UIMP Universidad Internacional Menéndez Pelayo

Práctica en R de Detección de Anomalías

Sergio Sáez Bombín [100007762@alumnos.uimp.es]

12-03-2023

Índice

1. Dataset y Selección de Variables **3**

2. Detección de outliers en una dimensión **5**
2.1. Outliers IQR . . . 5
2.1.1. Obención de los outliers IQR . . . 6
2.1.2. Índices y valores de los outliers IQR . . . 7
2.1.3. Cómputo de los outliers IQR con funciones . . . 8
2.1.4. Desviación de los outliers con respecto a la media de la columna . . . 8
2.1.5. Gráfico . . . 9
2.1.6. Diagramas de cajas . . . 11
2.2. Tests de hipótesis (OPCIONAL) . . . 13
2.2.1. Comprobación de la hipótesis de Normalidad . . . 13
2.2.2. Test de Grubbs . . . 14
2.2.3. Test de Normalidad . . . 14
2.3. Trabajando con varias columnas . . . 18
2.3.1. Outliers IQR . . . 18
2.3.2. Tests de hipótesis (OPCIONAL) . . . 20

3. Outliers multivariantes **23**
3.1 Métodos estadísticos basados en la distancia de Mahalanobis (OPCIONAL) . . . 23
3.1.1 Hipótesis de Normalidad . . . 23
3.1.2 Tests de hipótesis para detectar outliers . . . 23
3.2 Visualización de datos con un Biplot . . . 25
3.3 Métodos basados en distancias: LOF . . . 26
3.4 Métodos basados en Clustering . . . 31
3.4.1 Clustering usando k-means . . . 31
3.4.2 Clustering usando medoides (OPCIONAL) . . . 33
3.5 Análisis de los outliers multivariantes puros . . . 36

4. Análisis de resultados **39**
4.1. Conjunto de datos . . . 39
4.2. Outliers en una variable . . . 39
4.2.1. Método IQR . . . 39
4.2.2. Test de Hipótesis . . . 39
4.3. Outliers multivariantes . . . 39
4.3.1. Visualización con biplot . . . 39
4.3.2. Métodos estadísticos usando la distancia de Mahalanobis . . . 39
4.3.3. LOF . . . 39
4.3.4. Métodos basados en clustering . . . 40

1. Dataset y Selección de Variables

Para esta práctica se ha decidido optar por un dataset de los presentes en ODDS conformado por variables características de los vinos, amplia e históricamente utilizado. Este dataset es **Wine**, del UCI Machine Learning Repository. El fichero RMarkdown relativo a este trabajo se encuentra en el siguiente enlace de GitHub.

En primer lugar, vamos a cargar el dataset elegido en R, con el objetivo de comenzar la exploración de este y la toma de decisiones en cuanto a su preparación para nuestro modelo de detección de outliers:

```
datos <- read.csv('./wine.data')
names(datos) <- c('CLASS','ALC','ACMA','CEN','ALCEN','MAG','FET','FLA','FNF',
                  'PRO','IC','MAT','OD','PROL')
row.names(datos) <- as.character(c(1:nrow(datos)))
head(datos)
```

```
##   CLASS   ALC ACMA  CEN ALCEN MAG  FET  FLA  FNF  PRO   IC  MAT   OD PROL
## 1     1 13.20 1.78 2.14  11.2 100 2.65 2.76 0.26 1.28 4.38 1.05 3.40 1050
## 2     1 13.16 2.36 2.67  18.6 101 2.80 3.24 0.30 2.81 5.68 1.03 3.17 1185
## 3     1 14.37 1.95 2.50  16.8 113 3.85 3.49 0.24 2.18 7.80 0.86 3.45 1480
## 4     1 13.24 2.59 2.87  21.0 118 2.80 2.69 0.39 1.82 4.32 1.04 2.93  735
## 5     1 14.20 1.76 2.45  15.2 112 3.27 3.39 0.34 1.97 6.75 1.05 2.85 1450
## 6     1 14.39 1.87 2.45  14.6  96 2.50 2.52 0.30 1.98 5.25 1.02 3.58 1290
```

Como se puede ver, el dataset está conformado por trece variables descriptoras (y una variable de clase, correspondiente a tres tipos de vinos) relativas a las **características físicas y químicas** del vino:

- **Alcohol *(ALC)***: El alcohol es el componente principal del vino y se refiere a la cantidad de etanol presente en la bebida. El alcohol del vino es producido a través de la fermentación del azúcar.
- **Ácido málico *(ACMA)***: El ácido málico es uno de los ácidos orgánicos presentes en el vino. Aporta acidez y un sabor refrescante a la bebida.
- **Ceniza *(CEN)***: La ceniza se refiere a la cantidad total de minerales presentes en el vino, como calcio, potasio, sodio, hierro, entre otros.
- **Alcalinidad de la ceniza**: La alcalinidad de la ceniza es la medida de la capacidad de los minerales presentes en la ceniza para neutralizar ácidos.
- **Magnesio *(MAG)***: El magnesio es un mineral presente en el vino que contribuye a su sabor y aroma.
- **Fenoles totales *(FET)***: Los fenoles son compuestos orgánicos presentes en el vino que aportan sabor, aroma y color. La cantidad total de fenoles presentes en el vino es conocida como fenoles totales.
- **Flavonoides *(FLA)***: Los flavonoides son un tipo de fenol presente en el vino que aporta sabor, aroma y color.
- **Fenoles no flavonoides *(FNF)***: Los fenoles no flavonoides son otro tipo de fenol presente en el vino que también contribuye al sabor y aroma.
- **Proantocianidinas *(PRO)***: Las proantocianidinas son un tipo de flavonoide presente en el vino que aporta sabor y aroma.
- **Intensidad del color *(IC)***: La intensidad del color se refiere a la profundidad del color del vino.
- **Matiz *(MAT)***: El matiz se refiere al tono del color del vino.
- **OD280/OD315 de vinos diluidos *(OD)***: La relación OD280/OD315 se refiere a la absorbancia de la luz a una longitud de onda específica en el espectro del vino y se utiliza para medir la cantidad de proteína presente en la bebida.

- **Prolina*(PROL)***: La prolina es un aminoácido presente en el vino que contribuye al sabor y aroma de la bebida.

Además, si ejecutamos un `str(...)` sobre el dataframe, podremos ver que tiene 177 observaciones y son numéricas:

```
str(datos)
```

```
## 'data.frame':    177 obs. of  14 variables:
##  $ CLASS: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ ALC  : num  13.2 13.2 14.4 13.2 14.2 ...
##  $ ACMA : num  1.78 2.36 1.95 2.59 1.76 1.87 2.15 1.64 1.35 2.16 ...
##  $ CEN  : num  2.14 2.67 2.5 2.87 2.45 2.45 2.61 2.17 2.27 2.3 ...
##  $ ALCEN: num  11.2 18.6 16.8 21 15.2 14.6 17.6 14 16 18 ...
##  $ MAG  : int  100 101 113 118 112 96 121 97 98 105 ...
##  $ FET  : num  2.65 2.8 3.85 2.8 3.27 2.5 2.6 2.8 2.98 2.95 ...
##  $ FLA  : num  2.76 3.24 3.49 2.69 3.39 2.52 2.51 2.98 3.15 3.32 ...
##  $ FNF  : num  0.26 0.3 0.24 0.39 0.34 0.3 0.31 0.29 0.22 0.22 ...
##  $ PRO  : num  1.28 2.81 2.18 1.82 1.97 1.98 1.25 1.98 1.85 2.38 ...
##  $ IC   : num  4.38 5.68 7.8 4.32 6.75 5.25 5.05 5.2 7.22 5.75 ...
##  $ MAT  : num  1.05 1.03 0.86 1.04 1.05 1.02 1.06 1.08 1.01 1.25 ...
##  $ OD   : num  3.4 3.17 3.45 2.93 2.85 3.58 3.58 2.85 3.55 3.17 ...
##  $ PROL : int  1050 1185 1480 735 1450 1290 1295 1045 1045 1510 ...
```

Siguiendo la práctica, primero se comprueba que las variables sean numéricas. En el caso de este dataset, estrictamente hablando, inicialmente no se detectan como numéricas algunas de ellas, sino como enteros. Sin embargo, el tipo de dato numérico también contiene a los enteros, por lo tanto, al pasar estas columnas por la función `is.numeric(...)` se van a detectar como numéricas:

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

A continuación, se va a evaluar cada columna (variable) para determinar si contienen pocos valores distintos, además de la variable clase. Para calcular la varianza de cada columna se utiliza la función `var(...)`, y se va a considerar una varianza baja aquella que sea menor que el 10% (0.1):

```
## [1] "Varianza de cada columna (columnas.var):"
```

```
##          ALC         ACMA          CEN         ALCEN          MAG          FET
##   0.59906266   0.65417110   1.25286476   0.07566925  11.12937018 200.90279918
##          FLA          FNF          PRO           IC          MAT           OD
##   0.39245850   0.99731703   0.01553835   0.32666337   5.40305118   0.05250287
##         PROL
##   0.49717010
```

```
## [1] "Datos tras eliminar las columnas con poca varianza:"
```

```
##     ALC ACMA  CEN MAG  FET  FLA  FNF   IC  MAT PROL
## 1 13.20 1.78 2.14 100 2.65 2.76 0.26 4.38 1.05 1050
## 2 13.16 2.36 2.67 101 2.80 3.24 0.30 5.68 1.03 1185
## 3 14.37 1.95 2.50 113 3.85 3.49 0.24 7.80 0.86 1480
## 4 13.24 2.59 2.87 118 2.80 2.69 0.39 4.32 1.04  735
## 5 14.20 1.76 2.45 112 3.27 3.39 0.34 6.75 1.05 1450
## 6 14.39 1.87 2.45  96 2.50 2.52 0.30 5.25 1.02 1290
```

Vemos que se han eliminado, además de la variable clase, las variables *ALCEN*, *PRO* y *OD*. Por otro lado, en la propia descripción original del dataset se nos indica que no hay valores faltantes en ninguna de las observaciones, por lo que no sería necesario utilizar la función `na.omit(...)` (de todos modos la aplicamos para asegurarnos de que efectivamente no se utiliza ningún NA).

2. Detección de outliers en una dimensión

2.1. Outliers IQR

El primer apartado de outliers es el correspondiente al método IQR. Tal y como se indica en la práctica, este método solamente es aplicable con distribuciones normales (o semejantes), por lo que a continuación se muestran las distribuciones de todas las de nuestro dataset:

```
# COMPLETAR
par(mfrow=c(2,3))
histogramas <- sapply(c(1:ncol(datos.num)),
  function(x) hist(datos.num[, x], main="", xlab=names(datos.num)[x]))
```

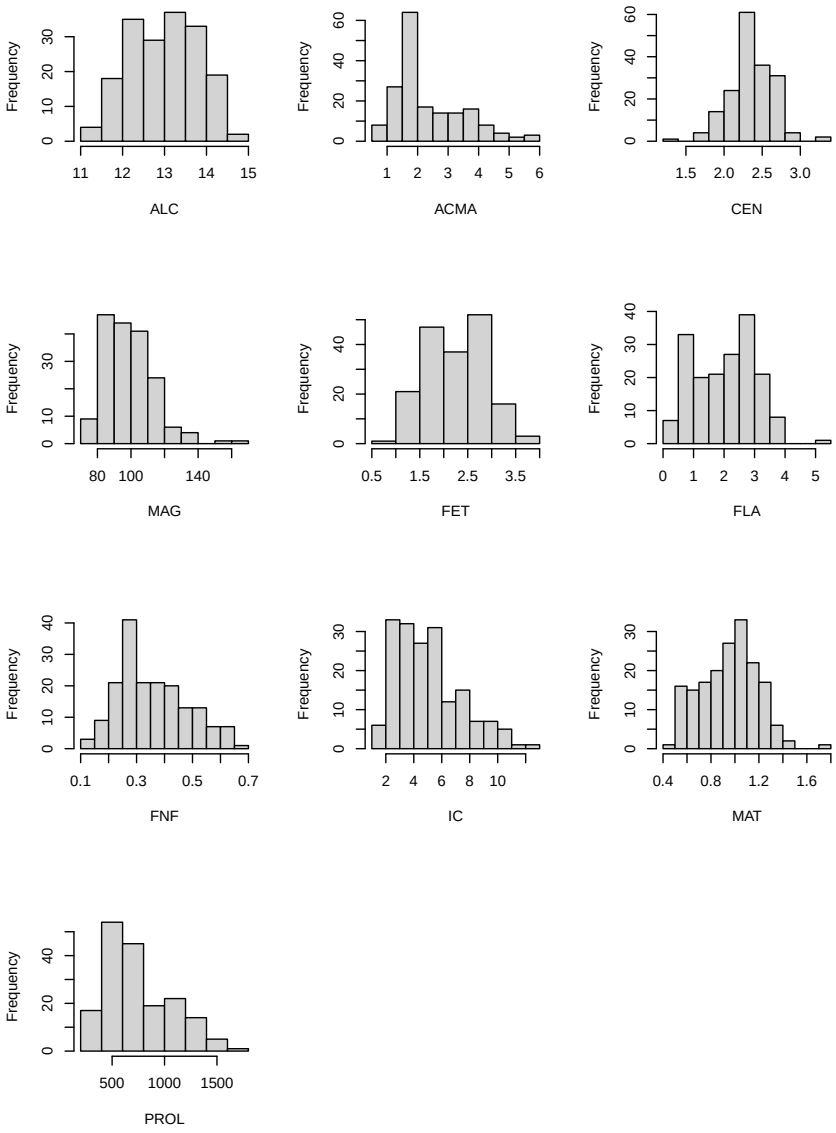

Figura 1: Histogramas de cada una de las variables

Ahora, se van a definir las variables `columna`, `indice.columna` y `datos.columna` para localizar la columna (variable) con la que se realizarán las operaciones en la primera parte de la práctica.

En este caso, se utilizará la variable *CEN*, que es una de las que presenta una distribución más cercana a la normal, si bien todas presentan distribuciones en las que se podría considerar que solo hay una moda presente y con una o dos colas de datos de poca frecuencia. En todo caso, cabría destacar la variable *FLA*, que presenta una distribución con dos cajas prominentes un tanto separadas, lo que podría indicar dos modas (se obvia este hecho y se continúa con el análisis, ya que la inspección de las distribuciones no es exhaustiva y esta no se ve como un caso claro de no normalidad).

2.1.1. Obención de los outliers IQR

A continuación, se dispone a calcular los outliers IQR, que se guardarán en las variables `son.outliers.IQR` y `son.outliers.IQR.extremos`, tal y como se vio en el tutorial de la práctica:

```
# COMPLETAR

cuartil.primero = quantile(columna, 0.25)
mediana = quantile(columna, 0.5)
cuartil.tercero = quantile(columna, 0.75)
iqr = IQR(columna)

extremo.superior.outlier.IQR = cuartil.tercero + 1.5*iqr
extremo.inferior.outlier.IQR = cuartil.primero - 1.5*iqr
extremo.superior.outlier.IQR.extremo = cuartil.tercero + 3*iqr
extremo.inferior.outlier.IQR.extremo = cuartil.primero - 3*iqr

son.outliers.IQR = columna > extremo.superior.outlier.IQR |
                   columna < extremo.inferior.outlier.IQR
son.outliers.IQR.extremos = columna > extremo.superior.outlier.IQR.extremo |
                            columna < extremo.inferior.outlier.IQR.extremo

indice.columna
```

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

```
nombre.columna
```

```
## [1] "CEN"
```

```
cuartil.primero
```

```
##  25%
## 2.21
```

```
cuartil.tercero
```

```
##  75%
## 2.56
```

```
iqr
```

```
## [1] 0.35
```

```
extremo.superior.outlier.IQR
```

```
##   75%
## 3.085
```

```
extremo.inferior.outlier.IQR
```

```
##   25%
## 1.685
```

```
extremo.superior.outlier.IQR.extremo
```

```
##  75%
## 3.61
```

```
extremo.inferior.outlier.IQR.extremo
```

```
##  25%
## 1.16
```

```
head(son.outliers.IQR)
```

```
## [1] FALSE FALSE FALSE FALSE FALSE FALSE
```

```
sum(son.outliers.IQR)
```

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

```
head(son.outliers.IQR.extremos)
```

```
## [1] FALSE FALSE FALSE FALSE FALSE FALSE
```

```
sum(son.outliers.IQR.extremos)
```

```
## [1] 0
```

El resultado obtenido es que tenemos tres outliers (ninguno extremo) en la variable *CEN*, esto es, que se encuentra bien en el rango $x > q_3 + 1.5IQR$, o bien en el rango $x > q_1 - 1.5IQR$.

2.1.2. Índices y valores de los outliers IQR

Ahora, es de interés encontrar estos outliers, es decir, obtener tanto su posición en la columna *CEN* como su valor. Para ello, se calculan las variables (tal y como se define en la práctica): `claves.outliers.IQR`, `df.outliers.IQR`, `nombres.outliers.IQR` y `valores.outliers.IQR` (de igual manera para los valores extremos):

```
# COMPLETAR

claves.outliers.IQR = which(son.outliers.IQR, arr.ind=TRUE)
df.outliers.IQR = datos.num[claves.outliers.IQR,]
nombres.outliers.IQR = row.names(df.outliers.IQR)
valores.outliers.IQR = df.outliers.IQR[,indice.columna]

claves.outliers.IQR.extremos = which(son.outliers.IQR.extremos, arr.ind=TRUE)
df.outliers.IQR.extremos = datos.num[claves.outliers.IQR.extremos,]
nombres.outliers.IQR.extremos = row.names(df.outliers.IQR.extremos)
valores.outliers.IQR.extremos = df.outliers.IQR.extremos[,indice.columna]

claves.outliers.IQR
```

```
## [1]  25  59 121
```

```
df.outliers.IQR
```

```
##       ALC ACMA  CEN MAG  FET  FLA  FNF   IC  MAT PROL
## 25  13.05 2.05 3.22 124 2.63 2.68 0.47 3.58 1.13  830
## 59  12.37 0.94 1.36  88 1.98 0.57 0.28 1.95 1.05  520
## 121 11.56 2.05 3.23 119 3.18 5.08 0.47 6.00 0.93  465
```

```
nombres.outliers.IQR
```

```
## [1] "25"  "59"  "121"
```

```
valores.outliers.IQR
```

```
## [1] 3.22 1.36 3.23
```

```
claves.outliers.IQR.extremos
```

```
## integer(0)
```

```
df.outliers.IQR.extremos
```

```
##  [1] ALC  ACMA CEN  MAG  FET  FLA  FNF  IC   MAT  PROL
## <0 rows> (or 0-length row.names)
```

```
nombres.outliers.IQR.extremos
```

```
## character(0)
```

```
valores.outliers.IQR.extremos
```

```
## numeric(0)
```

Como vemos, los tres outliers encontrados son los correspondientes a los vinos ***25***, ***59*** y ***121***, que se analizarán en las siguientes secciones.

2.1.3. Cómputo de los outliers IQR con funciones

Siguiendo la práctica, se obtienen estos mismos outliers pero haciendo uso de las funciones disponibles dadas en la práctica.

```
## [1] FALSE FALSE FALSE FALSE FALSE FALSE
```

```
## [1]  25  59 121
```

```
## [1] FALSE FALSE FALSE FALSE FALSE FALSE
```

```
## integer(0)
```

Como era de esperar, se obtienen los mismos tres outliers que antes.

2.1.4. Desviación de los outliers con respecto a la media de la columna

En este apartado se va a realizar el mismo estudio de outliers con el método IQR, pero esta vez sobre los datos normalizados (este análisis se realiza sobre la misma columna *CEN*):

```
##         ALC        ACMA        CEN         MAG       FET       FLA        FNF
## 1 0.2551008 -0.50020530 -0.8221529  0.02909756 0.5710456 0.7375437 -0.8208101
## 2 0.2056453  0.01796903  1.1045562  0.09964918 0.8104843 1.2181890 -0.4999191
## 3 1.7016732 -0.34832662  0.4865552  0.94626865 2.4865554 1.4685250 -0.9812556
## 4 0.3045563  0.22345196  1.8316163  1.29902677 0.8104843 0.6674496  0.2220856
## 5 1.4914875 -0.51807338  0.3047901  0.87571703 1.5607256 1.3683906 -0.1790282
## 6 1.7264010 -0.41979894  0.3047901 -0.25310893 0.3316069 0.4972211 -0.4999191
##            IC        MAT        PROL
## 1 -0.29030665  0.4059482  0.96830550
## 2  0.26896630  0.3186634  1.39703475
## 3  1.18101141 -0.4232572  2.33388755
## 4 -0.31611925  0.3623058 -0.03206274
## 5  0.72929095  0.4059482  2.23861439
## 6  0.08397601  0.2750210  1.73049083
```

Calculando los valores de los outliers, se obtiene, como era de esperar, el mismo resultado que antes, solamente que ahora los valores están normalizados frente a una $N(0,1)$, por lo que se pueden identificar más fácilmente:

```
# COMPLETAR

son.outliers.IQR.norm     = son_outliers_IQR (datos.num.norm, indice.columna)
head(son.outliers.IQR.norm)
```

```
## [1] FALSE FALSE FALSE FALSE FALSE FALSE
```

```
claves.outliers.IQR.norm  = claves_outliers_IQR (datos.num.norm, indice.columna)
claves.outliers.IQR.norm
```

```
## [1]  25  59 121
```

```
df.outliers.IQR.norm = datos.num.norm[claves.outliers.IQR.norm,]
nombres.outliers.IQR.norm = row.names(df.outliers.IQR.norm)
valores.outliers.IQR.norm = df.outliers.IQR.norm[,indice.columna]

valores.outliers.IQR.norm
```

```
## [1]  3.103971 -3.657687  3.140324
```

Finalmente, obtenemos todos los valores de todas las variables para cada uno de los outliers:

```
# COMPLETAR

datos.num.norm.outliers.IQR = df.outliers.IQR.norm

datos.num.norm.outliers.IQR
```

```
##            ALC       ACMA       CEN        MAG        FET        FLA        FNF
## 25   0.0696428 -0.2589862  3.103971  1.7223365  0.5391204  0.6574362  0.8638675
## 59  -0.7711002 -1.2506647 -3.657687 -0.8175219 -0.4984474 -1.4554000 -0.6603646
## 121 -1.7725734 -0.2589862  3.140324  1.3695784  1.4170624  3.0606623  0.8638675
##             IC        MAT       PROL
## 25  -0.6344746  0.7550872  0.2696356
## 59  -1.3357168  0.4059482 -0.7148538
## 121  0.4066335 -0.1177605 -0.8895212
```

Tras normalizarlos, podemos ver que los tres registros van a representar un outlier en otras variables distintas a la de *CEN*. Por ejemplo, considerando que -2.68 (+2.68) es el extremo que delimita un outlier IQR y -4.69 (+4.69) el del outlier extremo IQR, podemos decir que:

- Los vinos de etiqueta ***25*** y ***59*** son ***outliers IQR (no extremos)*** solamente en la variable ***CEN***.
- El vino de etiqueta ***121*** es un ***outlier IQR (no extremo)*** en las variables ***CEN*** y ***FLA***.

2.1.5. Gráfico

También es de interés mostrar visualmente la presencia de estos outliers (y los extremos) dentro del conjunto de datos de la propia variable *CEN*:

Con los datos normalizados se ve fácilmente la disposición de los 3 outliers. Sin embargo, también se puede ver que estos no son considerados outliers extremos.

```
# COMPLETAR

plot_2_colores(datos.num.norm[,3], claves.outliers.IQR, c('CEN'))
```

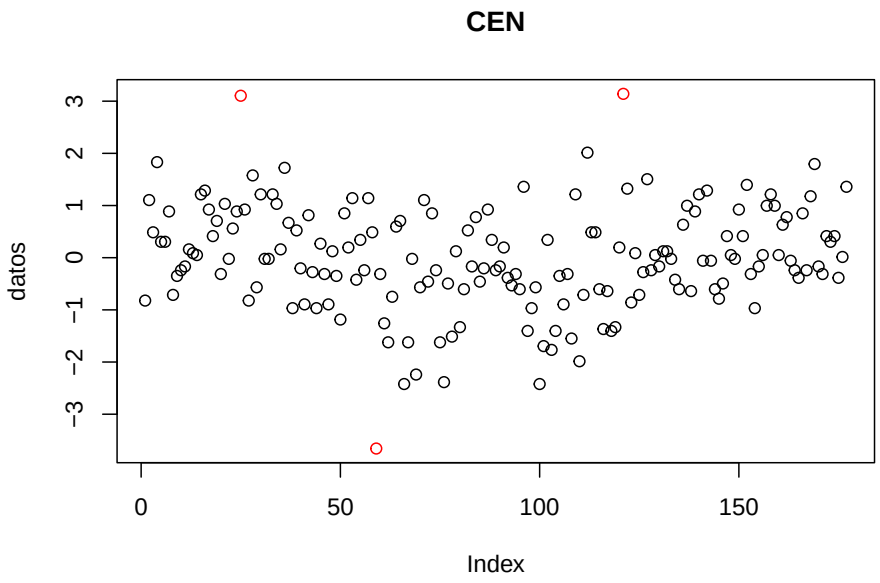

Figura 2: Datos en la variable CEN con outliers IQR

En rojo quedan marcados los tres outliers IQR calculados previamente. Con esta representación sí se ve a simple vista que, efectivamente son outliers, ya que se ve claramente que superan el límite de -2.68 (+2.68). Por otro lado, en el caso de outliers extremos:

```
# COMPLETAR

plot_2_colores(datos.num.norm[,3], claves.outliers.IQR.extremos, c('CEN'))
```

Efectivamente no hay puntos rojos porque no se ha localizado ningún outlier IQR extremo (no hay vlaores normalizados por encima/debajo de +4.69/-4.69).

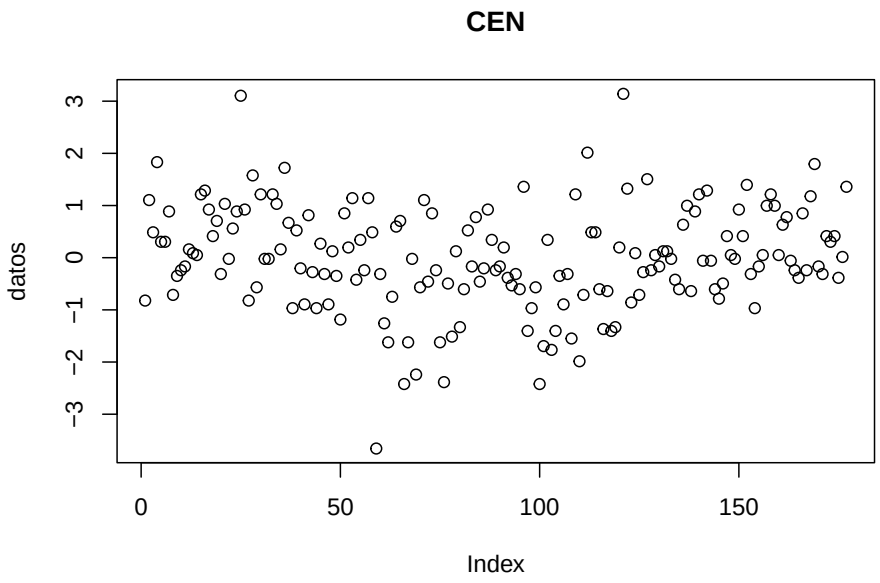

Figura 3: Datos en la variable CEN con outliers extremos IQR

2.1.6. Diagramas de cajas

Otro método de ver los outliers es a través de un diagrama de cajas. En la práctica ya se nos da una función para sacar estas cajas y ver de manera sencilla la presencia de los outliers:

```
# COMPLETAR

diag_caja_outliers_IQR(datos.num, indice.columna) +
  ggtitle("Boxplot con datos originales")
diag_caja_outliers_IQR(datos.num.norm, indice.columna) +
  ggtitle("Boxplot con datos normalizados")
```

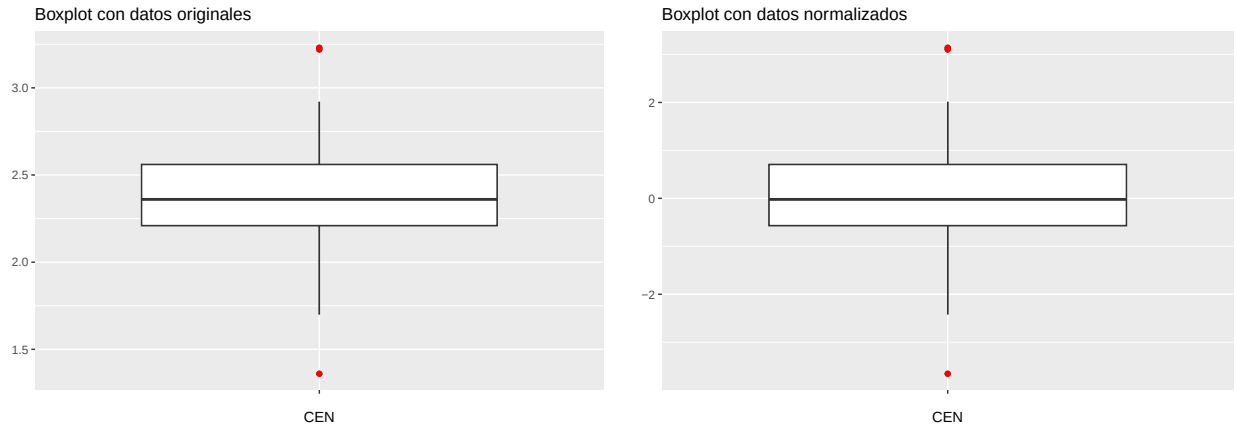

Figura 4: Boxplots de la variable CEN con datos originales y normalizados

Si les añadimos las etiquetas:

```
# COMPLETAR

diag_caja(datos.num, indice.columna, claves.outliers.IQR) +
  ggtitle("Boxplot con datos originales con etiquetas")
diag_caja(datos.num.norm, indice.columna, claves.outliers.IQR.norm) +
  ggtitle("Boxplot con datos normalizados con etiquetas")
```

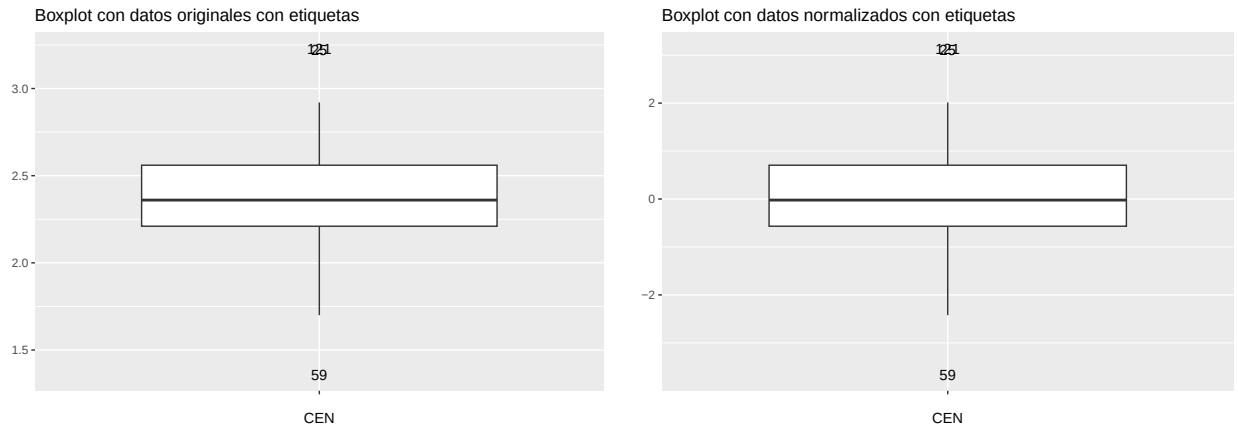

Figura 5: Boxplots de la variable CEN con datos originales y normalizados con etiquetas

Desafortunadamente, tenemos dos outliers demasiado juntos, y las etiquetas 121 y 25 se superponen (aunque se pueden intuir).

Y ya para terminar con esta sección, se obtienen las cajas de estos registros que presentan outliers para todas las variables:

```
# COMPLETAR

diag_caja_juntos(datos.num.norm, "Outliers en alguna columna", claves.outliers.IQR.norm)
```

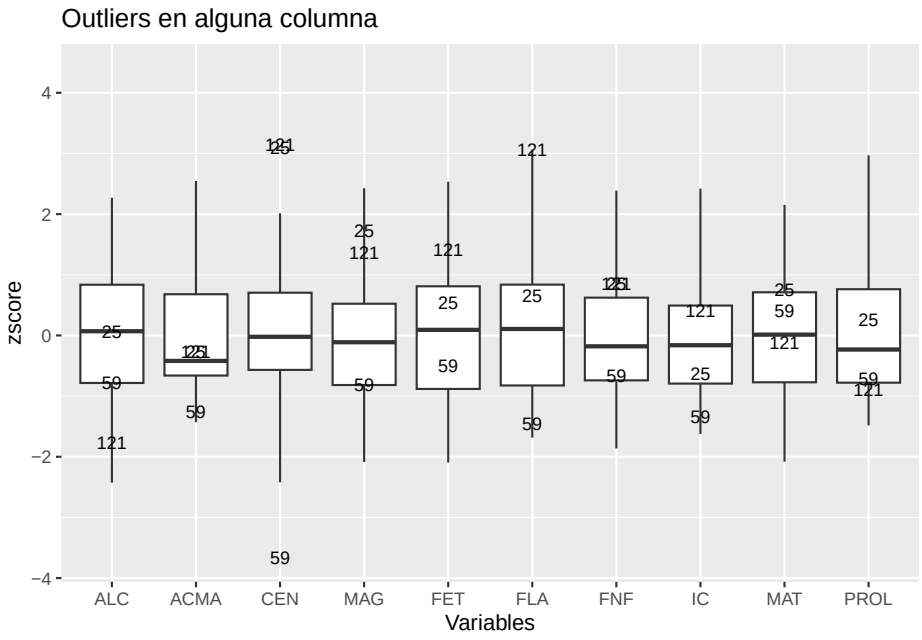

Figura 6: Boxplots de todas las variables con datos normalizados y etiquetas

Este plot nos ayuda a confirmar lo que se indicaba anteriormente, que los vinos ***25*** y **59** solo son outliers IQR en la variable ***CEN***, mientras que el vino de etiqueta ***121*** lo es en las variables ***CEN*** y ***FLA***.

2.2. Tests de hipótesis (OPCIONAL)

En este apartado se busca poder determinar de manera estadística si los puntos más alejados de la media distribucional son outliers o no.

2.2.1. Comprobación de la hipótesis de Normalidad

En primer lugar se comprueba si los datos siguen una distribución normal. Esto, tal y como se indica en el enunciado de la práctica, se realiza de manera informal.

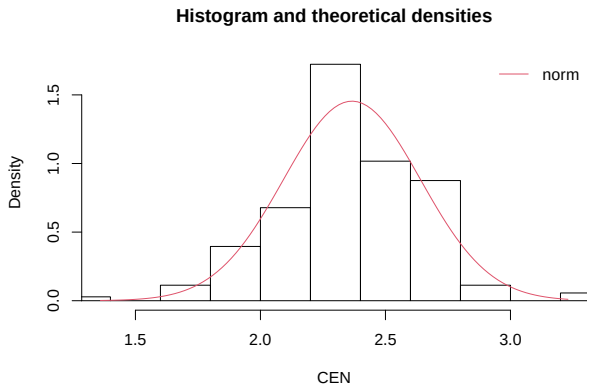

Figura 7: Histograma de CEN y distrbución teórica

A la vista de los resultados, se puede decir que la variable *CEN* sigue una distribución normal (o una no muy alejada de esta).

2.2.2. Test de Grubbs

A continuación, se aplica el test de Grubbs, para el cual se obtiene el siguiente valor de p-value:

```
## [1] 0.03453048
```

El valor obtenido es menor que 0.05, por lo tanto, podemos rechazar con seguridad desde un punto de vista estadístico, que uno los tres outliers (el más alejado de la media) de la variable *CEN* siga una distribución semejante al resto de los datos de dicha variable. Esto es, podemos asegurar que efectivamente es un outlier. A continuación, se muestra el valor que toma este outlier y su posición, donde se ve que este esl el vino con etiqueta ***59*** (como se puede ver también en la Figura 6, para la variable *CEN*).

```
## [1] "valor.posible.outlier"
```

```
## [1] 1.36
```

```
## [1] "clave.posible.outlier"
```

```
## [1] 59
```

2.2.3. Test de Normalidad

Con este test se puede comprobar si la distribución subyacente de datos tras eliminar el oulie anterior sigue una distribución normal o no. Repetimos el análisis anterior para nuestros datos y posteriormente se aplicarán los tests de normalidad utilizados en el enunciado de la práctica.

```
# COMPLETAR

test.de.Grubbs = grubbs.test(columna, two.sided = TRUE)
valor.posible.outlier = outlier(columna)
es.posible.outlier = outlier(columna, logical = TRUE)
clave.posible.outlier = which(es.posible.outlier == TRUE)

test.de.Grubbs$p.value
```

```
## [1] 0.03453048
```

```
valor.posible.outlier
```

```
## [1] 1.36
```

```
es.posible.outlier
```

```
##   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
##  [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [145] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [157] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
## [169] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
clave.posible.outlier
```

```
## [1] 59
```

Pasando los tests de normalidad de Shapiro-Wilks y Anderson-Darling, se obtiene lo siguiente:

```
##   [1] 2.14 2.67 2.50 2.87 2.45 2.45 2.61 2.17 2.27 2.30 2.32 2.41 2.39 2.38 2.70
##  [16] 2.72 2.62 2.48 2.56 2.28 2.65 2.36 2.52 2.61 3.22 2.62 2.14 2.80 2.21 2.70
##  [31] 2.36 2.36 2.70 2.65 2.41 2.84 2.55 2.10 2.51 2.31 2.12 2.59 2.29 2.10 2.44
##  [46] 2.28 2.12 2.40 2.27 2.04 2.60 2.42 2.68 2.25 2.46 2.30 2.68 2.50 2.28 2.02
##  [61] 1.92 2.16 2.53 2.56 1.70 1.92 2.36 1.75 2.21 2.67 2.24 2.60 2.30 1.92 1.71
##  [76] 2.23 1.95 2.40 2.00 2.20 2.51 2.32 2.58 2.24 2.31 2.62 2.46 2.30 2.32 2.42
##  [91] 2.26 2.22 2.28 2.20 2.74 1.98 2.10 2.21 1.70 1.90 2.46 1.88 1.98 2.27 2.12
## [106] 2.28 1.94 2.70 1.82 2.17 2.92 2.50 2.50 2.20 1.99 2.19 1.98 2.00 2.42 3.23
## [121] 2.73 2.13 2.39 2.17 2.29 2.78 2.30 2.38 2.32 2.40 2.40 2.36 2.25 2.20 2.54
## [136] 2.64 2.19 2.61 2.70 2.35 2.72 2.35 2.20 2.15 2.23 2.48 2.38 2.36 2.62 2.48
## [151] 2.75 2.28 2.10 2.32 2.38 2.64 2.70 2.64 2.38 2.54 2.58 2.35 2.30 2.26 2.60
## [166] 2.30 2.69 2.86 2.32 2.28 2.48 2.45 2.48 2.26 2.37 2.74
```

```
##
##  Shapiro-Wilk normality test
##
## data:  columna.sin.outlier
## W = 0.98728, p-value = 0.1131
```

```
##     1-mle-norm
## "not computed"
```

Al igual que ocurre en el enunciado de la práctica, el test Anderson-Darling no se puede aplicar porque hay pocos datos, y el test Shapiro-Wilks da un resultado de $p - value > 0.05$, por lo que no se puede rechazar la hipótesis de que los datos subyacentes sigan una distribución que no sea normal.

Por lo tanto, se puede asegurar, desde un punto de vista estadístico, que el vino ***59*** es efectivamente un outlier. A continuación, se presenta la función solicitada a desarrollar para implementar las operaciones recienmente realizadas:

```
# COMPLETAR

#############################################################################
# Aplica el test de Grubbs sobre la columna ind.col de datos y devuelve una lista con:

# nombre.columna: Nombre de la columna datos[, ind.col]
# clave.mas.alejado.media: Clave del valor O que está más alejado de la media
# valor.mas.alejado.media: Valor de O en datos[, ind.col]
# nombre.mas.alejado.media: Nombre de O en datos
# es.outlier: TRUE/FALSE dependiendo del resultado del test de Grubbs sobre O
# p.value:  p-value calculado por el test de Grubbs
# es.distrib.norm: Resultado de aplicar el test de Normalidad
#    de Shapiro-Wilks sobre datos[, ind.col]
#    El test de normalidad se aplica sin tener en cuenta el
#    valor más alejado de la media (el posible outlier O)
#    TRUE si el test no ha podido rechazar
#       -> Sólo podemos concluir que los datos no contradicen una Normal
#    FALSE si el test rechaza
#       -> Los datos no siguen una Normal
```

```
# Requiere el paquete outliers

test_Grubbs = function(data.frame, indice.columna, alpha = 0.05){

  columna = data.frame[,indice.columna]
  nombre.columna = colnames(data.frame)[indice.columna]

  test.de.Grubbs = grubbs.test(columna, two.sided = TRUE)
  p.value = test.de.Grubbs$p.value

  valor.mas.alejado.media = outlier(columna)
  es.posible.outlier = outlier(columna, logical = TRUE)
  clave.mas.alejado.media = which(es.posible.outlier == TRUE)

  nombre.mas.alejado.media = row.names(data.frame[clave.mas.alejado.media,])

  es.outlier <- ifelse(p.value<alpha,TRUE,FALSE)

  p.value.test.normalidad = shapiro.test(columna[-clave.mas.alejado.media])$p.value

  es.distrib.norm <- ifelse(p.value.test.normalidad>alpha,TRUE,FALSE)


  list(nombre.columna=nombre.columna,
       clave.mas.alejado.media=clave.mas.alejado.media,
       valor.mas.alejado.media = valor.mas.alejado.media,
       nombre.mas.alejado.media = nombre.mas.alejado.media,
       es.outlier = es.outlier,
       p.value = p.value,
       p.value.test.normalidad = p.value.test.normalidad,
       es.distrib.norm = es.distrib.norm
  )
}
```

Comprobamos su funcionamiento con el caso analizado anteriormente:

```
# COMPLETAR

test.Grubbs.datos = test_Grubbs(datos.num, 3)

test.Grubbs.datos
```

```
## $nombre.columna
## [1] "CEN"
##
## $clave.mas.alejado.media
## [1] 59
##
## $valor.mas.alejado.media
## [1] 1.36
##
## $nombre.mas.alejado.media
## [1] "59"
##
## $es.outlier
```

```
## [1] TRUE
##
## $p.value
## [1] 0.03453048
##
## $p.value.test.normalidad
## [1] 0.1131352
##
## $es.distrib.norm
## [1] TRUE
```

Por otro lado, para comprobar los otros dos outliers, es necesario realizar el mismo test que antes, eliminando el que hemos comprobado que es efectivamente outlier. Por lo tanto, hay que quitar el outlier (vino *59*) y comprobar si el ***121*** es efectivamente outlier:

```
# COMPLETAR

test.Grubbs.datos = test_Grubbs(datos.num[-59,], 3)

test.Grubbs.datos
```

```
## $nombre.columna
## [1] "CEN"
##
## $clave.mas.alejado.media
## [1] 120
##
## $valor.mas.alejado.media
## [1] 3.23
##
## $nombre.mas.alejado.media
## [1] "121"
##
## $es.outlier
## [1] FALSE
##
## $p.value
## [1] 0.1810993
##
## $p.value.test.normalidad
## [1] 0.2392921
##
## $es.distrib.norm
## [1] TRUE
```

Como se puede ver, al tener un $p-value > 0.05$ en el Test de Grubbs, no se puede asegurar que este valor sea efectivamente un outlier desde el punto de vista estadístico. Esto provoca que, como consecuencia, el siguiente outlier detectado (***25***) no pueda tampoco ser considerado como tal desde un punto de vista estadístico.

2.3. Trabajando con varias columnas

2.3.1. Outliers IQR

En esta sección se expande el análisis a más de una columna (variable) para determinar los outliers IQR en todas ellas, es decir, en todo el conjunto de datos.

Para ello, se hace uso de la función `claves outliers IQR en alguna columna(...)` siguiendo el código de la práctica:

```
## [1] "Los registros que son outliers en alguna columna son:"
##  [1] 123 137 173  25  59 121  69  73  78  95 151 158 159 115
```

Ahora, se obtienen muchos más registros (vinos) que son outliers IQR con respecto a alguna de las variables, hasta un total de 14. El detalle de estos outliers IQR es el siguiente:

```
## [1] "claves.outliers.IQR.en.mas.de.una.columna"
## integer(0)
## [1] "claves.outliers.IQR.en.alguna.columna"
##  [1] 123 137 173  25  59 121  69  73  78  95 151 158 159 115
## [1] "nombres_filas(datos.num, claves.outliers.IQR.en.mas.de.una.columna)"
## character(0)
## [1] "nombres_filas(datos.num, claves.outliers.IQR.en.alguna.columna)"
##  [1] "123" "137" "173" "25"  "59"  "121" "69"  "73"  "78"  "95"  "151" "158"
## [13] "159" "115"
```

En este caso, por la naturaleza del dataset, no se tiene un nombre propio para cada uno de los vinos observados, por lo que su etiqueta coincide con el nombre de estos.

Por otro lado, el valor normalizado de todos estos ouliers IQR es el siguiente:

```
# COMPLETAR

datos.num.norm[claves.outliers.IQR.en.alguna.columna, ]
```

```
##              ALC       ACMA        CEN        MAG        FET        FLA
## 123  0.069642799  3.0912789 -0.8585059 -0.9586252  0.5231578  0.6273959
## 137 -0.573278285  2.8321917  0.9954972 -0.2531089 -0.8017364 -1.4253597
## 173  0.885658020  2.9572683  0.3047901 -0.3236606 -0.9773248 -1.4153463
## 25   0.069642799 -0.2589862  3.1039714  1.7223365  0.5391204  0.6574362
## 59  -0.771100156 -1.2506647 -3.6576872 -0.8175219 -0.4984474 -1.4554000
## 121 -1.772573382 -0.2589862  3.1403244  1.3695784  1.4170624  3.0606623
## 69  -0.968922028 -1.0273137 -2.2399201  3.6272303 -0.7059610 -0.7444457
## 73  -0.004540403 -0.5984797  0.8500852  2.7806108  1.6086134  0.8677185
## 78  -0.820555624 -1.2059945 -1.5128600  2.5689560 -0.6261480 -0.1736795
## 95  -0.647461486 -0.7324903 -0.6040349  4.4032982  0.3316069  0.2468851
## 151 -0.251817743  0.2949243  0.4138492  0.8757170 -1.2965765 -0.6643381
## 158  1.664581640 -0.5895457  1.2136152 -0.1120057  0.8104843 -0.7144053
## 159  0.601289079 -0.5984797  0.9954972 -0.7469703  0.4912327 -0.9246876
## 115 -2.427858333 -0.7414244 -0.6040349 -1.0291768  0.2677565  0.1467507
##             FNF          IC        MAT        PROL
## 123 -0.49991911 -1.05608038 -0.9906082 -1.15946187
## 137  2.14743134 -0.02357648 -0.5978267 -0.73073262
## 173  1.26498119  1.13799041 -1.3833897 -0.01618388
## 25   0.86386748 -0.63447462  0.7550872  0.26963562
```

```
## 59  -0.66036460 -1.33571685  0.4059482 -0.71485376
## 121  0.86386748  0.40663348 -0.1177605 -0.88952123
## 69  -1.78348297 -0.94852789  1.4097230 -0.08605087
## 73  -1.22192378 -0.73342291  1.5406502  0.76188031
## 78  -0.09880541 -0.71191241  0.4495905  0.01557384
## 95  -0.33947363 -1.05608038  0.8860144  0.60944325
## 151 -0.98125556  2.47164128 -2.0816678 -0.84188465
## 158  1.34520393  3.41810319 -1.6888864 -0.27024565
## 159  1.26498119  2.88034074 -1.6888864 -0.39727654
## 115  1.26498119 -1.35722735  3.2863456 -1.07371602
```

Y representándolos de manera gráfica mediante cajas se pueden ver estos outliers. Para mayor comprensión de la imagen, se han añadido una serie de líneas indicando el umbral a partir del cual un valor normalizado se considera outlier (azul) y outlier extremo (rojo).

```
# COMPLETAR

diag_caja_juntos(datos.num.norm, "Outliers en alguna columna",
                 claves.outliers.IQR.en.alguna.columna) +
  geom_hline(yintercept=2.68, linetype="dashed", color="blue") +
  geom_hline(yintercept=-2.68, linetype="dashed", color="blue") +
  geom_hline(yintercept=4.69, linetype="dashed", color="red") +
  geom_hline(yintercept=-4.69, linetype="dashed", color="red")
```

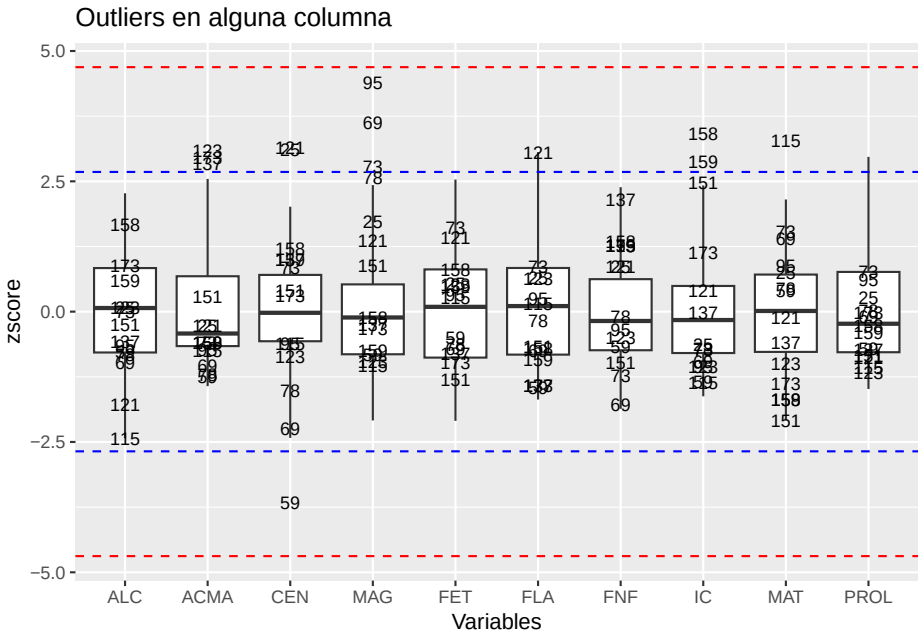

Figura 8: Boxlpots de todas las variables con datos normalizados

Se puede ver, tal y como se había calculado anteriormente, que no hay ningún outlier IQR extremo, pero sí se tienen varios outliers IQR no extremos. Por ejemplo, cabe destacar que el vino con etiqueta ***121*** es el único que es considerado outlier en más de una variable, en concreto ***CEN*** y ***FLA*** como se había analizado anteriormente. Esto nos indica que este vino aporta significativamente un mayor número minerales (*CEN*) y que tendrá mayor sabor, aroma e intensidad de color (*FLA*) que el resto de vinos.

En este último aspecto relativo a saber y aroma, probablemente esté en un nivel semejante a los vinos con etiquetas ***95*** y ***69***, ya que estos presentan valores muy elevados de magnesio (***MAG***), que también aporta sabor y aroma a los vinos.

2.3.2. Tests de hipótesis (OPCIONAL)

En esta ocasión, se va a aplicar el test de Grubbs sobre todas las variables. Para ello, como antes, vemos si todas ellas siguen una distribución normal:

```
# COMPLETAR

par(mfrow=c(2,3))
histogramas <- sapply(c(1:ncol(datos.num)),
  function(x) denscomp(fitdist(datos.num[, x] , "norm"), main="", xlab=names(datos.num)[x]))
```

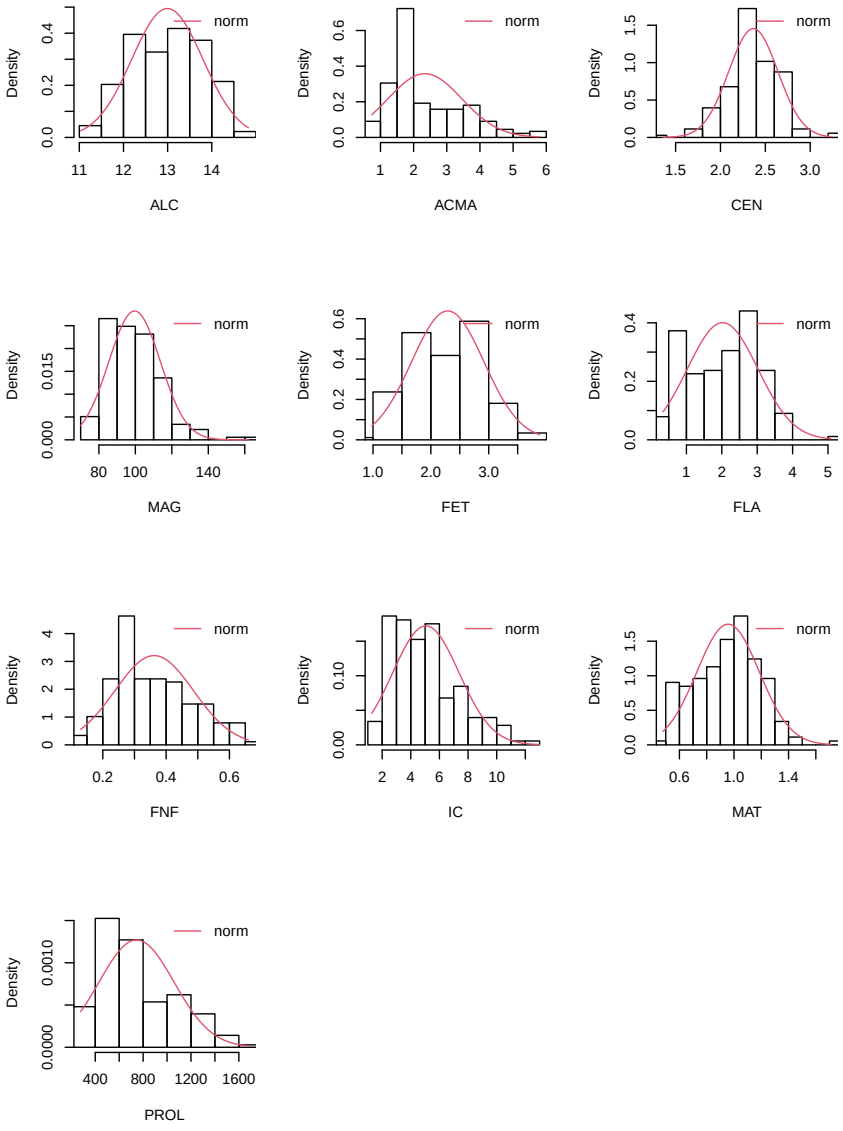

Figura 9: Histogramas de cada una de las variables junto a su teórica distribución

Aquí vemos que, de manera informal, se podría considerar que solo la variable *CEN* (y quizás también las *MAT*) siguen una distribución normal (no se puede considerar que no lo sean). Por ello, a continuación se aplica el test de Grubbs para todos ellos:

```
# COMPLETAR

sapply(c(1:ncol(datos.num)),
  function(x) test_Grubbs(datos.num, x))
```

```
##                           [,1]         [,2]          [,3]        [,4]
## nombre.columna            "ALC"        "ACMA"        "CEN"       "MAG"
## clave.mas.alejado.media   115          123           59          95
## valor.mas.alejado.media   11.03        5.8           1.36        162
## nombre.mas.alejado.media  "115"        "123"         "59"        "95"
## es.outlier                TRUE         FALSE         TRUE        TRUE
## p.value                   0            0.3091236     0.03453048  0.001060476
## p.value.test.normalidad   0.007794875  6.250607e-10  0.1131352   3.477779e-05
## es.distrib.norm           FALSE        FALSE         TRUE        FALSE
##                           [,5]         [,6]          [,7]         [,8]
## nombre.columna            "FET"        "FLA"         "FNF"        "IC"
## clave.mas.alejado.media   52           121           105          158
## valor.mas.alejado.media   3.88         5.08          0.66         13
## nombre.mas.alejado.media  "52"         "121"         "105"        "158"
## es.outlier                FALSE        FALSE         TRUE         FALSE
## p.value                   0.1177285    0.3443948     0            0.09132837
## p.value.test.normalidad   0.003001588  4.20182e-06   0.0001240141 2.156077e-06
## es.distrib.norm           FALSE        FALSE         FALSE        FALSE
##                           [,9]         [,10]
## nombre.columna            "MAT"        "PROL"
## clave.mas.alejado.media   115          18
## valor.mas.alejado.media   1.71         1680
## nombre.mas.alejado.media  "115"        "18"
## es.outlier                FALSE        FALSE
## p.value                   0.1515119    0.4729247
## p.value.test.normalidad   0.006135058  1.710521e-07
## es.distrib.norm           FALSE        FALSE
```

En primer lugar, se ve que solamente la variable *CEN* podría seguir una distribución normal, ya que el test de Shapiro-Wilks rechaza la hipótesis de que la distribución sea normal para el resto de variables. Por otro lado, únicamente el outlier IQR dado en el vino ***59*** puede considerarse realmente un outlier desde el punto de vista estadístico.

3. Outliers multivariantes

3.1 Métodos estadísticos basados en la distancia de Mahalanobis (OPCIONAL)

En esta sección se van a realizarutiliza tests estadísticos para asegurar que un outlier multivariante lo es desde un punto de vista estadístico.

3.1.1 Hipótesis de Normalidad

En primer lugar se analiza la hipótesis de normalidad basada en la distancia de Mahalanobis. Para ello, primero se utiliza la función realizada anteriormente para el test de Grubbs, con el objetivo de obtener las variables que siguen una distribución normal, ya que la primera premisa es que estas sigan una distribución 1-variante:

```
## [1] "son.col.normales"
```

```
##  [1] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
## [1] "datos.num.distrib.norm"
```

```
## [1] 2.14 2.67 2.50 2.87 2.45 2.45
```

Por lo tanto, no tiene sentido realizar el test de normalidad multivariante, ya que solamente disponemos de una única variable.

3.1.2 Tests de hipótesis para detectar outliers

Siguiendo el enunciado de la práctica, se va a realizar el test de hipótesis para detectar los outliers sobre las variables *FET* y *FLA* de nuestros datos (aunque ya sabemos que no siguen una distribución normal, lo que provoca que no podremos asegurar desde un punto de vista estadístico la detección de los outliers):

```
## $cor.cla
## [1] 0.8640455
##
## $cor.rob
## [1] 0.9160794
```

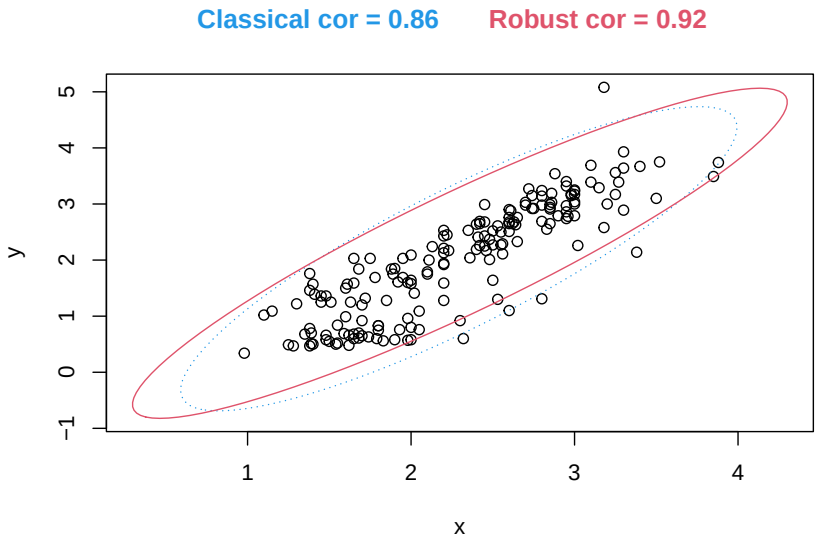

Figura 10: Variables FET y FLA junto a las elipses de distancia de Mahalanobis (azul) y robusta (roja).

La gráfica nos indica que las dos variables están altamente correlacionadas, de hecho, la mayoría de los puntos han sido considerados en la estimación robusta de la matriz de covarianzas (elipse en rojo).

Tras este análisis, el *test individual* y el *test de intersección* no van a poder ejecutarse ya que solamente tenemos una única variable que es normal. Sin embargo, a continuación se deja el código comentado:

```
# COMPLETAR

# set.seed(2)
#
# alpha.individual = 0.05
# test.individual = cerioli2010.fsrmcd.test(datos.num.distrib.norm,
#                                           signif.alpha=alpha.individual)
# son.outliers.test.individual = test.individual$outliers
# claves.test.individual = which(son.outliers.test.individual  == TRUE)
# nombres.test.individual = row.names(datos.num.distrib.norm[claves.test.individual,])
#
# alpha.interseccion = 1 - (1-0.05)^(1/nrow(datos.num.distrib.norm))
# test.interseccion = cerioli2010.fsrmcd.test(datos.num.distrib.norm,
#                                             signif.alpha=alpha.interseccion)
# son.outliers.test.interseccion = test.interseccion$outliers
# claves.test.interseccion = which(son.outliers.test.interseccion  == TRUE)
# nombres.test.interseccion = row.names(datos.num.distrib.norm[claves.test.interseccion,])
#
# claves.test.individual
# nombres.test.individual
# claves.test.interseccion
# nombres.test.interseccion
```

Por otro lado, si quisieramos mostrar las distancias de Mahalabonis ordenadas, por ejemplo, para el test individual, se tendría el siguiente código:

```
# COMPLETAR

# dist.individual.ordenados = sort(test.individual$mahdist.rw,
#                                  decreasing=TRUE,
#                                  index.return = TRUE)
# plot(dist.individual.ordenados$x)
```

3.2 Visualización de datos con un Biplot

A continuación, se presenta, al igual que en el tutorial de la práctica, un `biplot(...)` con los datos, que nos va a permitir entender qué variables (características del vino) explican mejor cada uno de estos:

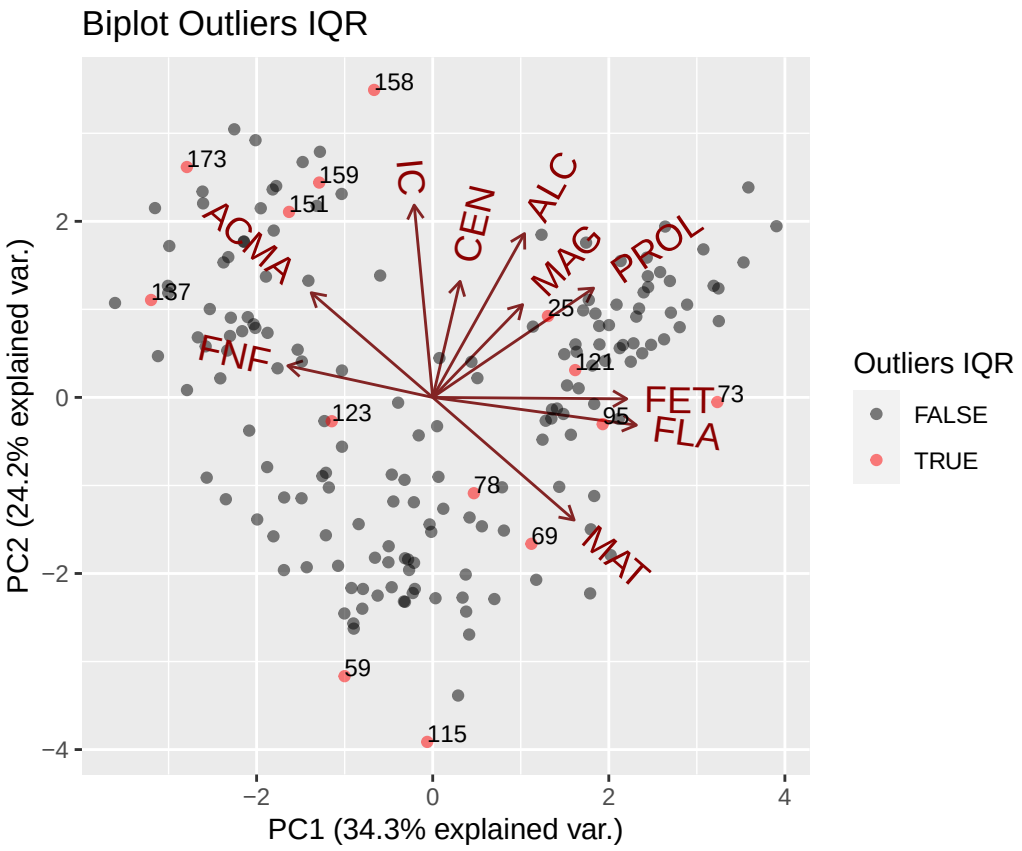

Figura 11: Biplot de las dos primeras ocmponentes principales con los outliers IQR

En primer lugar, podemos ver que este `biplot(...)`, con las dos primeras componentes principales, no va a explicar de manera suficientemente precisa nuestros datos, ya que solamente abarca el 58.5% de la varianza (explicación) de estos.

Por ejemplo, en el `biplot(...)` se puede ver que el vino de etiqueta ***121*** sí parece tomar una valor suficientemente elevado en la variable ***CEN*** como para considerarlo un outlier IQR. Sin embargo, este vino no parece, a simple vista, un outlier en la variable ***FLA***, cuando antes sí hemos comprobado que lo era con el método IQR. Esto nos indica, efectivamente, que la pérdida de información al utilizar solamente las dos primeras componentes principales es significativa, y se obtienen resultados diferentes a los obtenidos con todas las variables originales.

3.3 Métodos basados en distancias: LOF

En la siguiente sección se va a utilizar un método sin garantía estadística basado en distancias (euclídea en este caso) entre los distintos puntos de datos para obtener los outliers. Esto se realiza, tal y como se indica en la práctica, con la función `LOF` y considerando un valor de cinco vecinos.

Una vez calculados los *LOF scores*, los mostramos de manera ordenada:

```
# COMPLETAR

lof.scores.ordenados=sort(lof.scores, decreasing=TRUE, index.return = TRUE)
plot(lof.scores.ordenados$x)
```

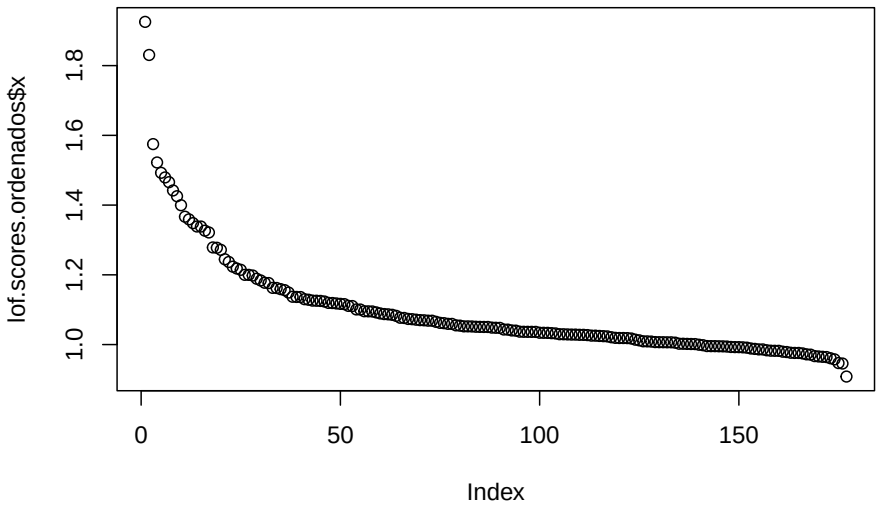

Figura 12: LOF socres ordenados

En la anterior gráfica se ve que hay dos valores con un *score* significativamente mayor que el resto, por lo tanto, se utilizará $num.outliers = 2$.

```
# COMPLETAR

num.outliers = 2
claves.lof.ordenados = lof.scores.ordenados$ix
claves.outliers.lof = claves.lof.ordenados[1:num.outliers]
nombres.outliers.lof = nombres_filas(datos.num.norm,claves.outliers.lof)
```

```
claves.outliers.lof
```

```
## [1] 121  71
```

```
nombres.outliers.lof
```

```
## [1] "121" "71"
```

Este resultado nos indica que tenemos dos outliers de tipo *LOF*, como son ***121*** (lo veíamos también con el método IQR) y ***71***. Los valores normalizados de estos son:

```
##           ALC       ACMA      CEN        MAG      FET       FLA        FNF
## 121 -1.772573 -0.2589862 3.140324  1.3695784 1.417062 3.0606623  0.8638675
## 71   1.071116 -0.7414244 1.104556 -0.9586252 1.049923 0.8376782 -1.2219238
##             IC        MAT       PROL
## 121  0.4066335 -0.1177605 -0.8895212
## 71  -0.7205166  1.7588621 -1.0641887
```

Seguramente, el caso de que el vino ***121*** haya obtenido un *score* alto sea por su valor alejado en las variables *CEN* y *FLA*. Sin embargo, más llamativo es el caso del vino ***71***, que no presenta aparentemente ningún valor alejado en ninguna de las variables.

A continuación se va a representar la posición de cada uno de estos dos vinos en la representación de las variables confrontadas entre sí dos a dos:

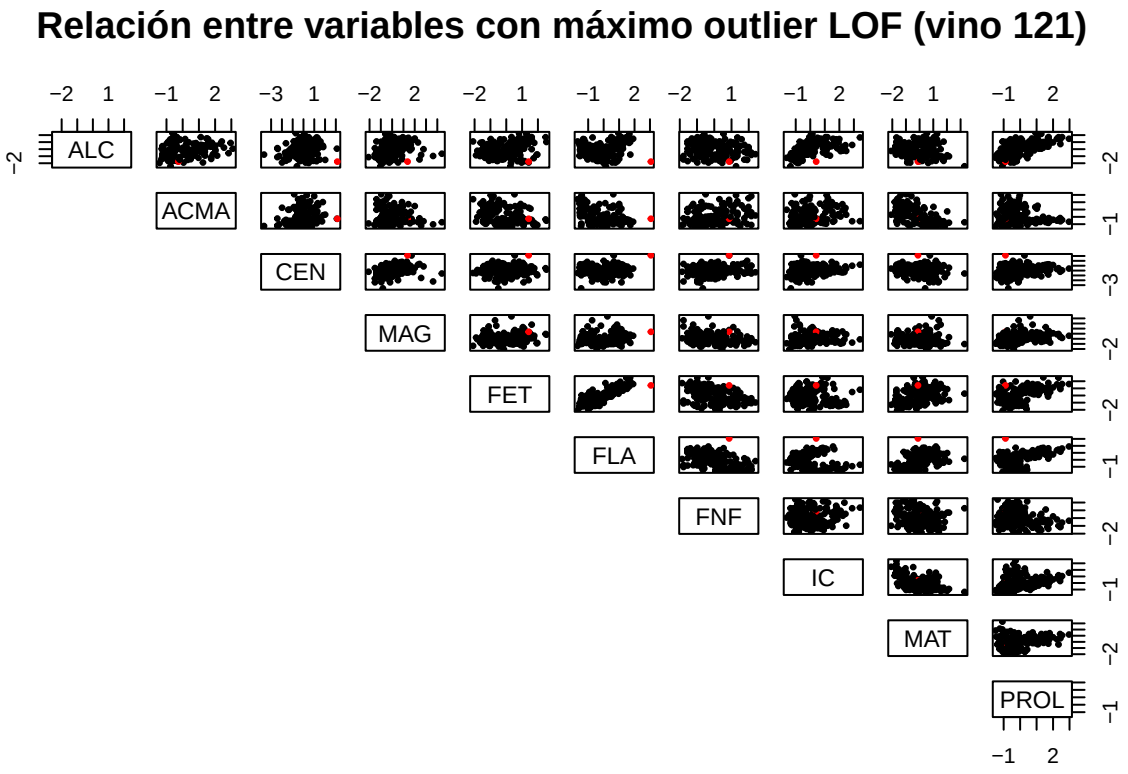

Figura 13: Relación entre variables marcando la posición del vino 121

En esta representación, se puede ver que el vino con etiqueta ***121*** está considerablemente alejado de la nube

de puntos en *CEN* y *FLA* contra cualquier otra variable (reforzando de nuevo la exlpicación de que es un outlier), pero también en otras relaciones como *ALC vs. MAG* o *ALVC vs. IC*.

Además, también se representa esta relación, pero mostrando el segundo máximo *score LOF*, es decir, el vino ***71***, que aparentemente no presenta valores extremos en ninguna variable:

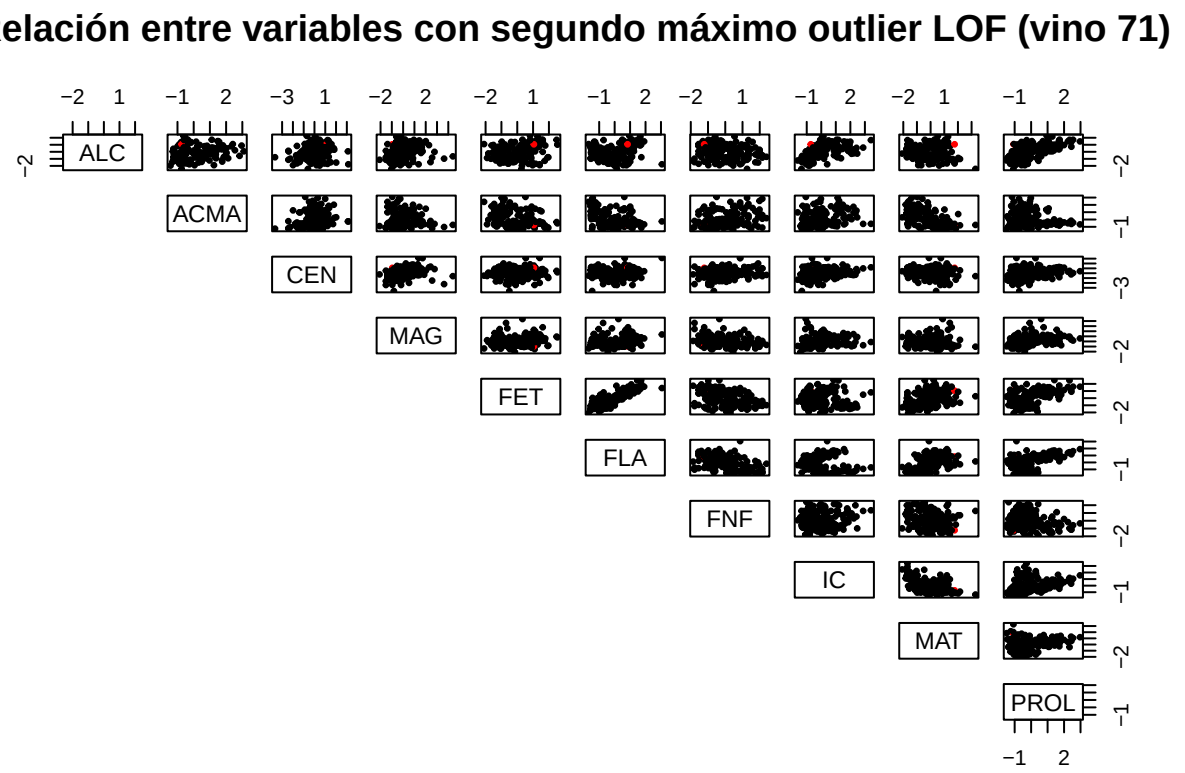

Figura 14: Relación entre variables marcando la posición del vino 71

Revisando todas las gráficas, se puede ver que este punto se encuentra, generalmente, en el "borde" de la nube de puntos, lo que podría hacer que la distancia a sus cinco vecions más cercanos sea mayor que la del resto de puntos de la nube entre sí, lo que lo dejaría en una zona de baja densidad frente a una de alta. La variable en la que más se clarifica este hecho es *MAT*, donde se puede ver el valor de ***71*** generalmente en zonas de baja densidad con respecto a la gran nube de puntos.

A continuación, representamos ambos valores con un `biplot(...)` considerando todas las variables (con las dos primeras componentes principales, que ya se ha comentado anteriormente que no suponen una representación fiable de los datos al no abarcar la suficiente varianza de los datos).

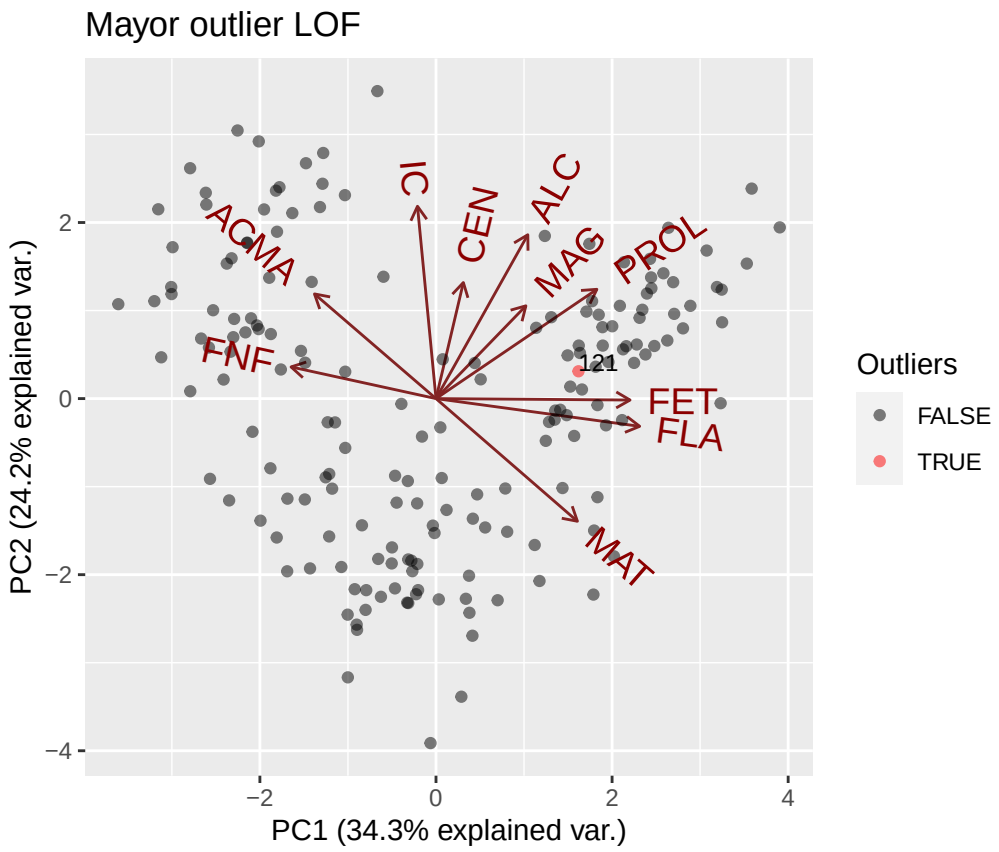

Figura 15: Biplot de los datos normalizados con el outlier de mayor LOF socre

Al no representar fiablemente la relación de las variables y los datos, no da la impresión de que el vino ***121*** esté en una zona de baja densidad de datos junto a otra que parezca mucho mayor. Por otro lado, sí se ve que en el `biplot(...)` del vino ***71***, este se presenta en una zona de baja densidad otras dos zonas de puntos que parecen de mayor densidad (aunque también puede ser que contenga una cominación de dos variables algo inusual, lo que lo convierta en outlier):

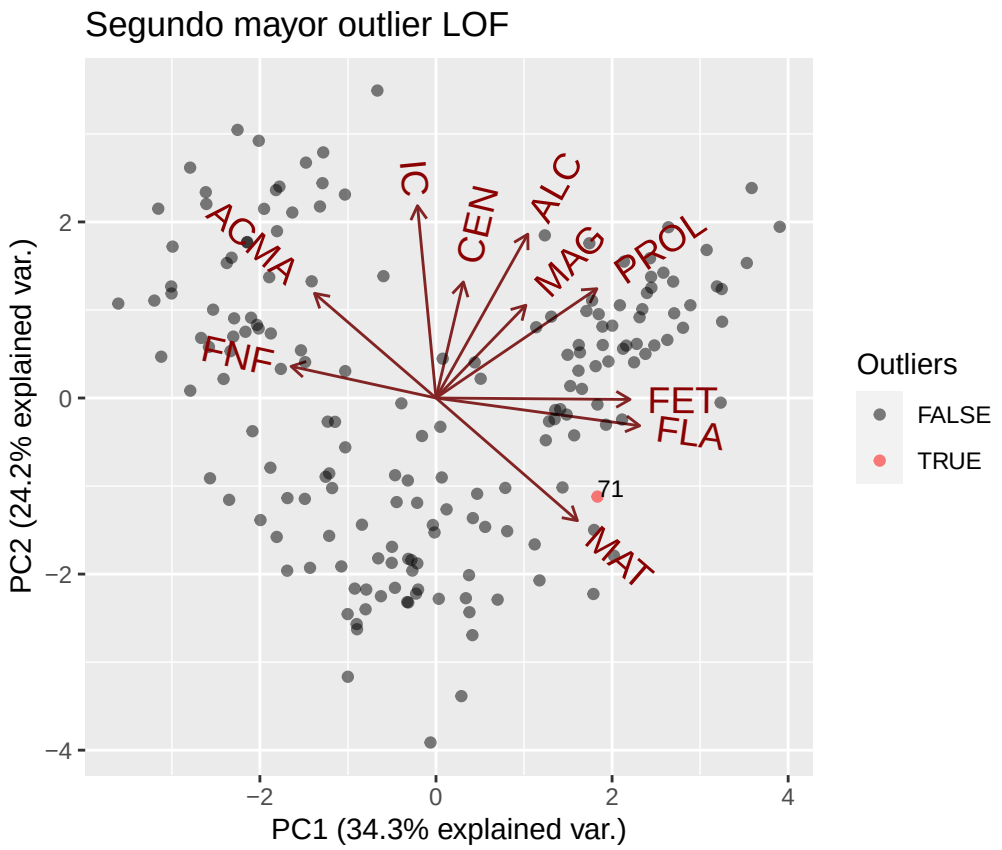

Figura 16: Biplot de los datos normalizados con el segundo outlier con mayor LOF score

3.4 Métodos basados en Clustering

3.4.1 Clustering usando k-means

En este apartado se va a utilizar el algoritmo *k-means*, también basado en distancias, para clusterizar los datos y encontrar los outliers. Para ello, al igual que en la práctica, se utilizarán tres clusters (además, ya que sabemos que hay tres clases de vinos en nuestro dataset) y se considerarán cinco outliers.

```
# COMPLETAR

modelo.kmeans = kmeans(datos.num.norm,num.clusters)
asignaciones.clustering.kmeans=modelo.kmeans$cluster
centroides.normalizados=modelo.kmeans$centers
```

```
# COMPLETAR

head(asignaciones.clustering.kmeans)
```

```
## 1 2 3 4 5 6
## 2 2 2 2 2 2
```

```
centroides.normalizados
```

```
##          ALC       ACMA        CEN         MAG        FET         FLA
## 1 -0.9200202 -0.3841961 -0.5345844 -0.45107468 -0.1255901 -0.01899421
## 2  0.8371996 -0.3084116  0.3780825  0.54457958  0.8902972  0.97786635
## 3  0.2028119  0.9346388  0.2578342 -0.07379022 -0.9746644 -1.23656461
##           FNF          IC        MAT       PROL
## 1 -0.00780648 -0.8703838  0.3933114 -0.7450473
## 2 -0.57237836  0.1558627  0.4960486  1.0942607
## 3  0.75021860  1.0135880 -1.1897266 -0.3734583
```

La versión desnormalizada de estos centroides sería la siguiente:

```
##        ALC     ACMA      CEN       MAG      FET       FLA       FNF       IC
## 1 12.24955 1.909851 2.219104  93.19403 2.213582 2.0044776 0.3613433 3.031642
## 2 13.67081 1.994677 2.470161 107.30645 2.850000 3.0000000 0.2909677 5.417097
## 3 13.15771 3.386042 2.437083  98.54167 1.681667 0.7885417 0.4558333 7.410833
##        MAT      PROL
## 1 1.047104  510.4925
## 2 1.070645 1089.6613
## 3 0.684375  627.5000
```

Para comprobar qué puntos son outliers, debemos calcular su distancia a estos centroides, en función de a qué cluster pertenece el punto:

```
#COMPLETAR

################################################################################
# Calcula las distancias de los datos a los centroides
# y se queda con los primeros (tantos como indica num.outliers)
# Devuelve una lista con las claves de dichos registros y las
# correspondientes distancias a sus centroides

top_clustering_outliers = function(datos.norm,
                                   asignaciones.clustering,
                                   datos.centroides.norm,
                                   num.outliers){
```

```
  distancias.centroides = distancias_a_centroides (datos.norm,
                                                   asignaciones.clustering,
                                                   datos.centroides.norm)

  dist.centroides.ordenadas = sort(distancias.centroides, decreasing=T, index.return=TRUE)

  list(distancias = distancias.centroides[dist.centroides.ordenadas$ix[1:num.outliers]],
       claves = dist.centroides.ordenadas$ix[1:num.outliers])
}
```

```
## [1] "claves.outliers.kmeans:"
## [1] 121  69  95  96 123
## [1] "nombres.outliers.kmeans:"
## [1] "121" "69"  "95"  "96"  "123"
## [1] "distancias.outliers.centroides:"
##      121       69       95       96      123
## 5.116378 5.044437 4.582026 4.208680 4.077158
```

Como era de esperar (ya que así se ha definido previamente), obtenemos cinco outliers, aquellos puntos más alejados del centroide de su cluster. Si los representamos con un biplot, obtenemos lo siguiente:

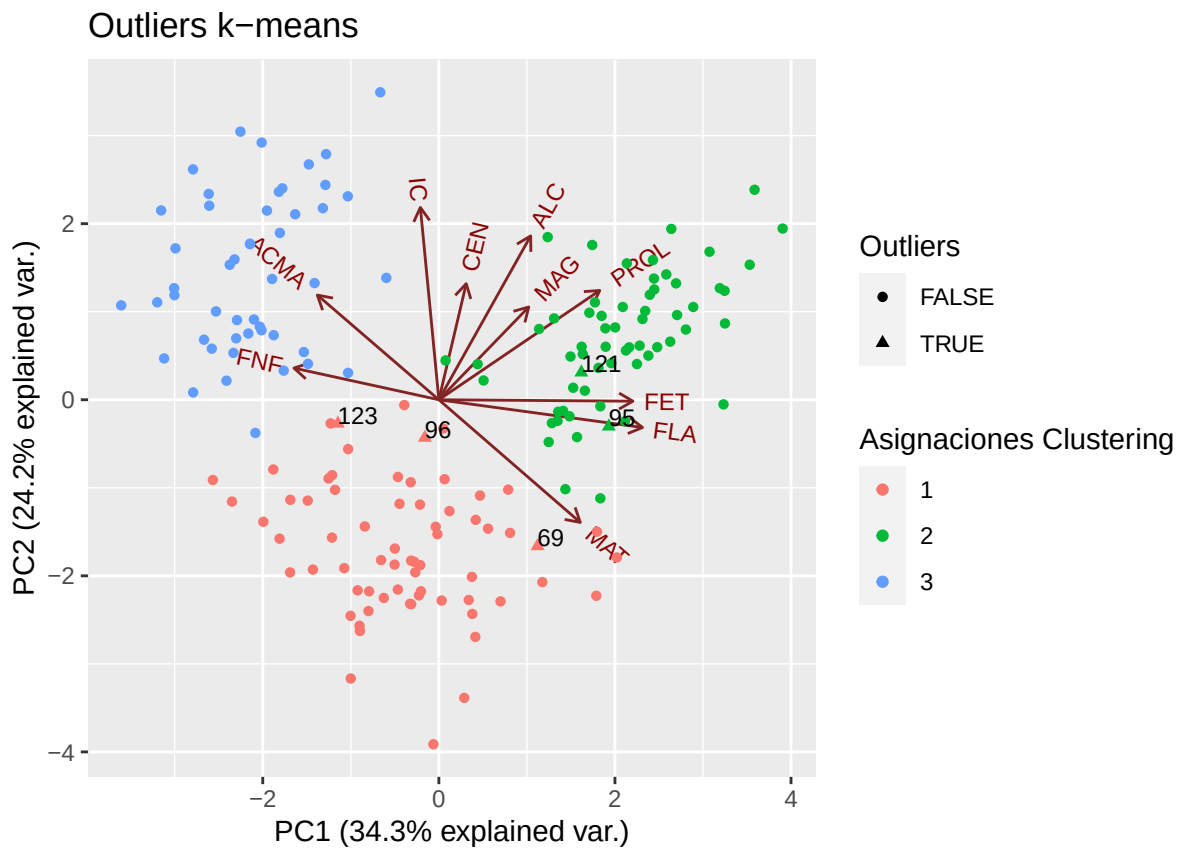

Figura 17: Outliers obtenidos con el método k-means

Si revisamos los outliers, encontramos que tres de ellos (vinos ***123***, ***96*** y ***69***) están en las zonas más exteriores

de su cluster (el número 1). Por otro lado, el cluster 2 no cuenta con ningún outlier, mientras que el 3 tiene dos outliers (vinos ***121***, que ha sido considerado outlier en todos los métodos vistos en la práctica, y el ***95***). Estos dos últimos outliers no parecen tan exteriores a su cluster asociado, pero esto puede ser efecto, de nuevo, de que las dos primeras componentes principales no son suficentemente explicativas para todo el conjunto de datos y variables orginal.

Para obtener más información, representamos sus *boxplots* siguiendo la práctica:

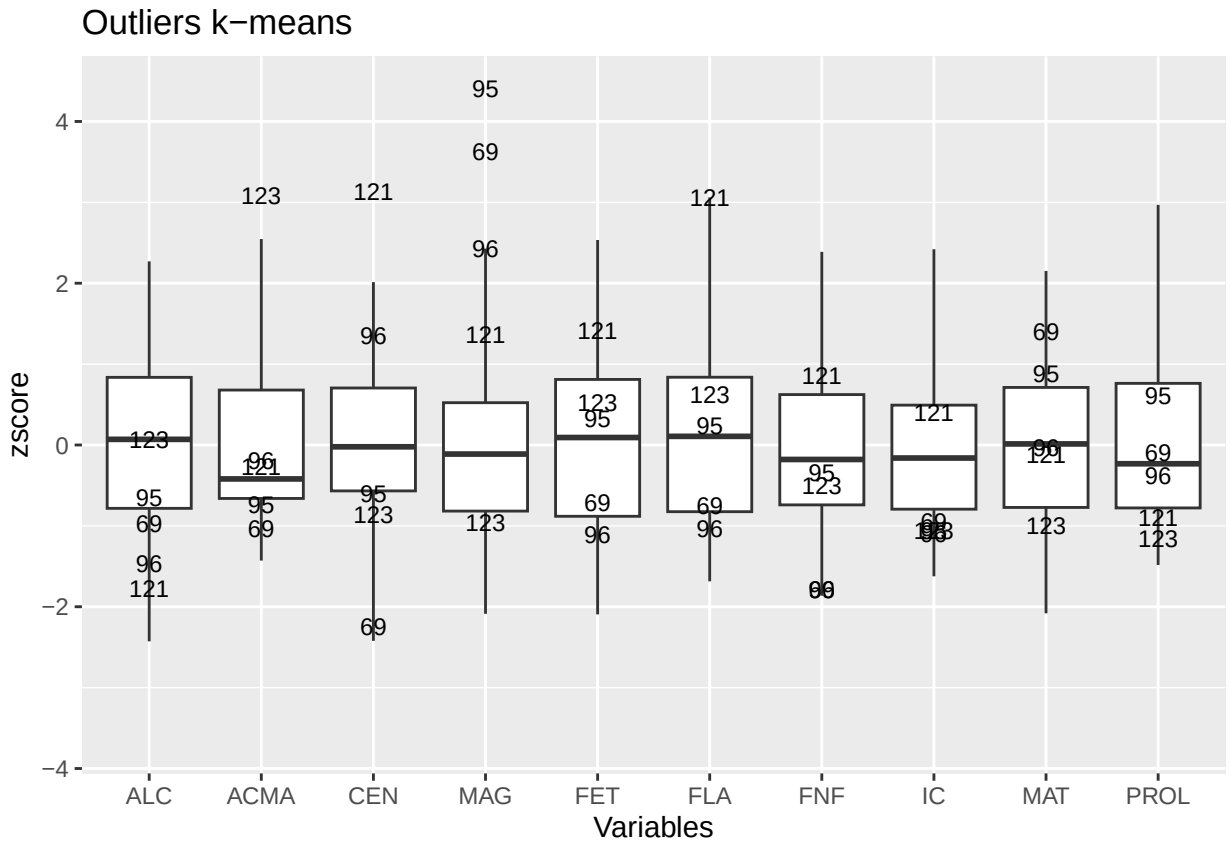

Figura 18: Boxplots de las variables con las etiquetas de los outliers k-means

Aquí se ve por ejemplo, que, además del ya conocido outlier ***121*** en las variables *CEN* y *FLA*, el vino ***95*** toma un valor muy diferente del resto para la variable *MAG*, por lo que este puede ser el motivo por el que, a pesar de no estar muy alejado de la nube de puntos de su cluster, sea considerado un outlier.

3.4.2 Clustering usando medoides (OPCIONAL)

Finalmente, el último método de clustering utilziada será el basado en medoides. Para ello, se utiliza el cñodigo del enunciado de la práctica , en el que se calculan los clusters y se muestra la información de los medoides:

```
set.seed(2)
matriz.distancias = dist(datos.num.norm)
modelo.pam        = pam(matriz.distancias , k = num.clusters)
```

```
asignaciones.clustering.pam = modelo.pam$clustering
nombres.medoides = modelo.pam$medoids
medoides = datos.num[nombres.medoides, ]
```

```
medoides.normalizados = datos.num.norm[nombres.medoides, ]

nombres.medoides
```

```
## [1] "35"  "128" "174"
```

```
medoides
```

```
##       ALC ACMA  CEN MAG  FET  FLA  FNF   IC  MAT PROL
## 35  13.48 1.81 2.41 100 2.70 2.98 0.26 5.10 1.04  920
## 128 12.37 1.63 2.30  88 2.22 2.45 0.40 2.12 0.89  342
## 174 13.40 3.91 2.48 102 1.80 0.75 0.43 7.30 0.70  750
```

```
medoides.normalizados
```

```
##            ALC       ACMA        CEN         MAG        FET        FLA
## 35   0.6012891 -0.4734032  0.1593781  0.02909756  0.6508585  0.9578395
## 128 -0.7711002 -0.6342159 -0.2405049 -0.81752191 -0.1153454  0.4271270
## 174  0.5023781  1.4027453  0.4138492  0.17020081 -0.7857739 -1.2751581
##            FNF          IC        MAT        PROL
## 35  -0.8208101  0.01944452  0.3623058  0.55545512
## 128  0.3023083 -1.26258116 -0.2923300 -1.28014122
## 174  0.5429765  0.96590643 -1.1215353  0.01557384
```

A continuación, se calculan los top outliers, haciendo uso de las funciones disponibles, y mostramos sy `biplot(...)`:

```
# COMPLETAR

top.outliers.pam = top_clustering_outliers(datos.num.norm ,
                                           asignaciones.clustering.pam,
                                           medoides.normalizados,
                                           num.outliers)
claves.outliers.pam = top.outliers.pam$claves
nombres.outliers.pam = nombres_filas(datos.num, claves.outliers.pam)
distancias.outliers.medoides = top.outliers.pam$distancias

claves.outliers.pam
```

```
## [1]  69 121  95  96  59
```

```
nombres.outliers.pam
```

```
## [1] "69"  "121" "95"  "96"  "59"
```

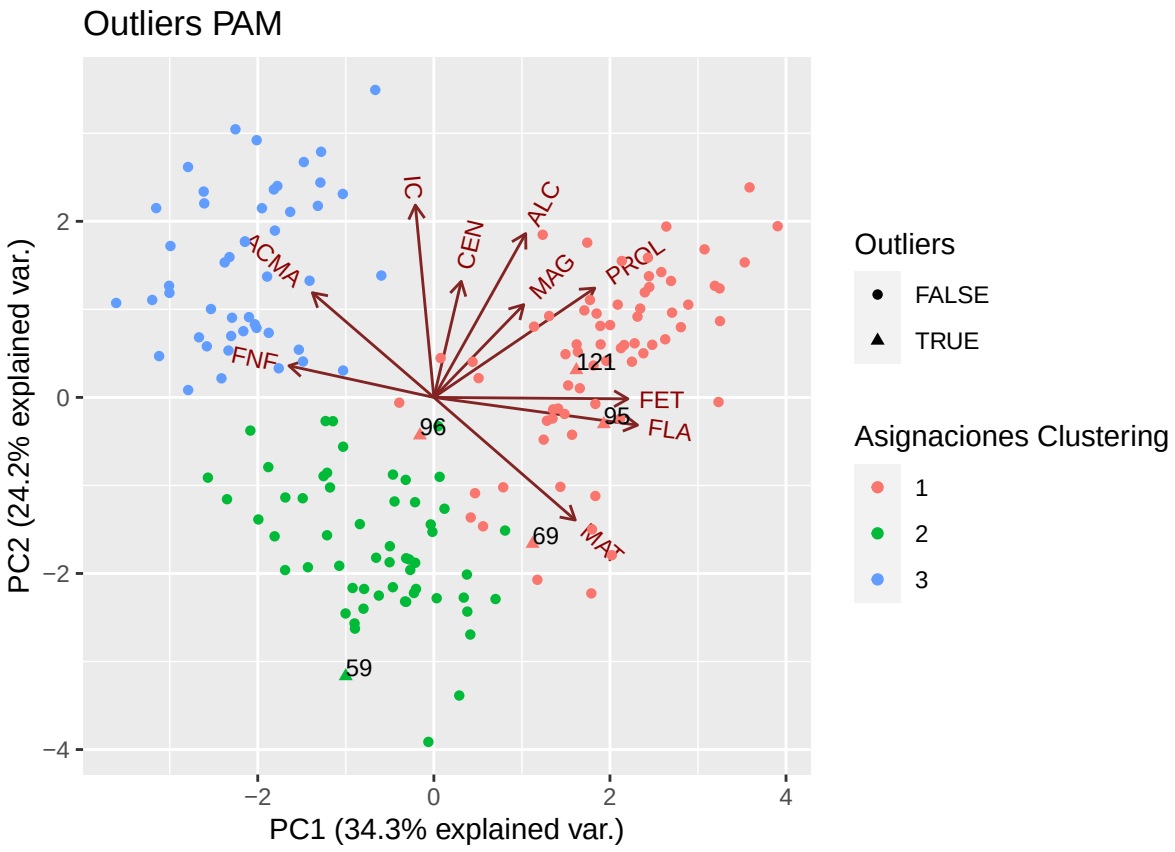

Figura 19: Outliers obtenidos con el método PAM

Estos resultados nos indican que los outliers obtenidos cuando se usa la partición creada por PAM, vienen dados, en su mayoría, por encontrarse en la periferia del cluster al que se han asociado. La única excepción sería el vino ***121***, que está en la zona central de su cluster, por lo que su característica de outlier vendrá dada porque tiene combinaciones inusuales de valores en distintas variables.

3.5 Análisis de los outliers multivariantes puros

A continuación, se presentan aquellos outliers multivariantes puros, es decir, que son variantes con respecto a más de una variable a la vez. Para ello, seguimos las indicaciones de la práctica y se obtiene lo siguiente:

```
# COMPLETAR

claves.outliers.lof.no.IQR=setdiff(claves.outliers.lof, claves.outliers.IQR.en.alguna.columna)
nombres.outliers.lof.no.IQR=nombres_filas(datos.num.norm,claves.outliers.lof.no.IQR)
```

```
## [1] "claves.outliers.IQR.en.alguna.columna"
##  [1] 123 137 173  25  59 121  69  73  78  95 151 158 159 115
## [1] "claves.outliers.lof"
## [1] 121  71
## [1] "claves.outliers.lof.no.IQR"
## [1] 71
## [1] "nombres.outliers.lof.no.IQR"
## [1] "71"
```

Tenemos que el único outlier multivariante puro es el vino ***71***. Esto puede darse, no porque tenga valores extremos (que ya se ha visto que no), sino porque se presentaba ciertamente aislado en la nube de puntos (con respecto al resto de distancias entre los valores), lo que nos indica que muestra una combinación inusual de valores en distintas variables.

También es de interés considerar un número mayor de outliers, como se indica en la práctica. Teniendo en cuenta el resultado obtenido de *scores LOF* con el conjunto de datos *wine* en esta práctica, en lugar de utilizar 11 outliers, se van a utilizar 16, que se corresponde con el segundo nivel distinguido de *scores* obtenido con el método *LOF* visto previamente. Por lo tanto, si ampliamos la selección de outliers se obtiene lo siguiente:

```
# COMPLETAR

claves.outliers.lof.no.IQR=setdiff(claves.lof.ordenados[1:16], claves.outliers.IQR.en.alguna.columna)
nombres.outliers.lof.no.IQR=nombres_filas(datos.num.norm,claves.outliers.lof.no.IQR)
```

```
## [1] "claves.outliers.IQR.en.alguna.columna"
##  [1] 123 137 173  25  59 121  69  73  78  95 151 158 159 115
## [1] "claves.outliers.lof"
## [1] 121  71
## [1] "claves.outliers.lof.no.IQR"
## [1]  71 110  96  13  41 112  46
## [1] "nombres.outliers.lof.no.IQR"
## [1] "71"  "110" "96"  "13"  "41"  "112" "46"
```

El número outliers multivariantes puros ha ascendido hasta siete, entre los que, obviamente, se incluye el vino ***71*** obtenido anteriormente. Si representamos estos outliers en un biplot, se obtiene:

```
#COMPLETAR

biplot.outliers.lof = biplot_2_colores(datos.num.norm,
                                       claves.outliers.lof.no.IQR,
                                       titulo.grupo.a.mostrar = "Outliers puros",
```

```
                                titulo ="Biplot Outliers puros")
biplot.outliers.lof
```

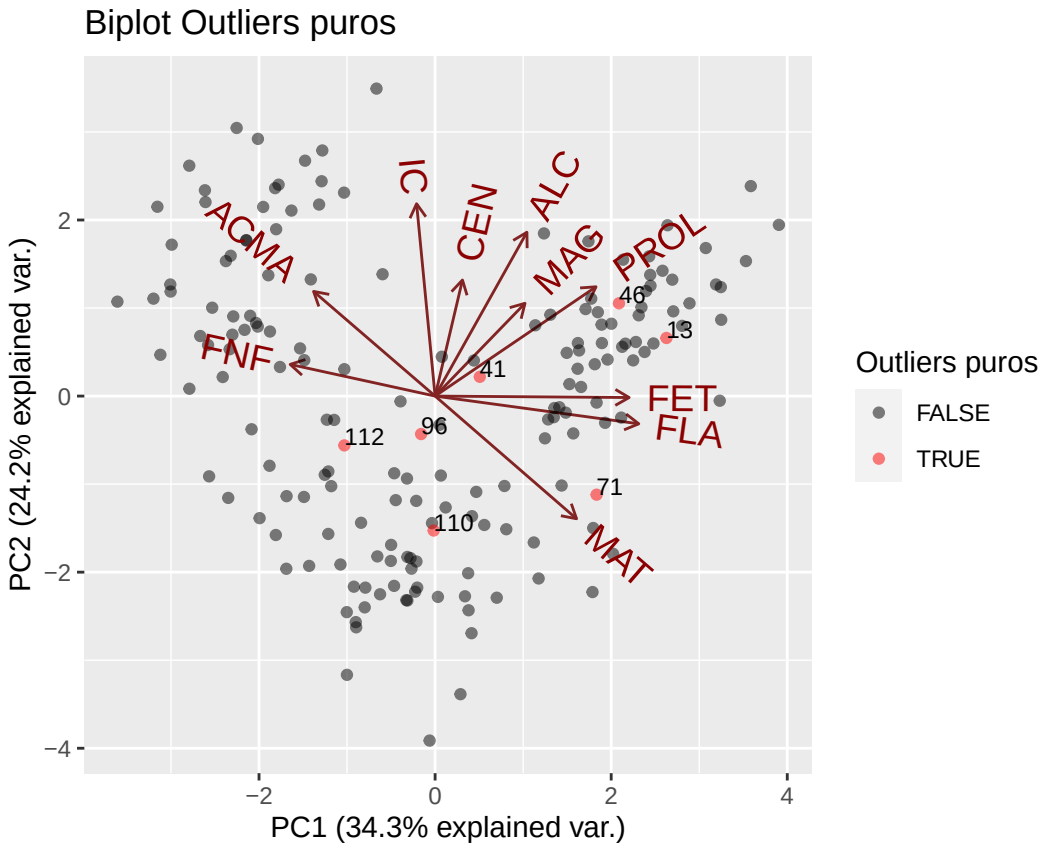

Figura 20: Biplot con los outliers multivarantes puros

Se puede ver que los vinos ***46*** y ***13*** son muy semejantes, ya que se sitúan cercanos entre sí. Por otro lado, los vinos ***41*** y ***96*** quedan cerca del centro del `biplot`, indicando que tienen valores moderados en todas las variables, y entre ellos presentan valores opuestos en algunas de ellas, como *MAG*, lo que nos indica que serán notablemente diferentes en cuenta a aroma y sabor.

Con respecto a los vinos ***112*** y ***110***, parecen semejantes en todas las variables, excepto en *ACMA* y *MAT*, en las que presentan valores opuestos. Es decir, el vino ***112*** será más ácido y refrescante que el ***110*** (*ACMA*), pero tendrá un color semejante (*IC*) aunque con un tono distinto (*MAT*).

Por último, el vino ***71*** parece ser el más distinto al resto en todas las variables, ya que es el más aislado de todos ellos.

Si vemos sus datos normalizados:

```
##           ALC       ACMA         CEN        MAG        FET          FLA
## 71   1.071116 -0.7414244  1.10455622 -0.9586252  1.0499230  0.837678159
## 110 -1.896212  1.2508666 -1.98544904  0.5229589  1.4170624  0.557301786
## 96  -1.463477 -0.1964479  1.35902724  2.4278527 -1.1050255 -1.034835475
## 13   2.171500 -0.5448755  0.08667213 -0.6058670  1.2893617  1.668793836
## 41   0.514742  1.3402070 -0.89485895 -0.6764187  0.2517939  0.657436205
## 112 -1.525296  0.3038583  2.01338129  0.2407524 -0.8655868  0.006562482
## 46   1.714037  1.1168560 -0.31321090  0.1702008  1.5288005  1.148094857
##            FNF          IC        MAT       PROL
## 71  -1.2219238 -0.72051661  1.7588621 -1.0641887
## 110 -0.9812556 -0.92701739 -0.9033234 -0.5814713
## 96  -1.7834830 -1.09910137 -0.0304757 -0.3813977
## 13   0.5429765  0.14850751  1.2787959  1.2858827
## 41  -0.7405873 -0.33332765 -0.2050452  0.9206689
## 112  1.9067631 -0.53982843  1.1915111 -0.4385616
## 46  -0.7405873 -0.06659747  0.3623058  1.0159421
```

Se puede ver cómo el vino ***71*** presenta valores normalizados $[-1, 1]$ (aproximadamente) para todas las variables, lo que lo hace único, y seguramente haga que este vino sea muy moderado en todos los sentidos (acidez, sabor, aromas, color, tono, etc.).

4. Análisis de resultados

4.1. Conjunto de datos

La práctica se ha realizado con el dataset `wine` disponible en UCI Machine Learning Repository. Se ha suprimido la variable de clase `CLASSS`, así como `ALCEN`, `PRO` y `OD` (estas por tener una varianza menor al 10% en los datos). Por lo tanto, el análisis de outliers se ha realizado con las variables `ALC`, `ACMA`, `CEN`, `FET`, `FLA`, `FNF`, `IC`, `MAT` y `PROL`, y siguiendo la práctica, se ha aplicado la normalización z-score cuando así ha sido requerida.

4.2. Outliers en una variable

4.2.1. Método IQR

Se han encontrado tres outliers (no extremos) que son el vino ***25***, ***59*** y ***121***. Estos comparten valores elevados en la variable *CEN*, lo que indica que son ricos en minerales como el calcio, potasio, hierro, etc. Además, en el resto de variables presentan valores más normales, a excepción del vino ***121***, que es outlier también con respecto a la variable *FLA*, lo que lo hará un vino con mucho aroma y sabor, especialmente en comparación con el resto.

Otras características de este vino ***121*** es que está dentro del grupo de los vinos con más alcohol (*ALC*) y más fenoles (*FET*), que aporta también en aroma y sabor, a la par que su color y matiz serán semejantes a la media del resto de vinos (*IC* y *MAT*).

4.2.2. Test de Hipótesis

El test de Shapir-Wilks ha rechazado la Normalidad en todas las variables, a excepción de la variable *CEN*, en la que concluimos que sigue dicha distribución.

Además, tras el análisis con ña variable *CEN*, se ha concluido que el único outlier IQR que también lo es desde un punto de vista estadístico, es el outlier ***59***.

4.3. Outliers multivariantes

4.3.1. Visualización con biplot

Desafortunadamente, la varianza explicada del biplot obtenido no es sufuciente (del 80%) como para considerarla una buena aporximación. Por lo tanto, todas las conclusiones sacadas de esta representación deberían recibir un análisis adicional (quizás incluyendo otra componente principal al estudio) para que sean más fiables.

4.3.2. Métodos estadísticos usando la distancia de Mahalanobis

La naturaleza de los datos, es decir, la no normalidad de todas sus variables a excepción de una, ha impedido este análisis de outliers multivariantes. Este hecho nos indica que no se puede asegurar que ninguno de los outliers multivariantes lo sea realmente desde un punto de vista estadístico.

4.3.3. LOF

Con este método hubo 2 vinos que destacaron sobre el resto como outliers: ***121*** y el ***71***. De estos dos, ya se vio en el método IQR que el vino ***121*** presentaba valores muy elevados para dos de las variables.

Sin embargo, el vino ***71*** no presenta valores elevados en ninguna de las variables, siendo todos sus valores bastante moderados alrededor de la media de la distribución de todo el conjunto de datos. Revisando el `biplot`, se puede ver que este se sitúa en una zona con un baja densidad junto a otras dos zonas con densidades aparentemente mayores:

- En una de ellas predominan vinos con alta gradación de alcohol, muchos fenoles y minerales, así como magnesio, lo que les aportarán aromas y sabores más fuertes).

- En la otra nube de alta densidad cercana a este vino, se caracterizan por todo lo contrario, bajos en alcohol y magnesio por ejemplo, pero presentan una mayor variedad de matices en el color y de acidez (la primera nube de puntos tiene un tono de color elevado y una acidez baja).

Por lo tanto, el vino ***71*** se coloca en medio de estas características. A excepción del tono del color e igual un elevado aroma, este vino presenta valores "medios" (moderados) en cuanto a minerales, magnesio, alcohol o prolina (que también contribuye al sabor), por lo que se coloca como un vino generalmente moderado.

Probablemente el hecho de que se considere un outlier multivariante (y por lo que se coloque en esa posición de baja densidad) es debido a la combinación de valores de algunas de sus variantes, ya que no se ven muchos valores (vinos) moderados en cuanto a la mayoría de las variables que tengan untono de color (MAT) y aromas (FLA) tan marcados y superiores a la media.

De todos modos, y como se ha indicado, sería necesaria una mayor explicación por parte de las componentes principales para secundar estas afirmaciones con mayor seguridad.

4.3.4. Métodos basados en clustering

En el caso de utilizar el método de k-means, se determina si un valor es outlier en base a su distnacia euclídea con respecto al centroide del cluster al que se ha asignado. Para este método se han detectado cinco outliers (como se ha fijado), que son los vinos ***121***, ***95***, ****123***,* 96*** y ***69***. Al representarlos con un `boxplot`, se puede ver que el motico de que se hayan considerado outliers es por sus valores elevados en alguna de las variables. En concreto:

- El vino de etiqueta ***121***, ya visto durante toda la práctica como outlier con varios métodos, presenta valores elevados en las variables CEN y FLA, lo que contribuye a que la suma del efecto de estas provoque que obtenga un *score* elevado para este método.
- Y de igual manera, el resto de vinos detectados como outliers en este método presentan valores que, igual no son lo suficientemente elevados con respecto a la media como para considerarse un outlier en términos IQR, pero sí lo suficientemente elevados como para destacar entre algunas de las variables: los vinos ***95***, ***69*** y ***96*** destacan en la variable MAG, mientras que el ***123*** lo hace en la variable $ACMA$. Estos valores elevados en una de las variables, unido a valores inusuales en otras, provoca que el método k-means los detecte como outliers. De hecho, en los casos de los vinos ***123***, ***96*** y ***69*** es muy representativo el `biplot`, ya que están cercanos a la frontera de su cluster.