UNIVERSIDAD DE GRANADA

Clustering

Carlos Morales Aguilera

24/11/2020

Lectura de datos

El primer paso en el problema es leer correctamente los datos, para ello se utiliza la función `read.csv` teniendo en cuenta que no posee nombre de columnas, por lo que se asignará la nomenclatura *V1* para la variable 1 del conjunto.

```
f <- "C:\\Users\\power\\Desktop\\TID\\practica2\\wine.data"
# Read dataframe
bd_wine <- read.csv(f, header=FALSE)
```

Tras cargar los datos, visualizamos cuantas instancias hay de cada clase utilizando para ello la primera variable del conjunto de datos, referente al tipo de alcohol. Además almacenamos esa columna y se extrae del conjunto de datos.

```
# Group by class
classes <- table(bd_wine$V1)
classes
```

```
##
##  1  2  3
## 59 71 48
```

```
bd_classes <- bd_wine[, "V1"]
bd_wine[, "V1"] <- NULL
```

Una de las buenas prácticas aprendidas de la práctica anterior es la de eliminar valores perdidos, por lo que previamente a tratar con los datos, nos aseguramos de eliminar los posibles valores perdidos si hubieran. Además, se realizará una normalización aplicando el algoritmo **Min-max**, para poder trabajar con los datos de forma adecuada.

```
# Omit NAs
bd_wine <- na.omit(bd_wine)

# Definition of min_max normalization function
min_max_norm <- function(x) {
    (x - min(x)) / (max(x) - min(x))
}

# Apply normalization to our dataframe
bd_wine <- as.data.frame(lapply(bd_wine, min_max_norm))
```

Clustering Jerárquico

Dentro de las técnicas de clustering, existe el **Clustering Jerárquico**, que consiste en un método de agrupamiento de aprendizaje no supervisado, en el que se pueden distinguir principalmente dos tipos:

- **Aglomerativo**: Cada elemento se agrupa individualmente. Inicialmente cada elemento es un clúster, y si varios clúster se encuentran muy cercanos, estos se fusionan formando un único clúster. Este procedimiento se produce de forma jerárquica hacia arriba hasta alcanzar un único clúster que contenga todos los elementos.

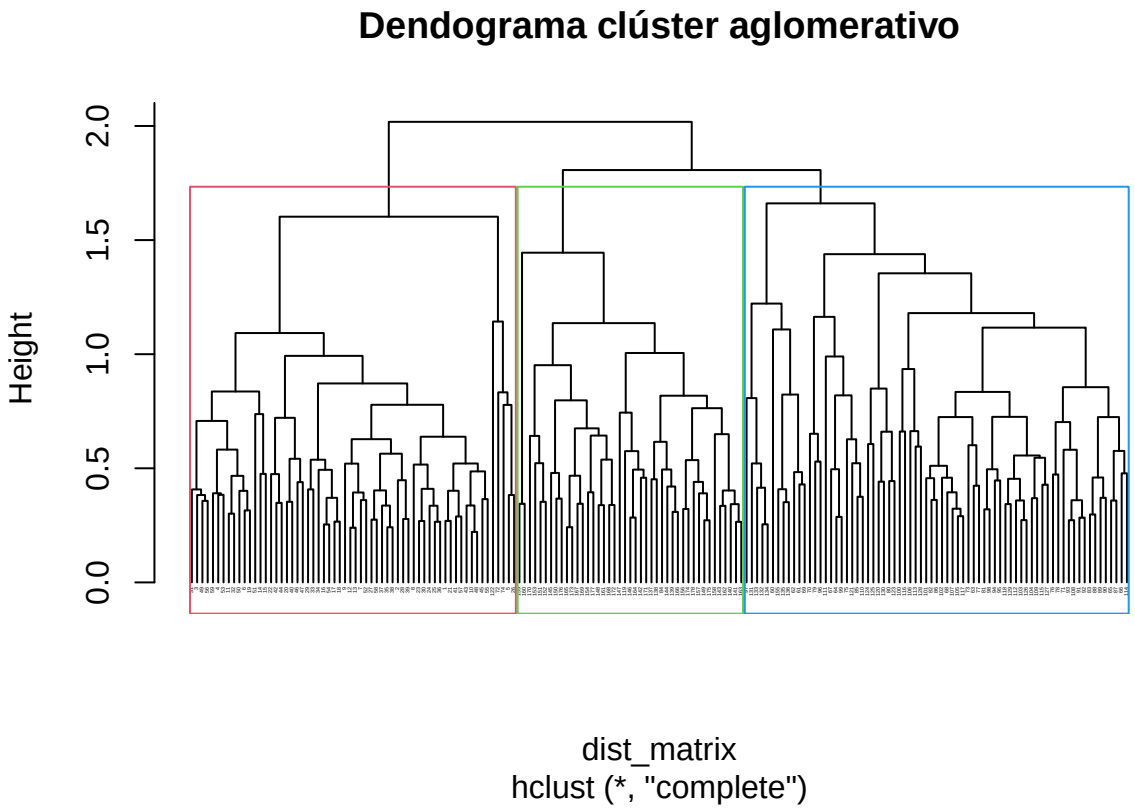
- **Divisivo**: Es el proceso opuesto al aglomerativo, ya que se parte de un único clúster que se va dividiendo en clúster inferiores más pequeños.

En esta práctica se ha decidido probar ambos tipos de clustering jerárquico y comprobar como clasifica cada uno de ellos.

Para la realización del clustering jerárquico aglomerativo, se ha decidido implementar la función hclust que se encuentra en el paquete stats. Para ello crearemos una matriz de valores de dimisilitud utilizando el método de distancia euclídea con la función dist.

Tras calcular dicha matriz de disimilitud, realizaremos el clustering con la función **hclust**, utilizando el método *completo*, que combina elementos como la media, mediana, etc. Tras realizar el clustering, se procede a dibujar el dendograma asociado con la elección de clusters.

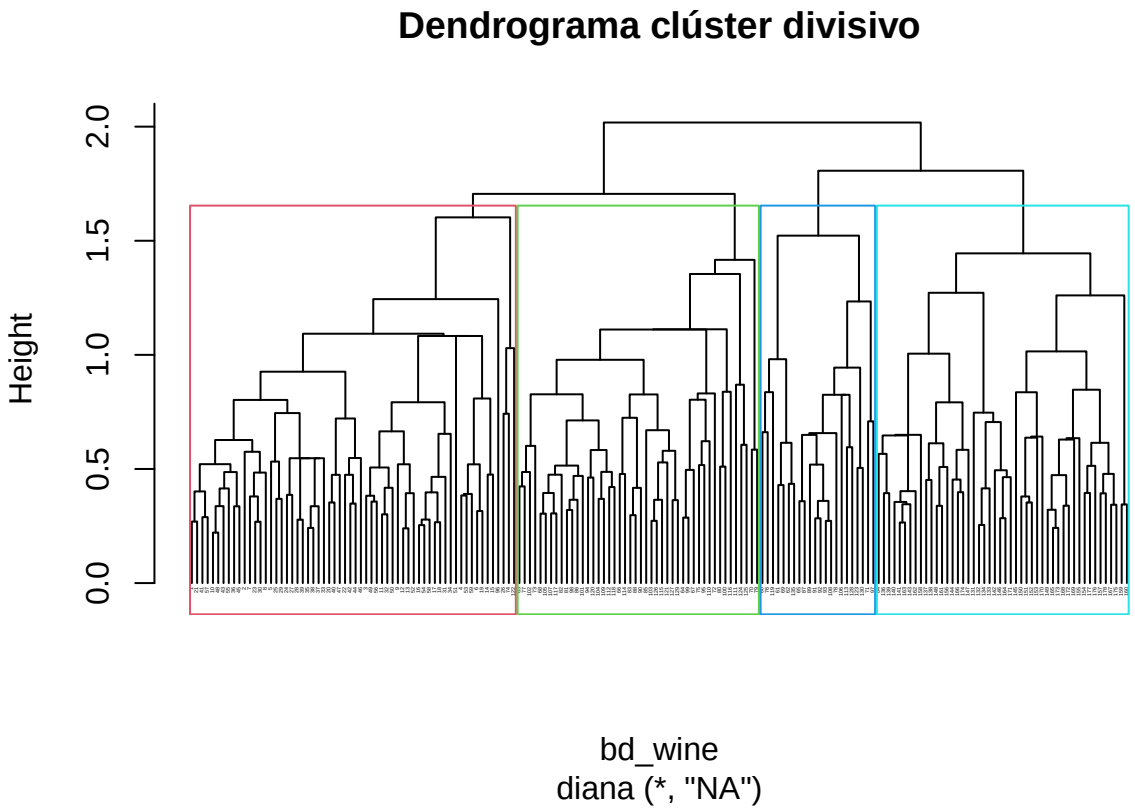
```
# Calculate the disimilarity matrix
dist_matrix <- dist(bd_wine, method = "euclidean")
# Compute clusters
a_cluster <- hclust(dist_matrix, method = "complete" )
# Plot dendogram
plot(a_cluster, cex = 0.2, hang = -1, main = "Dendograma clúster aglomerativo")
rect.hclust(a_cluster, k = 3, border = 2:5)
```



Para la realización del clustering jerárquico divisivo, se ha decidido implementar la función diana que se encuentra en el paquete cluster.

Tras realizar el clustering, se procede a dibujar el dendograma asociado con la elección de clusters.

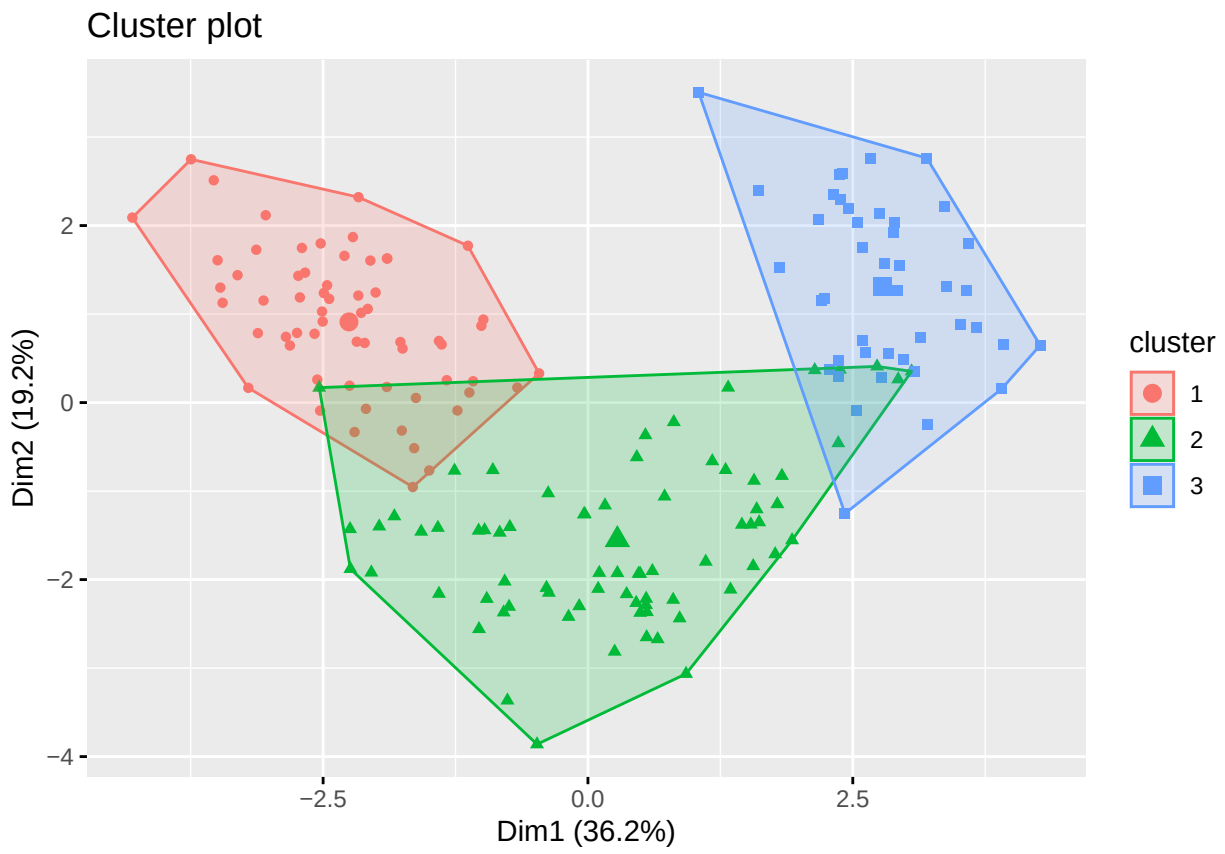
```r
# Compute clusters
d_cluster <- diana(bd_wine)

# Plot dendrogram
pltree(d_cluster, cex = 0.2, hang = -1, main = "Dendrograma clúster divisivo")
rect.hclust(as.hclust(d_cluster), k = 4, border = 2:5)
```



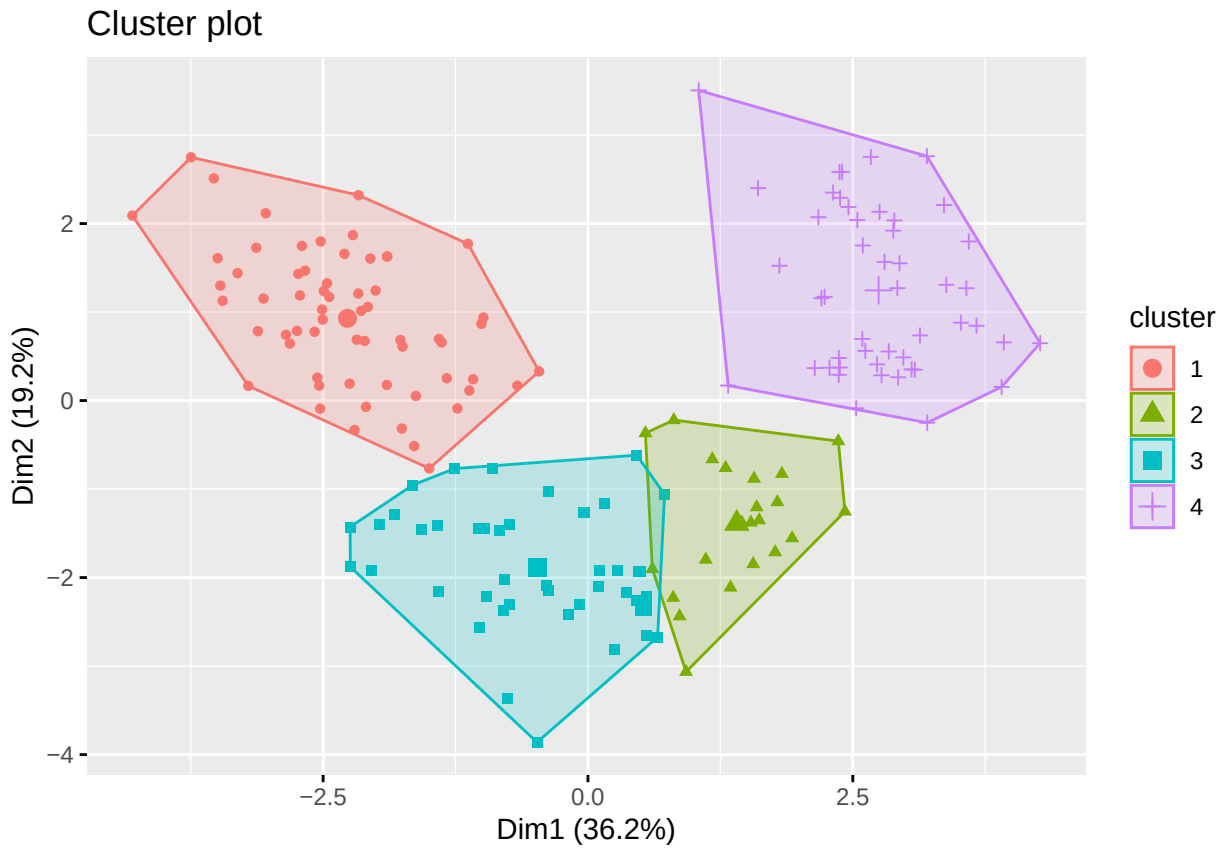
Otra forma de visualizar el clustering realizado es dibujar en un plano los diferentes elementos y dibujar los clusters que los agrupan, esto se puede realizar mediante la función fviz_cluster de la librería factoextra.

A continuación se muestra el clustering jerárquico aglomerativo:

```r
sub_grp <- cutree(a_cluster, k = 3)
fviz_cluster(list(data = bd_wine, cluster = sub_grp), geom = "point")
```




```
sub_grp <- cutree(as.hclust(d_cluster), k = 4)
fviz_cluster(list(data = bd_wine, cluster = sub_grp), geom = "point")
```

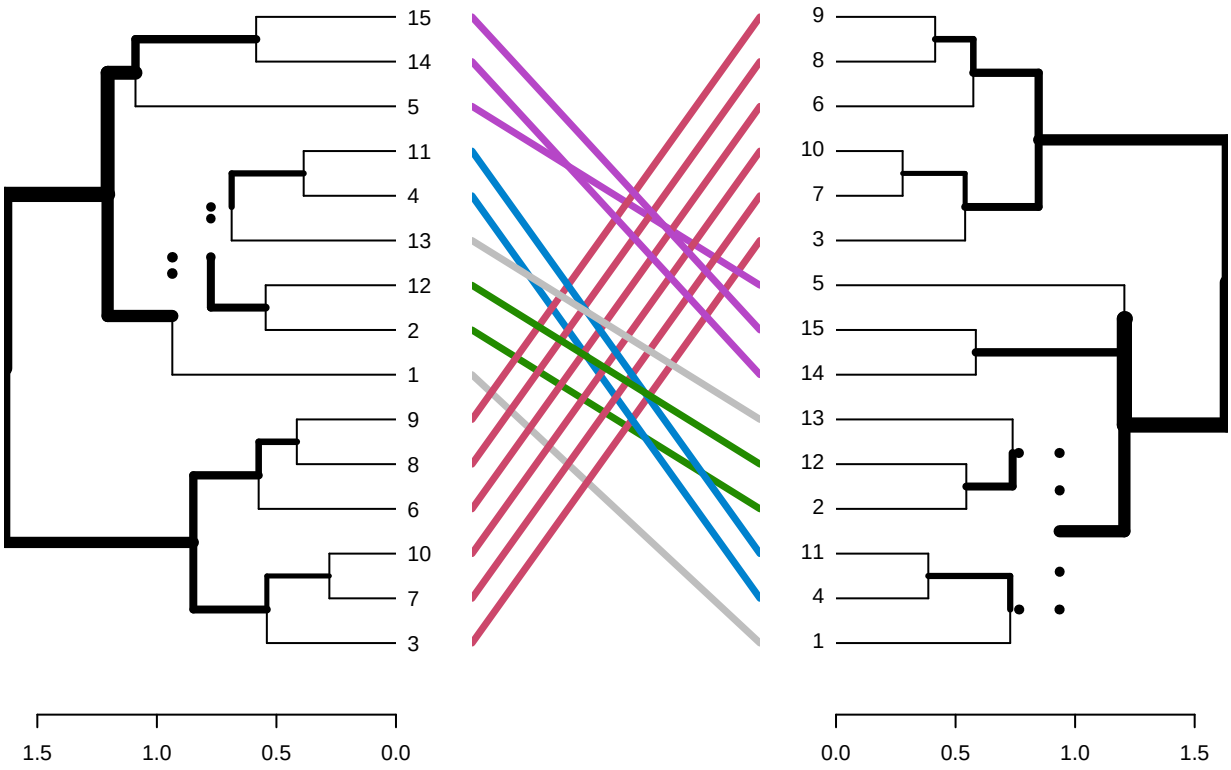


Por último, como observamos que existen diferencias en las clasificaciones en ambos dendogramas, se ha decidido mostrar dichas diferencias haciendo uso de la función tangleram de la librería dendextend.

En este diagrama se pueden observar las conexiones entre los distintos elementos clasificados, por lo que es una técnica interesante de validar las agrupaciones realizadas. En nuestro caso se obtiene un resultado "borroso", ya que la cantidad de datos manejada es bastante elevada para realizar un diagrama de este tipo, pero resulta interesante mencionarlo y además de puede apreciar ligeramente como se asocian mediante líneas de colores diferentes los distintos elementos dentro de ambos dendogramas.

Para poder comprobarlo, se ha realizado un pequeño ejemplo tomando únicamente 15 muestras aleatorias del conjunto de datos inicial.

```
sub_wine <- sample_n(bd_wine, size= 15)

# Calculate the disimilarity matrix
sub_dist_matrix <- dist(sub_wine, method = "euclidean")
# Compute clusters
sub_a_cluster <- hclust(sub_dist_matrix, method = "complete" )

# Compute clusters
sub_d_cluster <- diana(sub_wine)

# Create two dendrograms
dend1 <- as.dendrogram (sub_a_cluster)
dend2 <- as.dendrogram (as.hclust(sub_d_cluster))

tanglegram(dend1, dend2)
```



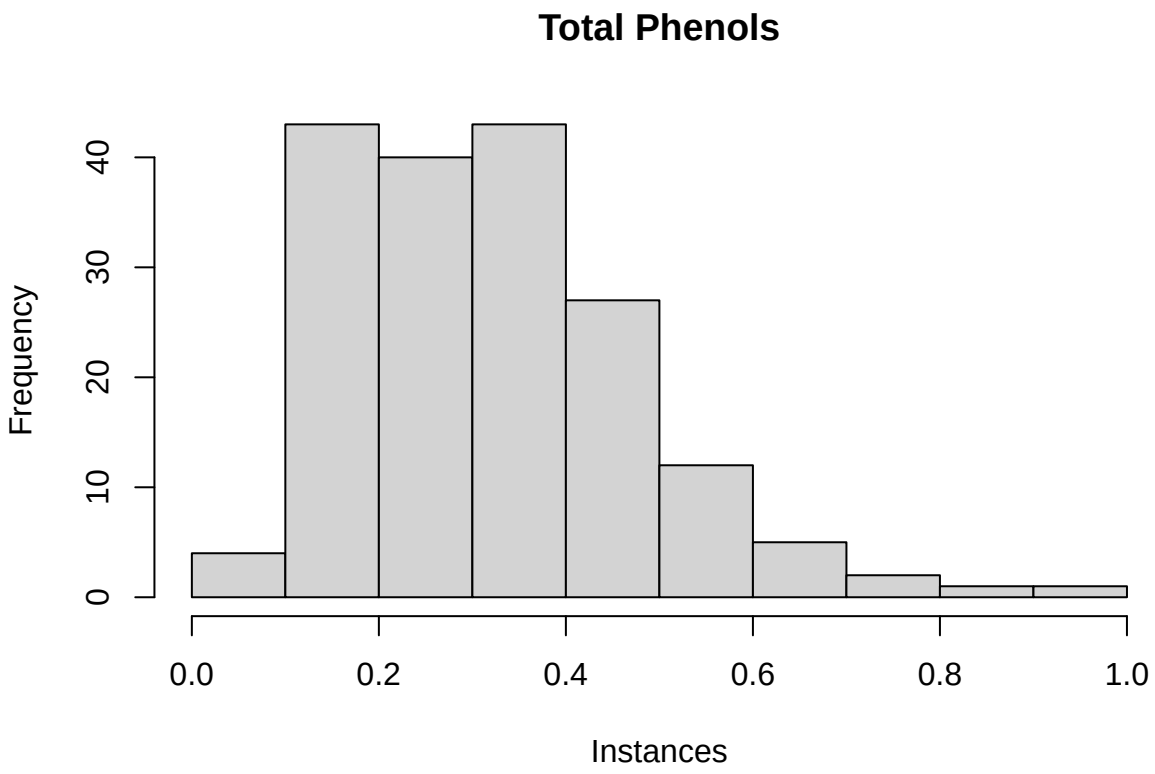
Outliers

En estadística, un outlier o valor atípico es una observación que es numéricamente diferente al resto de datos, por lo que dicha información a la hora de realizar un clustering puede desfavorecer el agrupamiento realizado, ya que esta información puede de cierta manera engañar a los algoritmos empleados, por lo que a la hora de realizar ciertos procedimientos, es mejor no considerar estos valores atípicos para realizar un agrupamiento en clúster adecuado a los datos.

Una de las técnicas más basicas a la hora de realizar la detección de *outliers* es la realización de *histogramas* que permitan ver si un valor sobresale por arriba o por debajo del resto de datos, sin embargo, esta técnica requiere de un análisis profundizado de los datos, por lo que aunque podría ser útil, se emplearán otras técnicas más efectivas y claras.

A continuación se muestra un ejemplo de un posible *histograma* donde podemos observar que existen ciertos datos que se producen rara vez respecto al resto:

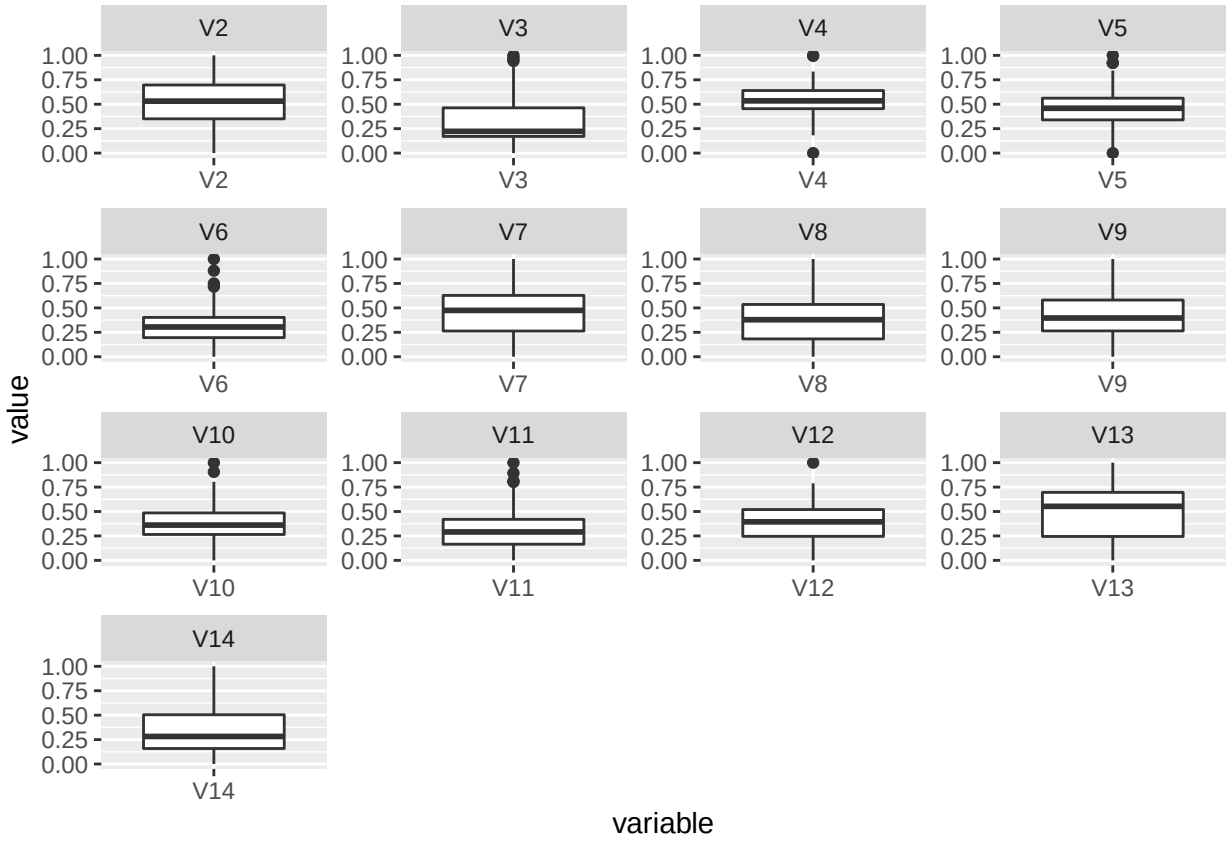
```
hist(bd_wine$V6,
  xlab = "Instances",
  main = "Total Phenols",
  breaks = sqrt(nrow(bd_wine))
)
```



Una de las técnicas más utilizadas para la detección de *outliers* es la realización de **boxplots** o diagramas de caja, los cuales muestran información estadística sobre los datos, obteniendo una representación gráfica de conceptos estadísticos como la mediana, cuartiles o valores atípicos En nuestro caso vamos a realizar un análisis de posibles *outliers* en todo el conjunto de datos, por lo que vamos a fijarnos en posibles valores atípicos en cada una de las columnas (variables) del conjunto.

Cabe destacar que el hecho de que un valor se encuentre fuera de la *caja* no implica que sea un *outlier*, puede tratarse de un caso particular o fuera de lo común, pero no debe eliminarse directamente, primero se debe hacer una análisis de las variables y comprobar si dicho elemento es realmente o no un *outlier*.

```
# Read dataframe
bd_wine <- read.csv(f, header=FALSE)
# Omit NAs
bd_wine <- na.omit(bd_wine)
# Apply normalitzaion to our dataframe
bd_wine[-1] <- as.data.frame(lapply(bd_wine[-1], min_max_norm))

bd_wine.melt <- melt(bd_wine[-1], measure.vars = colnames(bd_wine[-1]))
ggplot(data = bd_wine.melt, aes(x=variable, y=value)) + geom_boxplot() + facet_wrap( ~ variable, scales=
```



Como podemos observar, existen numerosos valores fuera de los rangos en las variables *V3*, *V4*, *V5*, *V6*, *V10*, *V11* y *V12*, por lo que se procede a examinar dichos *outliers* potenciales.

Tras examinar los datos se determina que efectivamente se tratan de *outliers*, por lo que se procede a extraerlos y eliminarlos del conjunto de datos:

```
outlier_values_v3 <- boxplot.stats(bd_wine$V3)$out  # outlier values.
outlier_values_v4 <- boxplot.stats(bd_wine$V4)$out  # outlier values.
outlier_values_v5 <- boxplot.stats(bd_wine$V5)$out  # outlier values.
outlier_values_v6 <- boxplot.stats(bd_wine$V6)$out  # outlier values.
outlier_values_v10 <- boxplot.stats(bd_wine$V10)$out  # outlier values.
outlier_values_v11 <- boxplot.stats(bd_wine$V11)$out  # outlier values.
outlier_values_v12 <- boxplot.stats(bd_wine$V12)$out  # outlier values.

for(i in 1:length(outlier_values_v3)) bd_wine <- bd_wine[!bd_wine$V3 == outlier_values_v3[i], ]
for(i in 1:length(outlier_values_v4)) bd_wine <- bd_wine[!bd_wine$V4 == outlier_values_v4[i], ]
for(i in 1:length(outlier_values_v5)) bd_wine <- bd_wine[!bd_wine$V5 == outlier_values_v5[i], ]
for(i in 1:length(outlier_values_v6)) bd_wine <- bd_wine[!bd_wine$V6 == outlier_values_v6[i], ]
for(i in 1:length(outlier_values_v10)) bd_wine <- bd_wine[!bd_wine$V10 == outlier_values_v10[i], ]
for(i in 1:length(outlier_values_v11)) bd_wine <- bd_wine[!bd_wine$V11 == outlier_values_v11[i], ]
for(i in 1:length(outlier_values_v12)) bd_wine <- bd_wine[!bd_wine$V12 == outlier_values_v12[i], ]

bd_classes <- bd_wine[, "V1"]
bd_wine[, "V1"] <- NULL
```

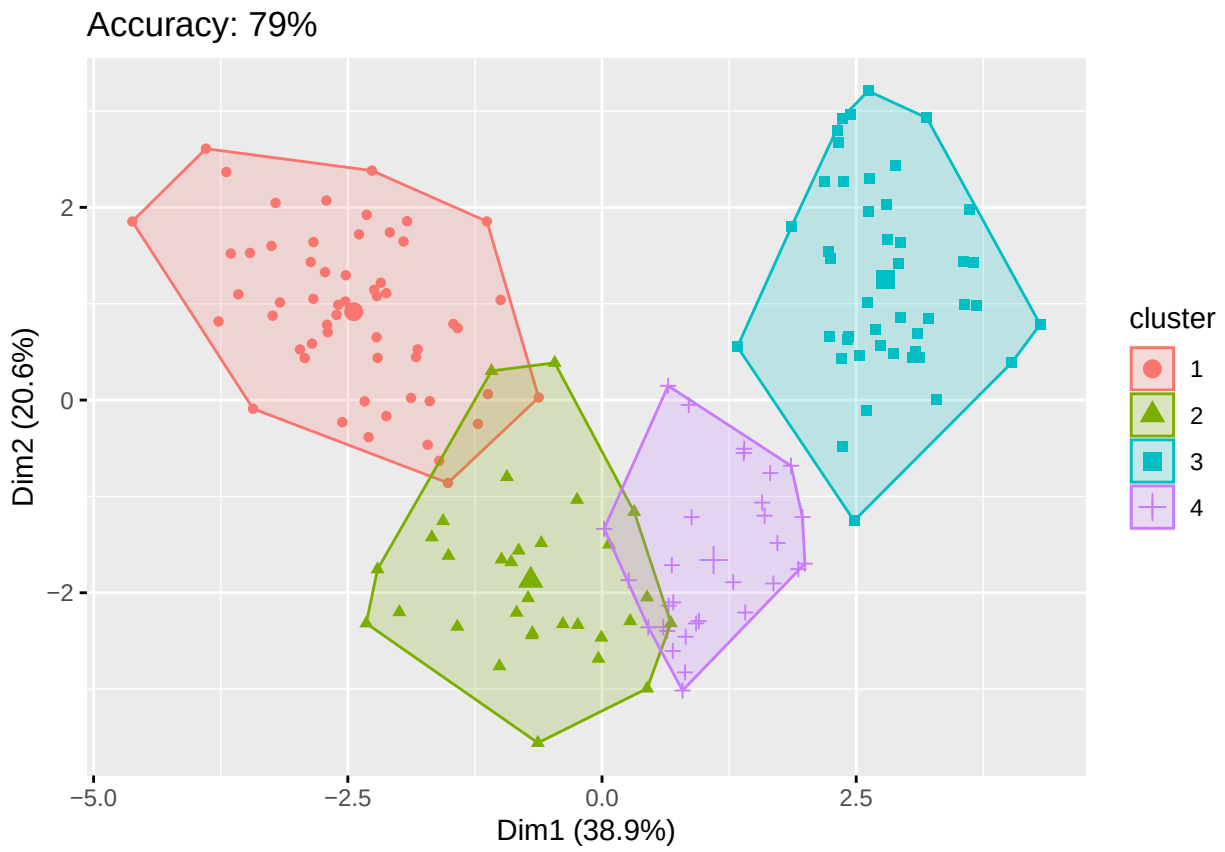
Algoritmo K-medias

Una vez se han eliminado los *outliers*, se procede a aplicar el algoritmo de K-medias. Para ello se empleará la función kmeans perteneciente a la librería stats. Para ello se utilizará el algoritmo con los siguientes argumentos:

- La base de datos de vinos: *bd_wine*.
- Número de clústers: 3.
- Número máximo de iteraciones: 10.

Una vez realizado el algoritmo de k-medias, se procede a visualizar los clusters y el acierto obtenido respecto a la clase original:

```
set.seed(1234)
km_cluster <- kmeans(bd_wine, 4, iter.max = 10)

acc <- paste0("Accuracy: ",round(100*length(which(km_cluster$cluster == bd_classes))/nrow(bd_wine)), "%

fviz_cluster(km_cluster, bd_wine, geom = "point", main=acc)
```



Reducción de dimensiones

La reducción de dimensión es el proceso de reducir el número de variables a un conjunto de valores de variables llamadas variables principales. Para esto se pueden utilizar dos metodologías principales de reducción de dimensiones:

- **Eliminación de características**: Consiste en eliminar características redundantes o que no aportan información nueva sobre el conjunto de datos.
- **Extracción de características**: Se basa en extraer información de las características obteniendo nuevas características a partir de estas. Si bien conserva la información original, las nuevas variables creadas pierden su interpretación.

En nuestro caso, vamos a realizar una **Eliminación de características**. Para ello realizaremos una matriz de correlación, y buscaremos aquellas variables que estén correladas con un porcentaje mayor al **60%**.

Para este proceso se crea inicialmente la matriz de correlación con la función cor, la cual nos permite partir de nuestro conjunto de datos obtener dicha matriz. Para encontrar las correlaciones superiores al porcentaje indicado, se utilizará la función findCorrelation del paquete caret.

A continuación se muestran las variables que se encuentran correladas y se eliminan del conjunto:

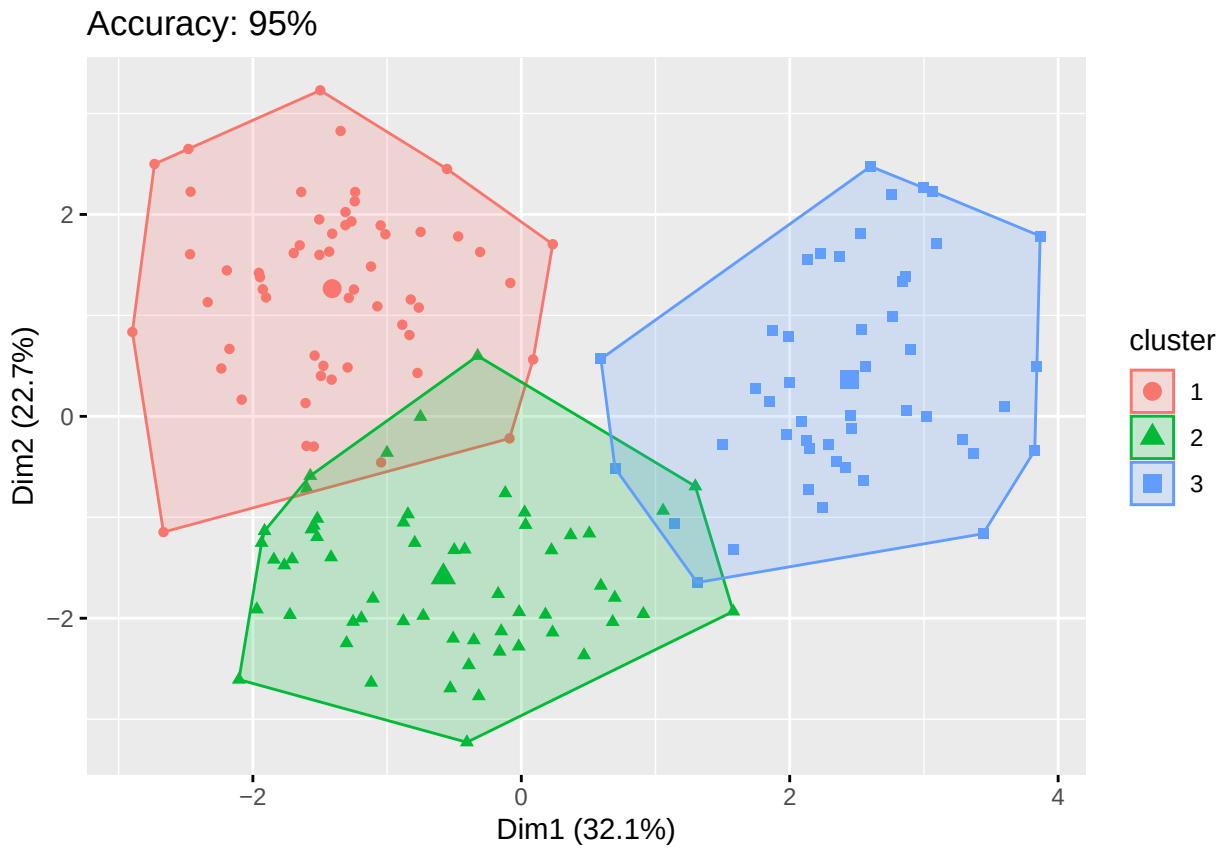
```
# Obtain correlation matrix
corr_matrix <- cor(bd_wine)
corr_matrix <- round(corr_matrix, 2)

# Find correlated variables
corr <- findCorrelation(corr_matrix, cutoff = .6, verbose = TRUE, names = TRUE)
```

```
## Compare row 7  and column  6 with corr  0.88
##   Means:  0.485 vs 0.32 so flagging column 7
## Compare row 6  and column  12 with corr  0.71
##   Means:  0.405 vs 0.295 so flagging column 6
## Compare row 13  and column  1 with corr  0.66
##   Means:  0.37 vs 0.281 so flagging column 13
## All correlations <= 0.6
```

```
# Erase correlated variables
for(i in 1:length(corr)) bd_wine[, corr[i]] <- NULL
```

A continuación se procede a realizar de nuevo el algoritmo de k-medias con 3 clústeres:

```
set.seed(1234)
km_cluster <- kmeans(bd_wine, 3, iter.max = 10)

acc <- paste0("Accuracy: ",round(100*length(which(km_cluster$cluster == bd_classes))/nrow(bd_wine)), "%

fviz_cluster(km_cluster, bd_wine, geom = "point", main=acc)
```



Algoritmo DBSCAN

Este es un método de agrupamiento adecuado para encontrar patrones de agrupamiento en el espacio físico. Este método es más adecuado para datos donde el método jerárquico no funciona bien debido al ruido y *outliers*. Este algoritmo agrupa los puntos más cercanos a una determinada métrica, generalmente utilizando la distancia euclidiana, y para este algoritmo, cada grupo debe contener la menor cantidad de puntos.

Para utilizar este algoritmo, se hará uso de la librería dbscan que contiene la función kNNdistplot que realiza una gráfica de las distancias k-nearest neighbor distances en una matriz de puntos, y que nos permitirá determinar el valor *eps* óptimo para la función dbscan, con la que podremos realizar el algoritmo.

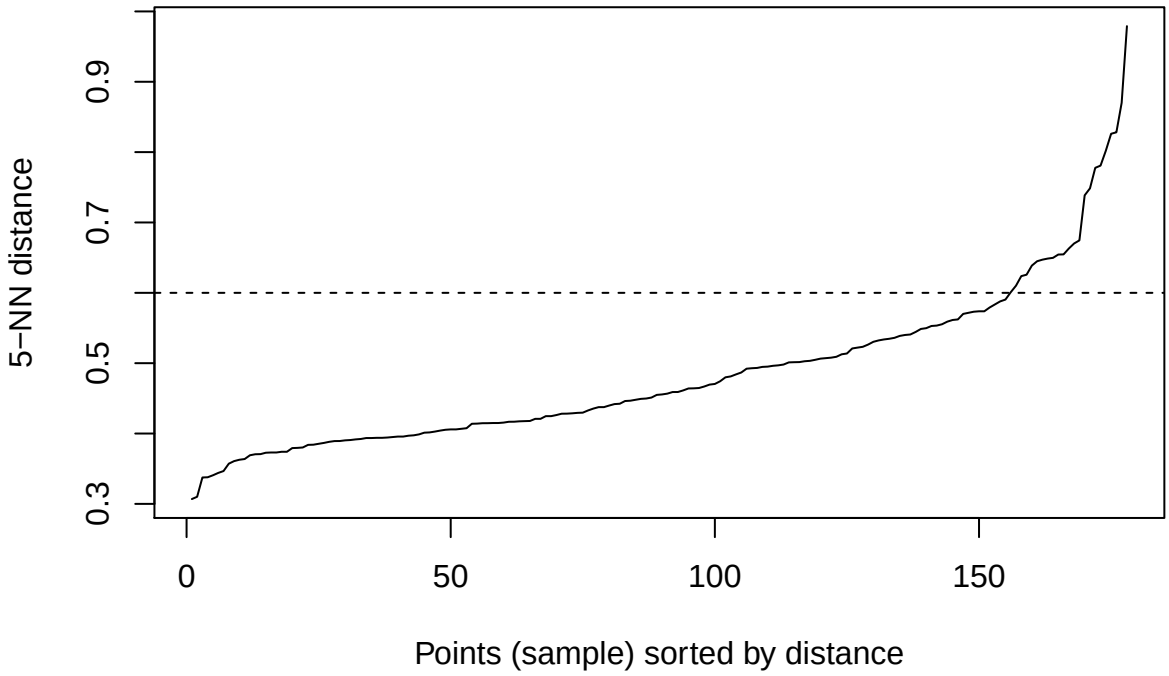
El principal problema a la hora de realizar el algoritmo de **DBSCAN** es la elección de parámetros adecuados, ya que una mala elección de estos realizará un agrupamiento no óptimo para el problema. El algoritmo utiliza los siguientes argumentos:

- **MinPts**: Número mínimo de puntos que deben estar en el algoritmo para formar un clúster. *MinPts >= D1*
- **eps**: Distancia de radio de los clúster.

Por norma general, el valor **MinPts** tomará el de mayor o igual que el número de dimensiones del conjunto de datos más uno. Sin embargo, una heurística muy utilizada es la de calcular el número de puntos como *ln(n)* donde *n* es el número de puntos. En este caso calcularemos nuestro **MinPts** como el redondeo del ln(n), siendo este 5.

Por otro lado, para el cálculo del valor **eps**, la técnica más común consiste en visualizar un histograma de **distancia kNN** y escoger un *knee*, que corresponde con el umbral donde se produce un cambio notable en la gráfica.

```r
# Read dataframe
bd_wine <- read.csv(f, header=FALSE)
# Omit NAs
bd_wine <- na.omit(bd_wine)
# Apply normalitzaion to our dataframe
bd_wine[-1] <- as.data.frame(lapply(bd_wine[-1], min_max_norm))

bd_classes <- bd_wine[, "V1"]
bd_wine[, "V1"] <- NULL

# Histogram of kNN distances
kNNdistplot(bd_wine, k = 5)
abline(h = 0.6, lty = 2)
```

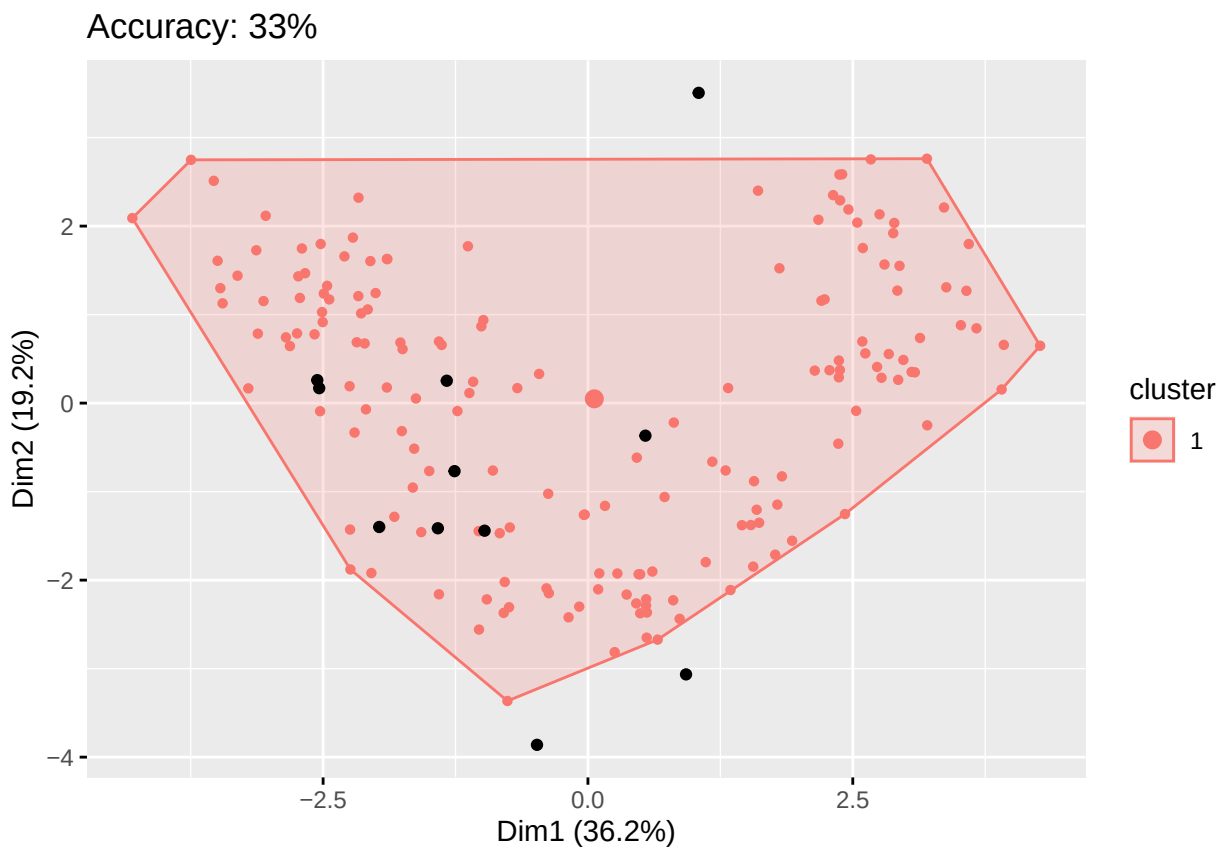


Como se puede observar existe una enorme cantidad de **ruido** en el conjunto de datos, por lo que resulta prácticamente imposible determinar un valor **eps** que sea válido, a la vez que resulta difícil para el algoritmo encontrar un agrupamiento correcto para los datos. Esto se puede observar ya que solo se crea un clúster, por lo que evidentemente no predice las clases correctamente:

```r
dbscan_cluster<-dbscan(bd_wine,eps=0.6,MinPts = 5)
```

```
## Warning in dbscan(bd_wine, eps = 0.6, MinPts = 5): converting argument MinPts
## (fpc) to minPts (dbscan)!
```

```
acc <- paste0("Accuracy: ",round(100*length(which(dbscan_cluster$cluster == bd_classes))/nrow(bd_wine)))

fviz_cluster(dbscan_cluster, bd_wine, geom = "point", main=acc)
```

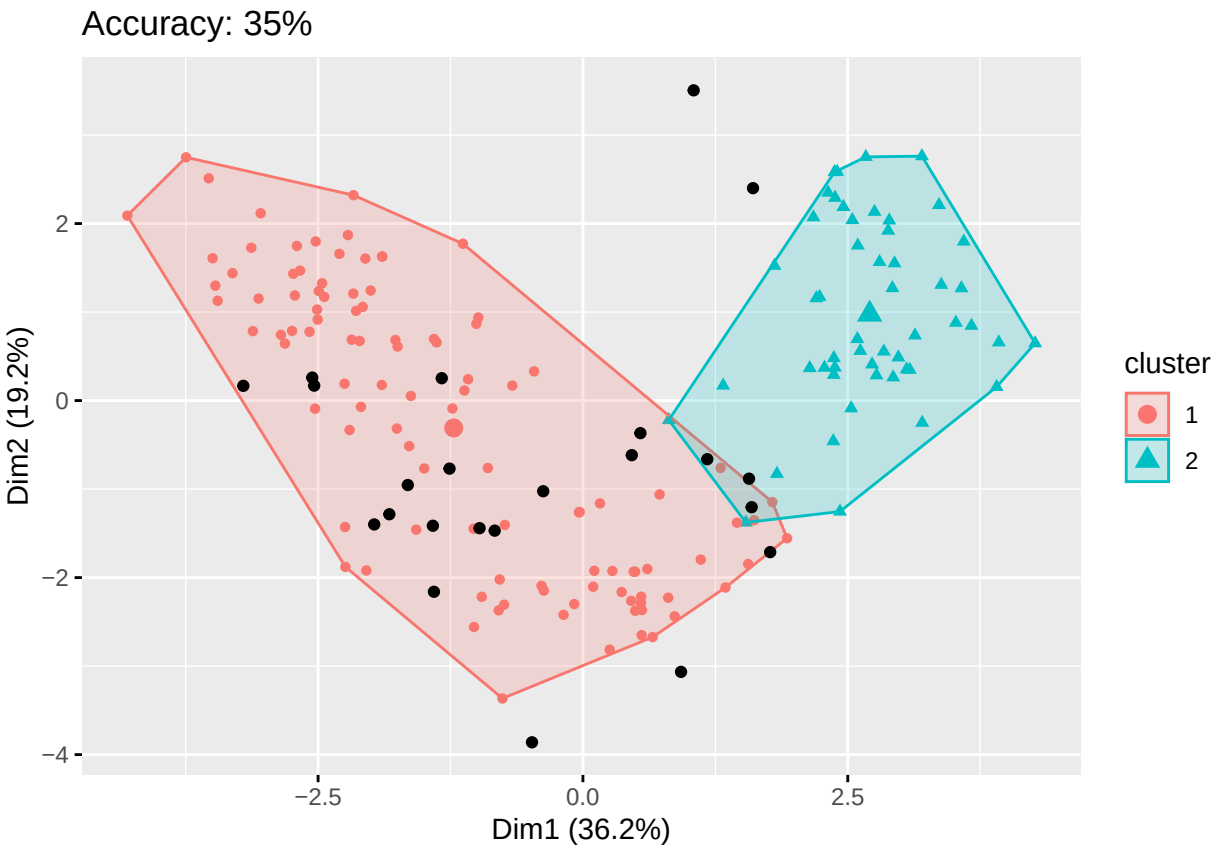


Por otro lado, podríamos intentar ajustar haciendo pruebas con los parámetros para intentar obtener un buen resultado con el algoritmo, pero tal y como se comenta previamente, con la enorme cantidad de ruido que posee el conjunto de datos, resulta casi imposible encontrar unos buenos parámetros que permitan encontrar una configuración adecuada. Se ha realizado a continuación un ejemplo con unos parámetros que se adecuan más al problema, pero que sin embargo resultarían impredecibles en primera instancia:

```
dbscan_cluster<-dbscan(bd_wine,eps=0.5,MinPts = 5)
```

```
## Warning in dbscan(bd_wine, eps = 0.5, MinPts = 5): converting argument MinPts
## (fpc) to minPts (dbscan)!
```

```
acc <- paste0("Accuracy: ",round(100*length(which(dbscan_cluster$cluster == bd_classes))/nrow(bd_wine)))

fviz_cluster(dbscan_cluster, bd_wine, geom = "point", main=acc)
```



Conclusiones

Tras realizar los diferentes puntos de la práctica se han llegado a las siguientes respuestas y conclusiones:

- A la pregunta de `Analizar en cuántos clusters diferentes podríamos agrupar los datos.` relativa al agrupamiento jerárquico, cabe destacar que no solo depende del algoritmo utilizado ni del dendograma obtenido como resultado, se puede observar que también hay un factor de decisión e interpretación de los datos, ya que al realizar un *Clustering Jerárquico aglomerativo* se ha obtenido que el mejor resultado podría ser agrupar en 3 clústeres, mientras que realizando un *Clustering Jerárquico divisivo* se ha obtenido como mejor opción agrupar en 4 clústeres. Realmente tendría más sentido escoger 3 clústeres de antemano ya que se conoce el número de clases, pero se trata de un problema en el que suponemos que no tenemos dicha información, por lo que realmente la elección final depende del conocimiento sobre el problema además de los resultados obtenidos.

- Tal y como se explica en la sección referente a **outliers**, no solo basta con ver que valores salen de la escala normal, sino también comprobar si dichos valores son realmente atípicos o no, por lo que de nuevo se necesita conocimiento sobre el problema a tratar para tomar la decisión final de si se pueden considerar dichos valores como *outliers* y deben eliminarse o no.

- En el caso del algoritmo **k-medias**, se ha podido observar que se desenvuelve con un gran resultado, incluso indicándole un número de clústeres mayor que el número de clases inicialmente conocido (supuestamente desconocido), por lo que en un problema en el que se desconoce completamente la clasificicación, y que posee una enorme cantidad de ruido, podría ser una gran opción a considerar. Eso sí, en este caso evidentemente se desenvuelve mejor con 3 clústeres que con 4.

- Tras entender y analizar las técnicas de reducción de dimensiones, se puede comprender la importancia de trasladar un problema con una cantidad excesiva de variables a un problema con un conjunto reducido de variables puede ser una herramienta muy útil para eliminar ruido e información innecesaria que puede entorpecer nuestro desempeño. Aunque existen numerosas técnicas, he decidido centrarme en el reducción de variables significativas ya que la extracción de información resulta en un proceso más complejo y en el que las variables pueden llegar a carecer de significado. Además se puede observar que el agrupamiento realizado con el algoritmo de *k-medias* ha obtenido un muy buen resultado bastante aproximado a la clasificación real.
- Aunque el algoritmo **DBSCAN** es una gran alternativa para conjuntos de datos con exceso de *outliers* y ruido, es un algoritmo capaz de detectar dichos valores atípicos y señalarlos, además de trabajar con el ruido e intentar disminuir su efecto lo máximo posible. Desgraciadamente, en el conjunto de datos propuesto, existe una gran cantidad de ruido que no permite ver un mejor desempeño del algoritmo. Dicho ruido es fácilmente identificable en el histograma realizado de *distancia kNN*.