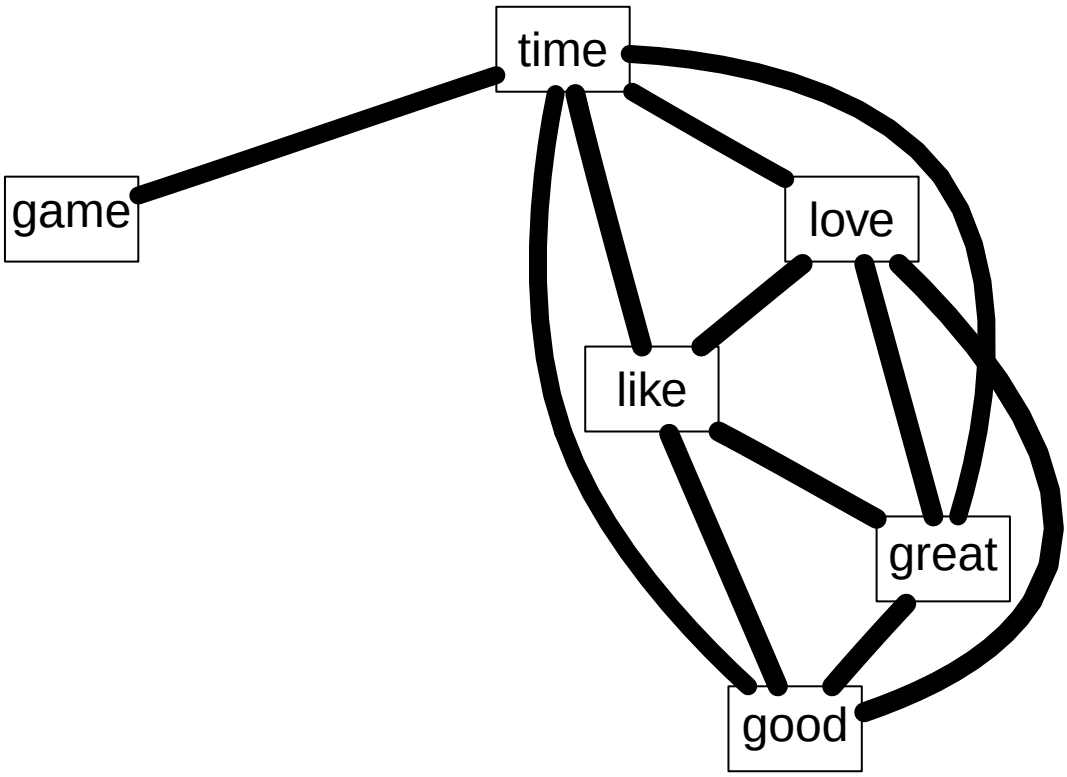


Se puede observar la frecuencia de términos en el documento, la cual como es de esperarse, se manifiesta de acuerdo a la Ley de Zipf: Algunos pocos términos son exponencialmente más frecuentes que los demás del documento.

3) Relaciones entre términos descriptivos

A continuación se presenta el gráfico de relación entre términos presentes en los documentos.

```
freq.terms <- findFreqTerms(dtm,lowfreq=4700)
plot(dtm, term = freq.terms, corThreshold = 0.90, weighting = T)
```



Se puede observar que los términos comunes estan fuertemente relacionados (correlation threshold = 0.90) en cuanto a ocurrencia conjunta en los reviews pues todas las aristas se muestran en negrita. Se puede ver que la palabra `game` solo esta fuertemente relacionada con `time`, probablemente debido a los reviews que comentan sobre el tiempo de juego. También se puede notar que existe una clique entre todos los términos, exceptuado `game`, y dichos términos son casi en su totalidad asociados a sentimientos positivos, luego, el conjunto de términos con mayor repetición en los reviews estan casi todos asociados a sentimientos positivos y tienen una ocurrencia altamente correlacionada.

4) Términos más repetitivos 3 categorías diferentes

Cargamos el corpus leyendo los textos de todas las reviews de las categorías ART_AND_DESIGN, EDUCATION, FINANCE.

4.1) Categoria *Art and Design*

```
positives_art <- categories_mongo$find('{"category": "ART_AND_DESIGN"}','{"cumulated_reviews.positives"
negatives_art <- categories_mongo$find('{"category": "ART_AND_DESIGN"}','{"cumulated_reviews.negatives"
neutrals_art <- categories_mongo$find('{"category": "ART_AND_DESIGN"}','{"cumulated_reviews.neutrals":1

rev_3_art <- c(positives_art[,1],negatives_art[,1],neutrals_art[,1])
vector_3_art_reviews <- VectorSource(rev_3_art)
```

Cargamos el corpus leyendo los textos de todas las reviews de estas categorías.

```
docs3 <- VCorpus(vector_3_art_reviews)

# Transformador de espacios
toSpace <- content_transformer(function(x, pattern) {return (gsub(pattern, " ", x))})
```

```r
# Limpieza de puntuacion
# Usar el transformador anterior para eliminar comas, dos puntos y otros...
docs3 <- tm_map(docs3, toSpace, "-")
docs3 <- tm_map(docs3, toSpace, ":")
docs3 <- tm_map(docs3, removePunctuation)
docs3 <- tm_map(docs3, toSpace, "’")
docs3 <- tm_map(docs3, toSpace, "‘")
docs3 <- tm_map(docs3, toSpace, " -")

# There are some parasite words that need cleaning
docs3 <- tm_map(docs3, toSpace, "data frame")
docs3 <- tm_map(docs3, toSpace, "Translated_Review")
docs3 <- tm_map(docs3, toSpace, "datafram")

# Transformar todo a minusculas
docs3 <- tm_map(docs3,content_transformer(tolower))

# Eliminar digitos
docs3 <- tm_map(docs3, removeNumbers)

# Remover stopwords usando la lista estándar de tm
docs3 <- tm_map(docs3, removeWords, stopwords("english"))

# Borrar todos los espacios en blanco extraños
docs3 <- tm_map(docs3, stripWhitespace)

# Stemming
docs3 <- tm_map(docs3,stemDocument)
#writeLines(as.character(docs[[1]]))

# Lemmatization (toma en cuenta el contexto) ... podrían hacerse varias más
docs3 <- tm_map(docs3, content_transformer(gsub), pattern = "organiz", replacement = "organ")
docs3 <- tm_map(docs3, content_transformer(gsub), pattern = "organis", replacement = "organ")
docs3 <- tm_map(docs3, content_transformer(gsub), pattern = "andgovern", replacement = "govern")
docs3 <- tm_map(docs3, content_transformer(gsub), pattern = "inenterpris", replacement = "enterpris")
docs3 <- tm_map(docs3, content_transformer(gsub), pattern = "team-", replacement = "team")

#writeLines(as.character(docs[[1]]))

# Matriz de documentos - términos (MDT)
dtm3 <- DocumentTermMatrix(docs3, control=list(wordLengths=c(4, 20)))
# Matriz de 30 x 4200, en la cual un 89% de filas son cero
dtm3
```

```
## <<DocumentTermMatrix (documents: 3, terms: 914)>>
## Non-/sparse entries: 1147/1595
## Sparsity           : 58%
## Maximal term length: 20
## Weighting          : term frequency (tf)
```

```r
freq3 <- colSums(as.matrix(dtm3))
ord3 <- order(freq3,decreasing=TRUE)
print("Términos más frecuentes para la categoría `Art and Design`: ")
```

```
## [1] "Términos más frecuentes para la categoría `Art and Design`: "
```

```
print(freq3[head(ord3)])
```

```
##  color   love   good   like pictur   make
##     99     84     64     60     55     39
```

Se puede ver que los términos más comunes para la cetagoría de arte y diseño son principalemente términos asociados a sentimientos positivos, con la excepción de los términos `make`, `color` y `picture`, además, el término más repetido es `color`, el cual tiene bastante relación con la categoría en sí por lo que es esperable.

4.2) Categoría *Education*:

```
positives_edu <- categories_mongo$find('{"category": "EDUCATION"}','{"cumulated_reviews.positives":1, "
negatives_edu <- categories_mongo$find('{"category": "EDUCATION"}','{"cumulated_reviews.negatives":1, "
neutrals_edu <- categories_mongo$find('{"category": "EDUCATION"}','{"cumulated_reviews.neutrals":1, "_i

rev_3_edu <- c(positives_edu[,1],negatives_edu[,1],neutrals_edu[,1])
vector_3_edu_reviews <- VectorSource(rev_3_edu)


docs3 <- VCorpus(vector_3_edu_reviews)

# Transformador de espacios
toSpace <- content_transformer(function(x, pattern) {return (gsub(pattern, " ", x))})

# Limpieza de puntuacion
# Usar el transformador anterior para eliminar comas, dos puntos y otros...
docs3 <- tm_map(docs3, toSpace, "-")
docs3 <- tm_map(docs3, toSpace, ":")
docs3 <- tm_map(docs3, removePunctuation)
docs3 <- tm_map(docs3, toSpace, "’")
docs3 <- tm_map(docs3, toSpace, "‘")
docs3 <- tm_map(docs3, toSpace, " -")

# There are some parasite words that need cleaning
docs3 <- tm_map(docs3, toSpace, "data frame")
docs3 <- tm_map(docs3, toSpace, "Translated_Review")
docs3 <- tm_map(docs3, toSpace, "datafram")

# Transformar todo a minusculas
docs3 <- tm_map(docs3,content_transformer(tolower))

# Eliminar digitos
docs3 <- tm_map(docs3, removeNumbers)

# Remover stopwords usando la lista estándar de tm
docs3 <- tm_map(docs3, removeWords, stopwords("english"))

# Borrar todos los espacios en blanco extraños
docs3 <- tm_map(docs3, stripWhitespace)

# Stemming
docs3 <- tm_map(docs3,stemDocument)
```

```
#writeLines(as.character(docs[[1]]))

# Lemmatization (toma en cuenta el contexto) ... podrían hacerse varias más
docs3 <- tm_map(docs3, content_transformer(gsub), pattern = "organiz", replacement = "organ")
docs3 <- tm_map(docs3, content_transformer(gsub), pattern = "organis", replacement = "organ")
docs3 <- tm_map(docs3, content_transformer(gsub), pattern = "andgovern", replacement = "govern")
docs3 <- tm_map(docs3, content_transformer(gsub), pattern = "inenterpris", replacement = "enterpris")
docs3 <- tm_map(docs3, content_transformer(gsub), pattern = "team-", replacement = "team")

#writeLines(as.character(docs[[1]]))

# Matriz de documentos - términos (MDT)
dtm3 <- DocumentTermMatrix(docs3, control=list(wordLengths=c(4, 20)))
# Matriz de 30 x 4200, en la cual un 89% de filas son cero
dtm3
```

```
## <<DocumentTermMatrix (documents: 3, terms: 1578)>>
## Non-/sparse entries: 2222/2512
## Sparsity           : 53%
## Maximal term length: 20
## Weighting          : term frequency (tf)
```

```
freq3 <- colSums(as.matrix(dtm3))
ord3 <- order(freq3,decreasing=TRUE)
print("Términos más frecuentes para la categoría `Education`: ")
```

```
## [1] "Términos más frecuentes para la categoría `Education`: "
```

```
print(freq3[head(ord3)])
```

```
##   learn languag    help    good    word   great
##     345     244     188     183     134     124
```

Se puede observar que el término más común es **learn** el cual tiene directa relación con la categoría, además, se puede ver nuevamente que hay términos asociados con sentimientos positivos como **great** y **good**, si embargo, en menor cantidad que para la categoría anterior. El término **help** es interesante pues puede tener relación con algún aspecto positivo que se esté tratando de explicar en una review, por ejemplo, "This app helped me to . . . ".

4.3) Categoría *Finance*:

```
positives_fin <- categories_mongo$find('{"category": "FINANCE"}','{"cumulated_reviews.positives":1, "_id
negatives_fin <- categories_mongo$find('{"category": "FINANCE"}','{"cumulated_reviews.negatives":1, "_id
neutrals_fin <- categories_mongo$find('{"category": "FINANCE"}','{"cumulated_reviews.neutrals":1, "_id"

rev_3_fin <- c(positives_fin[,1],negatives_fin[,1],neutrals_fin[,1])
vector_3_fin_reviews <- VectorSource(rev_3_fin)


docs3 <- VCorpus(vector_3_fin_reviews)

# Transformador de espacios
toSpace <- content_transformer(function(x, pattern) {return (gsub(pattern, " ", x))})

# Limpieza de puntuacion
```

```r
# Usar el transformador anterior para eliminar comas, dos puntos y otros...
docs3 <- tm_map(docs3, toSpace, "-")
docs3 <- tm_map(docs3, toSpace, ":")
docs3 <- tm_map(docs3, removePunctuation)
docs3 <- tm_map(docs3, toSpace, "’")
docs3 <- tm_map(docs3, toSpace, "‘")
docs3 <- tm_map(docs3, toSpace, " -")

# There are some parasite words that need cleaning
docs3 <- tm_map(docs3, toSpace, "data frame")
docs3 <- tm_map(docs3, toSpace, "Translated_Review")
docs3 <- tm_map(docs3, toSpace, "datafram")

# Transformar todo a minusculas
docs3 <- tm_map(docs3,content_transformer(tolower))

# Eliminar digitos
docs3 <- tm_map(docs3, removeNumbers)

# Remover stopwords usando la lista estándar de tm
docs3 <- tm_map(docs3, removeWords, stopwords("english"))

# Borrar todos los espacios en blanco extraños
docs3 <- tm_map(docs3, stripWhitespace)

# Stemming
docs3 <- tm_map(docs3,stemDocument)
#writeLines(as.character(docs[[1]]))

# Lemmatization (toma en cuenta el contexto) ... podrían hacerse varias más
docs3 <- tm_map(docs3, content_transformer(gsub), pattern = "organiz", replacement = "organ")
docs3 <- tm_map(docs3, content_transformer(gsub), pattern = "organis", replacement = "organ")
docs3 <- tm_map(docs3, content_transformer(gsub), pattern = "andgovern", replacement = "govern")
docs3 <- tm_map(docs3, content_transformer(gsub), pattern = "inenterpris", replacement = "enterpris")
docs3 <- tm_map(docs3, content_transformer(gsub), pattern = "team-", replacement = "team")

#writeLines(as.character(docs[[1]]))

# Matriz de documentos - términos (MDT)
dtm3 <- DocumentTermMatrix(docs3, control=list(wordLengths=c(4, 20)))
# Matriz de 30 x 4200, en la cual un 89% de filas son cero
dtm3
```

```
## <<DocumentTermMatrix (documents: 3, terms: 2372)>>
## Non-/sparse entries: 3763/3353
## Sparsity           : 47%
## Maximal term length: 20
## Weighting          : term frequency (tf)
```

```r
freq3 <- colSums(as.matrix(dtm3))
ord3 <- order(freq3,decreasing=TRUE)
print("Términos más frecuentes para la categoría `Finance`: ")
```

```
## [1] "Términos más frecuentes para la categoría `Finance`: "
```

```r
print(freq3[head(ord3)])
```

```
##    work    card    time    bank   updat account
##     326     286     274     273     258     244
```

En el caso de la categoría ***Finanzas*** no hay términos asociados a ningún sentimiento en particular, en lugar de eso, parece ser que los términos más populares son utilizaos en descripciónes utilitarias de las apps y características de las mismas. El término más común es `work` y tiene bastante sentido considerando que probablemente la mayoría de estas aplicaciones estan pensadas para asistir a las personas en el trabajo.

5) Terminos mas comunes en comentarios negativos y positivos

De una manera analoga a la anterior, se cargan los comentarios positivos y negativos de todas las categorias

```r
positives <- categories_mongo$find('{}','{"cumulated_reviews.positives":1, "_id":0}')
negatives <- categories_mongo$find('{}','{"cumulated_reviews.negatives":1, "_id":0}')

vector_positive_reviews <- VectorSource(c(positives[,1]))
vector_negative_reviews <- VectorSource(c(negatives[,1]))

Pdocs <- VCorpus(vector_positive_reviews)
Ndocs <- VCorpus(vector_negative_reviews)

# Limpieza de puntuacion
# Usar el transformador anterior para eliminar comas, dos puntos y otros...
Pdocs <- tm_map(Pdocs, toSpace, "-")
Pdocs <- tm_map(Pdocs, toSpace, ":")
Pdocs <- tm_map(Pdocs, removePunctuation)
Pdocs <- tm_map(Pdocs, toSpace, "’")
Pdocs <- tm_map(Pdocs, toSpace, "‘")
Pdocs <- tm_map(Pdocs, toSpace, " -")

Ndocs <- tm_map(Ndocs, toSpace, "-")
Ndocs <- tm_map(Ndocs, toSpace, ":")
Ndocs <- tm_map(Ndocs, removePunctuation)
Ndocs <- tm_map(Ndocs, toSpace, "’")
Ndocs <- tm_map(Ndocs, toSpace, "‘")
Ndocs <- tm_map(Ndocs, toSpace, " -")

# There are some parasite words that need cleaning
Pdocs <- tm_map(Pdocs, toSpace, "data frame")
Pdocs <- tm_map(Pdocs, toSpace, "Translated_Review")
Pdocs <- tm_map(Pdocs, toSpace, "datafram")

Ndocs <- tm_map(Ndocs, toSpace, "data frame")
Ndocs <- tm_map(Ndocs, toSpace, "Translated_Review")
Ndocs <- tm_map(Ndocs, toSpace, "datafram")

# Transformar todo a minusculas
Pdocs <- tm_map(Pdocs,content_transformer(tolower))
Ndocs <- tm_map(Ndocs,content_transformer(tolower))

# Eliminar digitos
Pdocs <- tm_map(Pdocs, removeNumbers)
```

```
Ndocs <- tm_map(Ndocs, removeNumbers)

# Remover stopwords usando la lista est?ndar de tm
Pdocs <- tm_map(Pdocs, removeWords, stopwords("english"))
Ndocs <- tm_map(Ndocs, removeWords, stopwords("english"))

# Borrar todos los espacios en blanco extraños
Pdocs <- tm_map(Pdocs, stripWhitespace)
Ndocs <- tm_map(Ndocs, stripWhitespace)

# Stemming
Pdocs <- tm_map(Pdocs,stemDocument)
Ndocs <- tm_map(Ndocs,stemDocument)

# Lemmatization (toma en cuenta el contexto) ... podr?an hacerse varias m?s
Pdocs <- tm_map(Pdocs, content_transformer(gsub), pattern = "organiz", replacement = "organ")
Pdocs <- tm_map(Pdocs, content_transformer(gsub), pattern = "organis", replacement = "organ")
Pdocs <- tm_map(Pdocs, content_transformer(gsub), pattern = "andgovern", replacement = "govern")
Pdocs <- tm_map(Pdocs, content_transformer(gsub), pattern = "inenterpris", replacement = "enterpris")
Pdocs <- tm_map(Pdocs, content_transformer(gsub), pattern = "team-", replacement = "team")

Ndocs <- tm_map(Ndocs, content_transformer(gsub), pattern = "organiz", replacement = "organ")
Ndocs <- tm_map(Ndocs, content_transformer(gsub), pattern = "organis", replacement = "organ")
Ndocs <- tm_map(Ndocs, content_transformer(gsub), pattern = "andgovern", replacement = "govern")
Ndocs <- tm_map(Ndocs, content_transformer(gsub), pattern = "inenterpris", replacement = "enterpris")
Ndocs <- tm_map(Ndocs, content_transformer(gsub), pattern = "team-", replacement = "team")

# Matriz de documentos - t?rminos (MDT)
Pdtm <- DocumentTermMatrix(Pdocs, control=list(wordLengths=c(4, 20)))
Ndtm <- DocumentTermMatrix(Ndocs, control=list(wordLengths=c(4, 20)))
# Matriz de 30 x 4200, en la cual un 89% de filas son cero
Pdtm
```

```
## <<DocumentTermMatrix (documents: 1, terms: 11148)>>
## Non-/sparse entries: 11148/0
## Sparsity           : 0%
## Maximal term length: 20
## Weighting          : term frequency (tf)
```

```
Ndtm
```

```
## <<DocumentTermMatrix (documents: 1, terms: 6321)>>
## Non-/sparse entries: 6321/0
## Sparsity           : 0%
## Maximal term length: 20
## Weighting          : term frequency (tf)
```

Los terminos de mayor Frecuencia de los terminos positivos estan dados por:

```
freqp <- colSums(as.matrix(Pdtm))
ordn <- order(freqp,decreasing=TRUE)
print(freqp[head(ordn)])
```

```
##  game  love  good great  like  time
##  5522  4690  4556  4462  4018  3450
```

El término más común en el caso de los comentarios positivos es `game`, lo que nos indica que al parecer la

mayoría de las reviews positivas se pueden encontrar en aplicaciones de juegos. Por otro lado, están varios términos ya vistos que hacen relación con la emocionalidad positiva del review. Finalmente, se puede ver que `time` esta asociado con reviews de sentimiento positivo.

Y los terminos de mayor Frecuencia de los terminos negativos estan dados por:

```
freqn <- colSums(as.matrix(Ndtm))
ordn <- order(freqn,decreasing=TRUE)
print(freqn[head(ordn)])
```

```
##  game  time  play  like  cant updat
##  4517  1753  1500  1310  1072  1032
```

En el caso de los reviews de sentimiento negativo se puede observar que curiosamente, al igual que en aquellos de sentimiento positivo, el término más común es `game`, esto nos hace contraste con la conclusión anterior puesto que gran parte de los reviews negativos hacen alusión a juegos también, concluimos entonces que los juegos son un tipo de aplicación con reviews bastante dispersos y extremos. Otro términos encontrado: `play`, también hacen alusión a juegos.

El resto de los términos curiosamente no expresan sentimientos negativos, probablemente tengan una connotación negativa en conjunto con otros términos de la misma oración.

6) Nube de palabras con N términos más comunes

Se decidió usar un N = 50 para los términos más comunes, los que serán obtenidos del primer análisis de frecuencias.

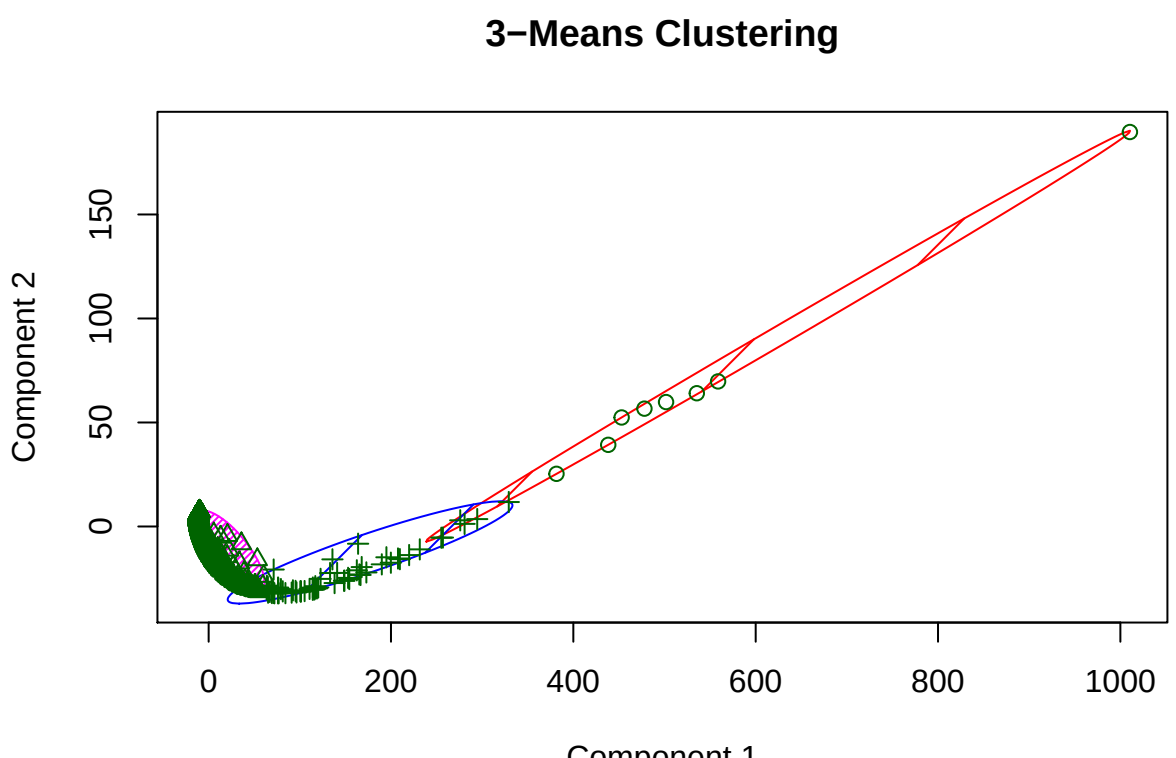
```
# Setear un valor semilla
set.seed(42)
# Nube de palabras en blanco y negro; palabras con frecuencia mínima de 1000
wordcloud(names(freq), freq, min.freq=1000, max.words= 60, colors=brewer.pal(6,"Dark2"))
```

Podemos ver claramente que los términos representativos que hemos encontrado en el análisis de sentimientos están presentes con mayor tamaño en la nube de palabras, los tèrminos con menor tamaño ortorgan poca información por si solos por lo que quizás son relevantes en un determinado contexto.

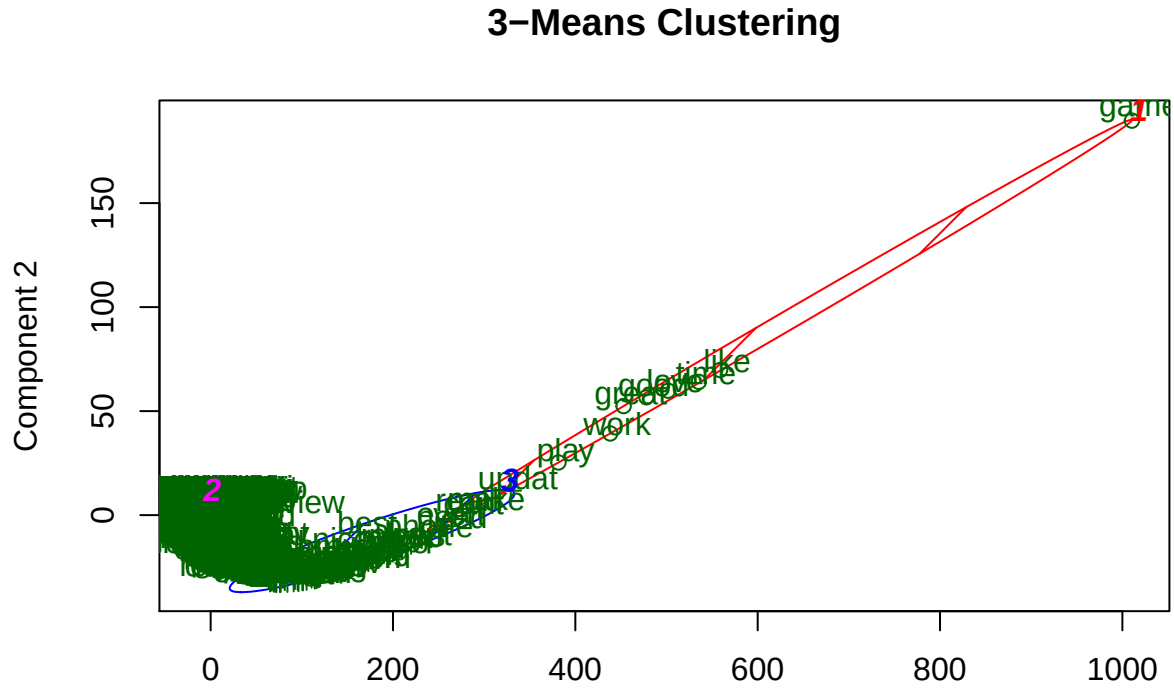
```
dtmr2 <- removeSparseTerms(dtm, sparse = 0.05)
inspect(dtmr2)
```

```
## <<DocumentTermMatrix (documents: 3, terms: 2180)>>
## Non-/sparse entries: 6540/0
## Sparsity           : 0%
## Maximal term length: 20
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs game good great like love make play time updat work
##    1 5522 4556  4462 4018 4690 2182 2371 3450  2200 2993
##    2 4517  396   231 1310  469  870 1500 1753  1032  981
##    3  102    4    27  421   37  140  163  321   295  609
```

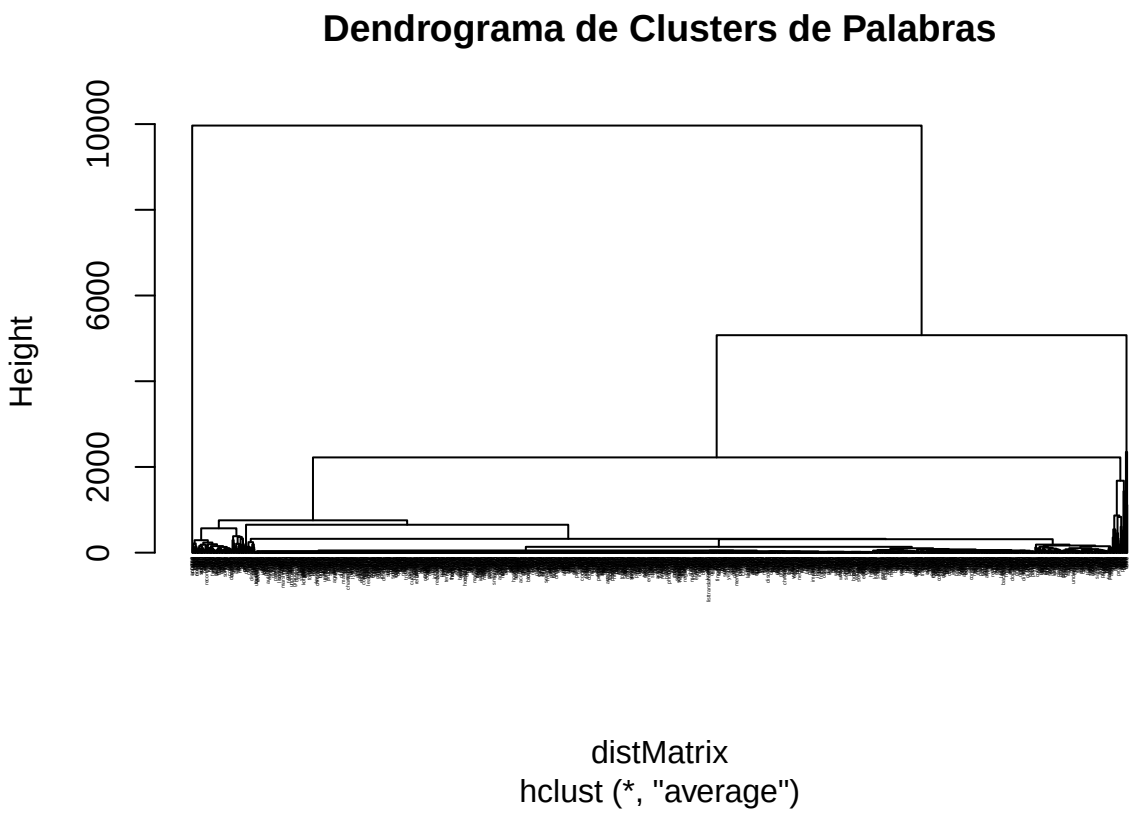
```
distMatrix <- dist(t(dtmr2), method= "manhattan")# probar con otros métodos

nroClusters <- 3 # as many as categories

kmeans_fit <- kmeans(distMatrix, nroClusters)
hclust_fit <- hclust(distMatrix, method = "average")

clusplot(as.matrix(distMatrix), kmeans_fit$cluster, color=T, shade=T, labels=1, lines=0, main = "3-Means
```




```
clusplot(as.matrix(distMatrix), kmeans_fit$cluster, color=T, shade=T, labels=2, lines=0, main = "3-Means
```




```r
# Dendrograma
plot(hclust_fit, cex=0.2, hang=-1, main = "Dendrograma de Clusters de Palabras")
```



```r
#clusplot(as.matrix(distMatrix), k_fitfit$cluster, color=T, shade=T, labels=3, lines=0, main = "Cluster
# Para chequear palabras representativas dentro de cada cluster
for (i in 1: nroClusters)
{
  cat(paste("cluster ", i, ": ", sep = ""))
  s <- sort(kmeans_fit$centers[i,], decreasing = T)
  cat(names(s)[1:5], "\n")
}
```

```
## cluster 1: airfar astro beij crook enterpris
## cluster 2: game like time love good
## cluster 3: game like time love good
```

Se puede ver que en el caso de clustering particional (KMeans) la mayoría de los clusters se concentran en la zona de alta densidad de la esquina inferior, mientras que un cluster centrado en un único término `game` nos muestra claramente que este término tiene características distintas a los demás, como se vio en el análisis de repetición de términos, `game` es un ´termino que se repite mucho tanto en sentimientos positivos como negativos.

Con respecto a clustering jerárquico, cortando al tercer nivel se puede extraer dos o tres clusters que parecen mostrar que un gran grupo de términos pueden ser aglomerados juntos y solo un pequeño conjunto está alejado del resto. Se utilizó la distancia media para calcular las aglomeraciones.