

# Tarea 2 - Minería de datos descriptiva

*José Manuel Alvarez García*

*Octubre 9, 2016*

Para instalar el paquete knitr se debe ejecutar lo siguiente:

```
knitr::opts_chunk$set(echo = TRUE)
install = function(pkg)
{
  # Si ya está instalado, no lo instala.
  if (!require(pkg, character.only = TRUE)) {
    install.packages(pkg)
    if (!require(pkg, character.only = TRUE)) stop(paste("load failure:", pkg))
  }
}
```

Inicialmente se debe estar posicionado en el directorio de la tarea:

```
setwd("C:/Users/José Manuel/Documents/ICD/mineria-de-datos-descriptiva-josemalvarezg1")
```

## Clustering

### Universidades

Para leer el dataset Universidades se debe ejecutar lo siguiente:

```
universidades = read.csv(file = "../data/Universidades.csv", header = T, dec = ".", sep = ",")
```

Esto indica que el archivo a leer posee en su primera fila los nombres de las columnas, el separador decimal es el punto (.), y el separador de cada elemento es la coma (,).

Seguidamente se debe preparar el dataset para que sólo contenga datos numéricos a trabajar:

```
#Se elimina la primera columna
universidades = universidades[,-1]
#Se colocan como nombres de fila los ID de cada universidad
rownames(universidades) <- universidades[,1]
#Se elimina la primera columna del dataset
universidades = universidades[,-1]
#Sólo se trabajará con las columnas de las 12 a la 15 que contienen datos numéricos
universidades = universidades[,12:15]
```

### K-medias

Ahora bien, se buscará el K más adecuado para aplicar el método de K-Medias. Esto se realizará mediante el Codo de Jambu:

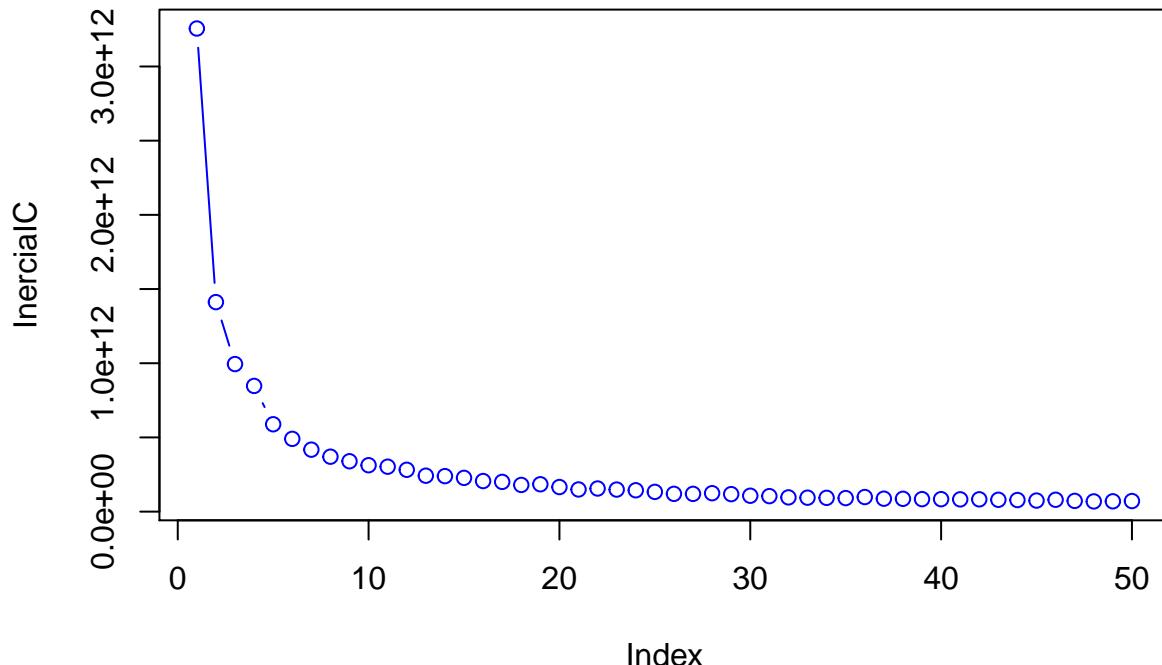
```

InerciaIC = rep(0,50)
for (k in 1:50) {
  grupos = kmeans(universidades, k)
  InerciaIC[k] = grupos$tot.withinss
}

```

Si se grafica la Inercia Inter-Clases se puede observar que cambia muy poco a partir de K=2 y K=3

```
plot(InerciaIC, col = "blue", type = "b")
```

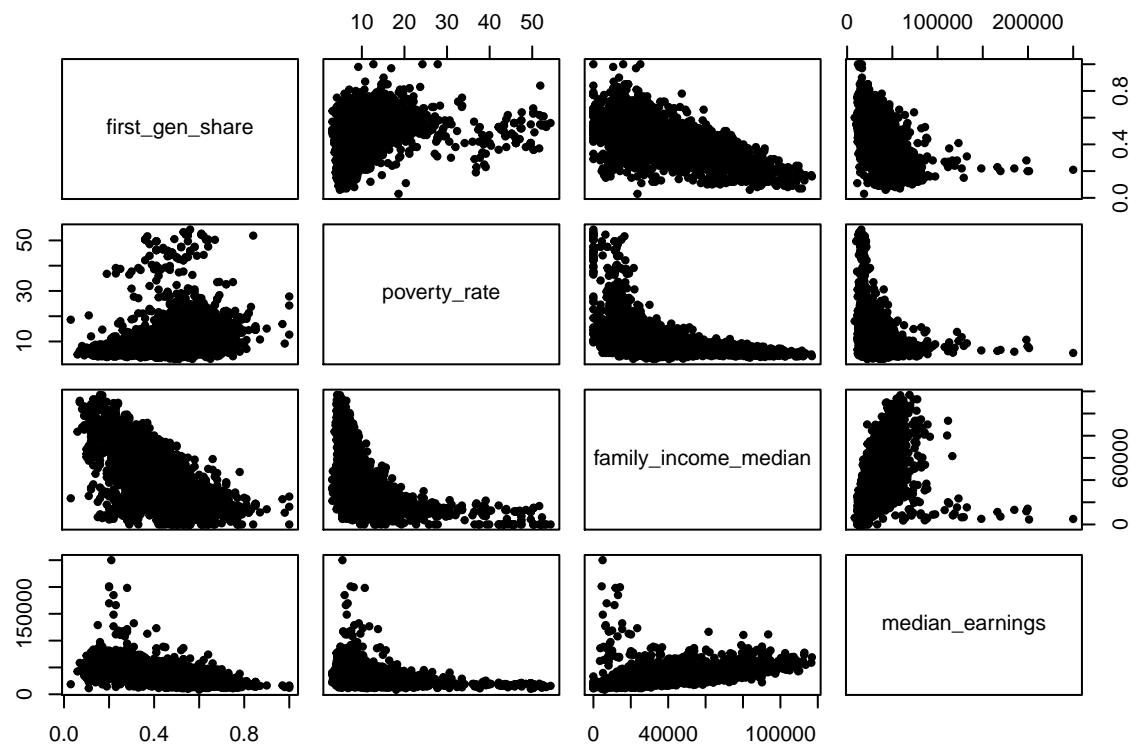


Por lo que calcula K-Medias con K=3 y 100 iteraciones, y se grafica:

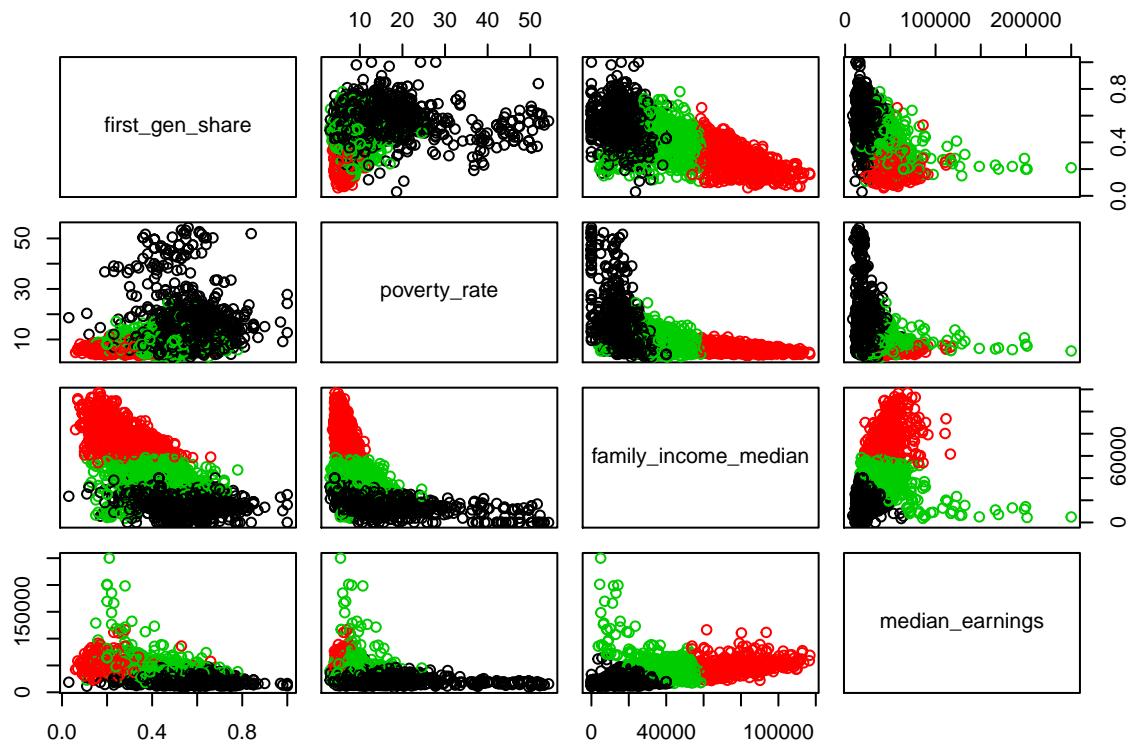
```

clusters <- kmeans(universidades, 3, iter.max = 100)
#Se grafica el dataset inicial
plot(universidades, pch = 20)

```



```
#Se colorean los tres grupos en el gráfico  
plot(universidades, col = clusters$cluster)
```



Si se desea realizar un análisis exploratorio para estudiar más a fondo el dataset se debe realizar lo siguiente:

```
library(FactoMineR)
#Se muestran valores de interés del dataset
# head(universidades)
# dim(universidades)
# names(universidades)
# str(universidades)
# attributes(universidades)
# summary(universidades)
# pca <- PCA(universidades)
```

### Clasificación Jerárquica

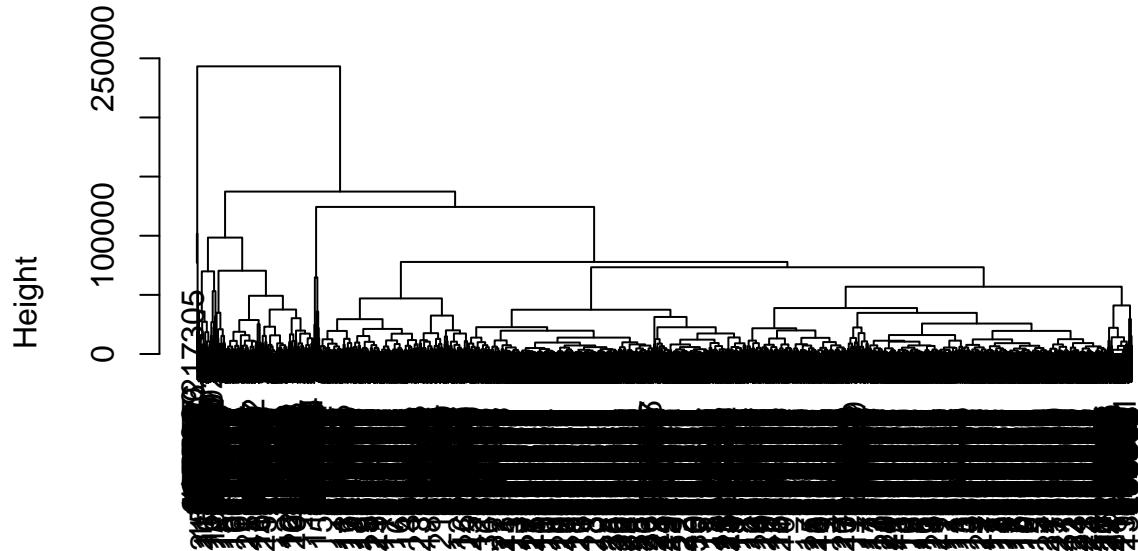
Se trabajará el dataset pre-procesado anteriormente como una matriz y luego se calculará la matriz de distancia:

```
datos = as.matrix(universidades)
distancia = dist(datos)
```

Se aplicarán y se graficarán los métodos de clasificación jerárquica. Para el método Complete:

```
cluster = hclust(distancia, method = "complete")
plot(cluster)
```

## Cluster Dendrogram

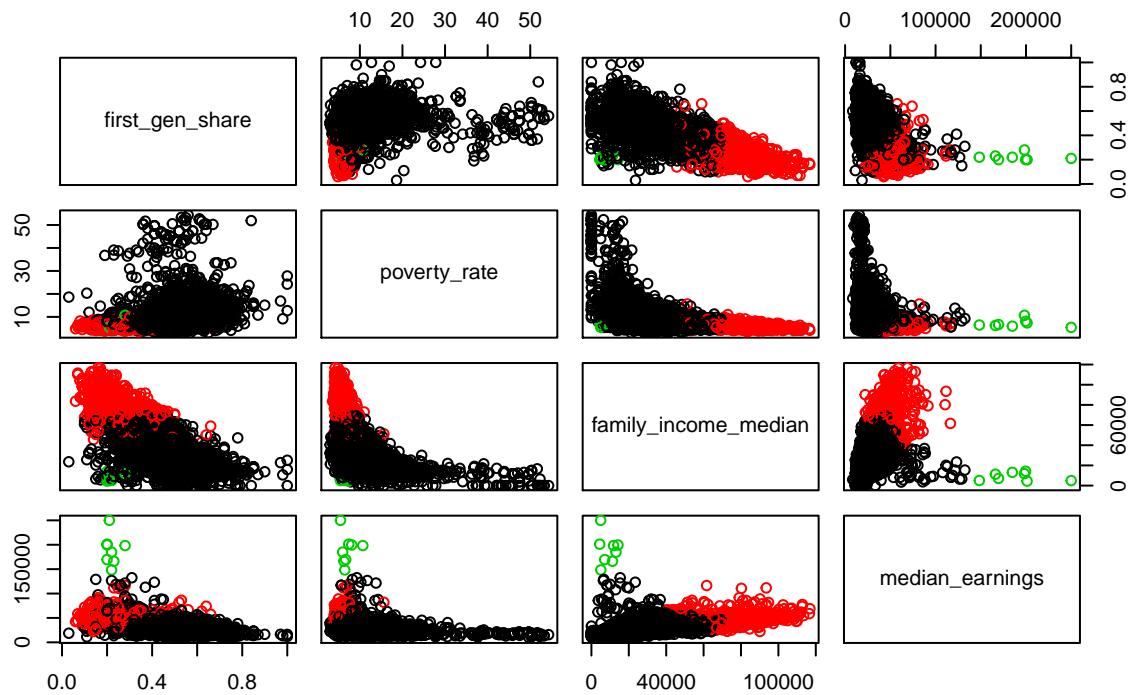


distancia  
hclust (\*, "complete")

```
#Se determina la altura requerida con k clusters, cortando el dendograma con k clases:  
corteD = cutree(cluster, k = 3)  
#Observamos la cantidad de clusters  
unique(corteD)
```

```
## [1] 1 2 3  
  
#Graficamos los clusters  
plot(universidades, col = corteD, main = "COMPLETE")
```

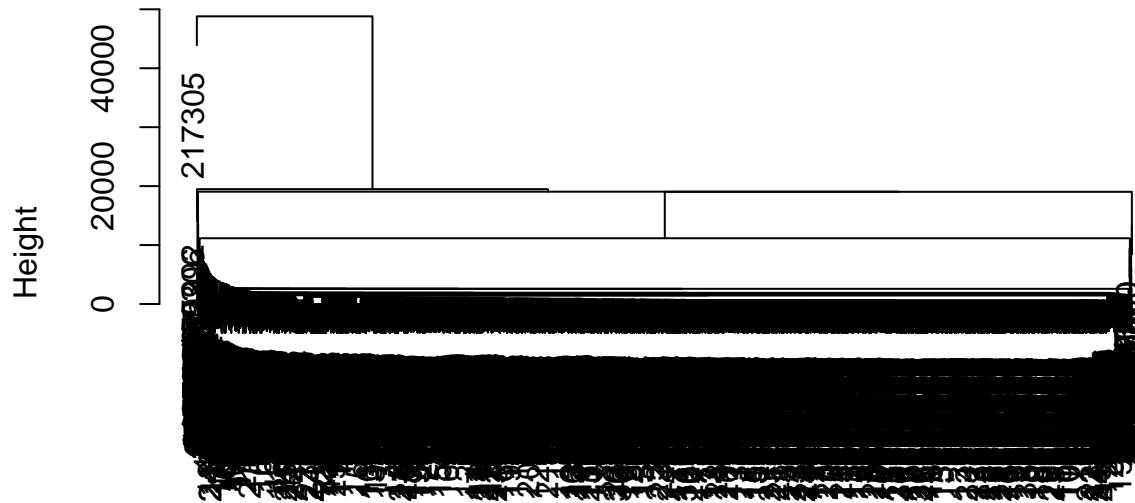
## COMPLETE



Para el método Single:

```
cluster = hclust(distancia, method = "single")
plot(cluster)
```

## Cluster Dendrogram

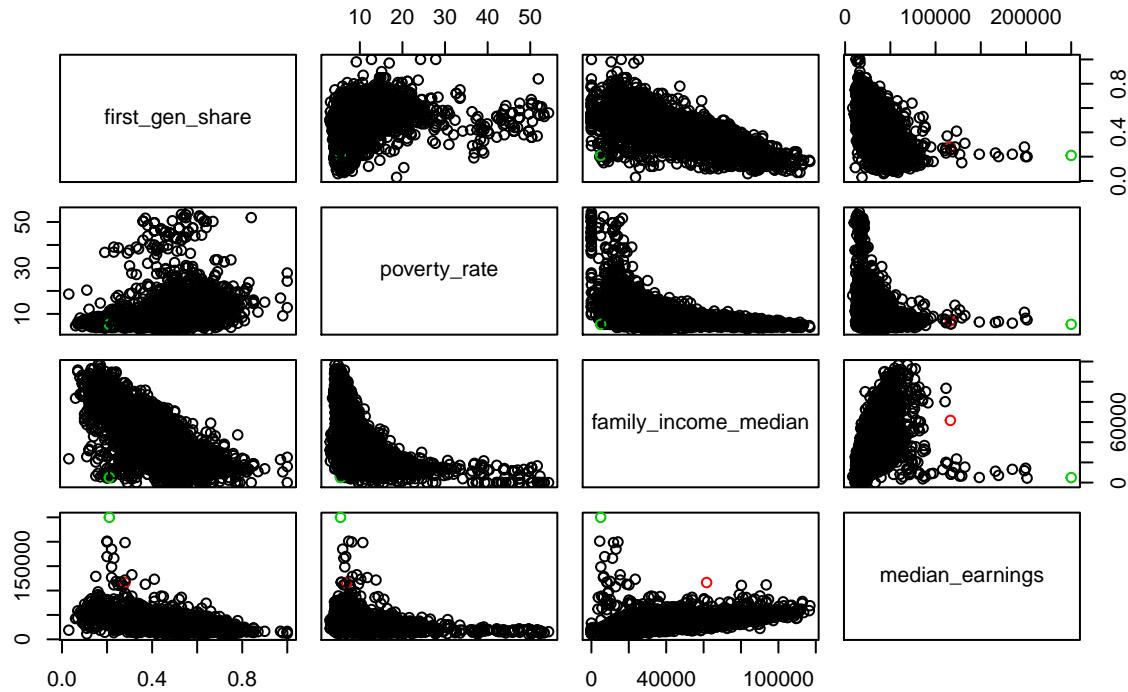


distancia  
hclust (\*, "single")

```
#Se determina la altura requerida con k clusters, cortando el dendograma con k clases:  
corteD = cutree(cluster, k = 3)  
#Observamos la cantidad de clusters  
unique(corteD)
```

```
## [1] 1 2 3  
  
#Graficamos los clusters  
plot(universidades, col = corteD, main = "SINGLE")
```

## SINGLE



Para el método Average:

```
cluster = hclust(distancia, method = "average")
plot(cluster)
```

## Cluster Dendrogram

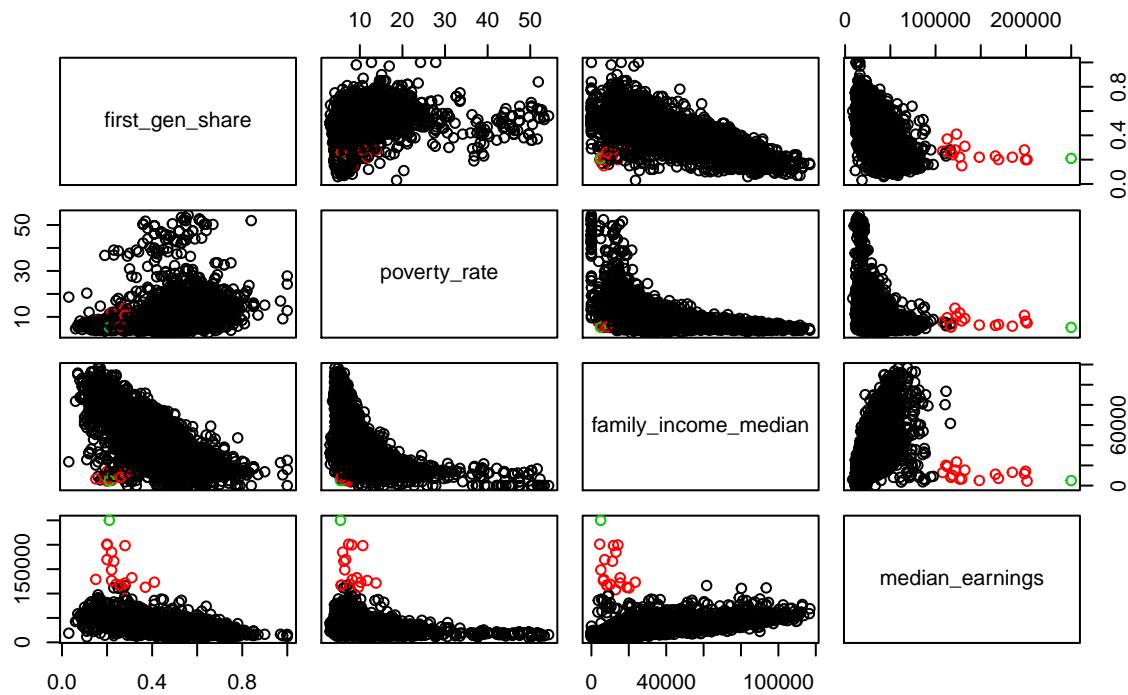


distancia  
hclust (\*, "average")

```
#Se determina la altura requerida con k clusters, cortando el dendograma con k clases:  
corteD = cutree(cluster, k = 3)  
#Observamos la cantidad de clusters  
unique(corteD)
```

```
## [1] 1 2 3  
  
#Graficamos los clusters  
plot(universidades, col = corteD, main = "AVERAGE")
```

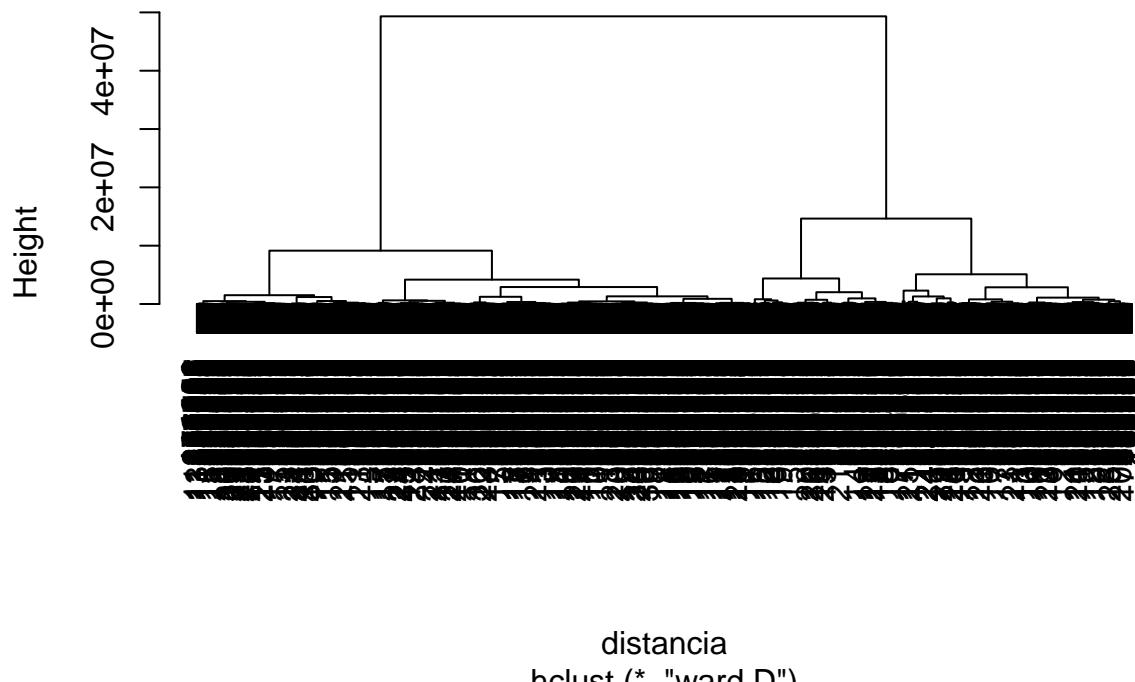
## AVERAGE



Para el método Ward:

```
cluster = hclust(distancia, method = "ward.D")
plot(cluster)
```

## Cluster Dendrogram



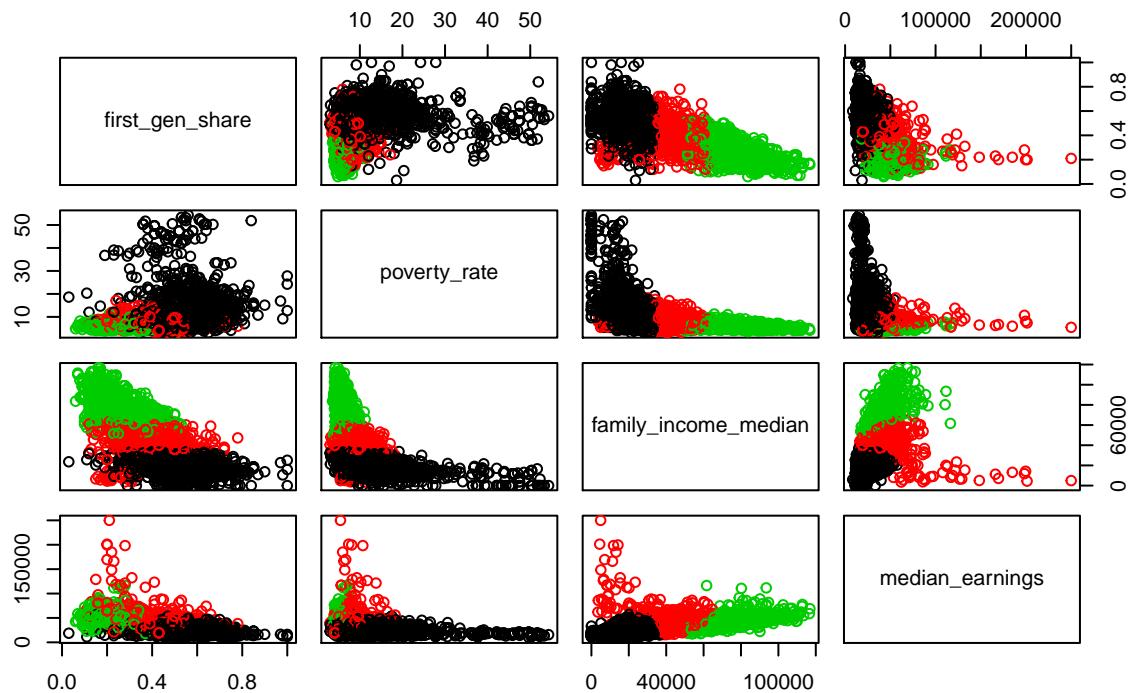
distancia  
hclust (\*, "ward.D")

```
#Se determina la altura requerida con k clusters, cortando el dendograma con k clases:  
corteD = cutree(cluster, k = 3)  
#Observamos la cantidad de clusters  
unique(corteD)
```

```
## [1] 1 2 3
```

```
#Graficamos los clusters  
plot(universidades, col = corteD, main = "WARD")
```

## WARD



Se puede observar que la clasificación por el método Ward es la que más se adapta al clustering del dataset utilizando K-Medias con K=3.

## Estudiantes

Para leer el dataset Estudiantes se debe ejecutar lo siguiente:

```
estudiantes = read.csv(file = "../data/Estudiantes.csv", header = F)
```

Seguidamente se debe preparar el dataset para que sólo contenga datos numéricos a trabajar:

```
estudiantes = estudiantes[,1:2]
```

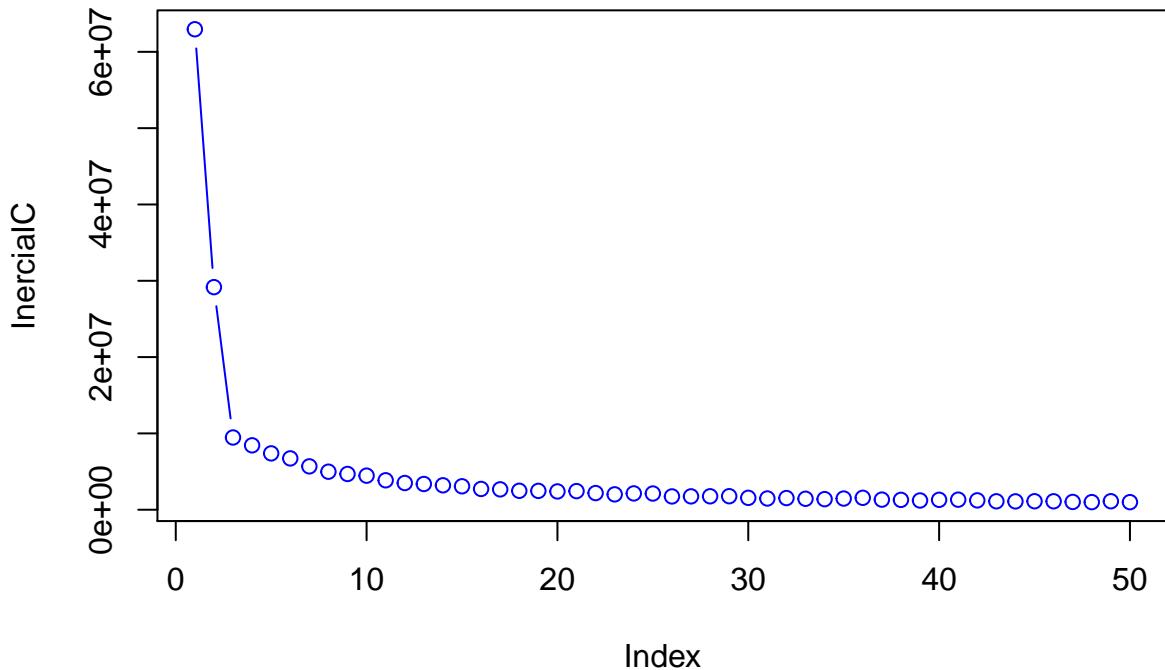
## K-medias

Ahora bien, se buscará el K más adecuado para aplicar el método de K-Medias. Esto se realizará mediante el Codo de Jambu:

```
InerciaIC = rep(0,50)
for (k in 1:50) {
  grupos = kmeans(estudiantes, k)
  InerciaIC[k] = grupos$tot.withinss
}
```

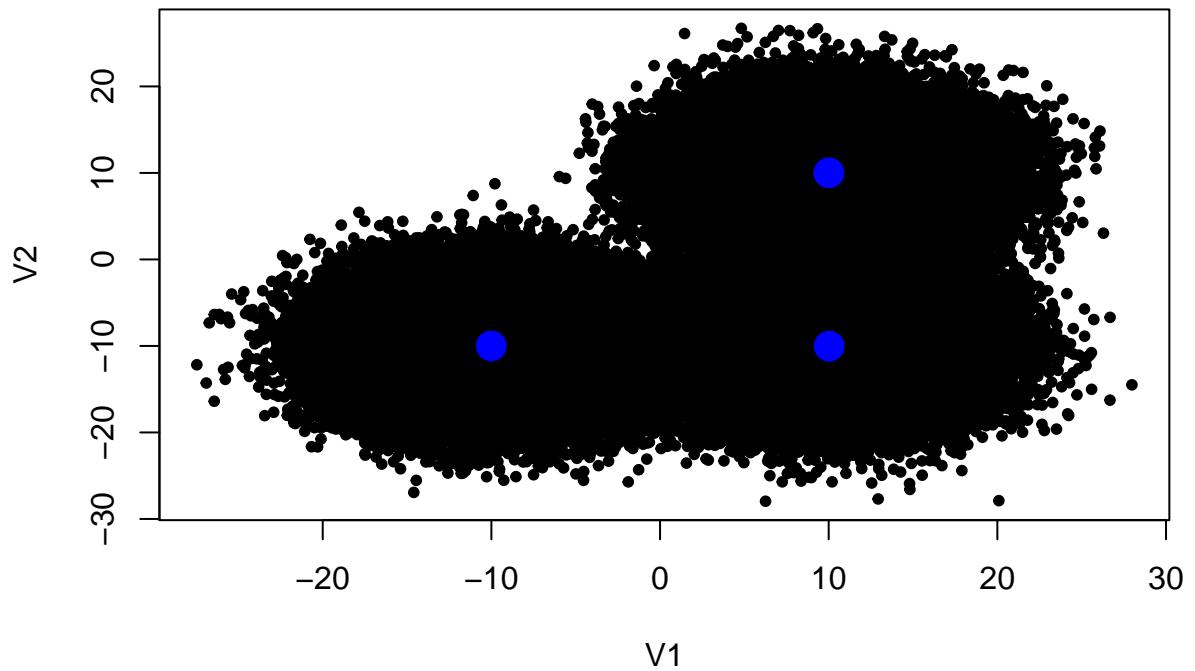
Si se grafica la Inercia Inter-Clases se puede observar que cambia muy poco a partir de K=2 y K=3

```
plot(InerciaIC, col = "blue", type = "b")
```

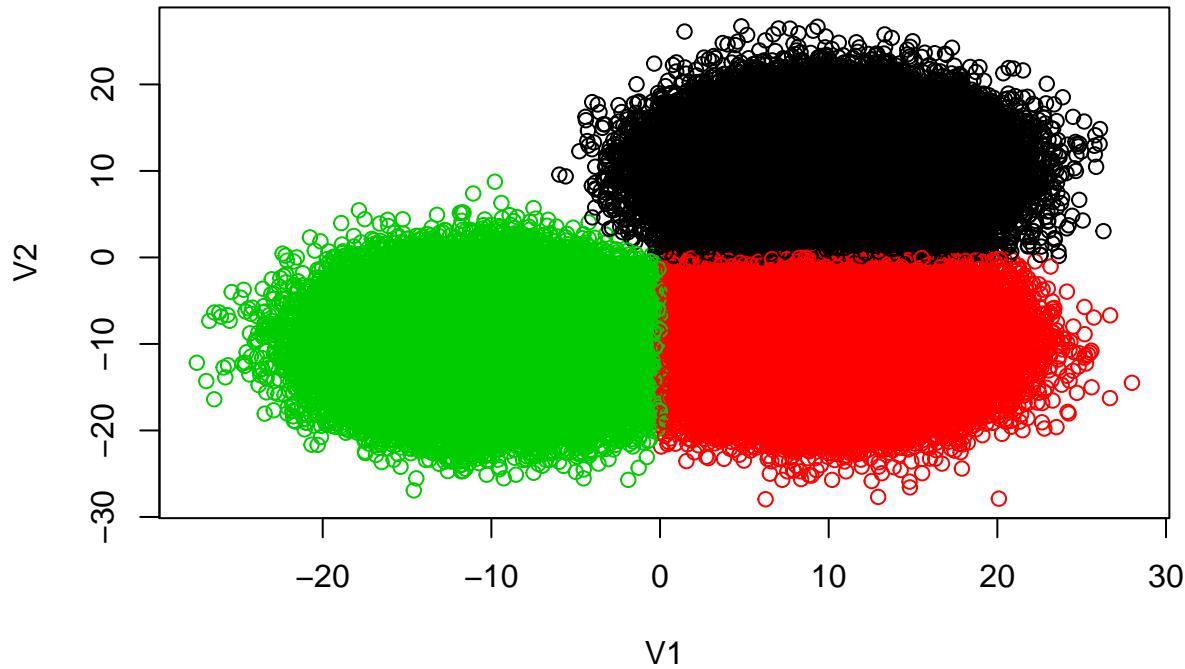


Por lo que calcula K-Medias con K=3 y 100 iteraciones, y se grafica:

```
clusters <- kmeans(estudiantes, 3, iter.max = 100)
plot(estudiantes, pch = 20)
#Se especifican los tres grupos (centroides) en el gráfico
points(clusters$centers, pch = 19, col = "blue", cex = 2)
```



```
#Se colorean los tres grupos en el gráfico  
plot(estudiantes, col = clusters$cluster)
```



Si se desea realizar un análisis exploratorio para estudiar más a fondo el dataset se debe realizar lo siguiente:

```
library(FactoMineR)
#Se muestran valores de interés del dataset
# head(estudiantes)
# dim(estudiantes)
# names(estudiantes)
# str(estudiantes)
# attributes(estudiantes)
# summary(estudiantes)
# pca <- PCA(estudiantes)
```

### Clasificación Jerárquica

Se trabajará el dataset pre-procesado anteriormente como una matriz y luego se calculará la matriz de distancia:

```
#datos = as.matrix(estudiantes)
#distancia = dist(datos)
```

Pero al momento de realizar esto, se mostrará un mensaje de error que indica “cannot allocate vector of size 335.3 Gb”. Esto ocurre porque se desean utilizar 335.3 GB de memoria RAM para calcular la matriz de distancias, pero esto es imposible en mi caso debido a que poseo una memoria RAM de 2GB y RStudio como tal ocupa 104,00MB de la misma.

## Reglas de asociación

Se deben incluir las bibliotecas “arules” y “arulesViz” de la siguiente manera:

```
library(arules)
library(arulesViz)
```

Para leer el dataset Alimentacion se debe ejecutar lo siguiente:

```
alimentacion <- read.transactions("../data/Alimentacion.csv", sep=',')
```

Seguidamente, se convierte el dataset a valores transaccionales para generar las reglas de asociación y se muestra un resumen de las mismas:

```
reglas <- apriori(alimentacion, parameter=list(support=0.001, confidence = 0.65))
```

```
## Apriori
##
## Parameter specification:
##   confidence minval smax arem aval originalSupport maxtime support minlen
##           0.65     0.1    1 none FALSE             TRUE      5   0.001      1
##   maxlen target ext
##       10   rules FALSE
##
## Algorithmic control:
##   filter tree heap memopt load sort verbose
##   0.1 TRUE TRUE FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 5
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[667 item(s), 5167 transaction(s)] done [0.00s].
## sorting and recoding items ... [290 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 done [0.02s].
## writing ... [1409 rule(s)] done [0.00s].
## creating S4 object ... done [0.00s].
```

```
summary(reglas)

## set of 1409 rules
##
## rule length distribution (lhs + rhs):sizes
##   2   3   4   5   6
##  24 405 723 238  19
##
##   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
##  2.000  3.000  4.000  3.874  4.000  6.000
##
## summary of quality measures:
##   support      confidence      lift
```

```

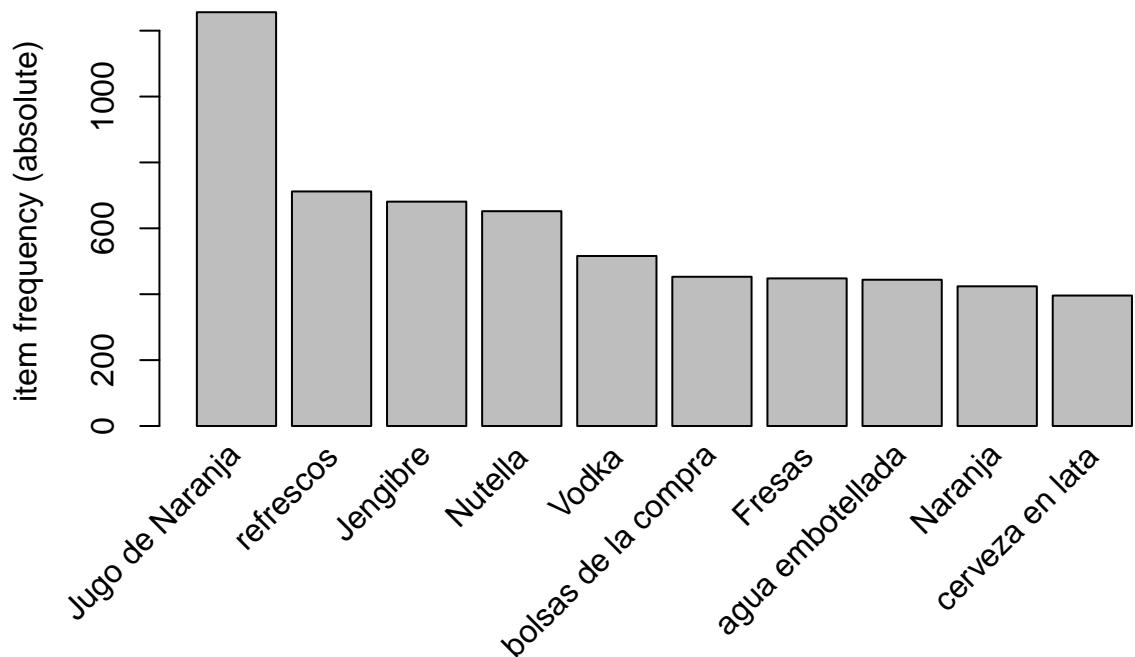
##  Min.   :0.001161   Min.   :0.6500   Min.   : 2.674
##  1st Qu.:0.001161  1st Qu.:0.7000  1st Qu.: 3.366
##  Median :0.001355  Median :0.7778  Median : 5.283
##  Mean    :0.001551  Mean   :0.8051  Mean   : 8.701
##  3rd Qu.:0.001548  3rd Qu.:0.8750  3rd Qu.: 6.934
##  Max.    :0.017031  Max.   :1.0000  Max.   :135.974
##
## mining info:
##           data ntransactions support confidence
## alimentacion      5167     0.001         0.65

```

Con esto, se genera un set de 1409 reglas donde se obtienen los valores MinSupport=0.001161, MinConfidence=0.6500 y MinLift=2.674. Se utilizaron los valores support=0.001 y confidence=0.65 para generar una gran cantidad de reglas que se den en proporción a las transacciones del dataset.

Para conocer las 10 transacciones con mayor número de apariciones en el dataset se realiza lo siguiente:

```
itemFrequencyPlot(alimentacion,topN=10,type="absolute")
```



Para obtener la probabilidad de que un estudiante consuma refrescos si consome chicles se realiza lo siguiente:

```

#Se obtienen las subreglas que cumplen chicles -> refrescos
subreglas = subset(reglas, subset=lhs %pin% "chicles" & rhs %pin% "refrescos")
#Se muestra la probabilidad de que un estudiante consuma refrescos dado que consume chicles
quality(subreglas)$confidence

```

```
## [1] 0.8571429
```

Para ordenar las reglas se tiene lo siguiente:

```
#Se ordenan las reglas por confianza
confianzaAlta <-sort(reglas, by="confidence", decreasing=TRUE)
inspect(head(confianzaAlta))
```

```
##      lhs                  rhs      support      confidence lift
## [1] {bebidas alcoholicas} => {Cerveza} 0.002128895 1       13.74202
## [2] {Helado}              => {galletas} 0.001161215 1       33.99342
## [3] {Cigarros}            => {postre}   0.001161215 1       29.86705
## [4] {Empanada}            => {Flips}    0.001741823 1       17.39731
## [5] {lechuga}              => {cebolla}  0.002515967 1       78.28788
## [6] {Jugo de Pera}         => {Vodka}    0.001935359 1       10.01357
```

Se observa que las primeras seis reglas tienen una confianza de 1, por lo que estas siempre serán verídicas.

```
#Se ordenan las reglas por soporte
supportAlto <-sort(reglas, by="support", decreasing=TRUE)
inspect(head(supportAlto))
```

```
##      lhs                  rhs      support      confidence
## [1] {agua mineral}        => {refrescos} 0.017031159 0.6929134
## [2] {Uvas}                => {Vodka}     0.010644475 0.9821429
## [3] {Jengibre,mantequilla} => {Jugo de Naranja} 0.010063867 0.7878788
## [4] {Harina de Trigo}     => {huevos}    0.007354364 1.0000000
## [5] {queso Crema}         => {Nutella}   0.007354364 0.7916667
## [6] {Jengibre,Nutella,Vodka} => {Jugo de Naranja} 0.006967292 0.6666667
##      lift
## [1] 5.028488
## [2] 9.834752
## [3] 3.241218
## [4] 74.884058
## [5] 6.273837
## [6] 2.742569
```

Se observa que las primeras seis reglas son las que más frecuentan en el dataset.

```
#Se ordenan las reglas por lift
liftAlto <-sort(reglas, by="lift", decreasing=TRUE)
inspect(head(liftAlto))
```

```
##      lhs                  rhs      support
## [1] {huevos,queso blando}    => {Harina de Trigo} 0.001161215
## [2] {huevos,pastes}         => {Harina de Trigo} 0.001548287
## [3] {huevos,Jengibre,pastes} => {Harina de Trigo} 0.001161215
## [4] {huevos,pollo}          => {Harina de Trigo} 0.001161215
## [5] {huevos,Jengibre,Nutella} => {Harina de Trigo} 0.001161215
## [6] {bolsas de la compra,huevos} => {Harina de Trigo} 0.001935359
##      confidence lift
## [1] 1.0000000 135.9737
## [2] 1.0000000 135.9737
## [3] 1.0000000 135.9737
```

```

## [4] 0.8571429 116.5489
## [5] 0.8571429 116.5489
## [6] 0.8333333 113.3114

```

Se observa que las primeras seis reglas son las más probables en ocurrir.

Si se obtienen todas las subreglas que contengan en su consecuente al producto “Naranja” y se muestran:

```

#Se obtienen las subreglas que cumplen ... -> Naranja
subreglas = subset(reglas, subset = rhs %ain% "Naranja")
#Se muestran las subreglas generadas
inspect(subreglas)

```

	lhs	rhs	support	confidence	lift
## [1]	{margarina, sopas}	=> {Naranja}	0.001161215	0.6666667	8.124214
## [2]	{aceite, detergente}	=> {Naranja}	0.001354751	0.8750000	10.663031
## [3]	{hortalizas de raiz, pollo}	=> {Naranja}	0.001161215	0.8571429	10.445418
## [4]	{Carne de res, Jengibre, uvas}	=> {Naranja}	0.001161215	0.8571429	10.445418
## [5]	{Carne de res, Jengibre, Vodka}	=> {Naranja}	0.001354751	0.8750000	10.663031
## [6]	{articulo de higiene, Nuggets, Nutella}	=> {Naranja}	0.001161215	0.6666667	8.124214
## [7]	{Jengibre, margarina, Queso Crema}	=> {Naranja}	0.001354751	0.7777778	9.478249
## [8]	{Jugo de Naranja, margarina, Queso Crema}	=> {Naranja}	0.001161215	0.8571429	10.445418
## [9]	{Jengibre, periodicos, vegetales de raiz}	=> {Naranja}	0.001161215	0.7500000	9.139741
## [10]	{Jengibre, Jugo de Naranja, margarina, Queso Crema}	=> {Naranja}	0.001161215	1.0000000	12.186321

Se observa que el producto que más se repite en las transacciones es “Jengibre”.

Si se obtienen todas las subreglas que contengan en su precedente a los productos “Flips”, “Nutella” y “Vodka” y se muestran:

```

#Se obtienen las subreglas que cumplen Flips, Nutella, Vodka -> ...
subreglas = subset(reglas, subset=lhs %ain% c("Flips", "Nutella", "Vodka"))
#Se muestran las subreglas generadas
inspect(subreglas)

```

	lhs	rhs	support
--	-----	-----	---------

```

## [1] {Flips,Nutella,Vodka}          => {Jugo de Naranja} 0.002322431
## [2] {Flips,Jengibre,Nutella,Vodka} => {Jugo de Naranja} 0.001548287
## [3] {Flips,Jugo de Naranja,Nutella,Vodka} => {Jengibre}      0.001548287
##   confidence lift
## [1] 0.8000000 3.291083
## [2] 0.8888889 3.656759
## [3] 0.6666667 5.058248

```

Se observa que el producto más recomendable es “Jugo de Naranja”.

## Metodología CRISP-DM

### Comprensión del negocio

Se desea que se apliquen técnicas descriptivas y de clustering a los datasets provistos por parte del Ministerio de Educación y Salud.

### Comprensión de datos

Se tienen 3 datasets (Universidades, Estudiantes y Alimentacion) a procesar.

### Preparación de datos

Al momento de leer los datasets, se deben pre-procesar; esto se realizó al sólo seleccionar los datos (columnas) numéricos de Universidades.csv y sólo trabajar con las dos primeras columnas de Estudiantes.csv para ignorar la columna clase.

### Modelado

Se desea buscar cuál es la técnica más adecuada para trabajar los datos; esto se realizó al obtener el K más apropiado para aplicar el método de K-Medias y al seleccionar la mejor clasificación jerárquica para Universidades.csv.

### Evaluación

Se considera que los datos provistos son adecuados para la aplicación de las técnicas impuestas en esta tarea.

### Despliegue

Se deberá explicar al cliente toda la implementación realizada para pre-procesar los datasets y la aplicación y la utilidad de las técnicas hacia los mismos.