Trabajo anomalías

2024-12-26

Contents

1 Dataset y Selección de Variables

NOTA: *He incluido algunas celdas de código ya resuelto porque creo que dichos resultados son relevantes para entender el contexto del problema. Las celdas que se pedía completar están indicadas mediante el comentario "`# COMPLETAR`".*

En esta práctica usaremos el conjunto de datos del cáncer de mama (breast cancer dataset), descargado de https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/OPQMVF, que contiene información sobre varias características calculadas a partir de imágenes digitales de mamas. Describimos aquí las columnas que serán el objetivo de nuestro estudio:

Variables relativas a las características de las células

- **radius_mean**: Radio medio de las células (promedio de las distancias desde el centro hasta los puntos del perímetro).
- **texture_mean**: Variación estándar de los valores de intensidad en la imagen.
- **perimeter_mean**: Perímetro medio de las células.
- **area_mean**: Área media de las células.
- **smoothness_mean**: Suavidad media, calculada como la variación local en la longitud del radio.
- **compactness_mean**: Compacidad media, calculada como $(perímetro^2)/área - 1.0$.
- **concavity_mean**: Media de las concavidades (profundidades de las zonas cóncavas en el contorno).
- **concave_points_mean**: Media de los puntos cóncavos en el contorno.
- **symmetry_mean**: Media de la simetría.
- **fractal_dimension_mean**: Media de la dimensión fractal (aproximación de la "complejidad" del contorno).

Variables relativas a los errores estándar

- **radius_se**: Error estándar del radio.
- **texture_se**: Error estándar de la textura.
- **perimeter_se**: Error estándar del perímetro.
- **area_se**: Error estándar del área.
- **smoothness_se**: Error estándar de la suavidad.
- **compactness_se**: Error estándar de la compacidad.
- **concavity_se**: Error estándar de las concavidades.
- **concave_points_se**: Error estándar de los puntos cóncavos.
- **symmetry_se**: Error estándar de la simetría.
- **fractal_dimension_se**: Error estándar de la dimensión fractal.

Variables relativas a los peores valores

- **radius_worst**: Valor más alto del radio.
- **texture_worst**: Valor más alto de la textura.
- **perimeter_worst**: Valor más alto del perímetro.
- **area_worst**: Valor más alto del área.
- **smoothness_worst**: Valor más alto de la suavidad.
- **compactness_worst**: Valor más alto de la compacidad.

- **concavity_worst**: Valor más alto de las concavidades.
- **concave_points_worst**: Valor más alto de los puntos cóncavos.
- **symmetry_worst**: Valor más alto de la simetría.
- **fractal_dimension_worst**: Valor más alto de la dimensión fractal.

Variable de clasificación

- **Class**: Clasificación del diagnóstico. Los posibles valores son:
  - `"o"`: outlier.
  - `"n"`: normal.

Este conjunto de datos permite analizar las diferencias entre tumores malignos y benignos. Utilizando 31 características tanto físicas como geométricas de las células mamarias, podemos hallar las instancias fuera de lo normal (outliers) y las clasificaremos como tumores malignos.

Empezamos cargando el conjunto de datos:

```
datos <- read.csv("breast-cancer-unsupervised-ad.csv", comment.char="@", header = FALSE)

#Asignamos manualmente los nombres de los atributos
names(datos) <- c("radius_mean",
                  "texture_mean",
                  "perimeter_mean",
                  "area_mean",
                  "smoothness_mean",
                  "compactness_mean",
                  "concavity_mean",
                  "concave_points_mean",
                  "symmetry_mean",
                  "fractal_dimension_mean",
                  "radius_se",
                  "texture_se",
                  "perimeter_se",
                  "area_se",
                  "smoothness_se",
                  "compactness_se",
                  "concavity_se",
                  "concave_points_se",
                  "symmetry_se",
                  "fractal_dimension_se",
                  "radius_worst",
                  "texture_worst",
                  "perimeter_worst",
                  "area_worst",
                  "smoothness_worst",
                  "compactness_worst",
                  "concavity_worst",
                  "concave_points_worst",
                  "symmetry_worst",
                  "fractal_dimension_worst",
                  "Class")

head(datos)
```

```
##   radius_mean texture_mean perimeter_mean area_mean smoothness_mean
## 1       17.99        10.38         122.80    1001.0         0.11840
## 2       20.57        17.77         132.90    1326.0         0.08474
## 3       19.69        21.25         130.00    1203.0         0.10960
## 4       11.42        20.38          77.58     386.1         0.14250
## 5       20.29        14.34         135.10    1297.0         0.10030
## 6       12.45        15.70          82.57     477.1         0.12780
##   compactness_mean concavity_mean concave_points_mean symmetry_mean
## 1          0.27760         0.3001             0.14710        0.2419
## 2          0.07864         0.0869             0.07017        0.1812
## 3          0.15990         0.1974             0.12790        0.2069
## 4          0.28390         0.2414             0.10520        0.2597
## 5          0.13280         0.1980             0.10430        0.1809
## 6          0.17000         0.1578             0.08089        0.2087
##   fractal_dimension_mean radius_se texture_se perimeter_se area_se
## 1                0.07871    1.0950     0.9053        8.589  153.40
## 2                0.05667    0.5435     0.7339        3.398   74.08
## 3                0.05999    0.7456     0.7869        4.585   94.03
## 4                0.09744    0.4956     1.1560        3.445   27.23
## 5                0.05883    0.7572     0.7813        5.438   94.44
## 6                0.07613    0.3345     0.8902        2.217   27.19
##   smoothness_se compactness_se concavity_se concave_points_se symmetry_se
## 1      0.006399        0.04904      0.05373           0.01587     0.03003
## 2      0.005225        0.01308      0.01860           0.01340     0.01389
## 3      0.006150        0.04006      0.03832           0.02058     0.02250
## 4      0.009110        0.07458      0.05661           0.01867     0.05963
## 5      0.011490        0.02461      0.05688           0.01885     0.01756
## 6      0.007510        0.03345      0.03672           0.01137     0.02165
##   fractal_dimension_se radius_worst texture_worst perimeter_worst area_worst
## 1             0.006193        25.38         17.33          184.60     2019.0
## 2             0.003532        24.99         23.41          158.80     1956.0
## 3             0.004571        23.57         25.53          152.50     1709.0
## 4             0.009208        14.91         26.50           98.87      567.7
## 5             0.005115        22.54         16.67          152.20     1575.0
## 6             0.005082        15.47         23.75          103.40      741.6
##   smoothness_worst compactness_worst concavity_worst concave_points_worst
## 1           0.1622            0.6656          0.7119               0.2654
## 2           0.1238            0.1866          0.2416               0.1860
## 3           0.1444            0.4245          0.4504               0.2430
## 4           0.2098            0.8663          0.6869               0.2575
## 5           0.1374            0.2050          0.4000               0.1625
## 6           0.1791            0.5249          0.5355               0.1741
##   symmetry_worst fractal_dimension_worst Class
## 1         0.4601                0.11890     o
## 2         0.2750                0.08902     o
## 3         0.3613                0.08758     o
## 4         0.6638                0.17300     o
## 5         0.2364                0.07678     o
## 6         0.3985                0.12440     o
```

Observamos que en este dataset, el atributo de la **clase** toma los valores 'o' y 'n', que corresponde al diagnóstico. Basándonos en el contexto del dataset de breast cancer, esto podría significar lo siguiente:

- 'o' probablemente indica "outlier", que podría ser equivalente a "maligno" (outlier en el sentido de un

crecimiento anormal de células que podría llevar al cáncer maligno).
- 'n' probablemente indica "normal", lo que podría referirse a un diagnóstico "benigno".

También podría ser que 'o' y 'n' se refieran únicamente a outlier y normal, en el sentido de que los outliers son datos que toman valores muy extremos, independientemente de si representan un tumor maligno o benigno.

Creamos un dataframe con solo las variables numéricas (omitimos el atributo Class porque queremos detectar los outliers nosotros). Partimos entonces de 30 columnas.

```
columnas.num = sapply(c(1:ncol(datos)) , function(x) is.numeric(datos[, x]))
datos.num = datos[, columnas.num]
head(datos.num)
```

```
##   radius_mean texture_mean perimeter_mean area_mean smoothness_mean
## 1       17.99        10.38         122.80    1001.0         0.11840
## 2       20.57        17.77         132.90    1326.0         0.08474
## 3       19.69        21.25         130.00    1203.0         0.10960
## 4       11.42        20.38          77.58     386.1         0.14250
## 5       20.29        14.34         135.10    1297.0         0.10030
## 6       12.45        15.70          82.57     477.1         0.12780
##   compactness_mean concavity_mean concave_points_mean symmetry_mean
## 1          0.27760         0.3001             0.14710        0.2419
## 2          0.07864         0.0869             0.07017        0.1812
## 3          0.15990         0.1974             0.12790        0.2069
## 4          0.28390         0.2414             0.10520        0.2597
## 5          0.13280         0.1980             0.10430        0.1809
## 6          0.17000         0.1578             0.08089        0.2087
##   fractal_dimension_mean radius_se texture_se perimeter_se area_se
## 1                0.07871    1.0950     0.9053        8.589  153.40
## 2                0.05667    0.5435     0.7339        3.398   74.08
## 3                0.05999    0.7456     0.7869        4.585   94.03
## 4                0.09744    0.4956     1.1560        3.445   27.23
## 5                0.05883    0.7572     0.7813        5.438   94.44
## 6                0.07613    0.3345     0.8902        2.217   27.19
##   smoothness_se compactness_se concavity_se concave_points_se symmetry_se
## 1      0.006399        0.04904      0.05373           0.01587     0.03003
## 2      0.005225        0.01308      0.01860           0.01340     0.01389
## 3      0.006150        0.04006      0.03832           0.02058     0.02250
## 4      0.009110        0.07458      0.05661           0.01867     0.05963
## 5      0.011490        0.02461      0.05688           0.01885     0.01756
## 6      0.007510        0.03345      0.03672           0.01137     0.02165
##   fractal_dimension_se radius_worst texture_worst perimeter_worst area_worst
## 1             0.006193        25.38         17.33          184.60     2019.0
## 2             0.003532        24.99         23.41          158.80     1956.0
## 3             0.004571        23.57         25.53          152.50     1709.0
## 4             0.009208        14.91         26.50           98.87      567.7
## 5             0.005115        22.54         16.67          152.20     1575.0
## 6             0.005082        15.47         23.75          103.40      741.6
##   smoothness_worst compactness_worst concavity_worst concave_points_worst
## 1           0.1622            0.6656          0.7119               0.2654
## 2           0.1238            0.1866          0.2416               0.1860
## 3           0.1444            0.4245          0.4504               0.2430
## 4           0.2098            0.8663          0.6869               0.2575
## 5           0.1374            0.2050          0.4000               0.1625
```

```
## 6           0.1791            0.5249          0.5355                0.1741
##   symmetry_worst fractal_dimension_worst
## 1         0.4601                 0.11890
## 2         0.2750                 0.08902
## 3         0.3613                 0.08758
## 4         0.6638                 0.17300
## 5         0.2364                 0.07678
## 6         0.3985                 0.12440
```

Eliminamos columnas con pocos valores distintos pues consideramos que no tienen valores anómalos. Para ello, eliminamos las columnas con menos de 10 valores distintos.

```
# Identificamos columnas con pocos valores distintos
valores.unicos <- sapply(datos.num, function(col) length(unique(col)))

# Mostrar columnas con pocos valores distintos
pocas.distintas <- names(valores.unicos[valores.unicos < 10])
print(paste("Columnas con pocos valores distintos:", paste(pocas.distintas, collapse = ", ")))
```

```
## [1] "Columnas con pocos valores distintos: "
```

```
# Eliminar columnas con pocos valores distintos
datos.num <- datos.num[, !(names(datos.num) %in% pocas.distintas)]

# Verificar el nuevo dataset
head(datos.num)
```

```
##   radius_mean texture_mean perimeter_mean area_mean smoothness_mean
## 1       17.99        10.38         122.80    1001.0         0.11840
## 2       20.57        17.77         132.90    1326.0         0.08474
## 3       19.69        21.25         130.00    1203.0         0.10960
## 4       11.42        20.38          77.58     386.1         0.14250
## 5       20.29        14.34         135.10    1297.0         0.10030
## 6       12.45        15.70          82.57     477.1         0.12780
##   compactness_mean concavity_mean concave_points_mean symmetry_mean
## 1          0.27760         0.3001              0.14710        0.2419
## 2          0.07864         0.0869              0.07017        0.1812
## 3          0.15990         0.1974              0.12790        0.2069
## 4          0.28390         0.2414              0.10520        0.2597
## 5          0.13280         0.1980              0.10430        0.1809
## 6          0.17000         0.1578              0.08089        0.2087
##   fractal_dimension_mean radius_se texture_se perimeter_se area_se
## 1                0.07871    1.0950     0.9053        8.589  153.40
## 2                0.05667    0.5435     0.7339        3.398   74.08
## 3                0.05999    0.7456     0.7869        4.585   94.03
## 4                0.09744    0.4956     1.1560        3.445   27.23
## 5                0.05883    0.7572     0.7813        5.438   94.44
## 6                0.07613    0.3345     0.8902        2.217   27.19
##   smoothness_se compactness_se concavity_se concave_points_se symmetry_se
## 1      0.006399        0.04904      0.05373           0.01587     0.03003
## 2      0.005225        0.01308      0.01860           0.01340     0.01389
## 3      0.006150        0.04006      0.03832           0.02058     0.02250
## 4      0.009110        0.07458      0.05661           0.01867     0.05963
```

```
## 5      0.011490        0.02461      0.05688           0.01885     0.01756
## 6      0.007510        0.03345      0.03672           0.01137     0.02165
##   fractal_dimension_se radius_worst texture_worst perimeter_worst area_worst
## 1             0.006193        25.38         17.33          184.60     2019.0
## 2             0.003532        24.99         23.41          158.80     1956.0
## 3             0.004571        23.57         25.53          152.50     1709.0
## 4             0.009208        14.91         26.50           98.87      567.7
## 5             0.005115        22.54         16.67          152.20     1575.0
## 6             0.005082        15.47         23.75          103.40      741.6
##   smoothness_worst compactness_worst concavity_worst concave_points_worst
## 1           0.1622            0.6656          0.7119               0.2654
## 2           0.1238            0.1866          0.2416               0.1860
## 3           0.1444            0.4245          0.4504               0.2430
## 4           0.2098            0.8663          0.6869               0.2575
## 5           0.1374            0.2050          0.4000               0.1625
## 6           0.1791            0.5249          0.5355               0.1741
##   symmetry_worst fractal_dimension_worst
## 1         0.4601                 0.11890
## 2         0.2750                 0.08902
## 3         0.3613                 0.08758
## 4         0.6638                 0.17300
## 5         0.2364                 0.07678
## 6         0.3985                 0.12440
```

Observamos que no se han eliminado columnas pues todas tienen alta variabilidad de los datos (todas toman valores reales continuos).

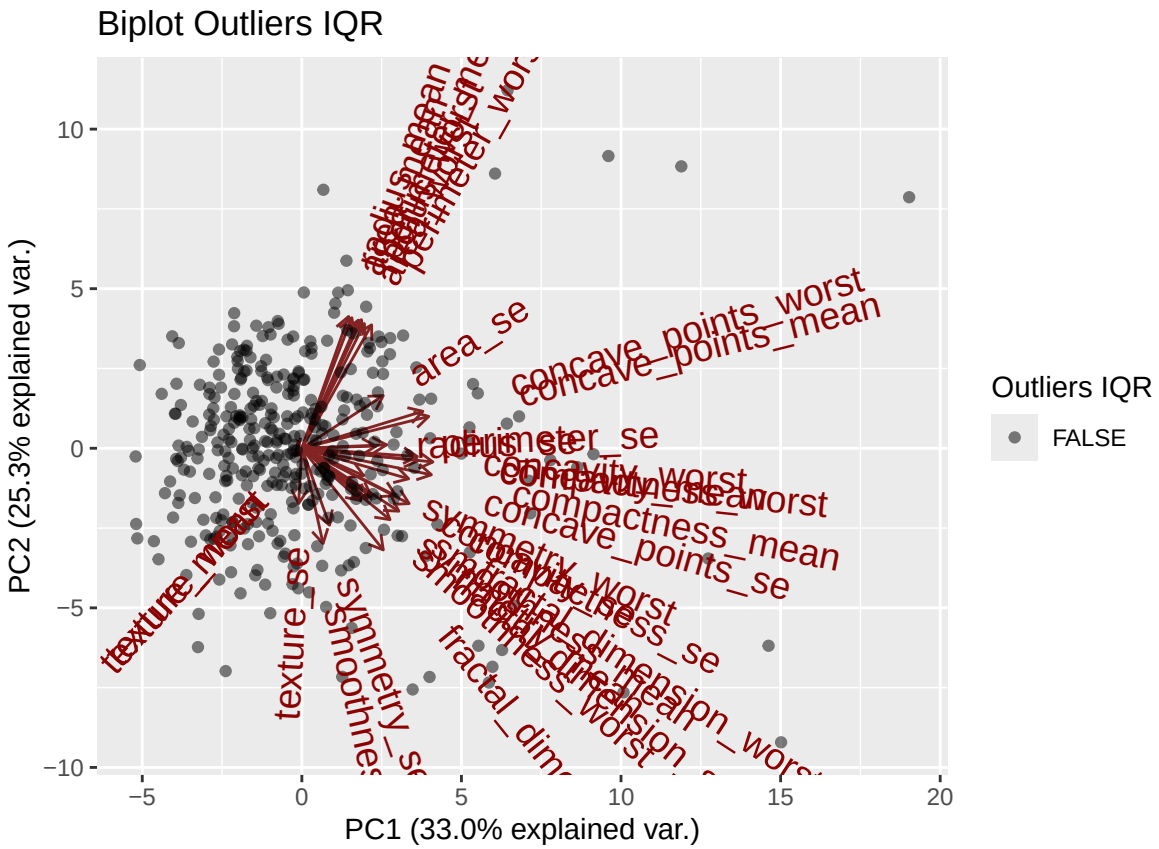
Eliminamos filas con valores nulos:

```
datos.num = na.omit(datos.num)

# Confirmamos que no hay valores nulos
print(sum(is.na(datos.num)))  # Debería devolver 0
```

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

Vamos a realizar una **reducción de dimensionalidad** temprana con el fin de no encontrar problemas en el futuro al intentar interpretar los gráficos, debido al alto número de variables. Para ello, vamos a eliminar variables que estén muy correladas entre sí. Por tanto, vamos a representar un **biplot** de las variables y las que sigan la misma dirección (mismo sentido o sentido inverso), estarán correladas:

```
biplot.outliers.IQR = biplot_2_colores(scale(datos.num),
                                       titulo.grupo.a.mostrar = "Outliers IQR",
                                       titulo ="Biplot Outliers IQR")
biplot.outliers.IQR
```



Las siguientes variables tienen una alta correlación positiva:

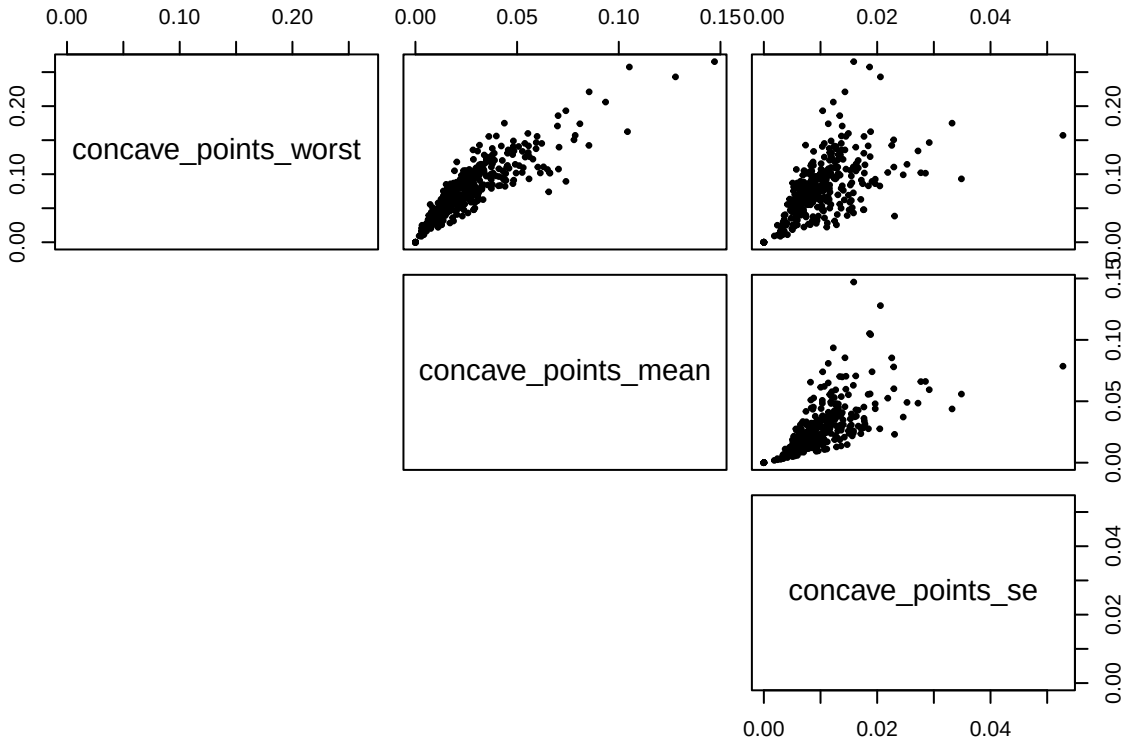
- `concave_points_worst`, `concave_points_mean`, `concave_points_se`.
- `compactness_mean`, `compactness_worst`, `compactness_se`.
- `texture_mean`, `texture_worst`, `texture_se`.
- `radius_mean`, `radius_worst`, `radius_se`, `perimeter_mean`, `perimeter_worst`, `perimeter_se`, `area_mean`, `area_se`, `area_worst`.
- `symmetry_mean`, `symmetry_worst`, `symmetry_se`.
- `smoothness_mean`, `smoothness_worst`, `smoothness_se`.
- `concavity_mean`, `concavity_worst`, `concavity_se`.
- `fractal_dimension_mean`, `fractal_dimension_se`, `fractal_dimension_worst`.

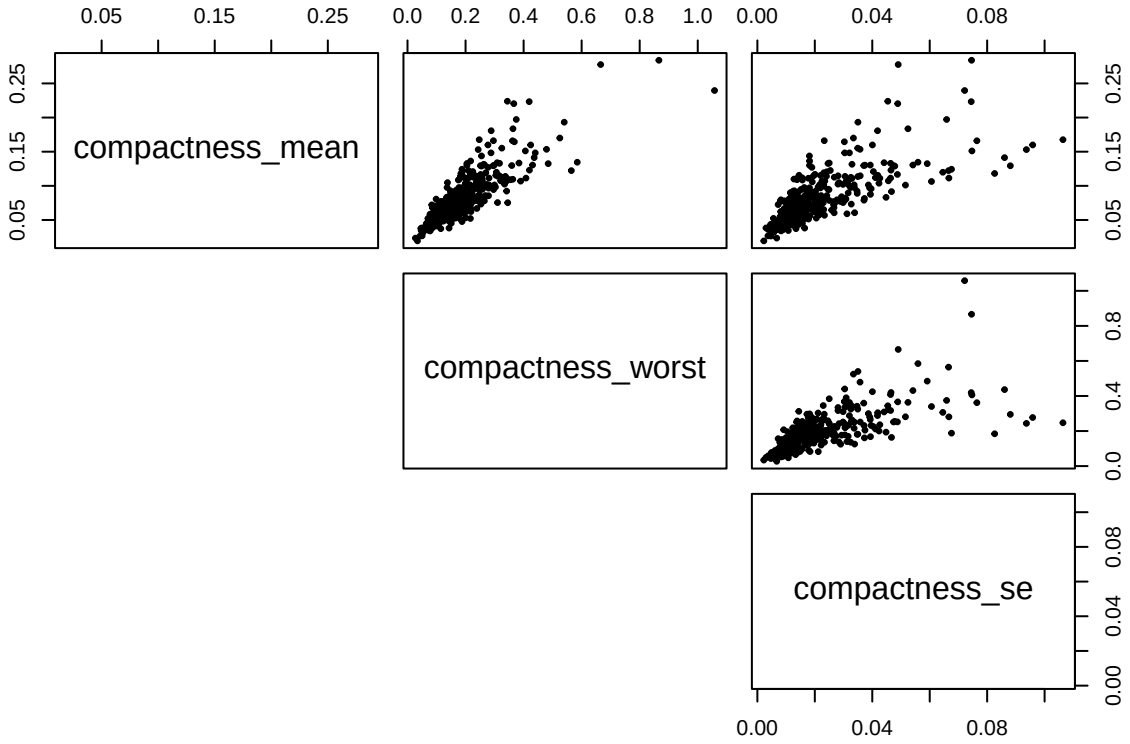
Por tanto, estas variables parecen ser redundantes entre sí pues contienen información muy similar. Vamos a seleccionar una variable representativa de cada grupo para mejorar la interpretabilidad.

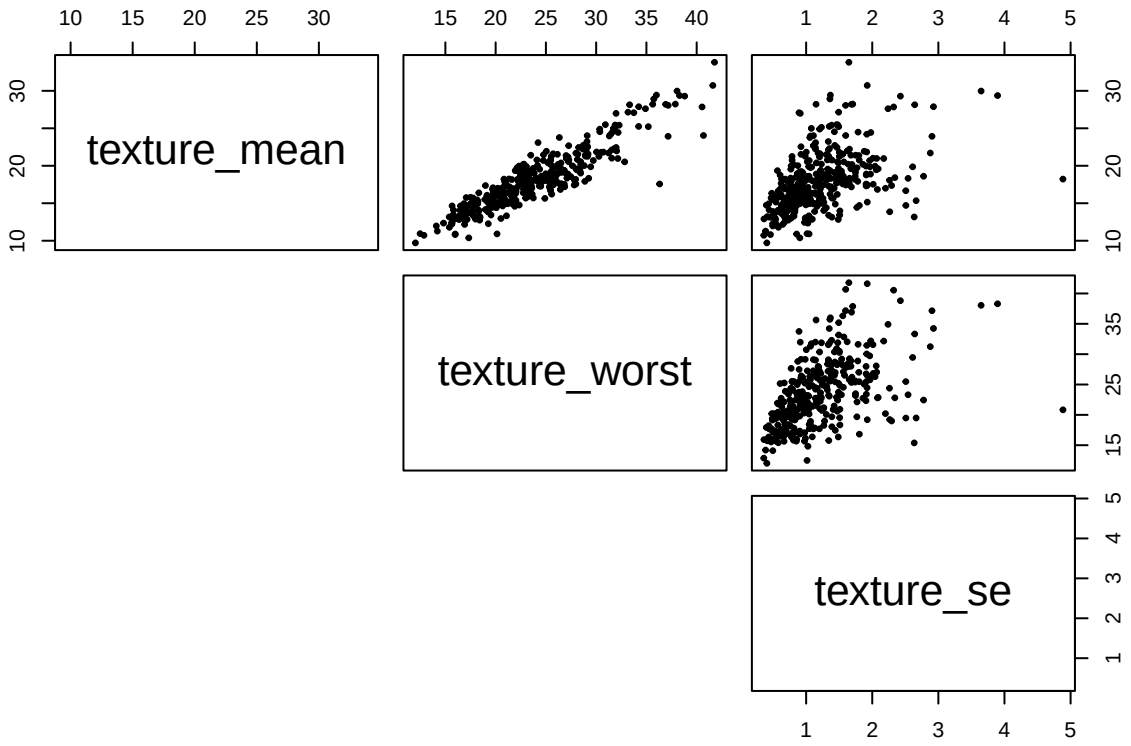
Para ello, primero vamos a representar **diagramas de dispersión** de cada grupo:

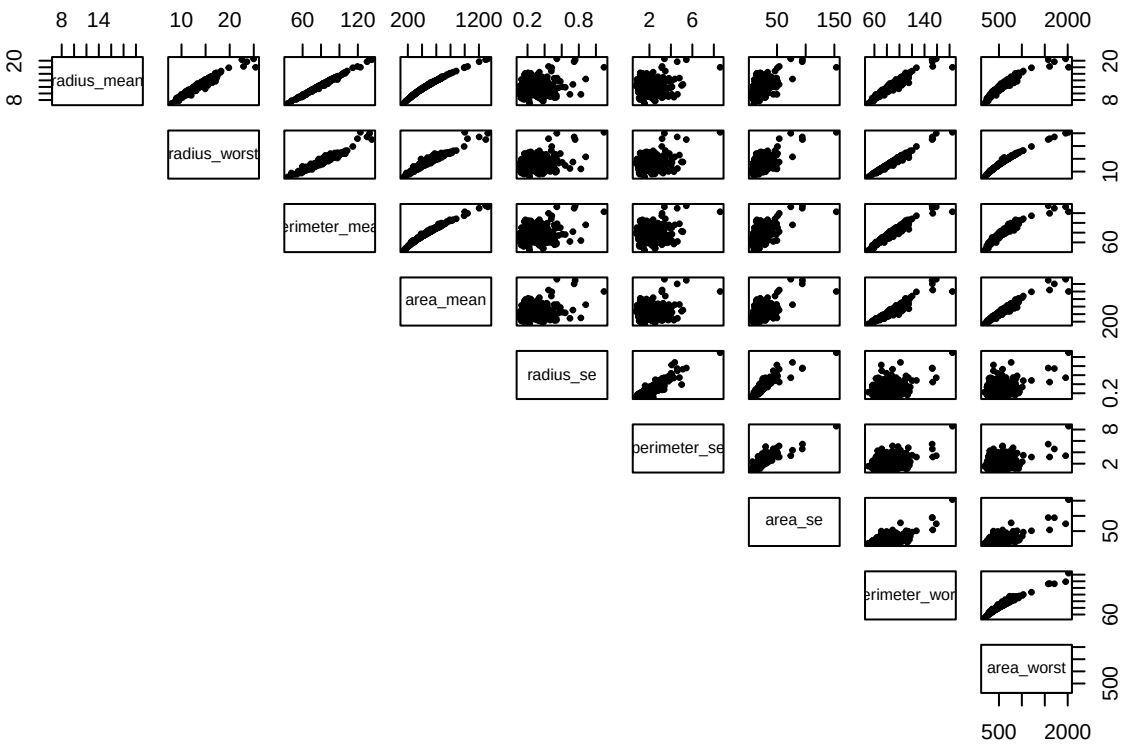
```
# Grupos de variables correlacionadas
grupos <- list(
  c("concave_points_worst", "concave_points_mean", "concave_points_se"),
  c("compactness_mean", "compactness_worst", "compactness_se"),
  c("texture_mean", "texture_worst", "texture_se"),
  c("radius_mean", "radius_worst", "perimeter_mean", "area_mean", "radius_se", "perimeter_se", "area_se
  c("symmetry_mean", "symmetry_worst", "symmetry_se"),
  c("smoothness_mean", "smoothness_worst", "smoothness_se"),
  c("concavity_mean", "concavity_worst", "concavity_se"),
  c("fractal_dimension_mean", "fractal_dimension_se", "fractal_dimension_worst")
```

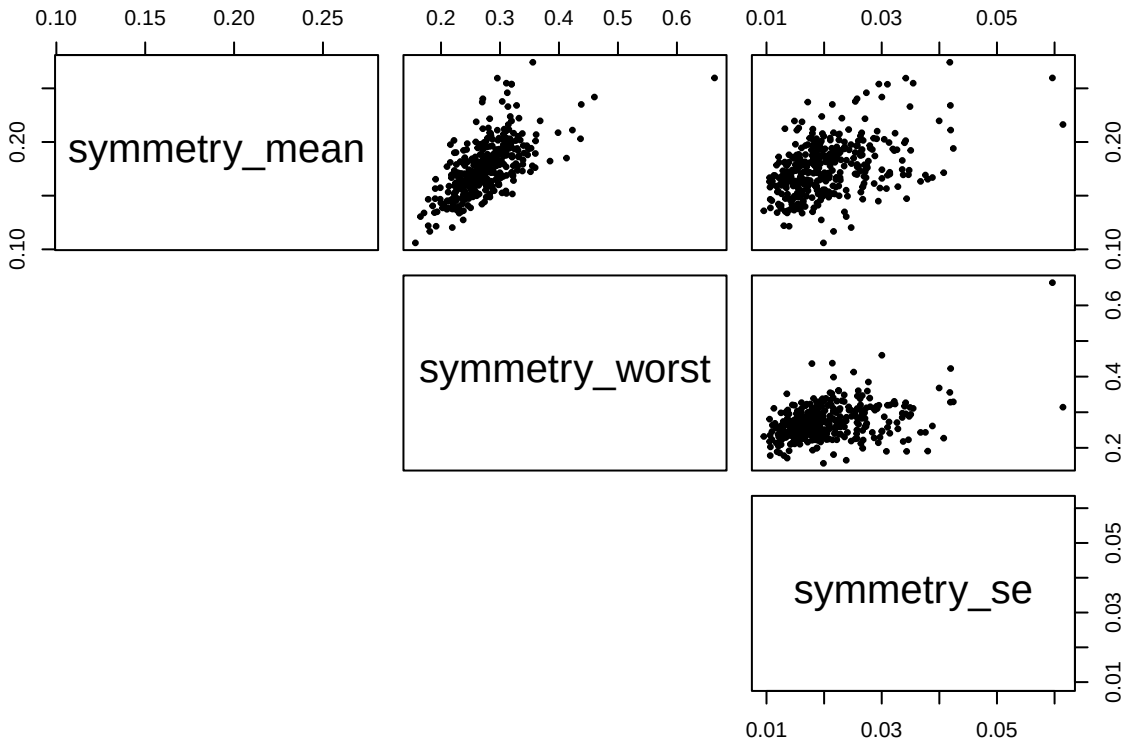
```
)

# Generar los pares para cada grupo
for (grupo in grupos) {
  pairs(datos.num[, grupo],
        pch = 19, cex = 0.5, lower.panel = NULL)
}
```

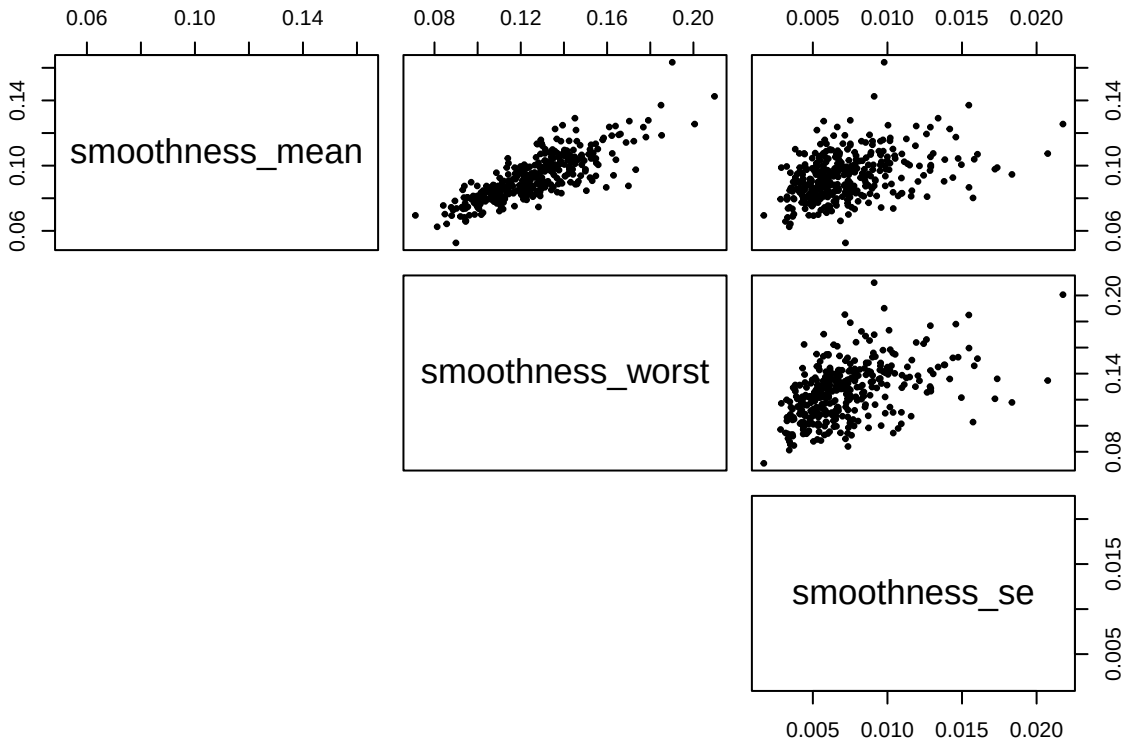


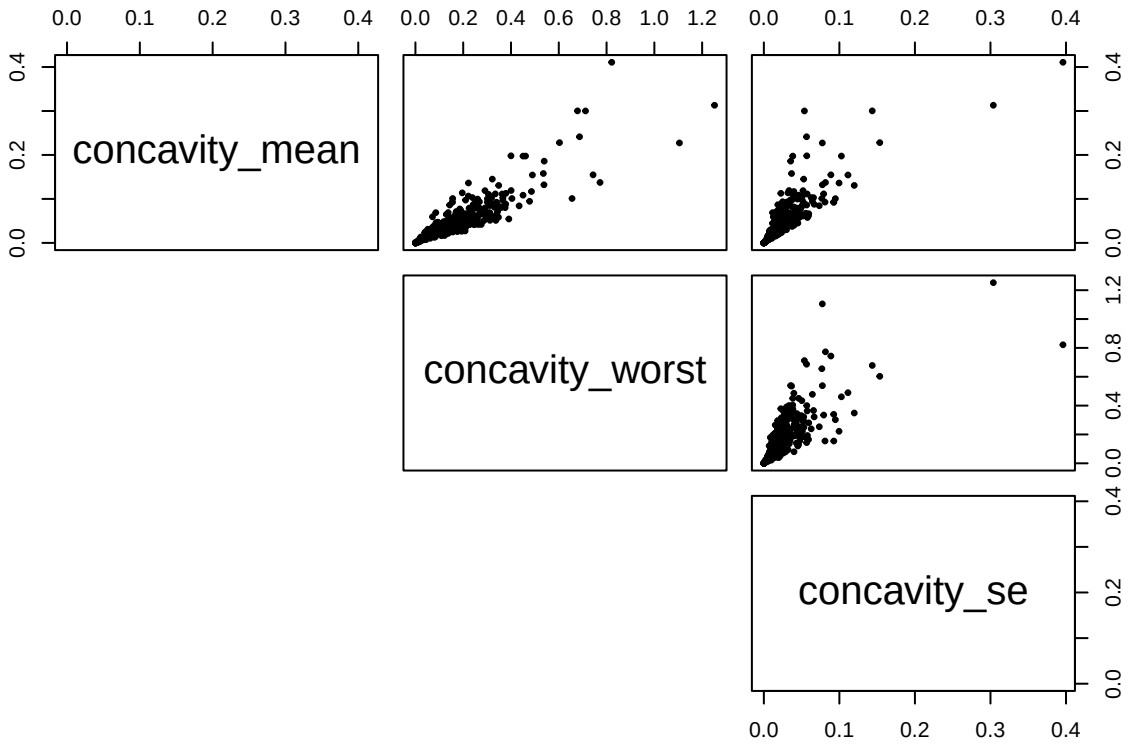

compactness_mean
compactness_worst
compactness_se
0.05
0.15
0.25
0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.04
0.08

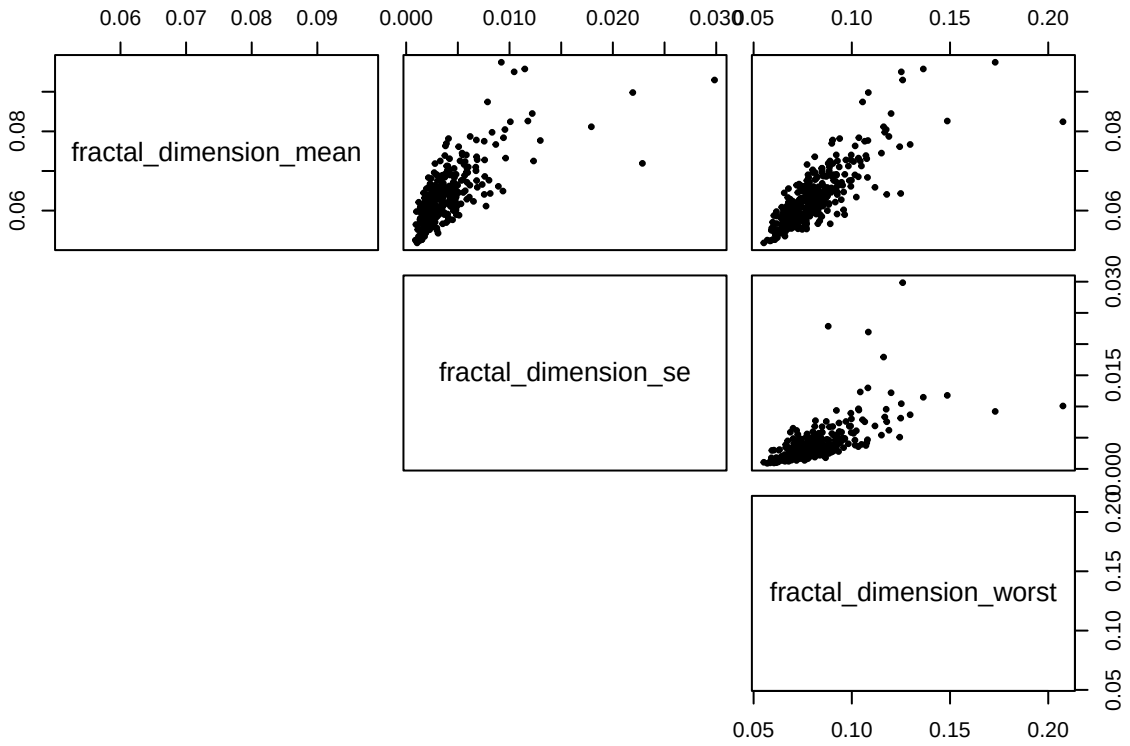

texture_mean
texture_worst
texture_se
10 15 20 25 30
15 20 25 30 35 40
1 2 3 4 5


radius_mean
radius_worst
perimeter_mea
area_mean
radius_se
perimeter_se
area_se
perimeter_wor
area_worst


symmetry_mean
symmetry_worst
symmetry_se
0.10
0.15
0.20
0.25
0.2
0.3
0.4
0.5
0.6
0.01
0.03
0.05


smoothness_mean
smoothness_worst
smoothness_se
0.06
0.10
0.14
0.08
0.12
0.16
0.20
0.005
0.010
0.015
0.020


concavity_mean
concavity_worst
concavity_se
0.0
0.1
0.2
0.3
0.4
0.0
0.2
0.4
0.6
0.8
1.0
1.2



Tras observar los diagramas de dispersión, he elegido eliminar las siguientes variables:

- `concave_points_worst`, `concave_points_se`
- `compactness_worst`, `compactness_se`
- `texture_worst`, `texture_se`.
- `radius_worst`, `perimeter_mean`, `area_mean`, `perimeter_se`, `area_se`, `perimeter_worst`, `area_worst`.
- `symmetry_worst`, `symmetry_se`
- `smoothness_worst`
- `concavity_worst`
- `fractal_dimension_worst`, `fractal_dimension_se`

Eliminamos las columnas elegidas:

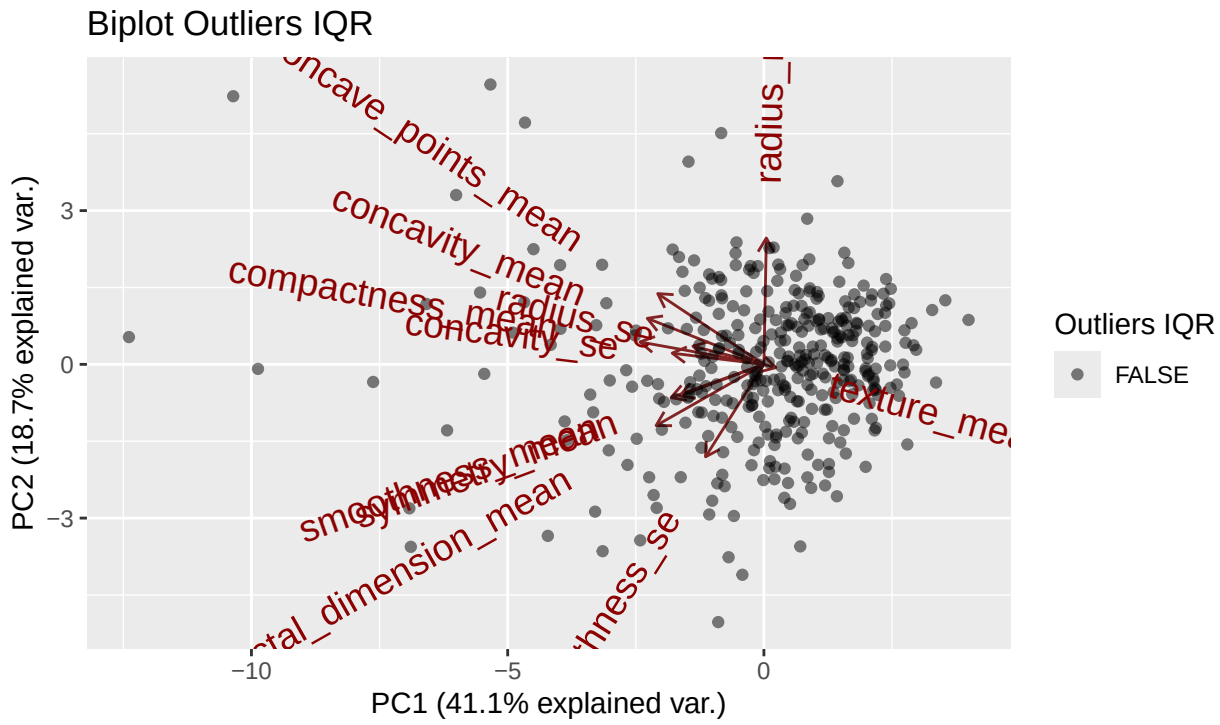
```
columnas.a.quitar <- c(
  "concave_points_worst", "concave_points_se",
  "compactness_worst", "compactness_se",
  "texture_worst", "texture_se",
  "radius_worst", "perimeter_mean", "area_mean", "perimeter_se", "area_se", "perimeter_worst", "area_wo
  "symmetry_worst", "symmetry_se",
  "smoothness_worst",
  "concavity_worst",
  "fractal_dimension_worst", "fractal_dimension_se"
)

# Eliminar las columnas seleccionadas
```

```r
datos.num <- datos.num[, !(colnames(datos.num) %in% columnas.a.quitar)]
head(datos.num)
```

```
##   radius_mean texture_mean smoothness_mean compactness_mean concavity_mean
## 1       17.99        10.38         0.11840          0.27760         0.3001
## 2       20.57        17.77         0.08474          0.07864         0.0869
## 3       19.69        21.25         0.10960          0.15990         0.1974
## 4       11.42        20.38         0.14250          0.28390         0.2414
## 5       20.29        14.34         0.10030          0.13280         0.1980
## 6       12.45        15.70         0.12780          0.17000         0.1578
##   concave_points_mean symmetry_mean fractal_dimension_mean radius_se
## 1             0.14710        0.2419                0.07871    1.0950
## 2             0.07017        0.1812                0.05667    0.5435
## 3             0.12790        0.2069                0.05999    0.7456
## 4             0.10520        0.2597                0.09744    0.4956
## 5             0.10430        0.1809                0.05883    0.7572
## 6             0.08089        0.2087                0.07613    0.3345
##   smoothness_se concavity_se
## 1      0.006399      0.05373
## 2      0.005225      0.01860
## 3      0.006150      0.03832
## 4      0.009110      0.05661
## 5      0.011490      0.05688
## 6      0.007510      0.03672
```

Mostramos el biplot de las variables restantes:

```r
biplot.outliers.IQR = biplot_2_colores(datos.num,
                                       titulo.grupo.a.mostrar = "Outliers IQR",
                                       titulo ="Biplot Outliers IQR")
biplot.outliers.IQR
```

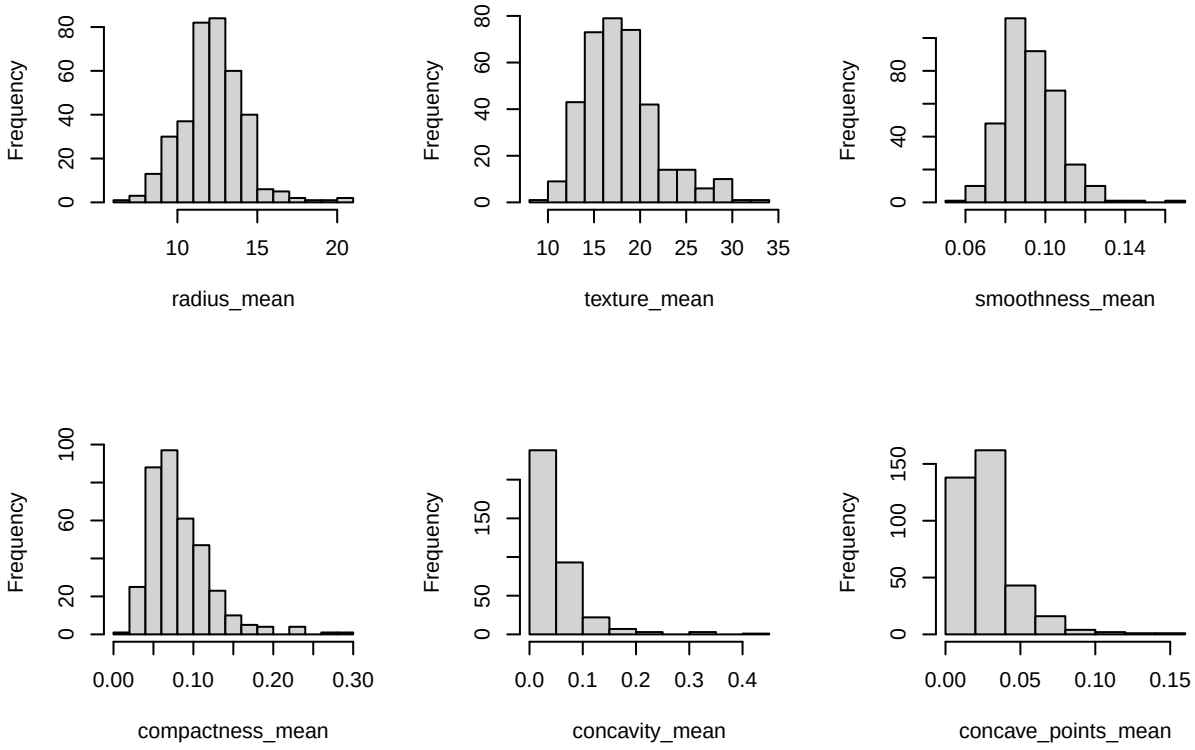


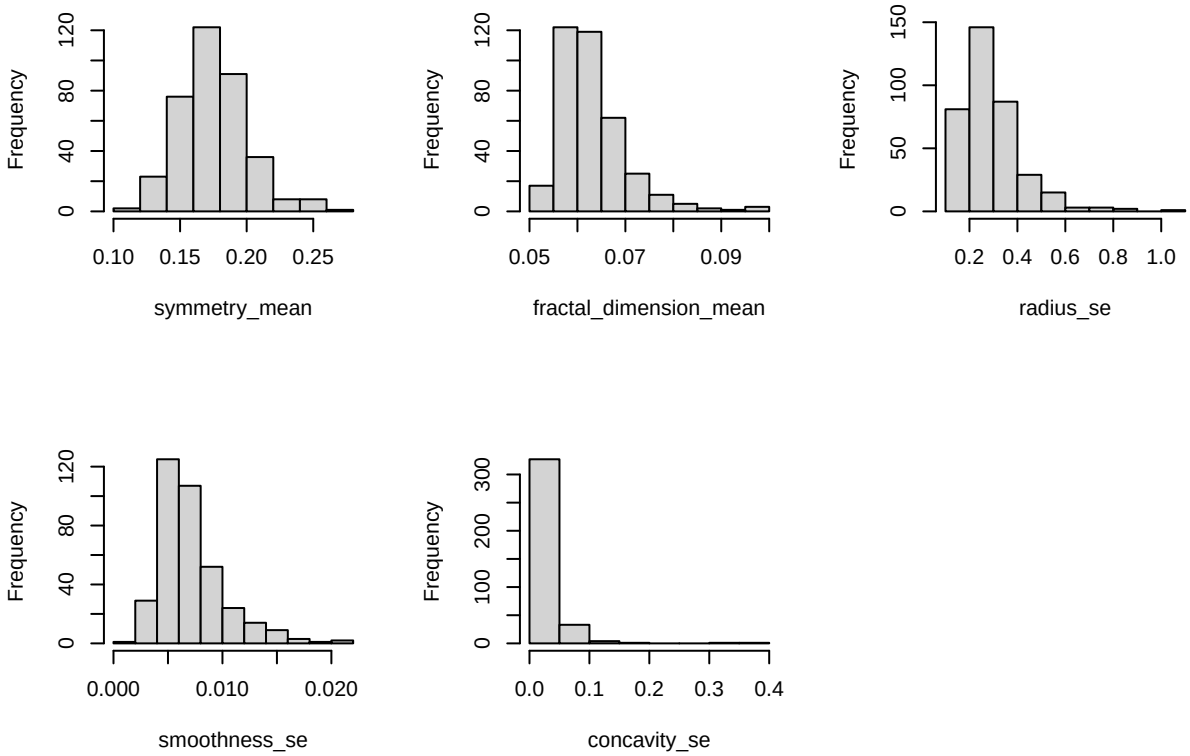
2 Detección de outliers en una dimensión

2.1 Outliers IQR

Observamos el histograma de cada variable para ver que los datos no sigan distribuciones raras (picos o forma de U).

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


Frequency
0
20
40
60
80
10
15
20
radius_mean
Frequency
0
20
40
60
80
10
15
20
25
30
35
texture_mean
Frequency
0
40
80
0.06
0.10
0.14
smoothness_mean
Frequency
0
20
60
100
0.00
0.10
0.20
0.30
compactness_mean
Frequency
0
50
150
0.0
0.1
0.2
0.3
0.4
concavity_mean
Frequency
0
50
100
150
0.00
0.05
0.10
0.15
concave_points_mean



Variables como `concavity_se` tienen muchas observaciones cercanas a cero, lo que puede generar dudas sobre la normalidad o incluso la relevancia de estas variables. Como disponemos de muchas columnas, vamos a eliminar esta columna con histograma muy desplazado a la izquierda.

```
columnas.a.quitar <- c("concavity_se")

datos.num <- datos.num[, !(colnames(datos.num) %in% columnas.a.quitar)]
head(datos.num)
```

```
##   radius_mean texture_mean smoothness_mean compactness_mean concavity_mean
## 1       17.99        10.38         0.11840          0.27760         0.3001
## 2       20.57        17.77         0.08474          0.07864         0.0869
## 3       19.69        21.25         0.10960          0.15990         0.1974
## 4       11.42        20.38         0.14250          0.28390         0.2414
## 5       20.29        14.34         0.10030          0.13280         0.1980
## 6       12.45        15.70         0.12780          0.17000         0.1578
##   concave_points_mean symmetry_mean fractal_dimension_mean radius_se
## 1             0.14710        0.2419                0.07871    1.0950
## 2             0.07017        0.1812                0.05667    0.5435
## 3             0.12790        0.2069                0.05999    0.7456
## 4             0.10520        0.2597                0.09744    0.4956
## 5             0.10430        0.1809                0.05883    0.7572
## 6             0.08089        0.2087                0.07613    0.3345
##   smoothness_se
## 1      0.006399
## 2      0.005225
## 3      0.006150
```

```
## 4      0.009110
## 5      0.011490
## 6      0.007510
```

Seleccionamos la primera columna (`radius_mean`):

```
indice.columna = 1
columna        = datos.num[, indice.columna]
nombre.columna = names(datos.num) [indice.columna]
```

2.1.1 Obtención de los outliers IQR

Calculamos el iqr:

```
# COMPLETAR
cuartil.primero= quantile(columna, 0.25)
cuartil.tercero = quantile(columna, 0.75)
iqr= cuartil.tercero - cuartil.primero
```

Calculamos los extremos que delimitan los outliers:

```
# COMPLETAR
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
```

Identificamos los outliers IQR

```
# COMPLETAR
son.outliers.IQR <- columna<extremo.inferior.outlier.IQR | columna>extremo.superior.outlier.IQR
son.outliers.IQR.extremos <- columna<extremo.inferior.outlier.IQR.extremo | columna>extremo.superior.ou
```

2.1.2 Índices y valores de los outliers IQR

Obtenemos los índices de las filas que contienen un outlier en la columna seleccionada, así como el valor correspondiente en dicha columna.

```
# COMPLETAR
claves.outliers.IQR <- which(son.outliers.IQR==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==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]   1   2   3   5   7  46 310
```

```
df.outliers.IQR
```

```
##     radius_mean texture_mean smoothness_mean compactness_mean concavity_mean
## 1        17.990        10.38         0.11840          0.27760        0.30010
## 2        20.570        17.77         0.08474          0.07864        0.08690
## 3        19.690        21.25         0.10960          0.15990        0.19740
## 5        20.290        14.34         0.10030          0.13280        0.19800
## 7        18.250        19.98         0.09463          0.10900        0.11270
## 46        6.981        13.43         0.11700          0.07568        0.00000
## 310      17.850        13.23         0.07838          0.06217        0.04445
##     concave_points_mean symmetry_mean fractal_dimension_mean radius_se
## 1               0.14710        0.2419                0.07871    1.0950
## 2               0.07017        0.1812                0.05667    0.5435
## 3               0.12790        0.2069                0.05999    0.7456
## 5               0.10430        0.1809                0.05883    0.7572
## 7               0.07400        0.1794                0.05742    0.4467
## 46              0.00000        0.1930                0.07818    0.2241
## 310             0.04178        0.1220                0.05243    0.4834
##     smoothness_se
## 1        0.006399
## 2        0.005225
## 3        0.006150
## 5        0.011490
## 7        0.004314
## 46       0.010190
## 310      0.004369
```

```
nombres.outliers.IQR
```

```
## [1] "1"   "2"   "3"   "5"   "7"   "46"  "310"
```

```
valores.outliers.IQR
```

```
## [1] 17.990 20.570 19.690 20.290 18.250  6.981 17.850
```

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

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

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

```
##   radius_mean texture_mean smoothness_mean compactness_mean concavity_mean
## 2       20.57        17.77         0.08474          0.07864         0.0869
##   concave_points_mean symmetry_mean fractal_dimension_mean radius_se
## 2             0.07017        0.1812                0.05667    0.5435
##   smoothness_se
## 2      0.005225
```

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

```
## [1] "2"
```

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

```
## [1] 20.57
```

Observamos que se detectan **7 outliers normales** (instancias 1,2,3,5,7,46,310) y solo **1 outlier extremo** (instancia 2) según el método IQR.

2.1.3 Cómputo de los outliers IQR con funciones

Usamos las funciones proporcionadas en `OutliersFunciones_byCubero.R`.

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

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

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

```
## [1]   1   2   3   5   7  46 310
```

```
son.outliers.IQR.extremos    = son_outliers_IQR (datos.num, indice.columna, 3)
head(son.outliers.IQR.extremos)
```

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

```
claves.outliers.IQR.extremos = claves_outliers_IQR (datos.num, indice.columna, 3)
claves.outliers.IQR.extremos
```

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

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

Se ve claramente que los datos se encuentran en distintas escalas, por lo que normalizamos los datos con el método z-score, mediante la función `scale`:

```
datos.num.zscore = scale(datos.num)
head(datos.num.zscore)
```

```
##   radius_mean texture_mean smoothness_mean compactness_mean concavity_mean
## 1   2.9405769  -1.89140898       1.8100258        5.0393325      5.0807588
## 2   4.2625437  -0.04125135      -0.5953986       -0.1084622      0.7546289
## 3   3.8116403   0.83000014       1.1811566        1.9940197      2.9968303
## 4  -0.4258270   0.61218727       3.5322696        5.2023357      3.8896526
```

```
## 5   4.1190744  -0.89998487       0.5165563        1.2928474      3.0090052
## 6   0.1019349  -0.55949578       2.4817723        2.2553422      2.1932903
##   concave_points_mean symmetry_mean fractal_dimension_mean radius_se
## 1           6.042488     2.6068463              2.1924204 6.3730316
## 2           2.152516     0.2342760             -0.9041620 1.9981313
## 3           5.071639     1.2388074             -0.4377077 3.6013359
## 4           3.923811     3.3025918              4.8239534 1.6181536
## 5           3.878303     0.2225499             -0.6006857 3.6933556
## 6           2.694574     1.3091637              1.8299348 0.3401909
##   smoothness_se
## 1    -0.2623344
## 2    -0.6489073
## 3    -0.3443247
## 4     0.6303394
## 5     1.4140220
## 6     0.1034939
```

```
#Seleccionamos nuestra columna elegida:
columna.norm   = datos.num.zscore[, indice.columna]
```

Se supone que las instancias deberían tomar valores normales (entre -2 y 2) en todas o casi todas las variables. Sin embargo, esto no ocurre aquí porque estamos mostrando las 6 primeras instancias (head), que casualmente son los outliers del dataset. Por eso vamos a mostrar otras instancias random del dataset:

```
datos.num.zscore[200:205,]
```

```
##     radius_mean texture_mean smoothness_mean compactness_mean concavity_mean
## 200   1.2855564   -0.7998410      -0.3088344       -0.1301960    -0.07366672
## 201  -0.4002075   -0.4768771      -0.1651950       -0.6202403    -0.48375624
## 202  -0.1542602   -0.7447619       1.6242235        0.9771905    -0.77108269
## 203  -0.3028533   -0.2165031      -1.2478503       -1.2039469    -0.84015467
## 204  -0.5692963   -0.9675820      -1.4529474       -0.5741855    -0.09456688
## 205   0.1582979    0.2842162      -0.3910162        0.5425155     1.08132067
##     concave_points_mean symmetry_mean fractal_dimension_mean  radius_se
## 200           0.3883007    -0.9031506            -0.55994121  0.4060325
## 201          -0.2215142    -0.4614695             0.08635094 -0.9591902
## 202          -0.4920374     2.7631936             0.37999237  0.5504082
## 203          -0.8080691    -0.3168482            -0.81424312  0.4901195
## 204          -0.6518230    -0.2425833            -0.31968913  1.0358913
## 205           0.8246774    -0.8562464            -0.17778585  0.5440620
##     smoothness_se
## 200    -0.8112415
## 201     0.6570109
## 202    -0.4595722
## 203    -0.2017472
## 204    -0.5448553
## 205     0.8749932
```

En efecto, aquí sí se cumple que se toman valores entre -2 y 2 para todas las variables.

Mostramos los valores normalizados de los outliers:

```
# COMPLETAR
valores.outliers.IQR.norm <- columna.norm[son.outliers.IQR]
valores.outliers.IQR.norm
```

```
##         1         2         3         5         7        46       310
##  2.940577  4.262544  3.811640  4.119074  3.073798 -2.700327  2.868842
```

Al haber normalizado, un valor de 2.94 o 4.26 nos dice directamente que es un valor inusual.

Mostramos los valores en todas las columnas de los outliers detectados (según la columna 1):

```
# COMPLETAR
datos.num.zscore.outliers.IQR <- datos.num.zscore[son.outliers.IQR, ]
datos.num.zscore.outliers.IQR
```

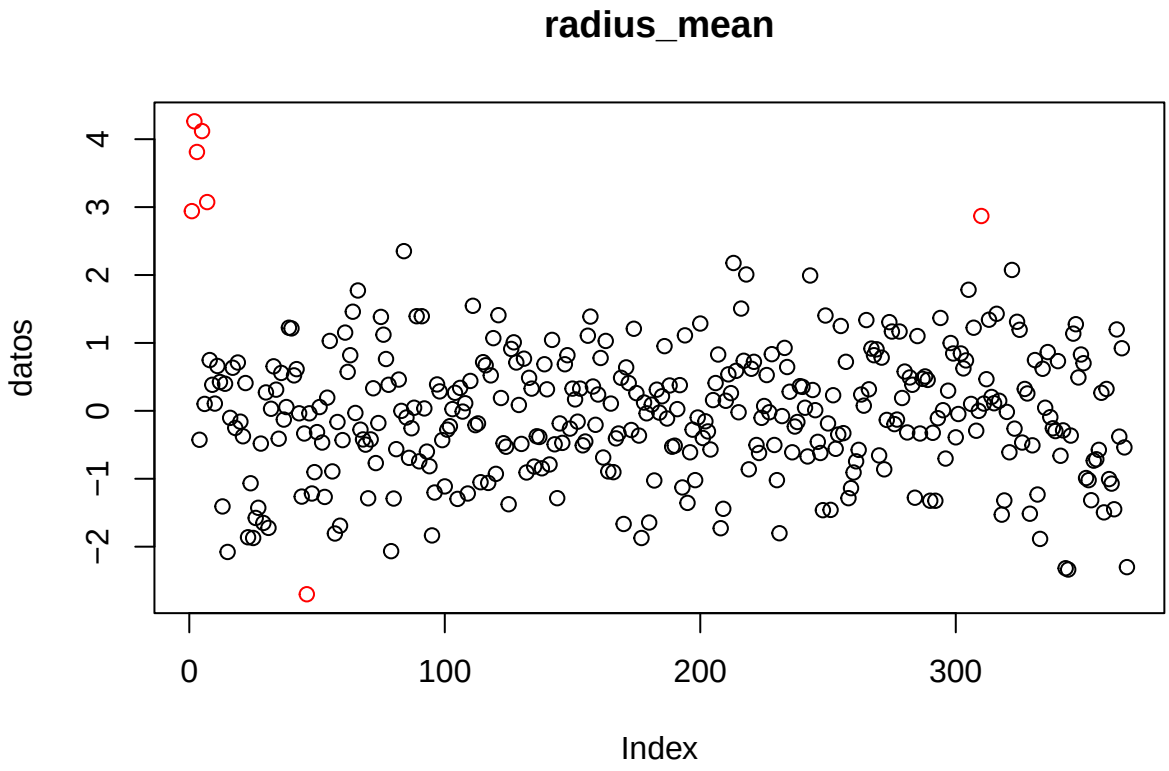
```
##     radius_mean texture_mean smoothness_mean compactness_mean concavity_mean
## 1      2.940577  -1.89140898       1.8100258        5.0393325      5.0807588
## 2      4.262544  -0.04125135      -0.5953986       -0.1084622      0.7546289
## 3      3.811640   0.83000014       1.1811566        1.9940197      2.9968303
## 5      4.119074  -0.89998487       0.5165563        1.2928474      3.0090052
## 7      3.073798   0.51204342       0.1113645        0.6770577      1.2781474
## 46    -2.700327  -1.12781213       1.7099784       -0.1850478     -1.0086952
## 310    2.868842  -1.17788405      -1.0498995       -0.5345990     -0.1067417
##     concave_points_mean symmetry_mean fractal_dimension_mean  radius_se
## 1             6.0424881     2.6068463              2.1924204  6.3730316
## 2             2.1525164     0.2342760             -0.9041620  1.9981313
## 3             5.0716386     1.2388074             -0.4377077  3.6013359
## 5             3.8783026     0.2225499             -0.6006857  3.6933556
## 7             2.3461807     0.1639197             -0.7987883  1.2302431
## 46           -1.3956354     0.6955005              2.1179563 -0.5355824
## 310           0.7169737    -2.0796641             -1.4998748  1.5213743
##     smoothness_se
## 1      -0.2623344
## 2      -0.6489073
## 3      -0.3443247
## 5       1.4140220
## 7      -0.9488799
## 46      0.9859601
## 310    -0.9307696
```

Observamos fácilmente que además de en la primera columna, los outliers tienen valores extremos en otras columnas.

2.1.5 Gráfico

Mostramos los valores y los outliers de la columna 1 (ya normalizada):

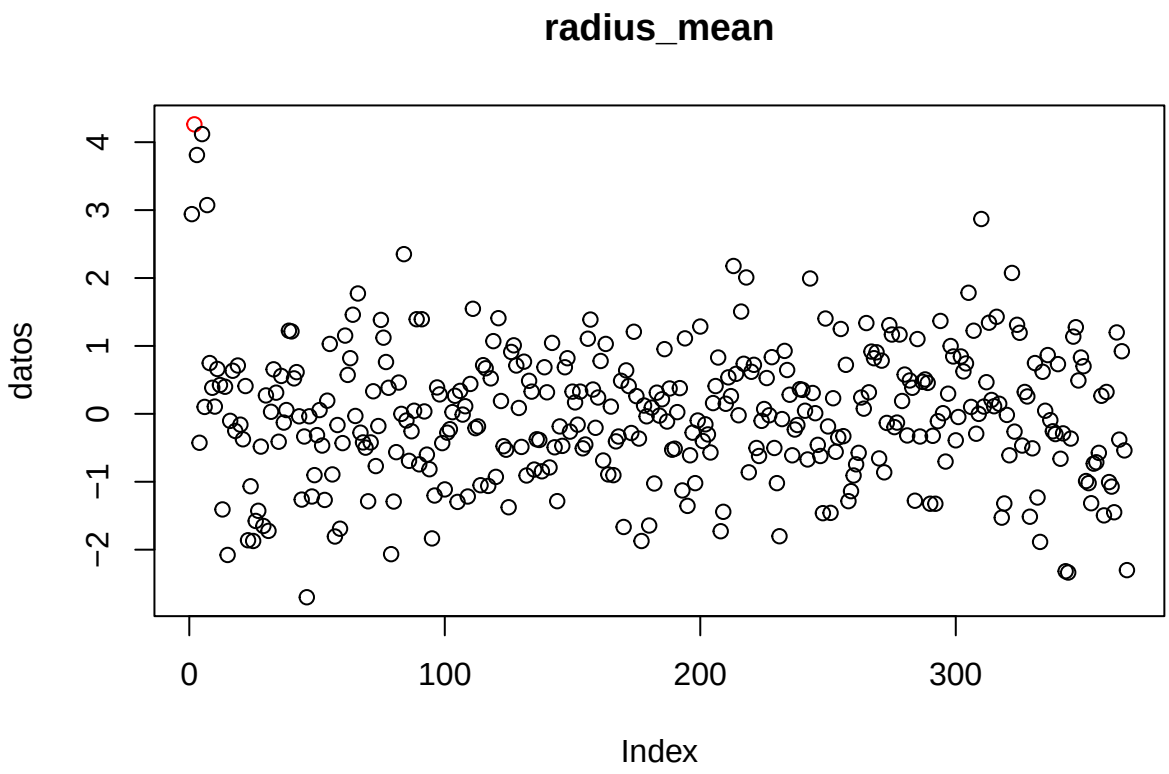
```
# COMPLETAR
plot_2_colores(columna.norm, claves.outliers.IQR, nombre.columna)
```



Vemos que hay 5 outliers que están muy claros (casualmente corresponde con las instancias numero 1,2,3,5 y 7), pues se encuentran todos juntos en una región aislada.

Representamos los outliers extremos:

```
# COMPLETAR
plot_2_colores(columna.norm, claves.outliers.IQR.extremos, nombre.columna)
```

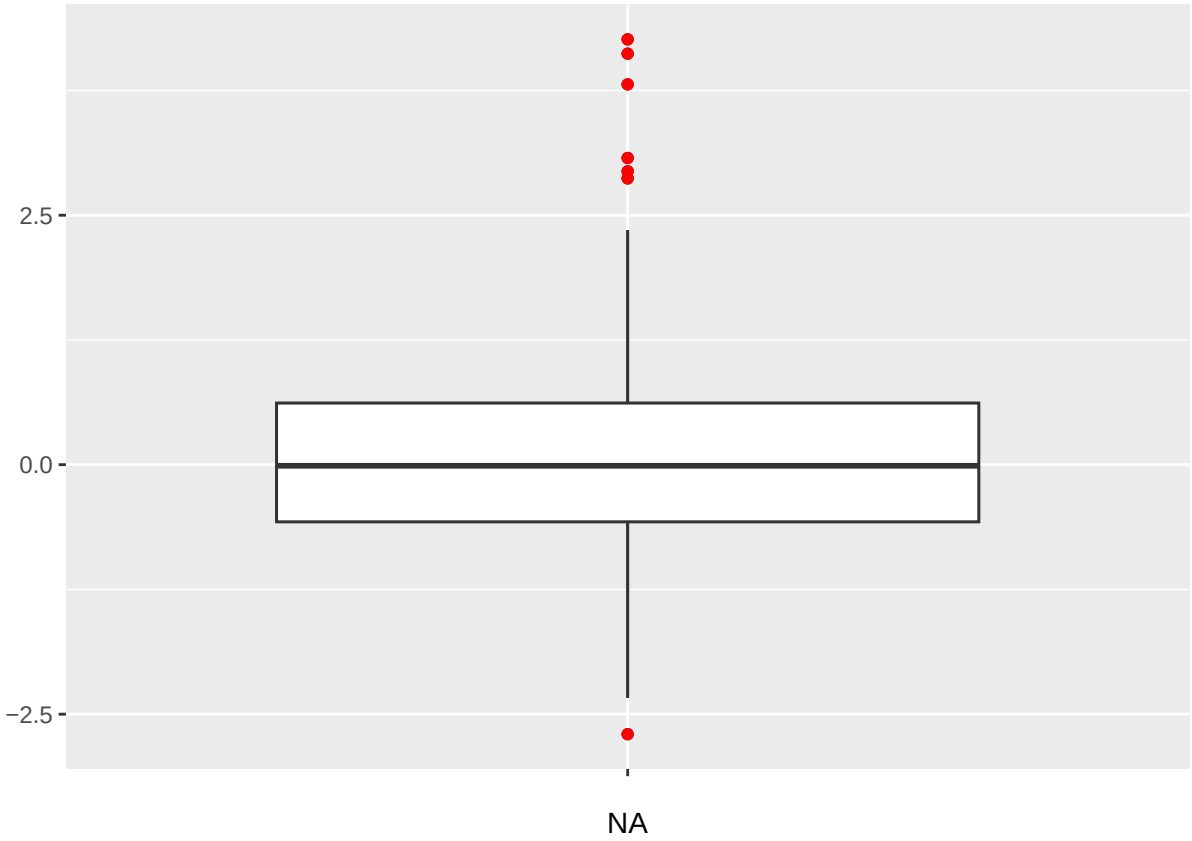


Solo hay un outlier extremo (instancia 2), que es el más alejado de los datos de la columna 'radius_mean".

2.1.6 Diagramas de cajas

Boxplot donde mostramos los outliers con puntos rojos:

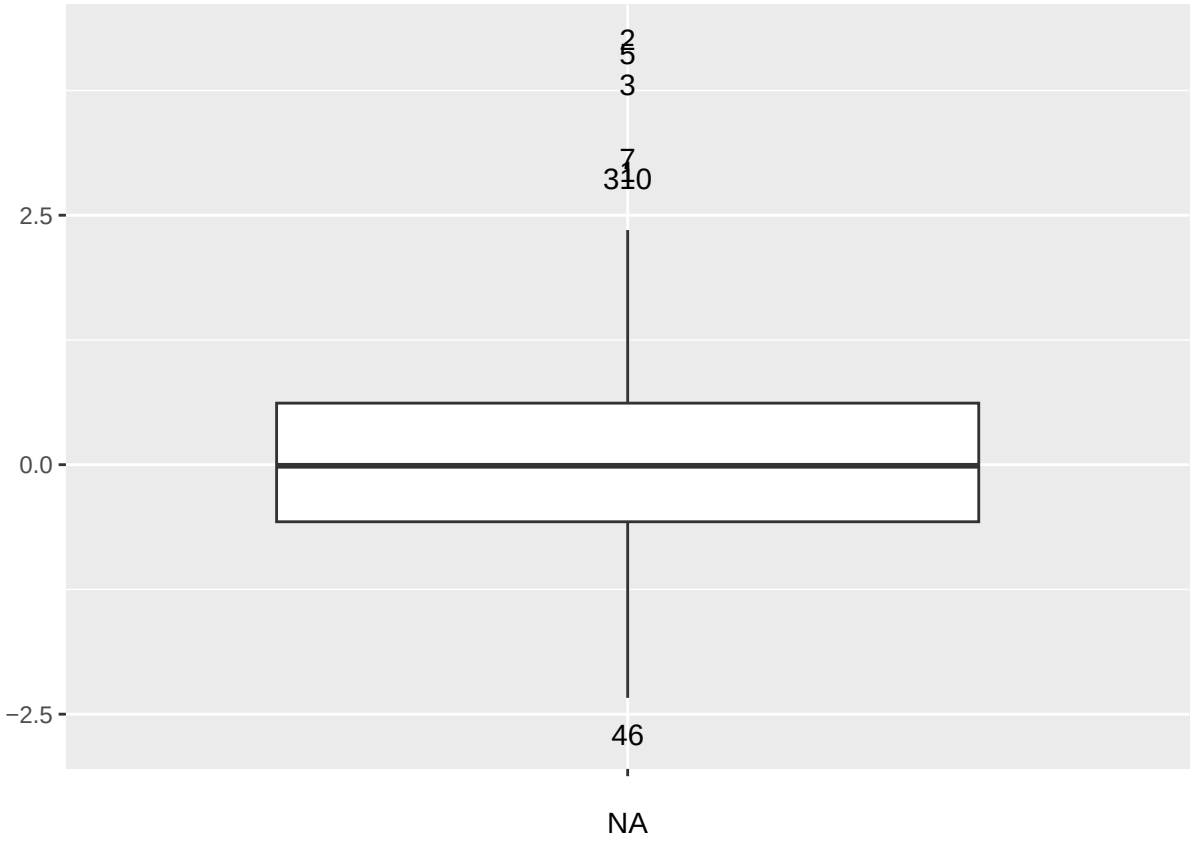
```
# COMPLETAR
diag_caja_outliers_IQR(datos.num.zscore, nombre.columna)
```



Se ven 6 outliers por arriba (valores extremos superiores) y 1 outlier por debajo (valor extremo inferior).

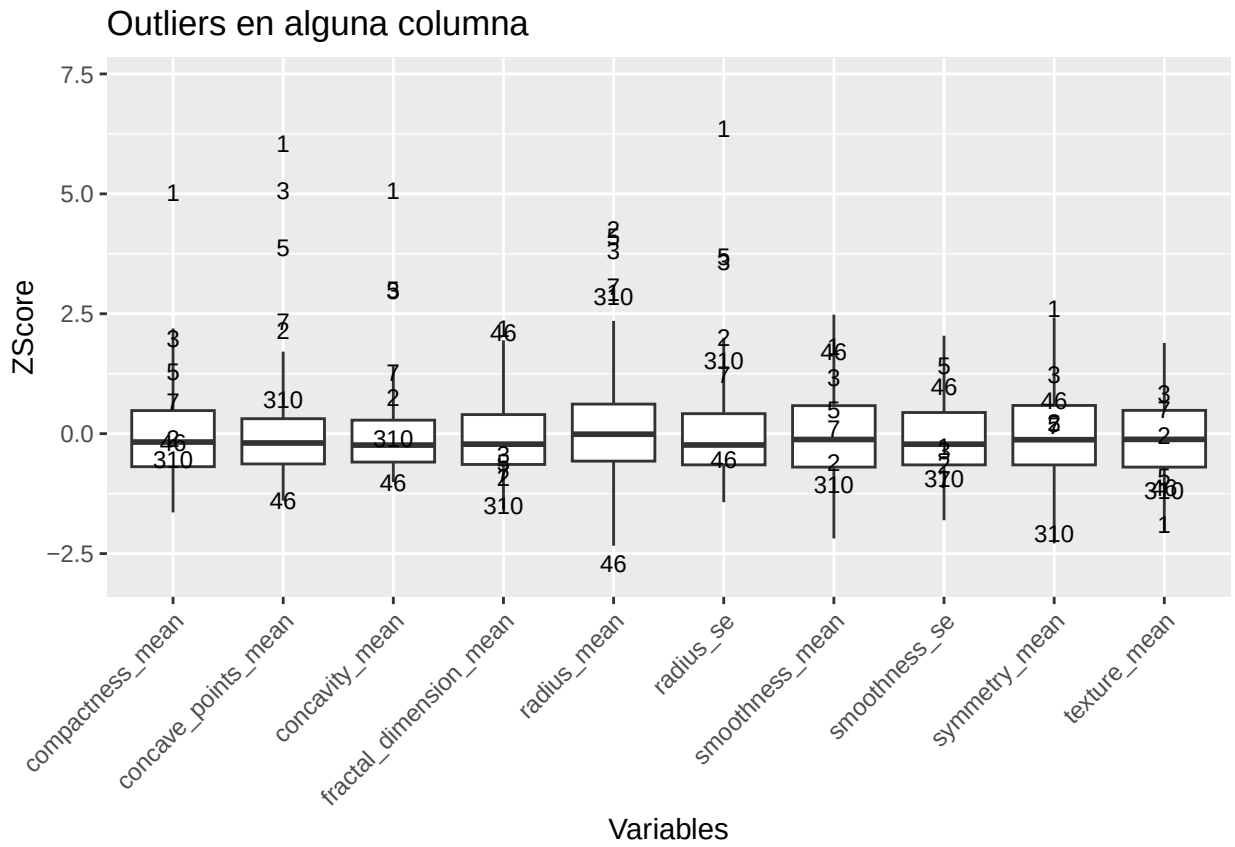
Para mostrar las etiquetas de los outliers (en este caso, las etiquetas serán el número de instancia):

```
# COMPLETAR
diag_caja(datos.num.zscore, nombre.columna, claves.outliers.IQR)
```



Vemos los valores que una instancia outlier toma en las otras columnas:

```
# COMPLETAR
diag_cajas <- diag_caja_juntos(datos.num.zscore, titulo = "Outliers en alguna columna", claves.outliers
diag_cajas + theme(axis.text.x = element_text(angle = 45, hjust = 1)) #Ponemos los nombres inclinados p
```



Vemos que, por ejemplo, la instancia 1 no solo toma un valor anormal (alto) en la columna 1, sino en la mayoría de columnas. Lo mismo ocurre con las instancias 3 y 5, e incluso con la 2. Con la instancia 46 ocurre también pero para valores muy bajos.

2.2 Test de hipótesis (opcional)

2.2.1 Test de Grubbs

Suponemos H0:El valor más alejado de la media NO es un outlier.

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

```
## [1] 0.005881068
```

El p-value es < 0.005, por lo que el test rechaza la hipótesis nula. Así pues, podemos concluir que el valor más alejado de la media (creo que era la instancia 2) es un outlier desde el punto de vista estadístico.

Para obtener el valor que toma el outlier:

```
valor.posible.outlier = outlier(columna)
valor.posible.outlier
```

```
## [1] 20.57
```

Clave del posible outlier:

```
es.posible.outlier = outlier(columna, logical = TRUE)
clave.posible.outlier = which(es.posible.outlier == TRUE)
clave.posible.outlier
```

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

Por tanto la instancia 2 es un outlier (siempre y cuando se cumpla la condición de normalidad, que veremos ahora).

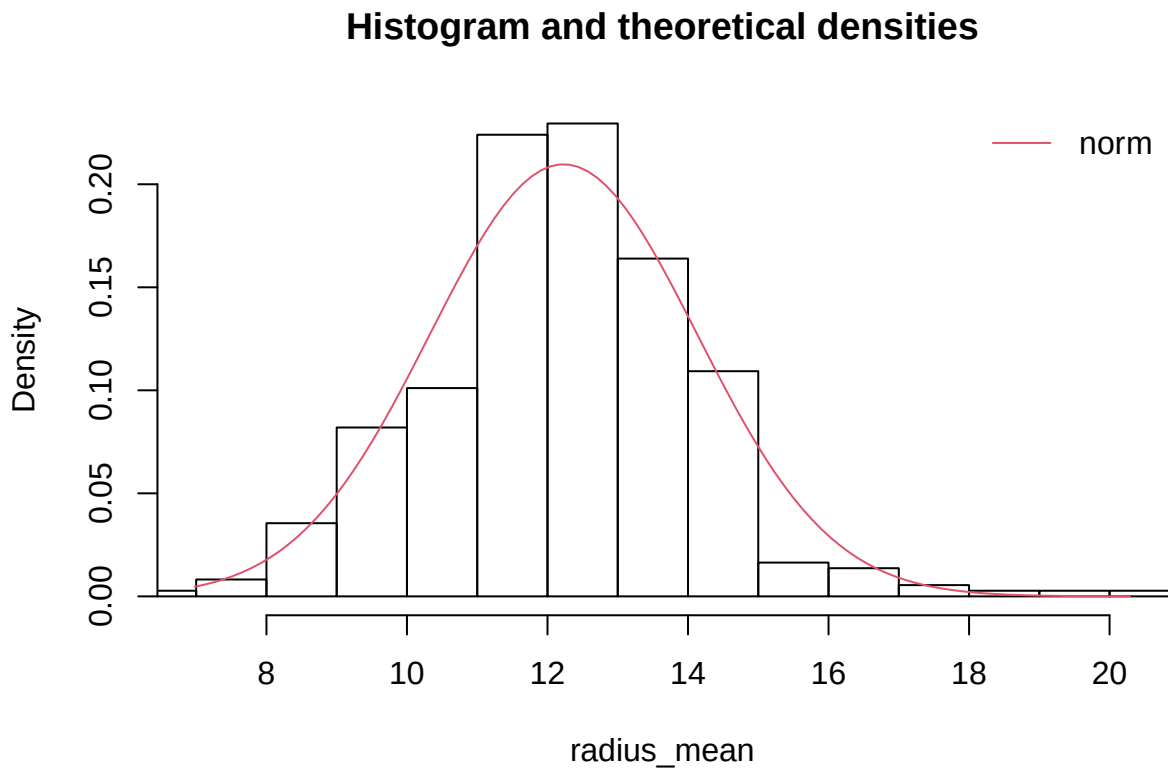
2.2.2 Comprobación de la hipótesis de Normalidad

Para aplicar el test hay que garantizar que se cumple la **condición de normalidad**. La hipótesis de normalidad se aplica sobre el dataset sin el outlier.

```
datos.num.sin.outlier = datos.num[-clave.posible.outlier,]
columna.sin.outlier = datos.num.sin.outlier[,indice.columna]
#columna.sin.outlier
```

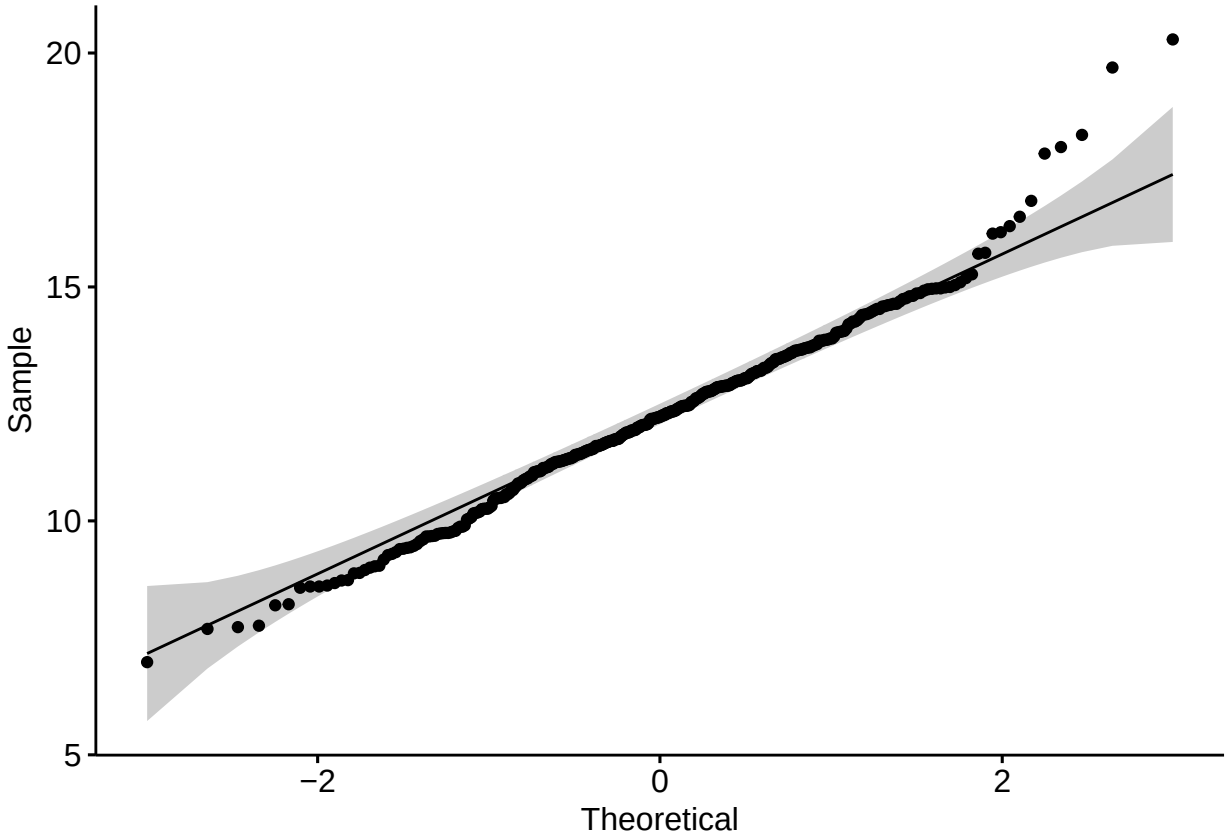
- **Forma 1: Visualizar histograma**

```
ajusteNormal = fitdist(columna.sin.outlier , "norm")
denscomp (ajusteNormal,  xlab = nombre.columna)
```



- **Forma 2: Visualizar el gráfico QQ**

```
ggqqplot(columna.sin.outlier)
```



Con los dos gráficos anteriores observamos que, efectivamente, podemos aceptar que la distribución subyacente es una normal. Por tanto, como el test de Grubbs ha rechazado H0, aceptamos que el valor más alejado de la media (instancia 2) es un outlier.

- **Forma 3: Aplicando un test de hipótesis** Test de hipótesis con H0: La distribución subyacente de la variable es una Normal. Si rechazamos H0, los datos no vienen de una normal. Aplicamos test de Shapiro-Wilk:

```
shapiro.test(columna.sin.outlier)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  columna.sin.outlier
## W = 0.98311, p-value = 0.000276
```

Como p<0.05, rechazamos H0 (columna NO compatible con una distrib normal, lo cual me resulta bastante curioso porque a simple vista sí lo parece...).

2.2.3 Construcción de una función propia

Construimos una función con el nombre test_Grubbs que devuelva una lista con los cómputos anteriores.

```r
# 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 que está más alejado de la media
# valor.mas.alejado.media: Valor de dicho elemento
# nombre.mas.alejado.media: Nombre de la fila del elemento más alejado
# es.outlier: TRUE/FALSE dependiendo del resultado del test de Grubbs
# p.value: p-value calculado por el test de Grubbs
# es.distrib.norm: Resultado del test de normalidad Shapiro-Wilk
# p.value.test.normalidad: p-value del test de normalidad

# Requiere el paquete "outliers" (library(outliers))
#############################################################################

test_Grubbs <- function(data.frame, indice.columna, alpha = 0.05) {
  # Extraer la columna correspondiente
  columna <- data.frame[, indice.columna]

  # Nombre de la columna
  nombre.columna <- colnames(data.frame)[indice.columna]

  # Clave del valor más alejado de la media
  clave.mas.alejado.media <- which.max(abs(columna - mean(columna)))

  # Valor del elemento más alejado de la media
  valor.mas.alejado.media <- columna[clave.mas.alejado.media]

  # Nombre de la fila correspondiente al valor más alejado de la media
  nombre.mas.alejado.media <- rownames(data.frame)[clave.mas.alejado.media]

  # Aplicar el test de Grubbs
  test.Grubbs <- grubbs.test(columna, two.sided = TRUE)
  es.outlier <- test.Grubbs$p.value < alpha

  # Eliminar el posible outlier de la columna
  columna.sin.outlier <- columna[-clave.mas.alejado.media]

  # Test de normalidad (Shapiro-Wilk) sin el outlier
  shapiro.result <- shapiro.test(columna.sin.outlier)
  p.value.test.normalidad <- shapiro.result$p.value
  es.distrib.norm <- p.value.test.normalidad > alpha

  # Devolver resultados en forma de lista
  return(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 = test.Grubbs$p.value,
```

```
    p.value.test.normalidad = p.value.test.normalidad,
    es.distrib.norm = es.distrib.norm
  ))
}
```

```
# Ejemplo de uso con datos.num y columna índice 1 ("mpg")
test.Grubbs.con.funcion <- test_Grubbs(datos.num, indice.columna)
test.Grubbs.con.funcion
```

```
## $nombre.columna
## [1] "radius_mean"
##
## $clave.mas.alejado.media
## [1] 2
##
## $valor.mas.alejado.media
## [1] 20.57
##
## $nombre.mas.alejado.media
## [1] "2"
##
## $es.outlier
## [1] TRUE
##
## $p.value
## [1] 0.005881068
##
## $p.value.test.normalidad
## [1] 0.000276
##
## $es.distrib.norm
## [1] FALSE
```

Como no tenemos suficiente evidencia estadística de que la variable siga una distribución normal, no se cumple la condición para aplicar el Test de Grubbs y por tanto no podemos afirmar que la instancia 2 sea un outlier.

2.3 Trabajando con varias columnas

Aplicamos lo anterior al resto de columnas.

2.3.1 Outliers IQR

Calculamos los outliers IQR con respecto a cada una de las columnas (puede haber instancias repetidas).

```
claves.outliers.IQR.en.alguna.columna =
  claves_outliers_IQR_en_alguna_columna(datos.num, 1.5)

claves.outliers.IQR.en.alguna.columna
```

```
##   [1]   1   2   3   5   7  46 310 123 124 127 220 277 280 281 284 285 293 295
##  [19] 348 357 359 360 362 365 366   4  33 318 329 367   1   4   6   9  10  55
##  [37]  64 180 255 318 319   1   3   4   5   6   9  10  29  55  79  80  96 130
##  [55] 134 156 180 219 255 306   1   2   3   4   5   6   7   9  10  33  36  39
##  [73]  55  64  80 145 156 255 278 305 319 336   1   4  24  33  78  80 154 202
##  [91] 258 318 366   1   4  10  29  31  46  55  56  74  79  80  96 180 219 318
## [109] 319 321 344   1   3   5   8  80 104 145 149 156 255 258 293 336  31  53
## [127]  59  93 100 132 144 145 154 177 198 231 251 291 319 329 344
```

Para devolver las intancias outlier sin duplicar:

```
claves.outliers.IQR.en.mas.de.una.columna =
  unique(
    claves.outliers.IQR.en.alguna.columna[
      duplicated(claves.outliers.IQR.en.alguna.columna)])
claves.outliers.IQR.en.alguna.columna =
  unique (claves.outliers.IQR.en.alguna.columna)

claves.outliers.IQR.en.mas.de.una.columna
```

```
##  [1]   1   4 318   3   5   6   9  10  55 180 255   2   7  33  64  80 156 319 366
## [20]  29  46  79  96 219 145 258 293 336  31 154 329 344
```

```
claves.outliers.IQR.en.alguna.columna
```

```
##  [1]   1   2   3   5   7  46 310 123 124 127 220 277 280 281 284 285 293 295 348
## [20] 357 359 360 362 365 366   4  33 318 329 367   6   9  10  55  64 180 255 319
## [39]  29  79  80  96 130 134 156 219 306  36  39 145 278 305 336  24  78 154 202
## [58] 258  31  56  74 321 344   8 104 149  53  59  93 100 132 144 177 198 231 251
## [77] 291
```

```
nombres_filas(datos.num, claves.outliers.IQR.en.mas.de.una.columna)
```

```
##  [1] "1"   "4"   "318" "3"   "5"   "6"   "9"   "10"  "55"  "180" "255" "2"
## [13] "7"   "33"  "64"  "80"  "156" "319" "366" "29"  "46"  "79"  "96"  "219"
## [25] "145" "258" "293" "336" "31"  "154" "329" "344"
```

```
nombres_filas(datos.num, claves.outliers.IQR.en.alguna.columna)
```

```
##  [1] "1"   "2"   "3"   "5"   "7"   "46"  "310" "123" "124" "127" "220" "277"
## [13] "280" "281" "284" "285" "293" "295" "348" "357" "359" "360" "362" "365"
## [25] "366" "4"   "33"  "318" "329" "367" "6"   "9"   "10"  "55"  "64"  "180"
## [37] "255" "319" "29"  "79"  "80"  "96"  "130" "134" "156" "219" "306" "36"
## [49] "39"  "145" "278" "305" "336" "24"  "78"  "154" "202" "258" "31"  "56"
## [61] "74"  "321" "344" "8"   "104" "149" "53"  "59"  "93"  "100" "132" "144"
## [73] "177" "198" "231" "251" "291"
```

Mostramos los valores normalizados de estos outlier:

```r
# COMPLETAR
as.data.frame(datos.num.zscore[claves.outliers.IQR.en.alguna.columna,])
```

```
##      radius_mean texture_mean smoothness_mean compactness_mean concavity_mean
## 1    2.940576866  -1.89140898      1.81002576       5.03933253     5.08075875
## 2    4.262543683  -0.04125135     -0.59539862      -0.10846220     0.75462887
## 3    3.811640273   0.83000014      1.18115664       1.99401970     2.99683034
## 5    4.119074416  -0.89998487      0.51655632       1.29284742     3.00900519
## 7    3.073798328   0.51204342      0.11136451       0.67705774     1.27814741
## 46  -2.700327275  -1.12781213      1.70997840      -0.18504781    -1.00869517
## 310  2.868842233  -1.17788405     -1.04989949      -0.53459901    -0.10674173
## 123 -0.477066028   2.28959676     -1.73236541      -1.15659833    -0.67733635
## 124 -0.528305052   3.97451704     -1.09134768      -1.21843604    -0.90790771
## 127  1.008865665   2.48237367     -0.77476925       0.54510281     1.22944801
## 220  0.619449083   2.57250314     -1.27929379      -0.92037313    -0.75079128
## 277 -0.128640665   2.57751033     -0.62183971      -0.47431372    -0.18587826
## 280  0.578457864   3.20090580     -0.04442352      -0.22178820    -0.43668016
## 281 -0.318225054   2.84289153      0.03561437       0.07523976     0.44417021
## 284 -1.278956752   2.56999954     -0.94556438      -0.94624665    -0.69600445
## 285  1.101095908   2.26706439     -1.65232752      -0.79178176    -0.30336555
## 293 -0.108145056   2.55497796     -0.39673315      -0.59074454    -0.52839736
## 295  0.009704699   3.01313608     -1.14923222      -1.26397342    -1.00869517
## 348  0.491351524   2.53494919     -0.45461770      -0.36383380    -0.40259058
## 357  0.265899819   2.87794188     -0.73689418      -1.04767083    -0.60347560
## 359  0.322262745   2.75025847     -0.84623165      -0.63628193     0.24835805
## 360 -1.004827974   2.42228736     -0.19806768      -0.16176164     0.20858687
## 362 -1.449070311   2.48988446     -0.84623165      -0.85698302    -1.00869517
## 365  0.921759324   2.30712193      0.44437930       0.77020240    -0.10329219
## 366 -0.538552857   2.86292030     -1.32788822      -1.22257580    -1.00869517
## 4   -0.425827004   0.61218727      3.53226960       5.20233568     3.88965264
## 33   0.655316400  -1.75120759      2.57467344       0.56580162     0.38674550
## 318 -1.528490798  -1.26801352      5.02583376       3.64992473     0.96565960
## 329 -1.514656262  -1.01014310      3.14637264       1.02635021    -0.33258519
## 367 -2.301175279   1.65368331     -2.89005629      -1.01455273    -1.00869517
## 6    0.101934942  -0.55949578      2.48177232       2.25534222     2.19329026
## 9    0.383749574   0.97270513      2.44604112       2.85560779     2.76347905
## 10   0.107058844   1.52850350      1.82431824       4.05613893     3.60354367
## 55   1.029361275   0.42942474     -1.05061411       3.63440062     5.08481704
## 64   1.459769075  -0.38674764      1.56705360       2.53218884     1.30046796
## 180 -1.644290992   0.24165502      0.47224963       2.95910185     2.99885948
## 255  1.249689077  -0.99011433      0.71665104       2.60722203     1.93356014
## 319 -1.319435581  -1.20041642      2.31740880       3.55936742     1.40192504
## 29  -1.650952066  -0.15140959      0.96676944       1.51277231     5.34251802
## 79  -2.065988159   0.69230235      0.06991632       1.23333834     1.67180087
## 80  -1.291254118  -0.64962524      1.00964688       1.99401970     7.32701850
## 96  -1.202610606   0.03135294      0.28001578       1.20487747     1.64339289
## 130 -0.487313833   0.06389969      0.20355101       1.28508537     2.13241602
## 134  0.327386647  -0.95756758     -0.39101616       1.33941975     1.77934538
## 156  1.106219811   0.44945351     -0.39316003       2.19324578     1.75499568
## 219 -0.861358707   0.57212973     -0.16733885       2.15184815     3.61774766
## 306  0.101934942  -0.38174044      0.14781034       1.76633276     2.12429945
## 36   0.557962255  -0.51943824      1.05252432       1.82842920     1.36337135
## 39   1.224069566  -0.67466121      1.43842128       1.32130829     1.01354734
```

```
## 145 -0.185003592  -0.14389880      2.10302160      -0.27767500     0.19438288
## 278  1.167706639  -0.99011433      1.20259536       0.24807485     0.39039796
## 305  1.782574926  -1.66608532      0.80240592       1.21781423     1.40801247
## 336  0.865396398  -1.19290563      2.26738512       0.38080598     1.04073784
## 24  -1.066314803  -0.76479067      1.45271376      -0.05749138    -0.78873622
## 78   0.383749574   0.71233112      1.45986000      -0.17961437    -0.37235637
## 154 -0.507809442   0.50703623     -0.91983792       0.91250674     0.87313074
## 202 -0.154260177  -0.74476190      1.62422352       0.97719053    -0.77108269
## 258 -1.285617825   0.29673414      1.03108560       0.01288459    -0.82741166
## 31  -1.723199089  -0.82487698      0.34004419       1.81807980     0.73758408
## 56  -0.892102122   0.56461894      1.36695888       1.22816363     0.30537692
## 74  -0.179879689  -0.82237338      1.58134608       1.21005217    -0.25588364
## 321 -0.610287490  -0.20398511      1.88148816       0.62789806    -0.18425494
## 344 -2.336530205   1.87900697     -0.45676157       0.95907907     0.86866663
## 8    0.747546643   0.72484910      1.84575696       2.11303788     0.89179885
## 104  0.265899819   0.87005768     -0.39887702      -0.43524471    -0.37722631
## 149 -0.261862128  -0.98009995     -1.06776509      -1.02024490    -0.55315288
## 53  -1.267684167  -0.23653186      0.75952848       0.03125478    -0.12926520
## 59  -1.691430894  -0.54447420      0.11064989       1.07292254     0.87089869
## 93  -0.600039685  -0.80735181      0.53799504      -0.65723948    -0.52920901
## 100 -1.112429924  -0.70720796     -0.02870179      -0.92839392    -0.97628978
## 132 -0.907473829   0.48200026      0.99535440      -0.59824786    -0.02841687
## 144 -1.285617825  -0.56700657     -0.19306531      -0.92994633    -0.78488085
## 177 -1.872304649   0.16654714      1.02393936      -0.63033102    -1.00869517
## 198 -1.020199681  -0.80735181      0.41079197       0.22659983    -0.28205957
## 231 -1.802107186  -0.27408580      0.77382096      -0.22127073    -1.00869517
## 251 -1.458293336   0.94266197      0.80955216      -0.54960565    -0.59332989
## 291 -0.323348956   0.06139609      1.74570960       1.69388692     1.06102925
##     concave_points_mean symmetry_mean fractal_dimension_mean   radius_se
## 1           6.042488133    2.60684632            2.192420366  6.37303160
## 2           2.152516398    0.23427600           -0.904162031  1.99813133
## 3           5.071638564    1.23880742           -0.437707696  3.60133594
## 5           3.878302636    0.22254995           -0.600685717  3.69335561
## 7           2.346180661    0.16391971           -0.798788311  1.23024313
## 46         -1.395635384    0.69550054            2.117956270 -0.53558244
## 310         0.716973729   -2.07966409           -1.499874797  1.52137429
## 123        -1.237619504    0.45707090           -0.958956366 -1.35344684
## 124        -1.070299648    0.36326252           -0.677959778 -0.53716899
## 127         0.833273417   -1.60280482           -0.255059914  0.34733034
## 220        -0.830823422   -1.29401889           -0.769283669 -0.97346910
## 277        -0.412144545   -0.53573447           -0.290184487 -0.79656924
## 280         0.254808882   -1.47381829           -0.413822986  0.39016708
## 281        -0.375737686    0.18346312           -0.203075545  0.17360357
## 284        -0.868241582   -0.51228238           -0.503741894 -0.90048799
## 285        -0.517319915   -0.17613567           -1.232928039 -0.47688024
## 293        -0.193703392    0.39844066           -0.860607560  2.49472019
## 295        -1.395635384   -0.19958777           -0.492502030  1.22072385
## 348         0.260371041   -0.48492160           -0.743993976 -0.44832242
## 357        -0.637664810   -0.83279435           -0.946311520 -0.40231259
## 359        -0.210895520   -0.72725992           -0.846557731 -0.63474156
## 360        -0.011163447   -0.62172549           -0.257869880 -0.56889990
## 362        -1.395635384   -0.03933178           -0.353408719  1.95053496
## 365         0.780685732   -0.84061171           -0.196050630  0.57817277
## 366        -1.395635384   -2.70505330           -1.135984216  0.17836321
```

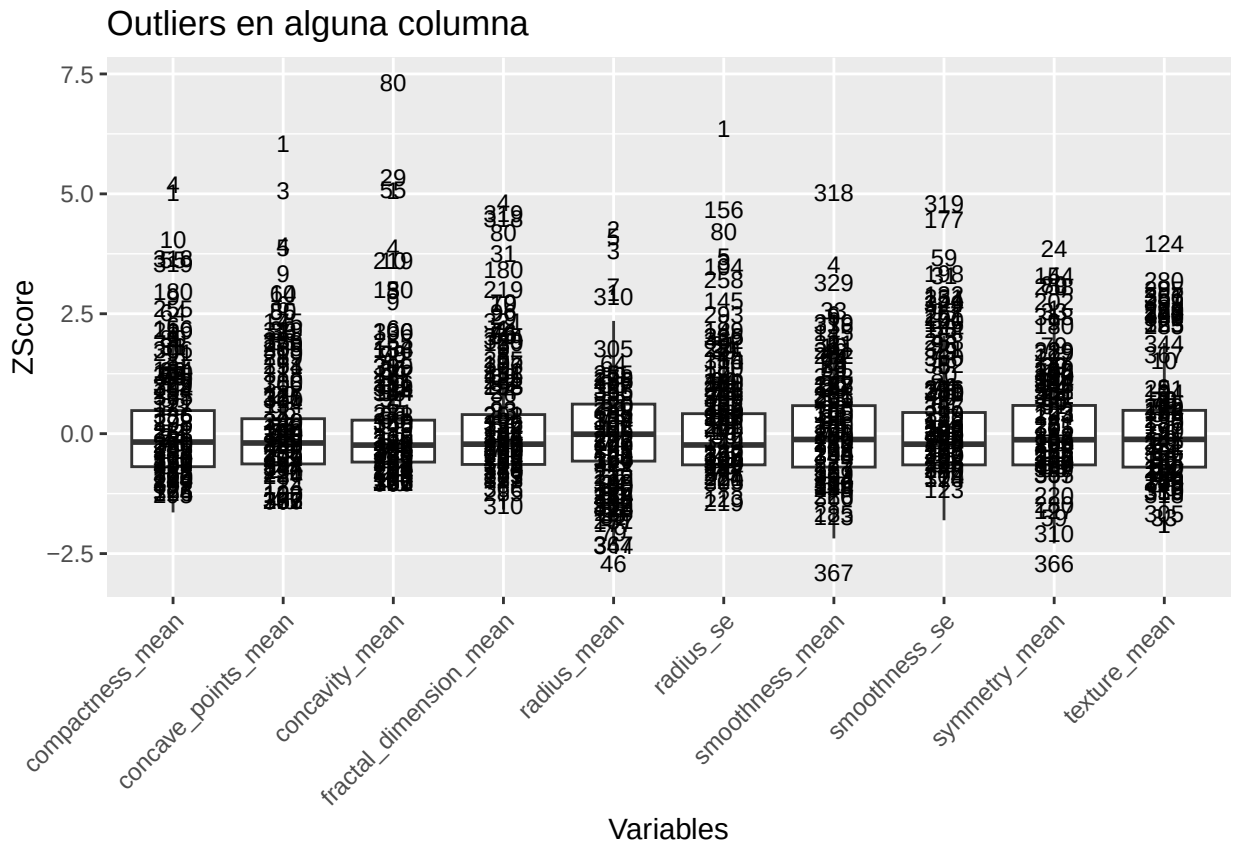
```
## 4          3.923811210    3.30259182          4.823953410  1.61815359
## 33         1.919411372    2.54430740          0.464291351  0.93990523
## 318        1.260042706    2.44659034          4.483947539  0.92007341
## 329       -0.171454756    1.73911879          1.946548352  0.49329256
## 367       -1.395635384   -0.64517758         -0.599280734  0.74634663
## 6          2.694574074    1.30916371          1.829934768  0.34019089
## 9          3.333716706    2.33714722          1.515218589  0.11648792
## 10         2.924139545    1.08636880          2.715074019  0.04747317
## 55         2.547429686   -0.18786172          2.049112106  0.56468713
## 64         2.919588687    0.97301700          0.219824320  1.10490600
## 180        1.086098825    2.25897357          3.417565489  1.37779189
## 255        1.789964762    1.30525502          1.539103299  2.01954970
## 319        2.163135065    1.19190323          4.586511294 -0.13656615
## 29         0.816586940    1.40297209          2.438292380  0.28386851
## 79        -0.299384413    1.83683585          2.740363712 -0.77832396
## 80         2.577263084    3.11106638          4.194521054  4.22722829
## 96         0.483363052   -0.32466561          2.536641186  1.10649255
## 130        0.047492047    1.18017718          1.908613812 -0.38724040
## 134        0.616854867   -0.60999944          0.138335309 -0.70692940
## 156        1.942671309   -0.14877490          1.238436950  4.67622078
## 219        1.608436119    1.70394065          3.005905487 -1.42880777
## 306        1.054748475    1.28962029          1.425299681  0.79711610
## 36         2.137346874    0.74240473          0.830991898 -0.04454649
## 39         2.176281987    1.42251550          0.049821384  1.74428399
## 145        2.348203264    1.02773856         -0.611925580  2.77315899
## 278        1.888566672   -0.39893058         -0.266299777  0.11410810
## 305        1.744961840    0.12483288         -0.072412132 -1.02027221
## 336        1.949244770    0.87529994          0.205774490  2.01875643
## 24        -0.743345830    3.87325948          0.912480909  1.77839473
## 78        -0.058188973    3.07979691         -0.314069197  1.02002580
## 154        1.429941381    3.29477446         -0.108941688  1.54675903
## 202       -0.492037374    2.76319363          0.379992375  0.55040821
## 258       -0.401020227    3.07197955          1.009424731  3.21183893
## 31         0.056593762    0.58605743          3.750546445  1.86089511
## 56         0.155701322    0.66423108          2.067376884  0.33305143
## 74         0.122834019    0.94956491          2.147460912  0.82964029
## 321        0.762482303    0.78930892          2.339943574 -0.90207454
## 344       -0.705927670    1.11372958          2.023822413 -0.57127972
## 8          1.630684755    1.73521011          1.602327532  2.31544051
## 104        0.052548556   -0.22694855         -0.033072609  3.48631137
## 149        0.001477823    1.36388526         -0.277539641  2.14409355
## 53        -0.496588232   -0.65690363          1.060004117  0.88358285
## 59        -0.228593299   -1.74742607          1.197692445  0.17122375
## 93        -0.089539324   -0.72725992          0.503630873 -0.66885229
## 100       -1.274076928   -0.19177040         -0.368863532  1.05413653
## 132        0.156712624   -0.05887520          0.181889780  0.63687497
## 144       -0.684184685    1.28571161          0.002051964 -0.18416253
## 177       -1.395635384    1.60622358          1.473069101  0.35843616
## 198       -0.365624670   -0.46537819          0.975705141  0.36795544
## 231       -1.395635384    0.91047808          1.106368554  1.78712073
## 251       -0.760032307    1.06682539          0.408092034  1.09935309
## 291        1.417805761    0.80103497          1.326950876  0.93990523
##     smoothness_se
## 1    -0.262334411
```

```
## 2    -0.648907273
## 3    -0.344324736
## 5     1.414022013
## 7    -0.948879911
## 46    0.985960070
## 310  -0.930769598
## 123  -1.166532945
## 124  -0.934062382
## 127  -0.292628025
## 220  -0.742093065
## 277   0.045540910
## 280  -0.446071768
## 281   0.877627471
## 284  -0.200100790
## 285  -0.605771800
## 293   0.105798860
## 295   0.047187302
## 348  -0.731885434
## 357   0.539787815
## 359   0.400503044
## 360   1.598417927
## 362   1.446949855
## 365   0.019857193
## 366   0.131153298
## 4     0.630339379
## 33    2.042943791
## 318   0.851943754
## 329   2.721257331
## 367  -0.002204461
## 6     0.103493911
## 9    -0.482292393
## 10   -0.015375597
## 55   -0.624869948
## 64    0.596094424
## 180   0.872359016
## 255   0.915494489
## 319   4.799004145
## 29    0.774892605
## 79    1.723543726
## 80    1.242797236
## 96    1.865133445
## 130  -0.175404908
## 134  -0.558026429
## 156   0.186472073
## 219   0.429150266
## 306   0.012613068
## 36   -0.132269436
## 39   -0.555721481
## 145   2.299780956
## 278  -0.349922470
## 305  -0.382521033
## 336   0.936568308
## 24    0.104811025
## 78   -0.058840349
```

```
## 154   2.813455288
## 202  -0.459572183
## 258   1.934281913
## 31    3.297494561
## 56    1.542440596
## 74    0.979374502
## 321   1.786106625
## 344   2.724550115
## 8     0.529909462
## 104  -0.864255358
## 149   0.276035802
## 53    2.191119078
## 59    3.672871957
## 93    2.556618122
## 100   2.368929424
## 132   2.912238813
## 144   2.174655158
## 177   4.463140159
## 198   3.346886324
## 231   2.839797561
## 251   2.484176870
## 291   2.434785107
```

Para ver esta información de forma gráfica, usamos la función `diag_caja_puntos`:

```
# COMPLETAR
diag_cajas <- diag_caja_juntos(datos.num.zscore, titulo = "Outliers en alguna columna", claves.outliers
diag_cajas + theme(axis.text.x = element_text(angle = 45, hjust = 1)) #Ponemos los nombres inclinados p
```

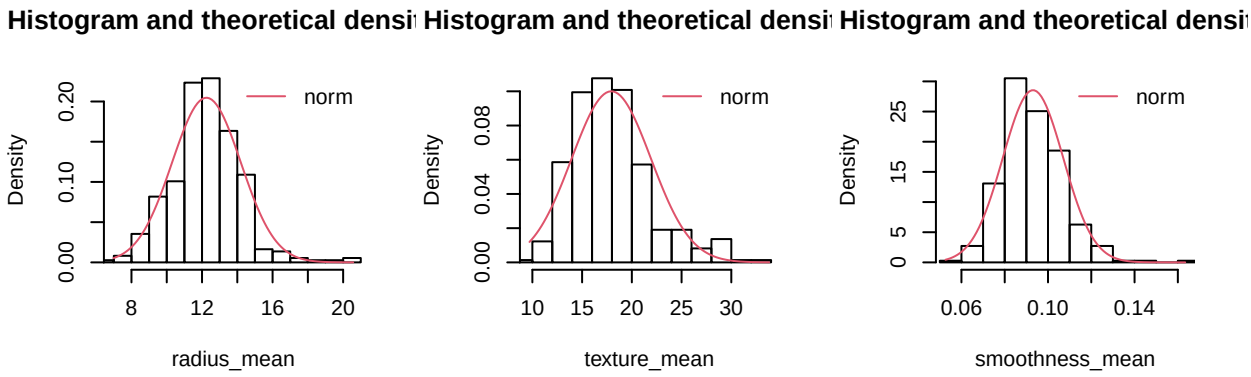


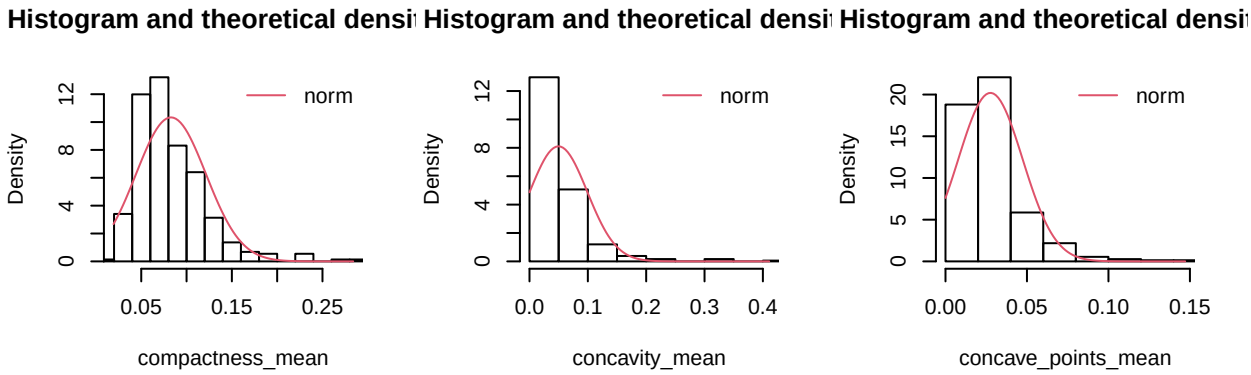
Vemos que en total 77 instancias son candidatas a outliers, por tener valores extremos en alguna de las columnas.

2.3.2 Test de hipótesis (Opcional)

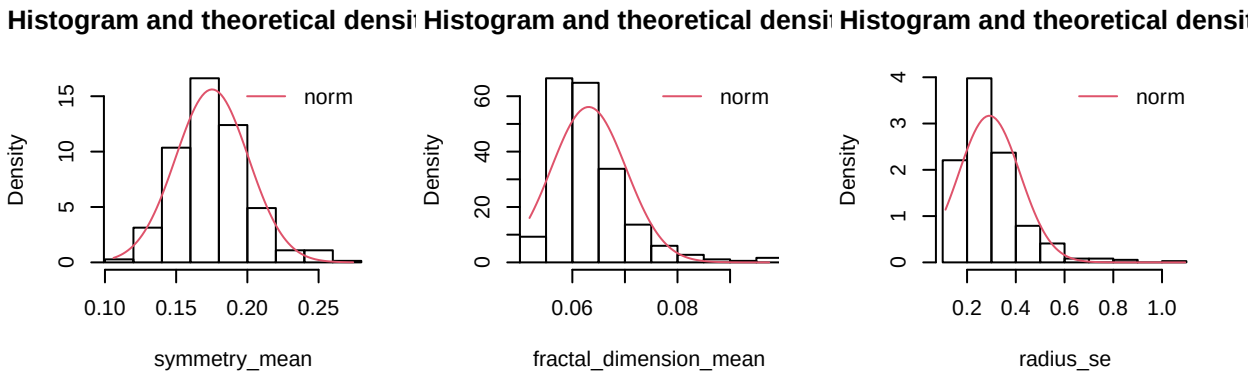
Primero vemos qué variables se alejan de una normal:

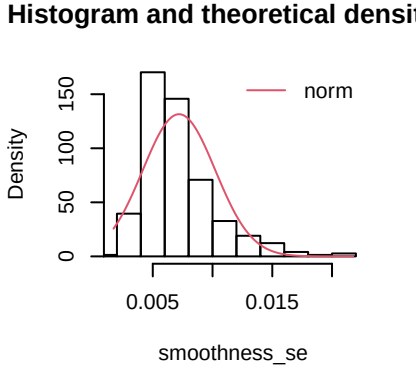
```
# COMPLETAR
par(mfrow = c(2, 3))

sapply(1:ncol(datos.num), function(i) {
  ajusteNormal = fitdist(datos.num[, i] , "norm")
  denscomp (ajusteNormal,  xlab = names(datos.num)[i])
})
```






```
##           [,1]        [,2]        [,3]        [,4]        [,5]
## hist      histogram,6 histogram,6 histogram,6 histogram,6 histogram,6
## densities numeric,101 numeric,101 numeric,101 numeric,101 numeric,101
##           [,6]        [,7]        [,8]        [,9]        [,10]
## hist      histogram,6 histogram,6 histogram,6 histogram,6 histogram,6
## densities numeric,101 numeric,101 numeric,101 numeric,101 numeric,101
```





La variable `concavity_mean` es la que más se aleja de una normal, por lo que la vamos a suprimir del estudio.

```
# COMPLETAR
columnas.no.normales <- c("concavity_mean") #Columnas que se alejan de una normal

datos.num.var.norm <- datos.num[, !(colnames(datos.num) %in% columnas.no.normales)]
head(datos.num.var.norm)
```

```
##   radius_mean texture_mean smoothness_mean compactness_mean concave_points_mean
## 1       17.99        10.38         0.11840          0.27760             0.14710
## 2       20.57        17.77         0.08474          0.07864             0.07017
## 3       19.69        21.25         0.10960          0.15990             0.12790
## 4       11.42        20.38         0.14250          0.28390             0.10520
## 5       20.29        14.34         0.10030          0.13280             0.10430
## 6       12.45        15.70         0.12780          0.17000             0.08089
##   symmetry_mean fractal_dimension_mean radius_se smoothness_se
## 1        0.2419                0.07871    1.0950      0.006399
## 2        0.1812                0.05667    0.5435      0.005225
## 3        0.2069                0.05999    0.7456      0.006150
## 4        0.2597                0.09744    0.4956      0.009110
## 5        0.1809                0.05883    0.7572      0.011490
## 6        0.2087                0.07613    0.3345      0.007510
```

Pasamos el test de Grubbs a todas las columnas.

```r
# COMPLETAR
sapply(1:ncol(datos.num.var.norm), function(i){
  test_Grubbs(datos.num.var.norm, i)})
```

```
##                           [,1]          [,2]           [,3]
## nombre.columna           "radius_mean" "texture_mean" "smoothness_mean"
## clave.mas.alejado.media  2             124            318
## valor.mas.alejado.media  20.57         33.81          0.1634
## nombre.mas.alejado.media "2"           "124"          "318"
## es.outlier               TRUE          TRUE           TRUE
## p.value                  0.005881068   0.02174254     0.0001165532
## p.value.test.normalidad  0.000276      2.879174e-09   0.001762445
## es.distrib.norm          FALSE         FALSE          FALSE
##                           [,4]               [,5]
## nombre.columna           "compactness_mean" "concave_points_mean"
## clave.mas.alejado.media  4                  1
## valor.mas.alejado.media  0.2839             0.1471
## nombre.mas.alejado.media "4"                "1"
## es.outlier               TRUE               TRUE
## p.value                  4.2681e-05         2.096715e-07
## p.value.test.normalidad  6.701489e-15       3.192756e-15
## es.distrib.norm          FALSE              FALSE
##                           [,6]             [,7]                    [,8]
## nombre.columna           "symmetry_mean"  "fractal_dimension_mean" "radius_se"
## clave.mas.alejado.media  24               4                       1
## valor.mas.alejado.media  0.2743           0.09744                 1.095
## nombre.mas.alejado.media "24"             "4"                     "1"
## es.outlier               TRUE             TRUE                    TRUE
## p.value                  0.03369295       0.000351259             2.009381e-08
## p.value.test.normalidad  1.523418e-05     1.60061e-15             3.205352e-15
## es.distrib.norm          FALSE            FALSE                   FALSE
##                           [,9]
## nombre.columna           "smoothness_se"
## clave.mas.alejado.media  319
## valor.mas.alejado.media  0.02177
## nombre.mas.alejado.media "319"
## es.outlier               TRUE
## p.value                  0.0004012179
## p.value.test.normalidad  1.175701e-14
## es.distrib.norm          FALSE
```

Habiendo aplicado el test de normalidad, se rechazan todas las variables pues p-value<0.05, por lo que ninguna variable parece seguir una normal. Por otro lado, todos los outliers han sido etiquetados como tal.

3 Outliers Multivariantes

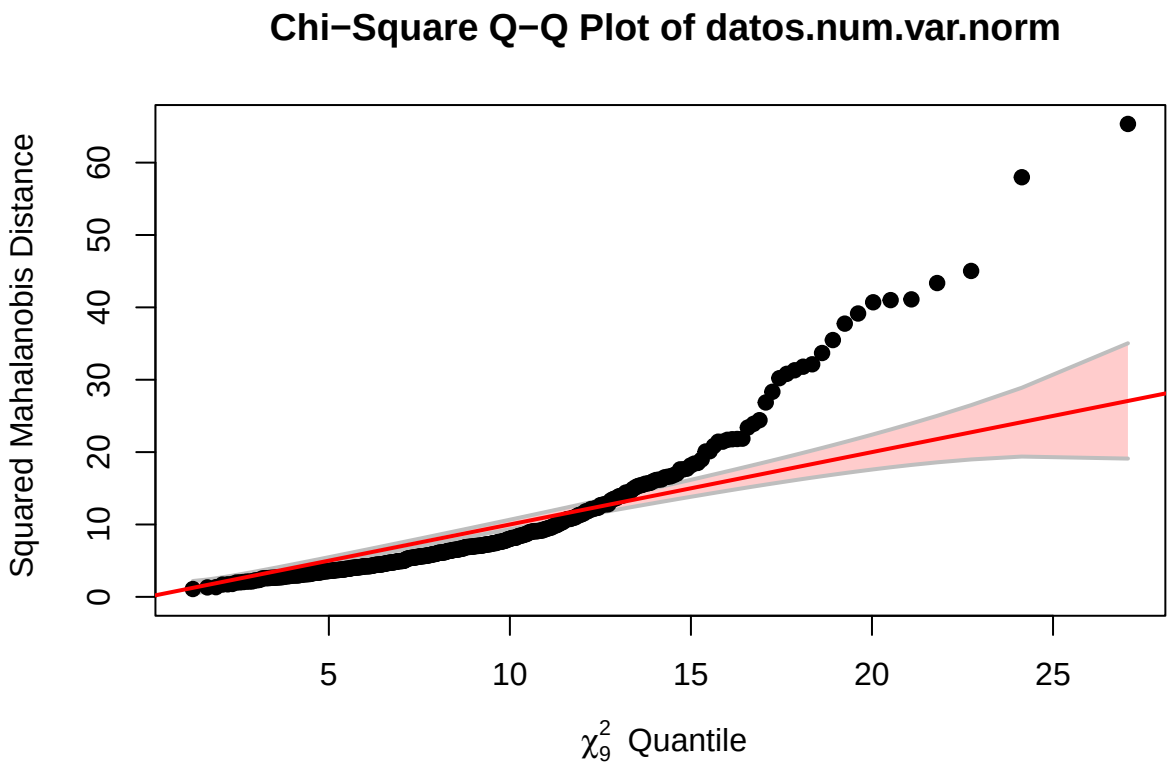
3.1 Métodos estadísticos basados en la distancia de Mahalanobis (OPCIONAL)

El requisito para aplicar estas técnicas es que la distribución subyacente de los datos sea una normal multivariante. Si la distribución conjunta de las variables es una normal multivariante, entonces la variable 1-dimensional formada al calcular las distancias de Mahalanobis, sigue una distribución chi-cuadrado con k grados de libertad, siendo k el número de variables (columnas).

3.1.1 Comprobación de la normalidad usando gráficos QQ

Los métodos basados en cuantiles calculan las distancias de Mahalanobis y ven si el gráfico QQ no se desvía demasiado de una χ^2.

```
cqplot(datos.num.var.norm , method = "classical")
```



Los datos se desvían demasiado de la línea de referencia, luego NO podemos afirmar que se verifique la hipótesis de normalidad.

3.1.2 Comprobación de la normalidad usando un test de hipótesis

La hipótesis nula planteada es H0:Los datos provienen de una distribución normal multivariante

```
test.MVN = mvn(datos.num.var.norm, mvnTest = "energy")
test.MVN$multivariateNormality["MVN"]
```

```
##   MVN
## 1  NO
```

```
test.MVN$multivariateNormality["p value"]
```

```
##   p value
## 1       0
```

El test rechaza la hipótesis nula (p value < 0.05), por lo que no podemos asumir que tengamos una distribución normal multivariante.

Conclusión: por un lado, el gráfico QQ muestras claras desviaciones de la normalidad, y por otro lado, el test de hipótesis rechaza que los datos sigan una normal multivariante. Por tanto, los valores detectados NO tendrán ninguna garantía estadística de ser outliers.

3.1.3 Detección de outliers usando cuantiles

```
dist.mah.clas = Mahalanobis(datos.num.var.norm, method = "classical")
dist.mah.clas
```

```
##   [1] 65.343013 37.753234 41.003804 35.482392 41.104826 12.792883 13.270925
##   [8] 10.934616 24.415801 23.902245  6.549059  7.836078  5.778663  9.131674
##  [15]  7.072646  4.956126  3.773629  5.613230  4.478089  3.323155  2.876340
##  [22]  6.900921 10.283187 26.859087 16.586192 18.400154  4.238821  5.649763
##  [29]  8.670637  4.040349 31.308382  1.302102 21.699906  4.318958  5.764187
##  [36]  6.942943  4.821371  2.964923  9.065973  5.328345  8.773078  4.252247
##  [43]  7.258251 12.321260  5.404309 14.863078  3.698696  9.169090  7.428959
##  [50]  3.781589  3.048196  3.884241  8.466327 11.123967 31.802607  6.918330
##  [57] 13.958085  2.116055 33.693622  5.918720  7.753380  6.136339  4.443836
##  [64] 15.468198  3.172139  6.342349 13.896572  5.509354 17.713729  7.322560
##  [71]  5.693655  2.046424  7.398755 12.798929  8.403155  4.407217  4.119792
##  [78] 18.990342 16.817146 57.981225  3.761931  1.982814  3.416229 13.570230
##  [85]  3.270008  6.850708  6.789315  7.032901  4.451590  8.508580  3.356887
##  [92]  6.138947 15.095075  5.607365  5.570250 12.796322  5.943900  7.960706
##  [99] 11.319697 13.452953  4.981143  3.596122  7.284174 23.391536 21.441086
## [106]  3.763798  4.220082  1.782471  5.913192  9.073287  6.109857  1.702349
## [113]  6.204181  7.597898  7.195330  4.230579  2.795817  2.004408 10.761243
## [120]  4.271945  7.074823  3.593510 12.716531 21.778235  8.909629  4.992170
## [127] 11.438374  2.587813  4.779748  9.060742  6.666648 14.039210  4.000961
## [134] 10.632282  5.518270  2.618127  3.253471  9.442036  3.564707  4.559812
## [141]  8.148647  8.104278  3.300840 11.524679 39.151860  5.861301  2.629941
## [148]  6.113191 15.963454  7.263775  3.534539  4.069727  6.355369 43.359752
## [155]  4.860221 32.134506  3.631617  3.704295  4.307348  3.786724  3.879546
## [162]  7.641179  6.111386  4.559596  2.702244  6.673201  4.466541  6.500514
## [169]  5.658472 16.136388  6.899283  8.004881  4.061857  4.674665  2.638238
## [176]  7.102215 30.813842  5.456639  3.622296 21.443212  4.715058  4.183803
## [183]  7.076908  1.340554  2.950811  5.997519  4.842614  2.535229 12.127209
## [190]  3.328343  3.123049 12.665770  3.150745  3.084664  9.339659  4.169323
## [197]  5.876379 17.024087  1.731026  3.530776  3.873137 21.858151  3.553403
## [204]  8.990178  6.555327  4.519161  3.633800 14.402076 18.508827  4.740370
## [211]  2.293645  2.341716  8.440632  2.656358  1.082541  7.405972  4.236835
## [218] 12.217502 30.215142 10.762459  4.398485 12.129210  7.490127 15.281935
## [225]  7.030351  6.654870  3.510913  5.216857  7.575501  5.115754 16.652382
## [232]  2.954527  6.103154  6.538889  7.217817  4.757778  2.270568  6.663434
## [239]  5.704567  4.469072  3.705354  5.629830  6.493665  8.580736  4.753438
## [246]  3.021328  5.298313 10.839865  5.526681  4.100021  9.960782  4.635389
## [253]  4.952200  3.738390 15.616890  7.946126  4.654484 21.813929  6.484245
## [260]  3.133113  4.932230  4.941579  2.748327  9.184088  2.966954  2.847511
## [267]  4.237449  2.135847  2.572969  6.874567  9.105991 20.083251  9.571249
## [274]  7.226563  2.581261 12.281823  8.327541 10.977095  3.352192 16.194599
## [281] 10.090366  5.607828  4.722904 11.810652  9.729955  2.107737  4.447667
```

```
## [288]  7.025894  5.356824  4.367273  9.068103  8.647497 15.753967  4.207491
## [295] 16.505861  6.965659  2.915663  7.125604  7.722984  3.988841  3.122102
## [302]  2.711333  6.435261  3.530023 10.424921  6.315925  3.870373  3.897798
## [309]  2.841876 18.144240 10.234414  4.944452  6.369756  5.269391  2.682853
## [316]  6.992946  5.431691 40.702452 45.033557  9.857454 20.853529 11.932382
## [323]  5.728137  5.448213  5.395821  3.156350 14.424442  6.444862 17.615906
## [330]  6.043299  4.114637  6.938866  8.179416  6.181598  5.816524 15.391950
## [337]  3.985800  8.139106  3.412761  4.046478  5.378576 13.662569 15.676689
## [344] 20.121160  7.587357  8.961721  6.874987  9.473961  7.672830  3.690044
## [351]  2.965028  6.322924  4.018864  9.867601  4.906679  8.159942 11.806329
## [358]  4.601720 10.704026 14.547038  7.429366 16.230902 11.297154  9.327917
## [365]  9.575840 17.617987 28.333620
```

Etiquetamos como outliers aquellos datos con una distancia de Mahalanobis por encima del cuantil 97.5.

```
# COMPLETAR
cuantil_975= quantile(dist.mah.clas, 0.975)
claves.outliers.mah.clas <- which(dist.mah.clas>cuantil_975)
```

```
claves.outliers.mah.clas
```

```
##  [1]   1   2   3   4   5  80 145 154 318 319
```

```
nombres_filas (datos.num, claves.outliers.mah.clas)
```

```
##  [1] "1"   "2"   "3"   "4"   "5"   "80"  "145" "154" "318" "319"
```

Vemos que se repiten las instancias 1, 2 y 5 (como vimos en el punto 2.1.2).

3.1.4 Test de hipótesis para detectar outliers

Establecemos H0:El valor más alejado del centro de la distribución no es un outlier (si se rechaza, es outlier). Considerando la distancia de Mahalanobis, se convierte en H0:El valor con mayor distancia de Mahalanobis al centro de la distribución viene de la misma distribución Normal multivariante que el resto de datos.

Aplicamos el **test individual** con un valor de significación de 0.05 y el **test de intersección** con un valor de

$$1 - (1 - 0.05)^{1/n}$$

(n es el número de registros del conjunto de datos). Obtenemos las claves de los outliers encontrados por ambos métodos.

```
# Establecemos la semilla
set.seed(2)

#test individual
test.individual.Cerioli = cerioli2010.fsrmcd.test(datos.num.var.norm, signif.alpha = 0.05)
son.posibles.outliers.individual.Cerioli = test.individual.Cerioli$outliers
claves.test.individual = which (son.posibles.outliers.individual.Cerioli == TRUE)
nombres.test.individual = nombres_filas(datos.num.var.norm, claves.test.individual)

# test interseccion
```

```
n = nrow(datos.num.var.norm)
signif.interseccion = 1. - ((1. - 0.05)^(1./n))
test.interseccion.Cerioli = cerioli2010.fsrmcd.test(datos.num.var.norm, signif.alpha = signif.intersecci
son.posibles.outliers.Cerioli.interseccion = test.interseccion.Cerioli$outliers
claves.test.interseccion = which (son.posibles.outliers.Cerioli.interseccion == TRUE)
nombres.test.interseccion = nombres_filas(datos.num.var.norm, claves.test.interseccion)
```

```
claves.test.individual
```

```
##  [1]   1   2   3   4   5   6   7   8   9  10  24  25  26  29  31  33  39  46  53
## [20]  54  55  56  57  59  64  67  69  74  75  78  79  80  84  93  96 100 104 105
## [39] 110 119 123 124 130 132 134 144 145 149 154 156 170 177 180 189 192 195 198
## [58] 202 208 209 218 219 222 224 229 231 251 255 258 264 272 276 278 280 291 293
## [77] 295 306 310 311 316 318 319 321 322 327 329 336 342 343 344 345 354 360 361
## [96] 362 366 367
```

```
nombres.test.individual
```

```
##  [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"   "10"  "24"  "25"
## [13] "26"  "29"  "31"  "33"  "39"  "46"  "53"  "54"  "55"  "56"  "57"  "59"
## [25] "64"  "67"  "69"  "74"  "75"  "78"  "79"  "80"  "84"  "93"  "96"  "100"
## [37] "104" "105" "110" "119" "123" "124" "130" "132" "134" "144" "145" "149"
## [49] "154" "156" "170" "177" "180" "189" "192" "195" "198" "202" "208" "209"
## [61] "218" "219" "222" "224" "229" "231" "251" "255" "258" "264" "272" "276"
## [73] "278" "280" "291" "293" "295" "306" "310" "311" "316" "318" "319" "321"
## [85] "322" "327" "329" "336" "342" "343" "344" "345" "354" "360" "361" "362"
## [97] "366" "367"
```

```
claves.test.interseccion
```

```
##  [1]   1   2   3   4   5   9  10  24  26  31  33  55  59  79  80  96 104 105 145
## [20] 149 154 156 177 180 198 219 255 258 318 319 321 336 344 367
```

```
nombres.test.interseccion
```

```
##  [1] "1"   "2"   "3"   "4"   "5"   "9"   "10"  "24"  "26"  "31"  "33"  "55"
## [13] "59"  "79"  "80"  "96"  "104" "105" "145" "149" "154" "156" "177" "180"
## [25] "198" "219" "255" "258" "318" "319" "321" "336" "344" "367"
```

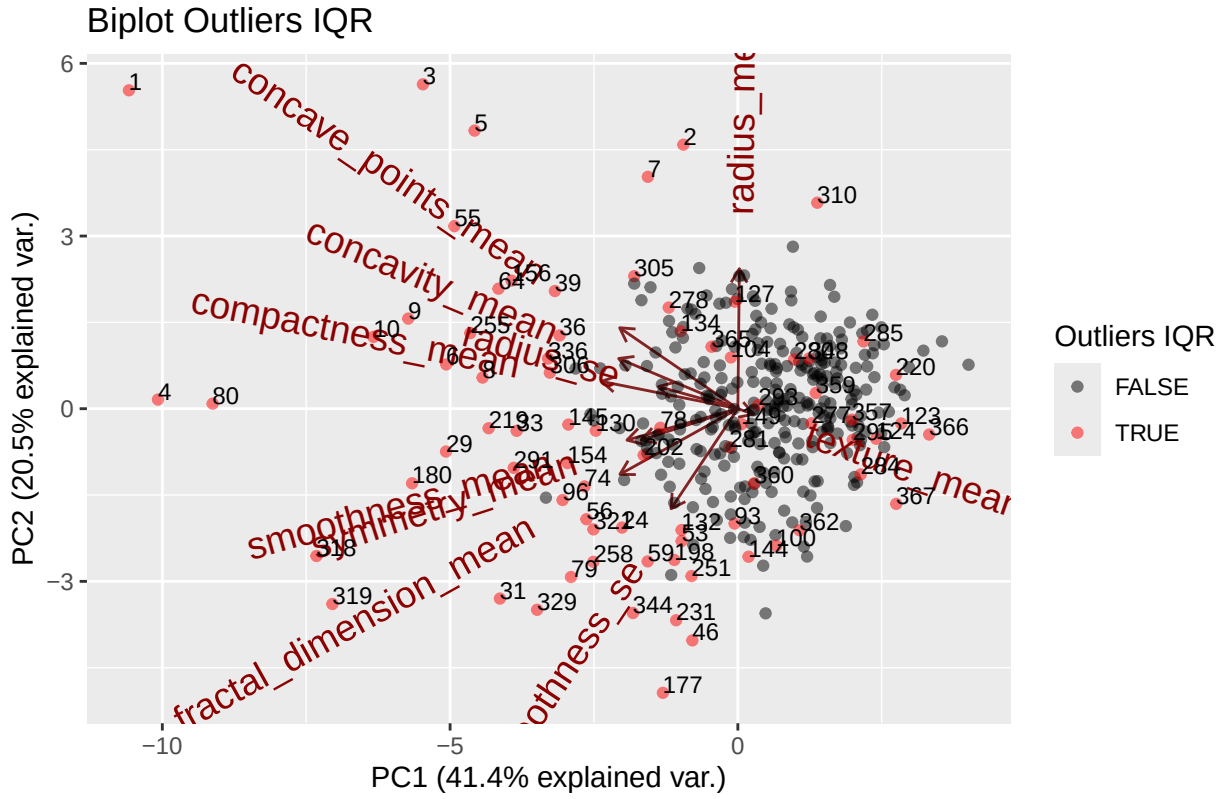
El test individual devuelve 98 outliers y el test de intersección 34 (es más conservador). Vimos antes que solo tenemos garantía estadística de que sea un outlier el que tiene mayor valor de distancia de Mahalanobis, que corresponde con la instancia 1.

Consideraciones finales:

- No hemos podido afirmar que la distribución fuera normal multivariante, por lo que no tenemos garantía estadística de que el el outlier más alejado sea un outlier (de que provenga de una distribución distinta del resto de los datos), a pesar de que toma valores extremos en varias variables.

3.2 Visualización de datos con un Biplot

```
biplot.outliers.IQR = biplot_2_colores(datos.num,
                                       claves.outliers.IQR.en.alguna.columna,
                                       titulo.grupo.a.mostrar = "Outliers IQR",
                                       titulo ="Biplot Outliers IQR")
biplot.outliers.IQR
```



La suma de los porcentajes explicados es ligeramente alta (20.5 + 41.4 = 61.9), por lo que la representación obtenida no parece ser mala aproximación. Puede apreciarse que las instancias 4 y 80 se encuentran en los valores más altos de un grupo de variables (no solo de una), al igual que la instancia 1.

3.3 Métodos basados en distancias: LOF

Vamos a aplicar otros métodos que no ofrecen garantía estadística, pero son capaces de determinar cómo de alejado está cada punto al resto de los datos. Usaremos LOF. Como operamos con distancias, usamos los datos normalizados.

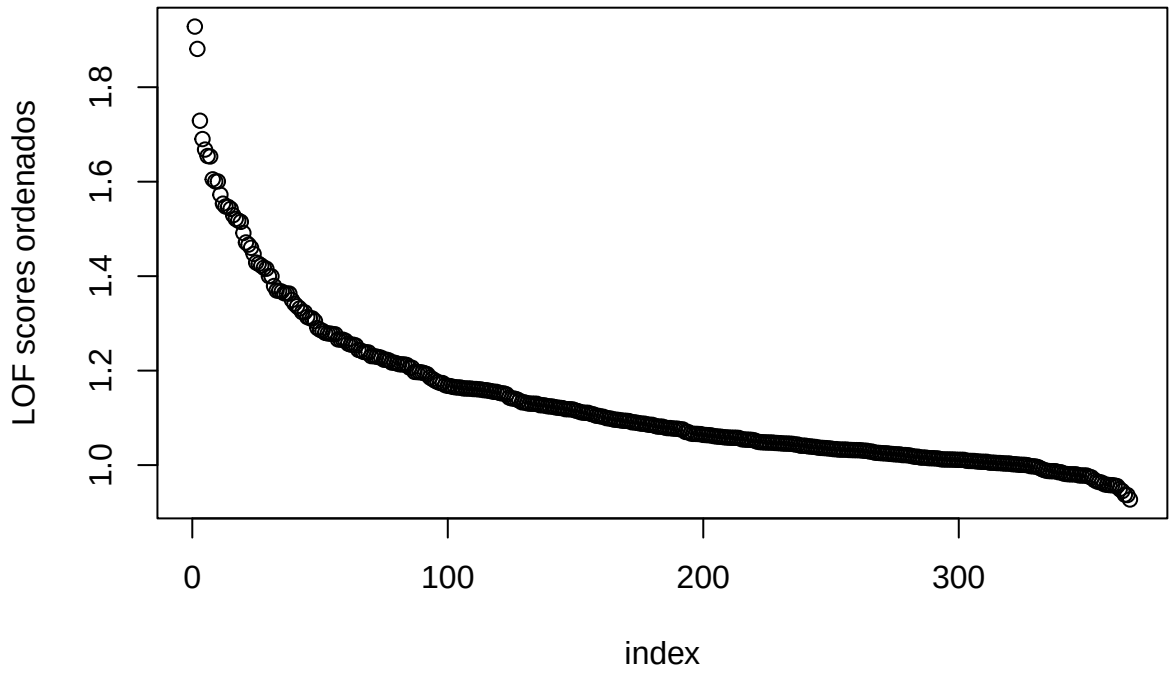
```
num.vecinos.lof = 5
lof.scores = LOF(dataset = datos.num.zscore, k = num.vecinos.lof)
lof.scores
```

```
##   [1] 1.6533700 1.5149571 1.6005625 1.3673465 1.3997044 1.1736081 1.5424850
##   [8] 1.1822995 1.2630878 1.0960287 1.0582127 1.1082939 1.1642743 1.1046576
##  [15] 1.0876077 0.9903592 1.0382808 1.2306370 1.0320903 0.9871094 0.9875246
##  [22] 1.0980883 1.0208070 1.3632836 1.2131990 1.3105107 1.0086515 1.1109923
```

```
##  [29] 1.3048575 1.0853178 1.6049576 1.0088079 1.5474856 0.9812193 1.0248706
##  [36] 1.1622773 1.0118816 1.0123591 1.2270449 1.0069193 1.2307588 1.0252123
##  [43] 1.1431387 1.1603767 1.0145529 1.4222828 1.0771051 1.0547632 1.1857542
##  [50] 1.0117962 1.0229165 1.1519133 1.0477291 1.2864975 1.4178685 1.0996030
##  [57] 1.2653782 1.1246224 1.2839959 0.9640600 1.0639446 1.0930941 0.9449559
##  [64] 1.1155294 0.9271875 1.0234123 1.2773452 0.9782490 1.2130066 1.0770226
##  [71] 1.0478928 1.0353566 1.0876021 1.2118692 1.1262531 1.0146223 1.0139051
##  [78] 1.4287825 1.3134900 1.5211342 1.0348935 1.1181599 0.9957835 1.6902476
##  [85] 1.0304493 1.0658339 1.1532966 1.0661024 1.1020803 1.3791491 1.0051374
##  [92] 1.0316734 1.1641671 1.1760093 1.0989349 1.2168171 1.5534646 1.0943614
##  [99] 1.2787762 1.2171110 1.1329393 1.1407410 1.0602317 1.4713732 1.6005940
## [106] 1.0003149 1.0302830 1.0212116 1.0087615 1.3416897 1.1671455 0.9849754
## [113] 1.0018104 1.2297302 1.3637767 0.9929487 1.0112910 0.9365701 1.5466697
## [120] 1.0019682 1.0361885 0.9686524 1.2661820 1.2060531 1.0247980 1.0523246
## [127] 1.1620718 1.0052029 1.0201177 1.2547936 1.0629234 1.0397354 1.0655205
## [134] 1.2532603 1.0577990 1.0157949 1.0176777 1.0793688 1.0011676 1.0852143
## [141] 1.0338682 1.1295971 1.0319210 1.1496558 1.4156715 1.0498557 0.9803350
## [148] 1.1412481 1.5176467 1.0936615 1.0410016 1.0092831 1.0907115 1.9283310
## [155] 1.0005708 1.7290855 1.0434048 1.0783314 0.9567618 0.9626612 0.9554979
## [162] 1.0633259 1.0477238 1.0265808 0.9804409 1.0783579 1.1215426 1.1624722
## [169] 1.0159531 1.2656352 1.1610793 1.1183235 0.9576406 1.0454757 1.0590401
## [176] 1.1564533 1.3362445 1.1600953 1.0316174 1.2440038 1.0337683 1.0702300
## [183] 1.1943385 0.9583128 1.0111130 1.1364521 1.0607651 1.0533129 1.1299476
## [190] 1.0448418 1.0393952 1.2153441 0.9996992 0.9821465 1.3630504 1.0116164
## [197] 1.0585937 1.1041100 0.9779770 1.0775279 1.0598837 1.3239346 1.0176441
## [204] 1.2550397 1.1205514 1.0885001 1.0078711 1.2799696 1.3315892 1.0832468
## [211] 0.9779035 0.9806813 1.2388861 1.0148792 0.9373879 1.0259717 0.9737932
## [218] 1.2384737 1.1958948 1.1074002 1.0031943 1.3686855 1.2285949 1.4470859
## [225] 1.0159400 0.9978606 1.1238855 1.0328083 1.1551408 1.0141791 1.1224451
## [232] 0.9593371 0.9574453 1.0317053 1.0953967 1.0452308 1.0387052 1.1516234
## [239] 1.1268853 1.0579202 1.0327492 1.1106137 1.0119202 1.1973261 0.9755067
## [246] 0.9492198 1.0310572 1.4667383 1.0370146 1.0636083 1.1028922 1.0117281
## [253] 1.1655798 1.0211993 1.0938522 1.0034004 1.0327266 1.4259709 1.1619787
## [260] 1.1587763 1.0487679 1.0005904 1.0238148 1.1176661 1.0043366 1.0455570
## [267] 1.0904684 1.0043749 0.9869345 1.0578400 1.0483914 1.5725213 1.1966885
## [274] 1.0814781 1.0813070 1.2901888 1.0435719 1.0466886 0.9804514 1.1922353
## [281] 1.1550746 1.0285864 1.0038593 1.1301371 1.0896188 0.9974166 1.0756748
## [288] 1.0226162 1.0069076 1.0364906 1.1788739 1.1969382 1.1647354 1.0816588
## [295] 1.2231339 1.1328283 1.0225260 1.0328828 1.0573573 0.9778695 1.0411211
## [302] 0.9655488 1.0111277 1.0256527 1.1575352 1.2230520 1.0754918 1.0032575
## [309] 1.0458534 1.8811122 1.2135444 1.1106997 1.1584735 1.0542206 0.9871328
## [316] 1.2424423 1.0109825 1.4001289 1.6543745 1.2783695 1.2768252 1.1120364
## [323] 1.1610137 1.0358909 1.1184718 0.9847965 1.4600603 1.0660402 1.3696322
## [330] 1.1242045 0.9970571 1.0860637 1.1738892 1.0464246 1.0410815 1.1691914
## [337] 1.0701757 1.1267286 1.0288526 1.0185673 1.0469483 1.3235062 1.3113141
## [344] 1.6679490 1.1210460 1.2570946 1.0536616 1.0048947 1.1392364 0.9884985
## [351] 1.0073989 1.0537754 1.1141577 1.3495117 1.1678595 1.2064599 1.0670653
## [358] 1.0069909 1.0615146 1.2214690 1.0481157 1.1299171 1.0963246 1.1310624
## [365] 1.2395813 1.4913854 1.5290517
```

La función LOF asigna un score a cada dato, indicando hasta qué punto es un outlier.

```r
# COMPLETAR
lof.scores.ordenados <- order(lof.scores, decreasing = TRUE)
#lof.scores.ordenados
#lof.scores[lof.scores.ordenados]
plot(1:length(lof.scores), lof.scores[lof.scores.ordenados], xlab="index", ylab="LOF scores ordenados")
```



Podemos apreciar que hay un grupo de unos 10 valores con scores más altos que el resto de datos. Vamos a analizar dichos valores. Establecemos la variable num.outliers en 8 y obtenemos sus claves junto con sus nombres.

```r
# COMPLETAR
num.outliers=10
claves.outliers.lof <- lof.scores.ordenados[1:num.outliers]
nombres.outliers.lof <- nombres_filas (datos.num.zscore, claves.outliers.lof)
```

```r
claves.outliers.lof
```

```
##  [1] 154 310 156  84 344 319   1  31 105   3
```

```r
nombres.outliers.lof
```

```
##  [1] "154" "310" "156" "84"  "344" "319" "1"   "31"  "105" "3"
```

Mostramos también los valores normalizados de dichos registros:

```
datos.num.zscore[claves.outliers.lof, ]
```

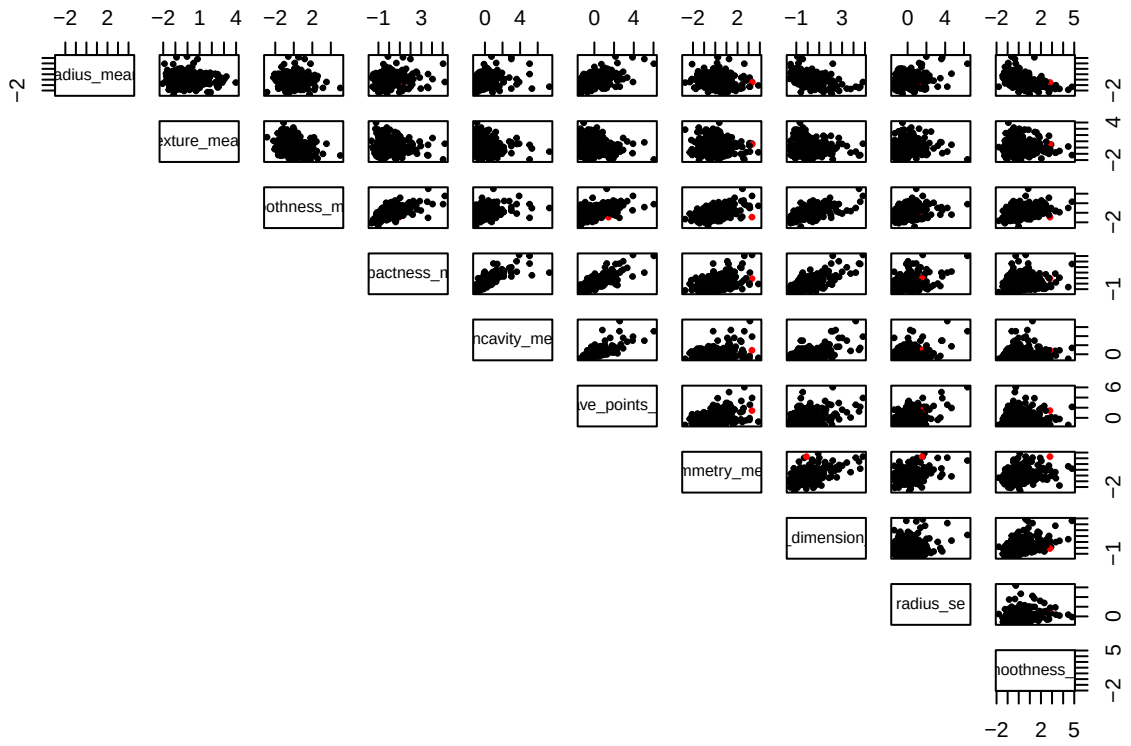
```
##     radius_mean texture_mean smoothness_mean compactness_mean concavity_mean
## 154  -0.5078094   0.50703623      -0.9198379        0.9125067     0.87313074
## 310   2.8688422  -1.17788405      -1.0498995       -0.5345990    -0.10674173
## 156   1.1062198   0.44945351      -0.3931600        2.1932458     1.75499568
## 84    2.3513281   0.38185641      -1.3307467       -0.2743114     0.03631275
## 344  -2.3365302   1.87900697      -0.4567616        0.9590791     0.86866663
## 319  -1.3194356  -1.20041642       2.3174088        3.5593674     1.40192504
## 1     2.9405769  -1.89140898       1.8100258        5.0393325     5.08075875
## 31   -1.7231991  -0.82487698       0.3400442        1.8180798     0.73758408
## 105  -1.2968904   0.07141048      -1.6844856       -1.5366803    -1.00869517
## 3     3.8116403   0.83000014       1.1811566        1.9940197     2.99683034
##     concave_points_mean symmetry_mean fractal_dimension_mean  radius_se
## 154          1.42994138     3.2947745             -0.1089417  1.5467590
## 310          0.71697373    -2.0796641             -1.4998748  1.5213743
## 156          1.94267131    -0.1487749              1.2384370  4.6762208
## 84           0.00552303     0.3593538             -1.4647502  1.4856770
## 344         -0.70592767     1.1137296              2.0238224 -0.5712797
## 319          2.16313507     1.1919032              4.5865113 -0.1365662
## 1            6.04248813     2.6068463              2.1924204  6.3730316
## 31           0.05659376     0.5860574              3.7505464  1.8608951
## 105         -1.39563538    -0.3872045              0.1917247  0.4940858
## 3            5.07163856     1.2388074             -0.4377077  3.6013359
##     smoothness_se
## 154     2.8134553
## 310    -0.9307696
## 156     0.1864721
## 84     -1.2356814
## 344     2.7245501
## 319     4.7990041
## 1      -0.2623344
## 31      3.2974946
## 105    -1.8053331
## 3      -0.3443247
```

Viendo estos datos, la mayoría de instancias toman un score alto porque toman valores extremos en más de una variable (y por tanto, son puntos alejados del resto de forma multivariante). Posteriormente analizaremos con más detalle esta cuestión.

Por ahora, vamos a analizar el registro que tiene el mayor score en LOF. Corresponde al registro 154. Podemos apreciar que tiene un valores extremos en varias variables (en `symmetry_mean`, `smoothness_se`).

Vamos a empezar viendo las posibles interacciones de dos variables. Para ello, mostramos los diagramas de dispersión corespondientes a los cruces 2 a 2 de las variables, mostrando en rojo el outlier con mayor score en LOF (registro 154).

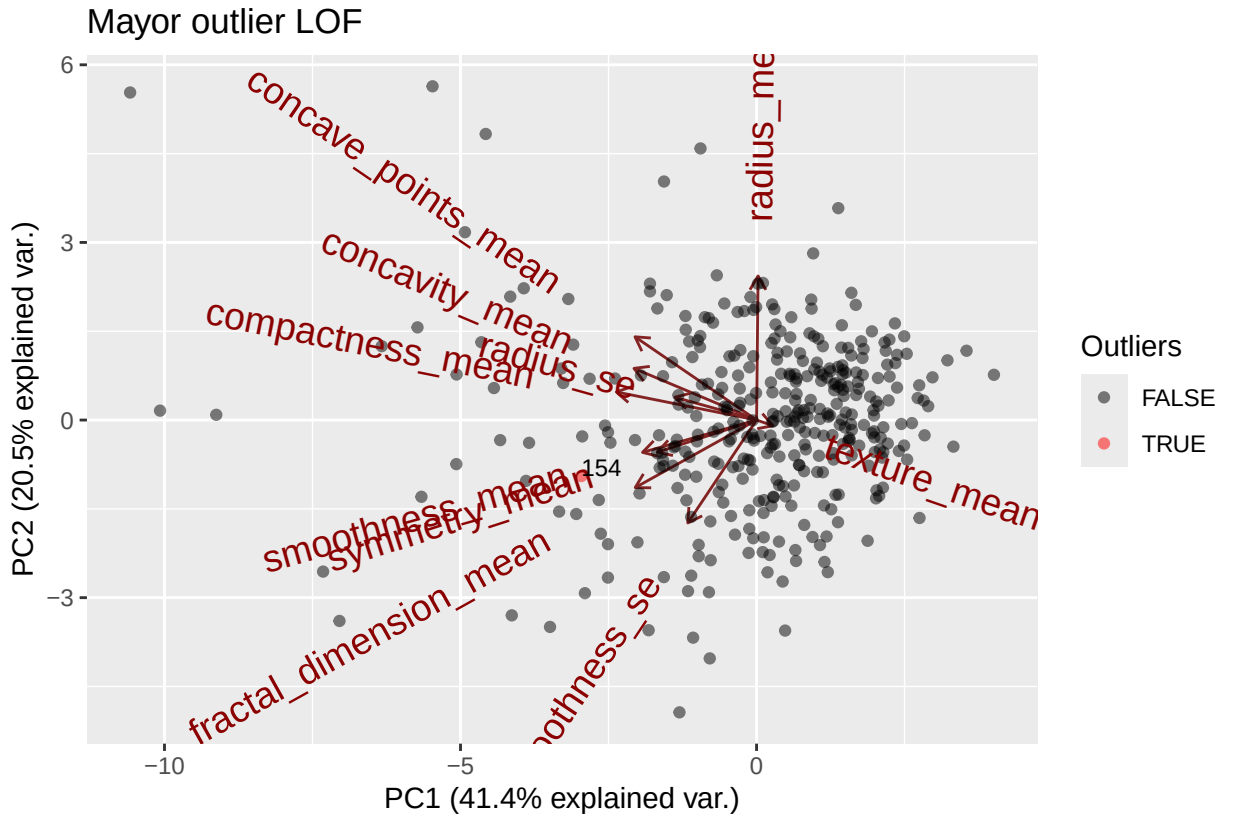
```
clave.max.outlier.lof = claves.outliers.lof[1]

colores = rep("black", times = nrow(datos.num.zscore))
colores[clave.max.outlier.lof] = "red"
pairs(datos.num.zscore, pch = 19,  cex = 0.5, col = colores, lower.panel = NULL)
```



Hay una correlación directa entre `compactness_mean`, `concavity_mean` y `concave_points_mean`, aunque el valor 154 se sitúa en zonas de alta densidad (no es outlier ahí), pero sí que se encuentra ligeramente aislado con respecto a los demás puntos (en los bordes de las regiones de alta densidad). Entre `radius_mean` y `simmetry_mean`, no hay correlación, pero el outlier se sitúa ahora sí en una zona aislada. Esta tendencia puede ser la que ha hecho que la instancia 154 haya obtenido un score tan alto en LOF.

Procedemos a ver la interacción de todas las variables (no sólo 2 a 2), con un biplot:

```
biplot.max.outlier.lof = biplot_2_colores(datos.num.zscore, clave.max.outlier.lof, titulo = "Mayor outli
biplot.max.outlier.lof
```



La suma de la variabilidad explicada por las dos componentes principales es medianamente alta (un 60%) y por tanto la aproximación es buena.

Observamos nuestro outlier en la parte central del diagrama, ligeramente alejado de la nube central de puntos, pero no completamente aislado como cabría esperar. De hecho, se encuentra en una zona de bastante alta densidad. Por tanto, esta instancia no parece tener las características de un outlier.

3.4 Métodos basados en Clustering

3.4.1 Clustering usando k-means

Fijamos 5 outliers y 3 centroides.

```
num.outliers = 5
num.clusters = 3
set.seed(2)
```

Construimos el modelo kmeans:

```
# COMPLETAR
modelo.kmeans <- kmeans(datos.num.zscore , num.clusters)
modelo.kmeans
```

```
## K-means clustering with 3 clusters of sizes 146, 25, 196
##
## Cluster means:
```

```
##   radius_mean texture_mean smoothness_mean compactness_mean concavity_mean
## 1  -0.5194611  -0.06760771       0.5666926        0.2268905   -0.009923327
## 2   0.3877257  -0.27678969       1.4267586        2.4390959    2.555329708
## 3   0.3374907   0.08566565      -0.6041127       -0.4801194   -0.318543046
##   concave_points_mean symmetry_mean fractal_dimension_mean   radius_se
## 1         0.003286737     0.4014926              0.4621165  0.03472647
## 2         2.356880232     1.4462803              1.9546973  1.59882946
## 3        -0.303070762    -0.4835456             -0.5935533 -0.22980001
##   smoothness_se
## 1     0.5659948
## 2     0.7861276
## 3    -0.5218797
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   2   3   2   2   2   2   3   2   2   2   1   1   1   3   1   1   3   3   3   3
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   1   3   1   1   1   1   1   3   2   3   2   3   2   3   1   2   1   1   2   3
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   3   3   1   1   3   1   3   1   1   1   3   3   1   1   2   1   1   1   1   3
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   1   3   3   2   1   3   3   3   1   1   1   3   3   1   3   3   3   1   1   2
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   1   1   3   3   3   3   1   1   3   3   3   1   1   3   1   1   3   3   3   1
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##   3   3   3   3   3   3   1   1   1   1   3   3   1   3   3   1   1   3   1   1
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##   3   3   3   3   1   3   3   3   3   1   3   1   3   1   1   1   3   1   3   3
## 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160
##   1   3   1   1   1   3   3   3   1   3   3   1   3   1   3   2   3   1   3   3
## 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
##   3   3   3   1   3   1   3   3   3   3   3   3   3   3   3   3   1   3   3   2
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200
##   3   1   1   3   3   3   3   1   1   3   3   1   1   1   1   1   1   1   3   3
## 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220
##   3   1   3   3   1   1   3   1   1   3   3   3   3   3   3   3   3   1   2   3
## 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240
##   3   1   3   3   1   3   3   3   1   1   1   1   3   1   3   3   3   3   3   3
## 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260
##   3   1   3   3   3   3   1   3   3   1   1   3   3   1   2   1   3   1   3   1
## 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280
##   3   3   3   1   3   3   3   3   3   1   3   1   1   3   3   3   3   1   3   3
## 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300
##   1   3   3   3   3   3   3   3   3   1   2   1   3   3   3   1   3   3   3   1
## 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320
##   3   3   1   3   1   2   3   1   3   3   3   3   3   1   3   1   1   2   2   1
## 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340
##   1   3   3   3   3   1   1   1   1   3   1   1   1   1   3   2   1   1   1   3
## 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
##   1   1   1   1   1   1   3   3   3   3   1   1   3   1   3   1   3   1   3   1
## 361 362 363 364 365 366 367
##   1   1   3   1   3   3   3
##
## Within cluster sum of squares by cluster:
```

```
## [1] 980.6102 467.2717 924.4179
##  (between_SS / total_SS =  35.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
# COMPLETAR
asignaciones.clustering.kmeans <- modelo.kmeans$cluster
centroides.normalizados <- modelo.kmeans$centers
```

```
head(asignaciones.clustering.kmeans)
```

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

```
centroides.normalizados
```

```
##   radius_mean texture_mean smoothness_mean compactness_mean concavity_mean
## 1  -0.5194611  -0.06760771       0.5666926        0.2268905   -0.009923327
## 2   0.3877257  -0.27678969       1.4267586        2.4390959    2.555329708
## 3   0.3374907   0.08566565      -0.6041127       -0.4801194   -0.318543046
##   concave_points_mean symmetry_mean fractal_dimension_mean   radius_se
## 1         0.003286737     0.4014926              0.4621165  0.03472647
## 2         2.356880232     1.4462803              1.9546973  1.59882946
## 3        -0.303070762    -0.4835456             -0.5935533 -0.22980001
##   smoothness_se
## 1     0.5659948
## 2     0.7861276
## 3    -0.5218797
```

Si queremos los centroides sin normalizar:

```
centroides.desnormalizados = desnormaliza(datos.num, centroides.normalizados)
centroides.desnormalizados
```

```
##   radius_mean texture_mean smoothness_mean compactness_mean concavity_mean
## 1    11.23726     17.66473      0.10100158       0.09160123     0.04922140
## 2    13.00776     16.82920      0.11303680       0.17710200     0.17564200
## 3    12.90972     18.27694      0.08461806       0.06427561     0.03401202
##   concave_points_mean symmetry_mean fractal_dimension_mean radius_se
## 1          0.02766577     0.1854781             0.06639452 0.2959932
## 2          0.07421160     0.2122080             0.07701800 0.4931640
## 3          0.02160710     0.1628352             0.05888077 0.2626469
##   smoothness_se
## 1   0.008914589
## 2   0.009583120
## 3   0.005610776
```

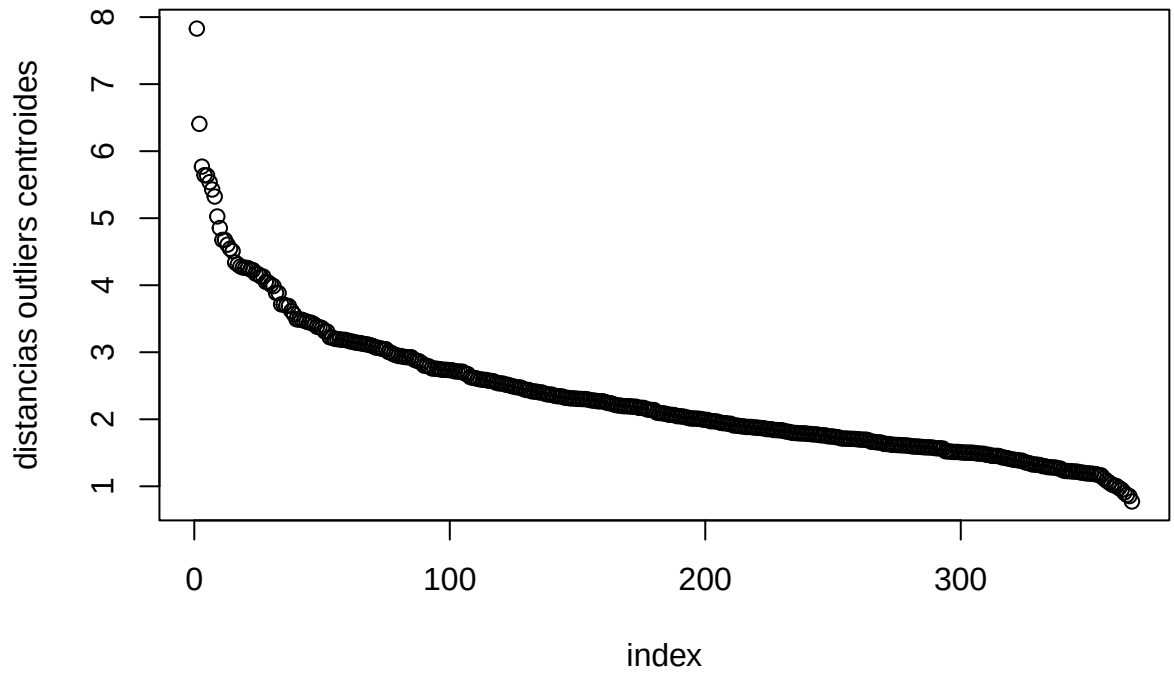
Calculamos los outliers como aquellos datos que más se alejan del centroide del cluster al que ha sido asignado.

```r
# 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.normalizados,
                                   asignaciones.clustering,
                                   datos.centroides.normalizados,
                                   num.outliers){
  distancias<- distancias_a_centroides(datos.normalizados, asignaciones.clustering, datos.centroides.no
  distancias.ordenadas <- order(distancias, decreasing = TRUE)
  claves <- distancias.ordenadas[1:num.outliers]
  list(distancias, claves)
}
```

Para elegir el número de outliers, seguimos el mismo procedimiento que se realizó con LOF, representamos todas las filas:

```r
# COMPLETAR
todos_outliers<- top_clustering_outliers(datos.num.zscore, asignaciones.clustering.kmeans, centroides.n
```

```r
plot(1:nrow(datos.num.zscore), todos_outliers[[1]][todos_outliers[[2]]], xlab="index", ylab="distancias
```



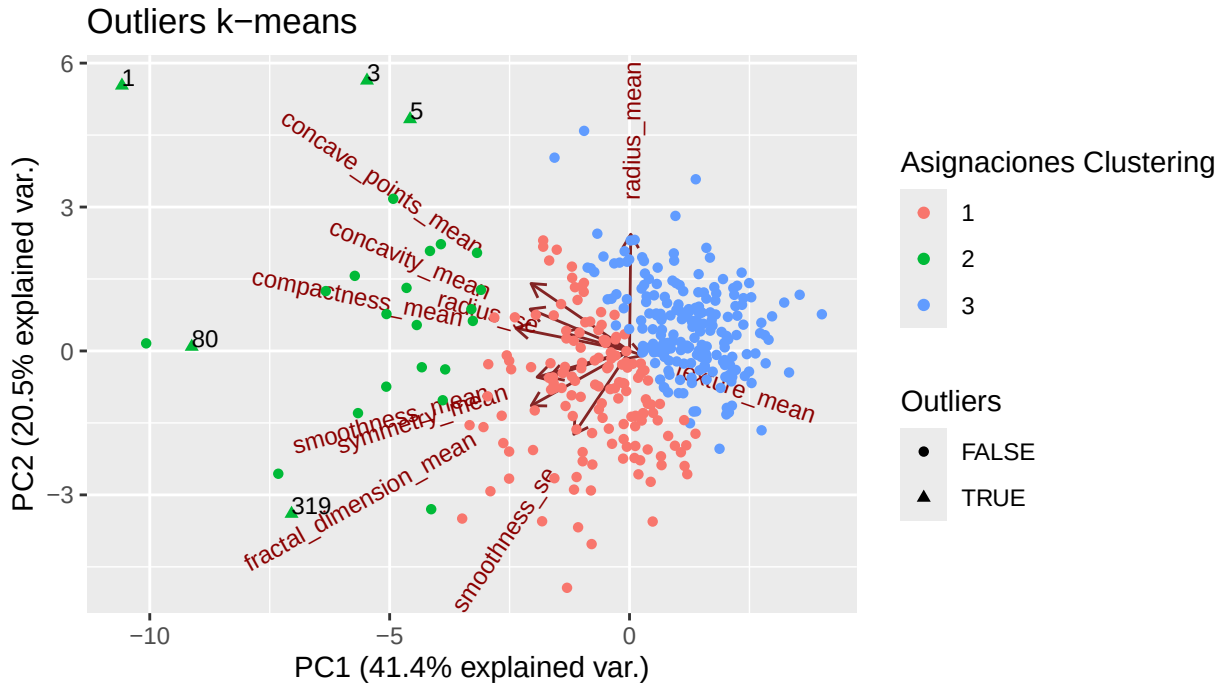
Según la gráfica obtenida, sería sensato elegir 2 como primera opción para la variable num.outliers. En cualquier caso, vamos a usar 5 para comparar con los resultados obtenidos en LOF.

```
distancias.outliers.centroides<- top_clustering_outliers(datos.num.zscore, asignaciones.clustering.kmean
claves.outliers.kmeans <- distancias.outliers.centroides[[2]]
claves.outliers.kmeans
```

```
## [1]   1  80 319   3   5
```

Vamos a mostrar un biplot con la información de los outliers y de los clusters.

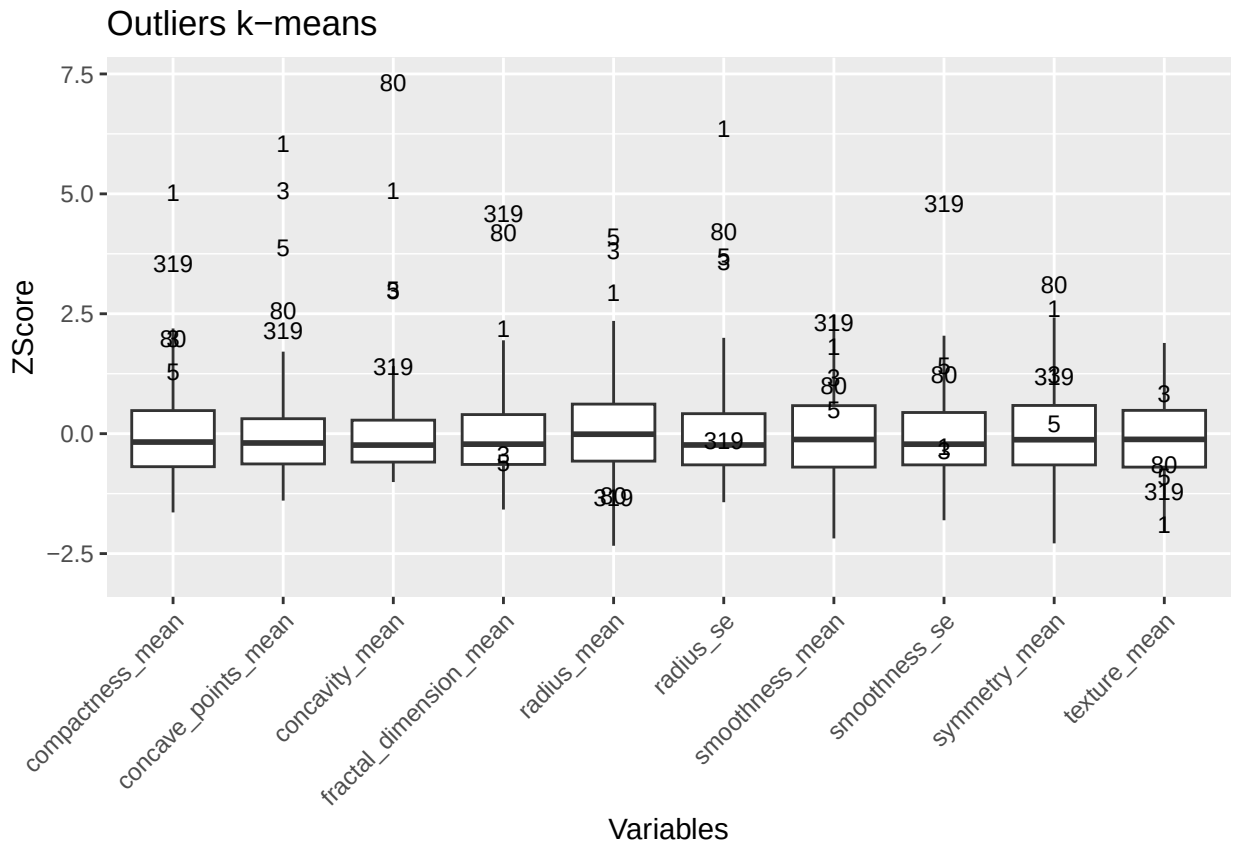
```
biplot_outliers_clustering(datos.num,
                           titulo = "Outliers k-means",
                           asignaciones.clustering = asignaciones.clustering.kmeans,
                           claves.outliers = claves.outliers.kmeans)
```



Podemos apreciar que cuatro de los cinco outliers detectados por este método están en el exterior de la nube de puntos, por lo que es muy posible que se hayan etiquetado como outliers porque tienen un valor muy alto en una o varias variables. Estas instancias son 1, 3, 5 y 80. Los registros 1, 3 y 5 ya los conocíamos porque eran outliers con respecto a 1 columna. Ahora han aparecido dos nuevos: 80 y 319.

Veamos el diagrama de cajas conjunto para hacernos una idea de los valores que toman:

```
diag_caja_juntos(datos.num, "Outliers k-means", claves.outliers.kmeans) + theme(axis.text.x = element_t
```



Parece ser que, efectivamente, los registros 80 y 319 han sido etiquetados como outliers porque tienen valores algo extremos (sin llegar a ser muy extremos) en varias variables, y tienen un valor muy extremo en una variable concreta (en este caso, en `fractal_dimension_mean`), por lo que la suma de los efectos de dichas variables (de forma individual) han podido determinar que tengan scores altos.

Por ejemplo, la instancia 1 tiene valores algo extremos en la mitad de variables. El registro que menos valores extremos posee es 5, pues solo destaca en la variable `radius_mean`, y no está tan alejado de la nube de puntos en el biplot. El diagrama de cajas también muestra que, efectivamente, no tiene un valores extremos en varias variables.

3.4.2 Clustering usando medoides (OPCIONAL)

En este apartado vamos a usar el método de clustering PAM (Partition around medoids):

```
set.seed(2)
matriz.distancias = dist(datos.num.zscore)
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.zscore[nombres.medoides, ]

nombres.medoides
```

```
## [1] "36"  "215" "35"
```

```
medoides
```

```
##     radius_mean texture_mean smoothness_mean compactness_mean concavity_mean
## 36        13.34        15.86         0.10780          0.15350        0.11690
## 215       12.21        18.02         0.09231          0.07175        0.04392
## 35        11.45        20.97         0.11020          0.09362        0.04591
##     concave_points_mean symmetry_mean fractal_dimension_mean radius_se
## 36              0.06987        0.1942                0.06902    0.2860
## 215             0.02027        0.1695                0.05916    0.2527
## 35              0.02233        0.1842                0.07005    0.3251
##     smoothness_se
## 36       0.006794
## 215      0.005833
## 35       0.010370
```

```
medoides.normalizados
```

```
##     radius_mean texture_mean smoothness_mean compactness_mean concavity_mean
## 36   0.55796225  -0.51943824      1.05252432        1.8284292     1.36337135
## 215 -0.02103872   0.02133855     -0.05442826       -0.2867307    -0.11749618
## 35  -0.41045530   0.75989945      1.22403408        0.2791231    -0.07711627
##     concave_points_mean symmetry_mean fractal_dimension_mean   radius_se
## 36            2.1373469     0.7424047              0.8309919 -0.04454649
## 215          -0.3706812    -0.2230399             -0.5543213 -0.30870638
## 35           -0.2665171     0.3515365              0.9757051  0.26562323
##     smoothness_se
## 36     -0.1322694
## 215    -0.4487060
## 35      1.0452302
```

Calculamos ahora los top outliers.

```
# COMPLETAR
distancias.outliers.medoides<- top_clustering_outliers(datos.num.zscore, asignaciones.clustering.pam, m
claves.outliers.pam <- distancias.outliers.medoides[[2]]
nombres.outliers.pam <- names(distancias.outliers.medoides[[1]][claves.outliers.pam])
```

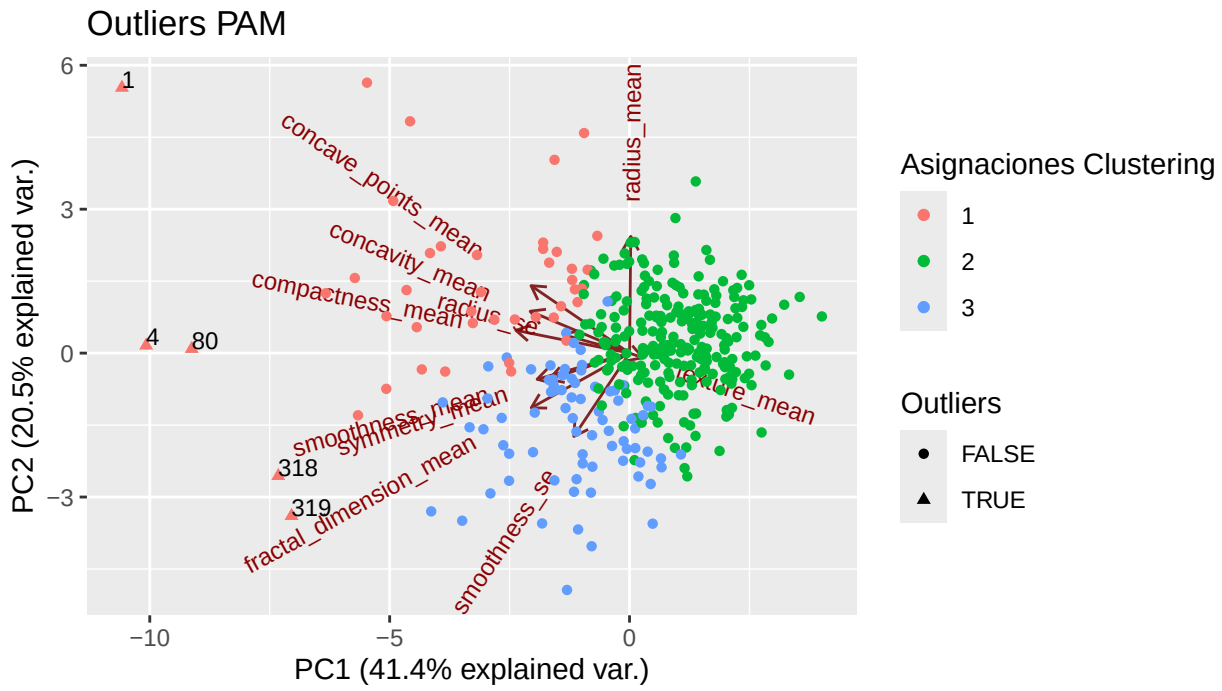
```
claves.outliers.pam
```

```
## [1]   1  80   4 319 318
```

```
nombres.outliers.pam
```

```
## [1] "1"   "80"  "4"   "319" "318"
```

```
biplot_outliers_clustering(datos.num,
                           titulo = "Outliers PAM",
                           asignaciones.clustering = asignaciones.clustering.pam,
                           claves.outliers = claves.outliers.pam)
```



Al usar la partición creada por pam, los outliers encontrados corresponden a registros que están en la periferia del biplot, es decir que son outliers porque tienen un valor extremo en alguna de las variables y no tanto porque tengan una combinación anómala de valores en distintas variables.

3.5 Análisis de los outliers multivariantes puros

Los outliers multivariantes puros son aquellos que no son outliers con respecto a una única variable, si no con respecto a varias. Ya hemos detectado varios de estos outliers en apartados anteriores. Por ejemplo, los registros 1, 3, y 5 han sido identificados como outliers por k-means por presentar valores extremos en varias variables. Por otro lado, los registros 1 y 3 fueron identificados por LOF como outliers por estar en zonas de muy baja densidad (aislados del resto de valores). Otros ejemplos, como 318 y 319 han sido clasificados como outliers por tener un valor extremo en una variable.

Vamos a automatizar la identificación de outliers puros (restando los outliers en alguna columna obtenidos mediante el método IQR):

```
# 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, claves.outliers.lof.no.IQR)
```

```
claves.outliers.IQR.en.alguna.columna
```

```
##  [1]   1   2   3   5   7  46 310 123 124 127 220 277 280 281 284 285 293 295 348
## [20] 357 359 360 362 365 366   4  33 318 329 367   6   9  10  55  64 180 255 319
## [39]  29  79  80  96 130 134 156 219 306  36  39 145 278 305 336  24  78 154 202
```

```
## [58] 258  31  56  74 321 344   8 104 149  53  59  93 100 132 144 177 198 231 251
## [77] 291
```

```
claves.outliers.lof
```

```
##  [1] 154 310 156  84 344 319   1  31 105   3
```

```
claves.outliers.lof.no.IQR
```

```
## [1]  84 105
```

```
nombres.outliers.lof.no.IQR
```

```
## [1] "84"  "105"
```

Antes habíamos visto en la gráfica de scores LOF que había un grupo de 3 registros con un score muy alto, y unos 7-8 registros con scores notablemente superiores al resto. Estos eran los registros 154, 310, 156, 84, 344, 319, 1, 31, 105 y 3. Analizamos de nuevo los resultados del método LOF aumentando el número total de outliers a 10. Calcularemos los registros que son outliers LOF pero no 1-variantes.

```
# COMPLETAR
num.outliers=10
claves.outliers.lof <- lof.scores.ordenados[1:num.outliers]
claves.outliers.lof.no.IQR <- setdiff(claves.outliers.lof, claves.outliers.IQR.en.alguna.columna)
nombres.outliers.lof.no.IQR <- nombres_filas(datos.num, claves.outliers.lof.no.IQR)
```

```
claves.outliers.IQR.en.alguna.columna
```

```
##  [1]   1   2   3   5   7  46 310 123 124 127 220 277 280 281 284 285 293 295 348
## [20] 357 359 360 362 365 366   4  33 318 329 367   6   9  10  55  64 180 255 319
## [39]  29  79  80  96 130 134 156 219 306  36  39 145 278 305 336  24  78 154 202
## [58] 258  31  56  74 321 344   8 104 149  53  59  93 100 132 144 177 198 231 251
## [77] 291
```

```
claves.outliers.lof
```

```
##  [1] 154 310 156  84 344 319   1  31 105   3
```

```
claves.outliers.lof.no.IQR
```

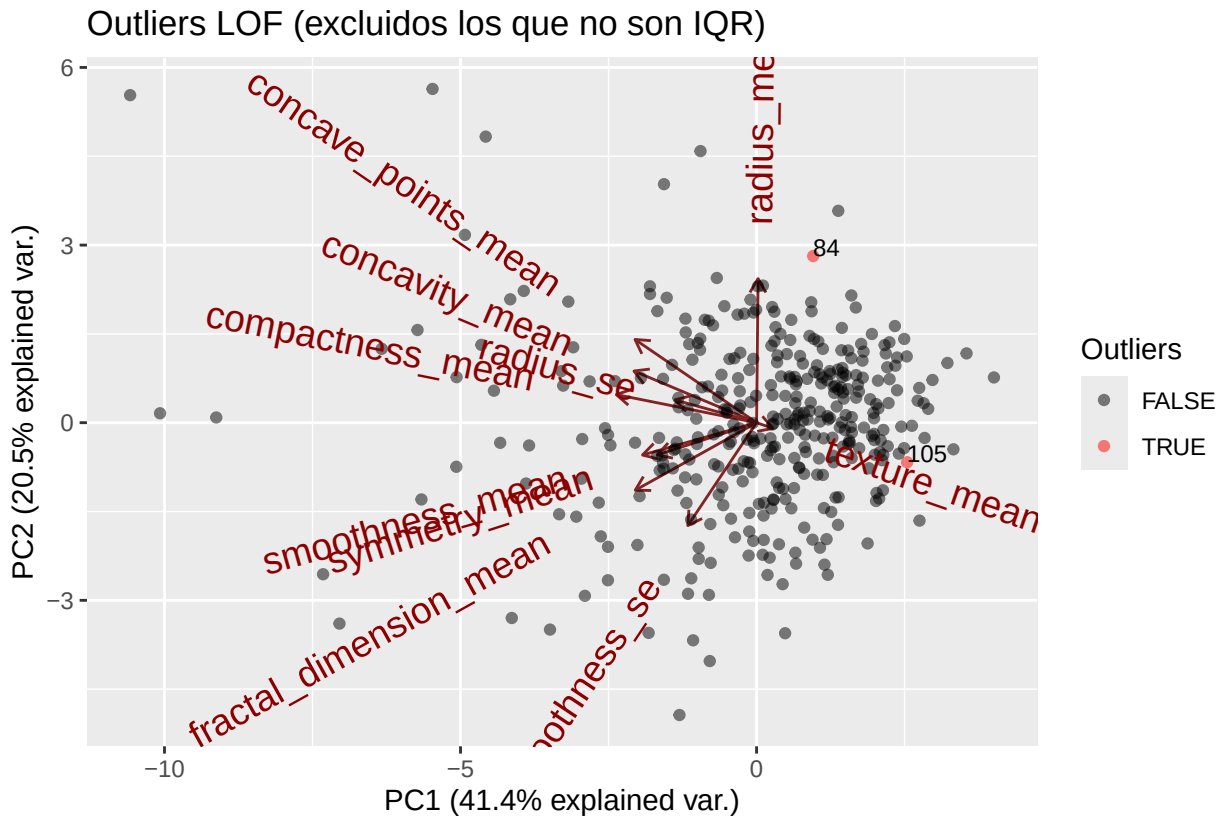
```
## [1]  84 105
```

```
nombres.outliers.lof.no.IQR
```

```
## [1] "84"  "105"
```

Mostramos el biplot de los outliers puros:

```
biplot.outliers.lof.no.IQR <- biplot_2_colores(datos.num, claves.outliers.lof.no.IQR, titulo="Outliers l
biplot.outliers.lof.no.IQR
```



El registro 105 parece haber sido seleccionado por presentar un valor alto en `texture_mean` y muy bajo en `concavity_mean` y `radius_se`, por lo que posiblemente el efecto sumado de estas variables haya contribuido a su alto score. El registro 84 parece tomar un valor alto en `radius_mean` y bajo en `smoothness_se`. Nos fijamos en el registro 272, que no parece tener un valor extremo en ninguna variaable, pues se sitúa en la zona central del biplot, de hecho en una zona de dispersión media.

Veamos los datos normalizados:

```
datos.num.zscore[claves.outliers.lof.no.IQR, ]
```

```
##     radius_mean texture_mean smoothness_mean compactness_mean concavity_mean
## 84     2.351328   0.38185641       -1.330747       -0.2743114     0.03631275
## 105   -1.296890   0.07141048       -1.684486       -1.5366803    -1.00869517
##     concave_points_mean symmetry_mean fractal_dimension_mean radius_se
## 84           0.00552303     0.3593538             -1.4647502 1.4856770
## 105         -1.39563538    -0.3872045              0.1917247 0.4940858
##     smoothness_se
## 84      -1.235681
## 105     -1.805333
```

Observamos que el valor de 105 en `texture_mean` es de hecho cercano a cero, y toma valores bajos en en `smoothness_se`, `compactness_mean` y `smoothness_mean`. El registro 272 toma un valor bastante extremo

en `symmetry_mean`, pero valores normales en las demás variables.
Concluimos que 105 y 84 son valores anómalos puros por tomar valores extremos en varias variables, pero 272 no lo es tanto.

4 Resumen final

4.1 Metodología

Objetivo. El objetivo del trabajo ha sido detectar outliers en un conjunto de datos de cáncer de mama. Asumimos que los datos no contienen errores de medición, por lo que el análisis se centra en identificar valores inusuales que aporten información relevante para la clasificación de tumores malignos y benignos. Para ello, se han utilizado técnicas 1-variantes (técnicas para encontrar datos con valores extremos con respecto a una única variable) y técnicas multivariantes (para encontrar datos con combinaciones anómalas de dos o más variables).

Preparación de datos. Inicialmente, hemos seleccionado únicamente las variables numéricas del dataset. Se eliminaron aquellas variables con baja variabilidad y se imputaron valores faltantes, si era necesario. También se realizó una normalización de los datos mediante el método z-score para estandarizar las escalas de las variables. Finalmente, eliminamos variables altamente correlacionadas para reducir la dimensionalidad y evitar redundancias.

Procedimientos aplicados. El análisis se dividió en técnicas univariantes y multivariantes:

1. **Análisis univariante**
   - **Método IQR:** Identificamos outliers en cada variable basándonos en el rango intercuartil. Los resultados se han representado mediante diagramas de caja para facilitar la visualización y el análisis. Vemos qué registros toman valores alejados de la media.
   - **Test de hipótesis:** Aplicamos el test de Grubbs para confirmar estadísticamente si los valores más alejados de la media eran outliers. El test tiene como hipótesis nula que los valores más extremos no son outliers. Este test exige que la variable siga una distribución normal, por lo que tuvimos que realizar un análisis de normalidad para validar los supuestos requeridos. Para confirmar este supuesto, hemos realizado tests de normalidad como Shapiro-Wilk. En aquellos casos donde la normalidad no ha sido rechazada, el test de Grubbs puede proporcionar información relevante para identificar outliers con evidencia estadística. En los casos donde se ha rechazado la normalidad, hemos interpretado los resultados del test con cierta cautela.
   - **Visualización:** Generamos gráficos para observar cómo se distribuyen los valores de las variables y los outliers detectados (diagramas de caja, biplots...).
2. **Análisis multivariante**
   - **Métodos estadísticos:** hemos utilizado la distancia de Mahalanobis para detectar outliers multivariantes en un conjunto reducido de variables clave. Este método, basado en una distribución multivariante normal, asigna puntuaciones a los registros en función de su distancia al centroide del grupo de datos. Para validar los resultados, hemos verificado previamente si las variables seleccionadas cumplían con el supuesto de normalidad multivariante.
   - **Métodos basados en densidad (LOF):** este método evalúa la densidad local de cada registro en comparación con sus vecinos. Los registros con puntuaciones altas de LOF se han etiquetado como outliers, ya que se encontraban en regiones de baja densidad del espacio multivariado. Este método ha sido útil para identificar registros aislados que no necesariamente presentaban valores extremos en variables individuales, pero sí en combinaciones específicas de ellas.
   - **Métodos basados en clustering:** Hemos aplicado el algoritmo de k-means para agrupar los datos en clústeres y detectar registros que se encontraban significativamente alejados de los centroides de sus respectivos clústeres. Estos registros se han considerado como outliers potenciales. El análisis de los clústeres nos ha permitido interpretar los patrones generales de los datos y ha destacado registros con combinaciones inusuales de características.

- En una etapa final, hemos explorado los **outliers multivariantes puros**, que aunque no han sido etiquetados como outliers univariantes, presentaban combinaciones de valores inusuales en varias variables. Este tipo de outliers es relevante porque reflejan casos en los que los registros no destacan en un único aspecto, sino en una combinación única de características. Asimismo, hemos detectado registros que, si bien tenían valores extremos en una única variable, su impacto en el análisis multivariante era menor.

4.2 Análisis de resultados

Conjunto de datos. El dataset original contenía 31 variables relacionadas con características de los tumores (como dimensiones, textura, compactación, entre otras), de las cuales hemos seleccionado una representación reducida tras identificar grupos de variables altamente correlacionadas (por ejemplo, `radius_mean`, `radius_worst`, y `radius_se` han sido representadas únicamente por `radius_mean`). Esta selección inicial nos ha facilitado la interpretación de los resultados.

Además, antes de aplicar los procedimientos, se han normalizado las variables numéricas mediante el método z-score para poder realizar comparaciones entre ellas, dado que tienen escalas distintas.

Resultados univariantes. Estudiamos los datos con respecto a sus valores en la primera variable: `radius_mean`.

- **Método IQR:** El análisis basado en el rango intercuartil ha identificado varios registros como valores extremos. Por ejemplo, el registro 2, correspondiente a un tumor con valores muy altos en `radius_mean` y `texture_mean` ha sido clasificado como un outlier extremo. Por otro lado, las instancias 1,3,5,7,46 y 310, aunque también tienen valores altos en las mismas variables no llegaron a clasificarse como outliers extremos. Los diagramas de caja nos han permitido destacar registros específicos. Por ejemplo, el tumor 2, con un valor extremadamente alto en `radius_mean`, resalta significativamente en su respectivo diagrama de caja, sugiriendo un posible tumor maligno con una extensión inusualmente grande.
- **Test de hipótesis:**
  Para validar estadísticamente los outliers identificados mediante IQR, hemos aplicado el test de Grubbs a cada variable. Sin embargo, debido a que ninguna variable cumple el supuesto de normalidad (según el test de Shapiro-Wilk), los resultados del test de Grubbs deben interpretarse con precaución. No obstante, todos los outliers fueron detectados como tal por el test de Grubbs. Estos outliers fueron 2, 124, 318, 4, 1 y 24.

Resultados multivariantes.

- **Visualización con biplot:** La suma de los porcentajes explicados es muy alta, explicando el 60% de la varianza acumulada, lo que ha permitido visualizar los registros en un espacio bidimensional. Hay registros claramente visibles en los extremos del biplot, como 1, 3, 5, 2, 7, 310, 4, 80...
- **Métodos estadísticos usando la distancia de Mahalanobis:** La distribución conjunta de las variables no es una normal multivariante. Por lo tanto, no deberíamos aplicar el método basado en la distancia de Mahalanobis. No obstante, lo hemos aplicado para ver si nos proporcionaba información relevante. Vimos que solo tenemos garantía estadística de que sea un outlier el que tiene mayor valor de distancia de Mahalanobis, que corresponde con la instancia 1. Este registro ya fue previamente etiquetado como outlier 1-variante en dos casos (variables `concave_points_mean` y `radius_se`) por el test de Grubbs. De todas formas, al no haber podido demostrar que la distribución sea normal multivariante, no podemos concluir con garantía estadística que sea outlier.
- **LOF:** El método LOF, basado en la densidad local, ha detectado registros interesantes que podrían ser indicadores de tumores malignos (al encontrarse aislados de los demás datos). Por ejemplo, el ejemplo 154 fue el ejemplo con score más alto detectado en una región de baja densidad, por tener valores muy

elevados en `smoothness_mean` y `symmetry_mean`. Otros registros destacables son 1 y 3, que ya los habíamos detectado como posibles outliers en otros apartados, con valores muy altos en `radius_mean` y `concave_points_mean`. En definitiva, el radio del tumor y su simetría parecen ser variables bastante determinantes a la hora de clasificar un tumor como outlier.

- **Métodos basados en clustering:** Los registros más alejados de los centroides indican posibles anomalías. Algos ejemplos a destacar al aplicar k-means son: 1, 80, 319, 3, 5. De estos, los que se encuentran más alejados de los clusters formados son 1, 3 y 5. Posteriormente, al aplicar clustering PAM, destacaron los registros 1, 80, 4, 219 y 318, que fueron clasificados como outliers porque tienen un valor extremo en alguna de las variables y no tanto porque tengan una combinación anómala de valores en distintas variables.
- **Análisis de los outliers multivariantes puros:** excluyendo los registros con altos scores en LOF que no toman valores altos en alguna variable por separado, nos hemos quedado con los registros 105 y 84, que son valores anómalos puros por tomar valores extremos en varias variables.

Conclusión

El análisis de los datos de cáncer de mama nos ha permitido identificar posibles outliers mediante técnicas tanto univariantes como multivariantes. En el análisis univariante, registros como el 1, 2, y 3 han destacado por valores excepcionalmente altos en variables como `radius_mean` y `texture_mean`, sugiriendo posibles tumores malignos debido a características como el tamaño y la textura de los tejidos. El método IQR ha identificado estos registros como valores extremos, y aunque el test de Grubbs confirmó algunos de ellos como outliers estadísticos, las pruebas de normalidad indican que no podemos afirmar que lo son.

Por otro lado, en el análisis multivariante, los biplots nos han permitido reducir la dimensionalidad y observar patrones claros, destacando registros como el 1, 3 y 5 en el biplot, con combinaciones únicas de valores que los separan del resto. Métodos basados en densidad, como LOF, han detectado registros como el 1, 3 y el 154, que se encontraban en regiones de baja densidad, lo que refuerza la hipótesis de que representan tumores con características atípicas. Asimismo, los algoritmos de clustering, como k-means, identificaron registros como el 1, 3, 4 y 5, alejados de los centroides. Finalmente, los análisis de outliers multivariantes puros destacaron registros como el 105 y 84, que, aunque no presentaban valores extremos en una única variable, mostraban combinaciones inusuales en múltiples dimensiones. En conjunto, estos resultados señalan una tendencia a identificar los outliers que los registros del 1 al 10, que de hecho son los outliers reales, ya que presentan características que los separan del comportamiento típico de los tumores analizados.