Fundamentos de la Ciencia de Datos Práctica 5

Fernández Díaz, Daniel
Cano Díaz, Francisco
Fernández Hernández, Alberto

10 de noviembre del 2019

Índice

1. Apartado 1 **3**
1.1. Caja y Bigotes . . . 3
1.2. Desviación Típica . . . 5
1.3. Regresión . . . 6
1.4. K-Vecinos . . . 7

2. Apartado 2 **9**
2.1. Análisis de incendios en la región de Amazonas, Brasil . . . 9
2.1.1. Datos Iniciales . . . 9
2.1.2. Detección de outliers con Caja y Bigotes . . . 10
2.1.3. Detección de outliers con Desviación Típica . . . 11
2.1.4. Detección de outliers con Regresión . . . 13
2.2. Otros algoritmos . . . 15
2.2.1. Algoritmo LOF para detección de *outliers* . . . 15
2.2.2. Extensión del algoritmo *LOF*: *LoOP* . . . 23

1. Apartado 1

La primera parte consistirá en la realización de un análisis de detección de **datos anómalos** con R aplicando todos los conceptos vistos en teoría. Aplicaremos los siguientes métodos:

1.1. Caja y Bigotes

Técnica **estadística** basada en la **ordenación**. La muestra que utilizaremos será la siguiente:

```
> muestra = t(matrix(c(3,2,3.5,12,4.7,4.1,5.2,4.9,7.1,6.1,6.2,5.2,14,5.3),2,7,
+ dimnames = list(c("r","d"))))
> # Convertimos a dataframe
> (muestra = data.frame(muestra))

     r    d
1  3.0  2.0
2  3.5 12.0
3  4.7  4.1
4  5.2  4.9
5  7.1  6.1
6  6.2  5.2
7 14.0  5.3
```

Para aplicar esta técnica a la muestra anterior seguiremos los siguientes pasos:

1. Determinación del **grado de *outliers*** o distancia a la que un suceso se considera un *outlier*. Para nuestro ejemplo utilizaremos el siguiente grado de *outlier*:

   ```
   > d = 1.5
   ```

2. **Ordenación** de los datos y obtención de los **cuartiles** utilizando las siguientes ecuaciones:

   $$\overline{X_c} = X_{[nc]+1} \text{ si nc} \notin N \text{ siendo [nc] parte entera de } n * c$$

   $$\overline{X_c} = \frac{X_{[nc]} + X_{[nc]+1}}{2} \text{ si nc } \in N$$

   Sacamos los cuartiles:

   ```
   > (cuar1 = quantile(muestra$r,0.25,names=FALSE))

   [1] 4.1

   > (cuar3 = quantile(muestra$r,0.75,names=FALSE))

   [1] 6.65
   ```

3. Se calcula los límites del **intervalo** para los valores atípicos utilizando la siguiente ecuación:

   $$(Q_1 - d * (Q_3 - Q_1), Q_3 + d * (Q_3 - Q_1))$$

   donde d es el grado de *outlier* definido anteriormente.

Sacamos el intervalo:

```
> (int = c(cuar1 - d*(cuar3 - cuar1),cuar1 + d*(cuar3 - cuar1)))

[1] 0.275 7.925
```

4. **Identificación** de los *ouliers* como los valores que quedan fuera del intervalo calculado en el paso anterior. Para ello utilizaremos el siguiente *script*:

```
> for(i in 1:length(muestra$r))
+         if(muestra$r[i]<int[1] || muestra$r[i]>int[2])
+                 print(sprintf("El suceso %d con resistencia %d es un suceso anómalo o outlier",
+                       i,muestra$r[i]))

[1] "El suceso 7 con resistencia 14 es un suceso anómalo o outlier"
```

Como podemos observar el único *outlier* de la muestra es el suceso 7 cuya resitencia es 14. Esto se debe principalmente a que este suceso es el único que se encuentra fuera del intervalo definido.
Además, en *R* existe una función denominada ***boxplot*** que realiza todo el proceso anterior mediante la siguiente llamada:

```
> # Parámetros:
> # 1. Muestra a utilizar.
> # 2. Grado de outlier.
> # 3. Flag para indicar si queremos o no que dibuje
> #    el diagrama de caja y bigotes.
> (boxplot(muestra$r,range=1.5,plot=FALSE))

$stats
     [,1]
[1,] 3.00
[2,] 4.10
[3,] 5.20
[4,] 6.65
[5,] 7.10

$n
[1] 7

$conf
         [,1]
[1,] 3.677181
[2,] 6.722819

$out
[1] 14

$group
[1] 1

$names
[1] "1"
```

¿Qué significa cada uno de estos parámetros de salida?

1. ***stats***: indica cada uno de los valores que definen el diagrama, es decir, el limite inferior, el cuartil 1, la mediana, el cuartil 3 y el limite superior.
2. ***n***: número de sucesos de la muestra.
3. ***conf***: intervalo de confianza alrededor de la mediana.
4. ***out***: el valor de cada suceso anómalo.
5. ***group***: clase a la que pertenecen cada uno de los datos anómalos.
6. ***names***: nombre de la clase a la que pertenecen cada uno de los datos anómalos.

Cómo podemos observar en el parámetro de salida *out* nos da un valor de 14 lo que corresponde con el suceso número 7 de la muestra, siendo el mismo suceso anómalo detectado anteriormente.

1.2. Desviación Típica

Técnica **estadística** basada en la **dispersión**. La muestra que utilizaremos será la que utilizamos anteriormente. Para aplicar esta técnica a la muestra seguiremos los siguientes pasos:

1. Determinación del **grado de *outliers*** o distancia a la que un suceso se considera un *outlier*. Para nuestro ejemplo utilizaremos el siguiente grado de *outlier*:

   ```
   > d = 2
   ```

2. Obtención de la **media aritmética** utilizando la siguiente ecuación:

   $$\overline{X_a} = \frac{\sum_{i=1}^{n} f_i * x_i}{\sum_{i=1}^{n} f_i}$$

   ```
   > (media = mean(muestra$d))

   [1] 5.657143
   ```

3. Obtención de la **desviación típica** utilizando la siguiente ecuación:

   $$S_a = \sqrt{\frac{\sum_{i=1}^{n} f_i * (x_i - \overline{x_a})^2}{\sum_{i=1}^{n} f_i}}$$

   ```
   > # No utilizamos la función sd de R ya que es para poblaciones y divide entre n-1
   > # por lo que la corregimos para que se divida entre n.
   > (sdd = sqrt((var(muestra$d))*((length(muestra$d)-1) / (length(muestra$d)))))

   [1] 2.857
   ```

4. Se calcula los límites del **intervalo** para los valores atípicos utilizando la siguiente ecuación:

   $$(\overline{X_a} - d * S_a, \overline{X_a} + d * S_a)$$

   ```
   > (intdes = c(media - d*sdd,media + d*sdd))

   [1] -0.05685714 11.37114285
   ```

Cómo podemos observar el intervalo comienza en un número negativo por lo que lo modificaremos y lo estableceremos en 0 ya que no podemos tener una densidad negativa.

```
> intdes[1] = 0
> intdes

[1]  0.00000 11.37114
```

5. Se **identifican** los *outliers* como los valores que quedan fuera del intervalo calculado en el paso anterior. Para ello utilizaremos el siguiente *script*:

```
> for(i in 1:length(muestra$d))
+         if(muestra$d[i]<intdes[1] || muestra$d[i]>intdes[2])
+                 print(sprintf("El suceso %d con densidad %d es un suceso anómalo o outlier",
+                       i,muestra$d[i]))

[1] "El suceso 2 con densidad 12 es un suceso anómalo o outlier"
```

Como podemos observar el único *outlier* de la muestra es el suceso 2 cuya densidad es 12. Esto se debe principalmente a que este suceso es el único que se encuentra fuera del intervalo definido.

1.3. Regresión

Técnica **estadística** basada en el **error estándar de los residuos**. La muestra que utilizaremos será la que utilizamos anteriormente. Para aplicar esta técnica a la muestra seguiremos los siguientes pasos:

1. Determinación del **grado de *outliers*** o distancia a la que un suceso se considera un *outlier*. Para nuestro ejemplo utilizaremos el siguiente grado de *outlier*:

```
> d = 2
```

2. Obtención de la **regresión lineal** utilizando las siguientes ecuaciones:

$$S_{xy} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} x_i * y_i * f_{ij}}{\sum_{i=1}^{n} f_i} - \overline{x} * \overline{y}; r_{xy} = \frac{S_{xy}}{S_x * S_y}; b = \frac{S_{xy}}{S_x^2} = r_{xy} * \frac{S_y}{S_x}; a = \overline{y} - b * \overline{x}$$

```
> (dfr = lm(muestra$d~muestra$r))

Call:
lm(formula = muestra$d ~ muestra$r)

Coefficients:
(Intercept)    muestra$r
    6.01445     -0.05723
```

3. Obtención del **error estándar de los residuos** utilizando la siguiente ecuación:

$$S_r = \sqrt{\frac{\sum_{i=1}^{n}(y_i - y_{ci})^2}{n}}$$

```
> # Obtenemos los residuos de la variable dfr
> (res = summary(dfr)$residuals)
```

```
         1          2          3          4          5          6          7
-3.8427477  6.1858698 -1.6454482 -0.8168308  0.4919157 -0.4595958  0.0868370

> # Error estándar de los residuos
> (sr = sqrt(sum(res^2)/7))

[1] 2.850242
```

4. Se calcula los límites del **intervalo** para los valores atípicos utilizando la siguiente ecuación:

$$d * sr$$

5. Se **identifican** los *outliers* como aquellos tales que:

$$|y_i - y_{ci}| > d * sr$$

```
> for(i in 1:length(res))
+         if(res[i]>d*sr)
+                 print(sprintf("El suceso %d, (%.1f,%d) es un suceso anómalo o outlier",
+                       i,muestra$r[i],muestra$d[i]))

[1] "El suceso 2, (3.5,12) es un suceso anómalo o outlier"
```

Como podemos observar el único *outlier* de la muestra es el suceso 2 cuya resistencia es 3.5 y densidad 12. Esto se debe principalmente a que este suceso es el único que se encuentra fuera del intervalo definido.

1.4. K-Vecinos

Técnica **estadística** basada en la **proximidad**. La muestra que utilizaremos será la siguiente:

```
> (muestra = matrix(c(4,4,4,3,5,5,1,1,5,4),2,5))

     [,1] [,2] [,3] [,4] [,5]
[1,]    4    4    5    1    5
[2,]    4    3    5    1    4

> (muestra = t(muestra))

     [,1] [,2]
[1,]    4    4
[2,]    4    3
[3,]    5    5
[4,]    1    1
[5,]    5    4
```

Para aplicar esta técnica a la muestra anterior seguiremos los siguientes pasos:

1. Determinación del **grado de *outliers*** o distancia a la que un suceso se considera un *outlier*. Para nuestro ejemplo utilizaremos el siguiente grado de *outlier*:

   ```
   > d = 2.5
   ```

2. Determinación del número de orden, o $\boldsymbol{k}$. Para nuestro ejemplo utilizaremos:

```
> # Al comenzar en 0 deberemos poner k+1
> k=4
```

3. Cálculo de las **distancias euclídeas** entre todos los puntos:

```
> (distancias = as.matrix(dist(muestra)))

         1        2        3        4        5
1 0.000000 1.000000 1.414214 4.242641 1.000000
2 1.000000 0.000000 2.236068 3.605551 1.414214
3 1.414214 2.236068 0.000000 5.656854 1.000000
4 4.242641 3.605551 5.656854 0.000000 5.000000
5 1.000000 1.414214 1.000000 5.000000 0.000000

> (distancias = matrix(distancias,5,5))

         [,1]     [,2]     [,3]     [,4]     [,5]
[1,] 0.000000 1.000000 1.414214 4.242641 1.000000
[2,] 1.000000 0.000000 2.236068 3.605551 1.414214
[3,] 1.414214 2.236068 0.000000 5.656854 1.000000
[4,] 4.242641 3.605551 5.656854 0.000000 5.000000
[5,] 1.000000 1.414214 1.000000 5.000000 0.000000
```

4. **Ordenación** por distancia de los vecinos de cada punto hasta llegar al k

```
> for(i in 1:5)
+     # Para cada columna lo ordeno en función de las filas
+         distancias[,i] = sort(distancias[,i])
> (distancias.ordenadas = distancias)

         [,1]     [,2]     [,3]     [,4]     [,5]
[1,] 0.000000 0.000000 0.000000 0.000000 0.000000
[2,] 1.000000 1.000000 1.000000 3.605551 1.000000
[3,] 1.000000 1.414214 1.414214 4.242641 1.000000
[4,] 1.414214 2.236068 2.236068 5.000000 1.414214
[5,] 4.242641 3.605551 5.656854 5.656854 5.000000
```

5. **Identificación** de los *outliers* como aquellos sucesos cuyo k vecinos se encuentra a una distancia mayor que el grado *outlier* definido.

```
> for(i in 1:5)
+         if(distancias.ordenadas[4,i]>d)
+                 print(sprintf("El suceso %d es un suceso anómalo o outlier",i))

[1] "El suceso 4 es un suceso anómalo o outlier"
```

Como podemos observar el único *outlier* de la muestra es el suceso 4. Esto se debe principalmente a que la distancia del tercer vecino más cercano al suceso 4 es mayor que el grado de *outlier* definido: 5>2.5

2. Apartado 2

La segunda parte consistirá en la realización de un análisis de detección de **datos anómalos** con *R* aplicando modificaciones sobre lo visto anteriormente.

2.1. Análisis de incendios en la región de Amazonas, Brasil

En este apartado realizaremos un análisis de detección de **datos anómalos** utilizando un *dataset* que contiene el número de incendios forestales de Brasil clasificados por región en un período de 10 años.[1] Una vez descargado el *dataset* procederemos a leerlo mediante el siguiente comando:

```
> incendios = read.csv("forest_fires_brazil.csv")
> # Numero de datos
> (nrow(incendios))

[1] 6454

> # Número de regiones
> (length(unique(incendios$state)))

[1] 23
```

Debido a la gran cantidad de regiones que tenemos en el *dataset* nos quedaremos con la región del Amazonas.

```
> # Número de incendios en el Amazonas
> (sum(incendios$state=="Amazonas"))

[1] 239

> # Obtenemos los datos que vamos a utilizar en el analisis
> incendios = subset(incendios,incendios$state=='Amazonas')
> incendios = incendios[order(incendios$year),]
> incendios.data = incendios$number
> incendios.year = unique(incendios$year)
> incendios.month = incendios$month
```

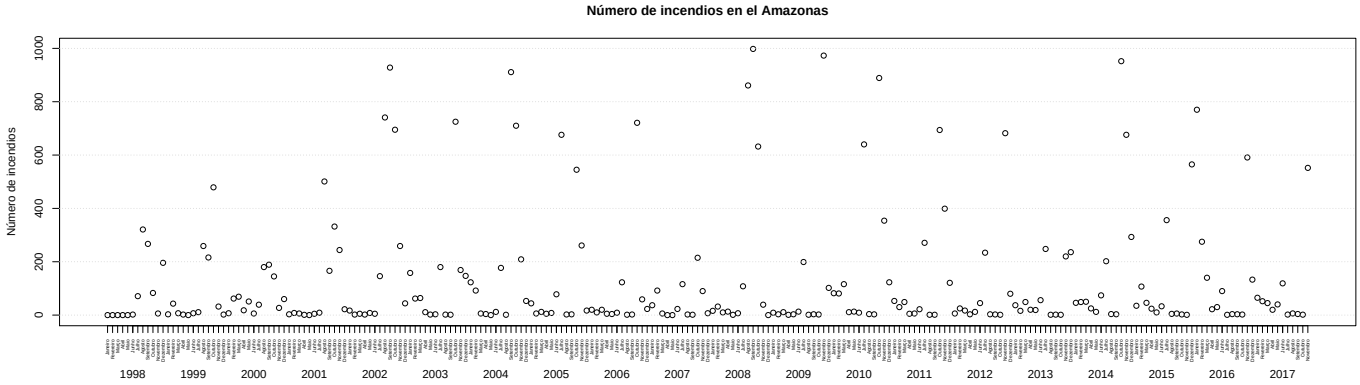
2.1.1. Datos Iniciales

Veamos gráficamente los datos que tenemos:

```
> plot(incendios.data,xlab="",ylab="Número de incendios",main="Número de incendios en el Amazonas",
+         xaxt='n')
> axis(1,at = 1:length(incendios.month),labels = as.character(incendios.month),las=2,cex.axis=0.4)
> axis(1,at=((1:length(incendios.year))-1)*12+6,labels = as.character(incendios.year),mgp=c(3,3,0))
> grid(NA,NULL)
```

Como podemos observar, contamos con datos de incendios de 20 años: desde el 1998 hasta el 2017. La cantidad de incendios, por regla general, suele estar entre los 400 en la segunda mitad del año y ningún incendio en meses de la primera mitad. Parece ser una tendencia a lo largo de esos 20 años que puede deberse a una mayor temperatura en esos meses, épocas de sequía, etc.

[1] https://www.kaggle.com/gustavomodelli/forest-fires-in-brazil



Pero además, encontramos algunas cantidades excesivamente altas de incendios cada cierto tiempo, por encima de los 500 incendios en un mes, que es donde nos vamos a centrar.

El objetivo de este apartado es utilizar técnicas de detección de outliers para encontrar sucesos aislados o anómalos en grandes cantidades de datos, no para descartarlos. A menudo son estos hechos aislados los que mayor información pueden aportar a un estudio, todo depende, claro, de nuestro problema en ese momento.

En estos casos el análisis de outliers también puede ser muy útil. En caso de contar con una gran cantidad de datos podemos estar más interesados en estos outliers o datos anómalos que pueden indicar algún hecho o suceso sobre el que prestar atención por salirse de lo "normal". Es el caso de los incendios: puede haber habido un período de tiempo con las condiciones climáticas adecuadas, una disminución de la "limpieza"del bosque (maleza, hojarasca, leña seca, etc.), o incluso un aumento de la cantidad de pirómanos.

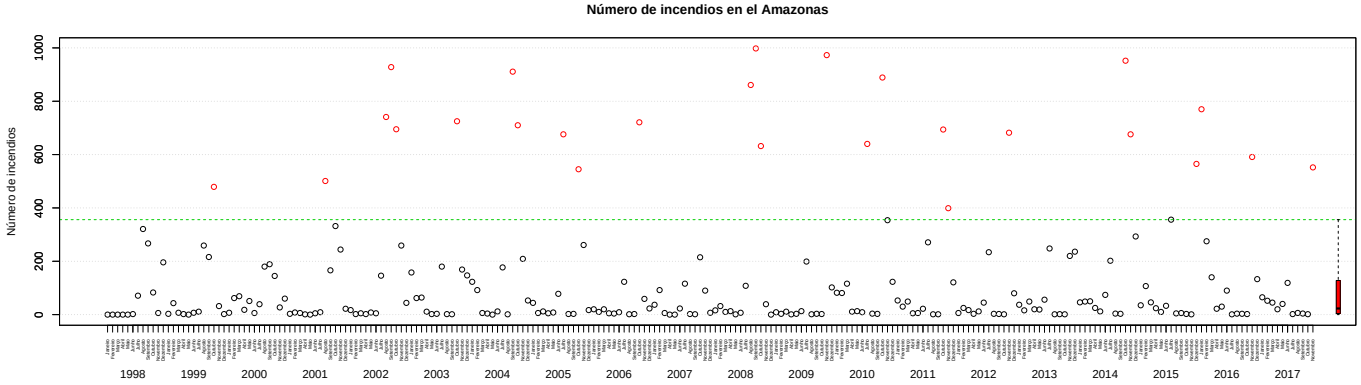
Otro ejemplo de este fenómeno puede ser a la hora de analizar los movimientos en una cuenta bancaria: la detección de outliers puede ser una buena forma de detectar el robo de una tarjeta, un fraude, etc.

2.1.2. Detección de outliers con Caja y Bigotes

Utilizaremos en primer lugar el método estadístico de Caja y Bigotes visto en el apartado anterior. Para ello usaremos de nuevo el comando **boxplot**.

```
> bp = boxplot(incendios.data,range=2,plot=FALSE)

Análisis de Caja y bigotes
        Limite inferior: 0.000000
        Primer cuartil: 4.016500
        Mediana: 23.000000
        Tercer cuartil: 128.000000
        Limite superior: 356.000000

Outliers:

479 501 741 928 695 725 911 710 676 545 721 861 998 632 973 640 889 694 399 682
952 676 565 770 591 552
```

```
> aux = as.data.frame(cbind(data=incendios.data,month=1:length(incendios.data)))
> normales = subset(aux,aux$data <= bp$stats[5,1])
> anomalos = subset(aux,aux$data > bp$stats[5,1])
> plot(normales$month,normales$data,xlab="",ylab="Número de incendios",
+ main="Número de incendios en el Amazonas",xaxt='n',col=1,ylim=c(0,max(incendios.data)))
> points(anomalos$month,anomalos$data,col=2)
> axis(1,at = 1:length(incendios.month),labels = as.character(incendios.month),las=2,cex.axis=0.4)
> axis(1,at=((1:length(incendios.year))-1)*12+6,labels = as.character(incendios.year),mgp=c(3,3,0))
> grid(NA,NULL)
> abline(h=bp$stats[5,1],col=3,lty='dashed')
> boxplot(incendios.data,range=2,add=T,outline=F,at=length(incendios.data)+5,varwidth=T,col=2)
```



Como resultado del análisis, obtenemos una serie de *outliers*. Estos no son datos erroneos ni tampoco deben ser eliminados. Son mediciones válidas pero que, gracias al análisis de outliers hemos podido extraer y destacar en rojo de una gran cantidad de datos.

Podemos ver, por ejemplo, que en el mes de noviembre del año 2017 hubo una cantidad de incendios inusitadamente alta (552 incendios), tal y como reflejan los medios de comunicación.

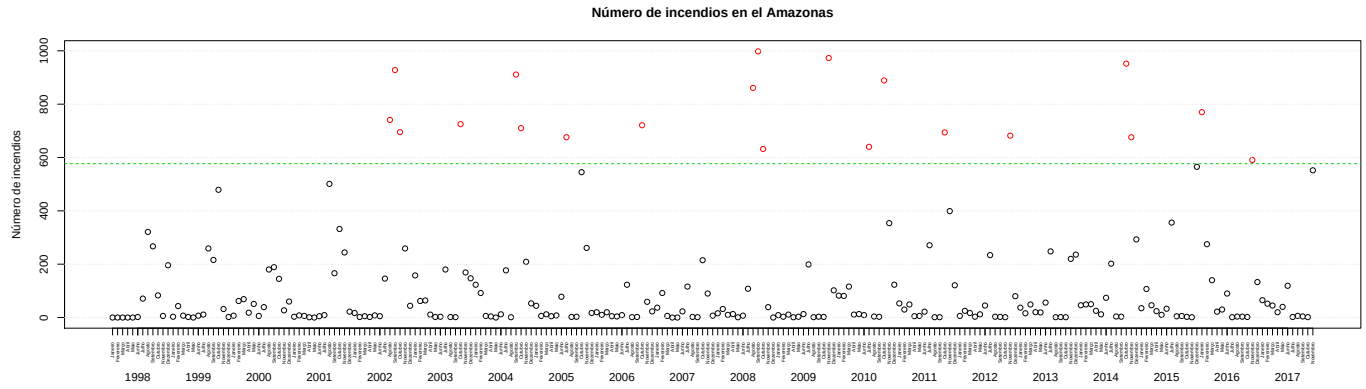
En la gráfica hemos representado además el diagrama de **caja y bigotes** resultante a la derecha de la gráfica, así como una línea verde orientativa que indica el final del rango fuera del cual consideramos los sucesos como outliers.

2.1.3. Detección de outliers con Desviación Típica

Utilizaremos ahora el método estadístico de detección de outliers que hace uso de la **desviación típica** para comprobar que, efectivamente podemos obtener resultados similares e intentar comparar este método y el anterior.

```
> # Determinacion del grado de outlier
> d = 2
> # Media aritmetica
> media = mean(incendios.data)
> # Desviacion tipica
> sdd = sqrt(var(incendios.data)*((length(incendios.data)-1)/(length(incendios.data))))
```

```
> # Limites del intervalo
> liminf = media - (d * sdd)
> if(liminf < 0) liminf = 0
> limsup = media + (d * sdd)
> aux = as.data.frame(cbind(data=incendios.data,month=1:length(incendios.data)))
> normales = subset(aux,aux$data <= limsup)
> anomalos = subset(aux,aux$data > limsup)

Análisis de Desviacion Tipica
        Grado de outlier: 2.000000
        Media aritmetica: 128.243218
        Desviacion tipica: 224.268314
        Limite inferior: 0.000000
        Limite superior: 576.779847

Outliers:

741 928 695 725 911 710 676 721 861 998 632 973 640 889 694 682 952 676 770 591


> plot(normales$month,normales$data,xlab="",ylab="Número de incendios",
+ main="Número de incendios en el Amazonas",xaxt='n',col=1,ylim=c(0,max(incendios.data)))
> points(anomalos$month,anomalos$data,col=2)
> axis(1,at = 1:length(incendios.month),labels = as.character(incendios.month),las=2,cex.axis=0.4)
> axis(1,at=((1:length(incendios.year))-1)*12+6,labels = as.character(incendios.year),mgp=c(3,3,0))
> grid(NA,NULL)
> abline(h=limsup,col=3,lty='dashed')
```



Como podemos comprobar, con un mismo grado de outlier $\mathbf{d = 2}$, ambos métodos crean intervalos diferentes y, por tanto, se seleccionan más o menos sucesos como outliers. En este caso el rango para considerar un suceso como *normal* es mayor, por lo que tenemos menos outliers que cuando lo hemos resuelto con el método de Caja y Bigotes. Podemos ver que, efectivamente, la línea verde indicando el límite superior del intervalo se sitúa ahora más arriba.

En este caso, el suceso de 552 incendios en el mes de Noviembre del año 2017 ya no se considera outlier. Ahora sería necesario mirar nuestro problema y preguntarnos si ese suceso en concreto fue anómalo o no. En caso de serlo sería necesario probar de nuevo este método con un **grado de outlier** menor.

Por eso en cada problema es recomendable tener algunos sucesos de control que sepamos de antemano si son outliers o no para así poder ayudarnos a determinar correctamente el grado de outlier o, en caso de no tener, extraer algunos resultados y comprobar si tiene sentido que sean outlier o no dentro del dominio del problema. Esta técnica es efectiva por grande que sea el conjunto de datos.

2.1.4. Detección de outliers con Regresión

Por último, utilizaremos el método estadístico de detección de outliers por medio de la **Regresión** y el **Error estándar de los residuos**.

```
> aux = as.data.frame(cbind(data=incendios.data,month=1:length(incendios.data)))
> # Determinacion del grado de outlier
> d = 2
> # Regresion lineal
> dfr = lm(aux$data ~ aux$month)
> a = (dfr$coefficients)[1]
> b = (dfr$coefficients)[2]
> # Residuos
> res = summary(dfr)$residuals
> # Error estandar de los residuos
> sr = sqrt(sum(res^2)/length(res))
> # Limites del intervalo
> limres = d * sr
> normales = subset(aux,res <= limres)
> anomalos = subset(aux,res > limres)
```

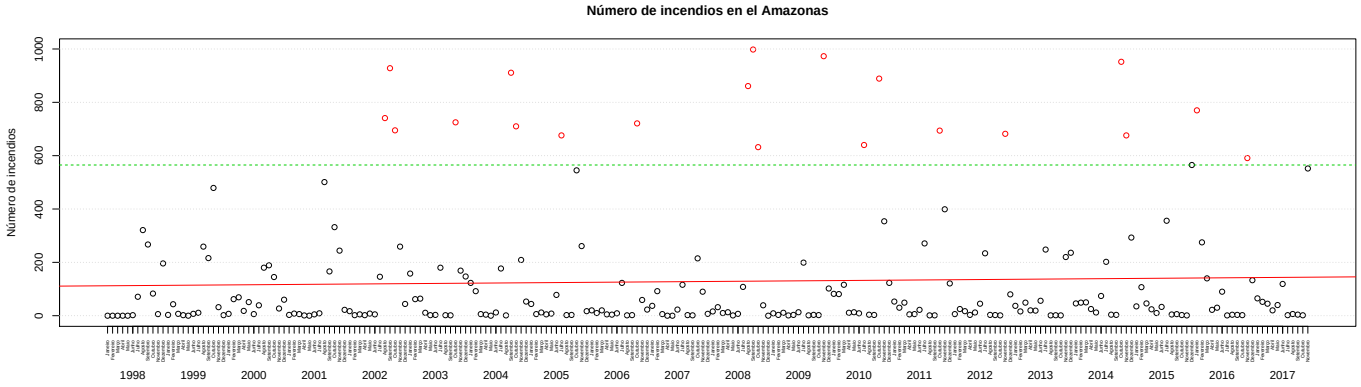
```
Análisis de Desviacion Tipica
        Grado de outlier: 2.000000
        Coeficientes:
                a = 112.057685
                b = 0.134879
        Desviacion tipica: 224.268314
        Limite d * sr: 448.150335
        Valor normal maximo: 565.000000

Outliers:

741 928 695 725 911 710 676 721 861 998 632 973 640 889 694 682 952 676 770 591
```

Lo mejor del método por regresión es que, además de la línea verde con el límite del intervalo, disponemos también de la **regresión lineal** en sí (línea roja). Con ella podemos ver la tendencia de los datos o la zona de mayor concentración. Vemos que en nuetros datos esta está bastante abajo, indicando que la mayoría de meses la cantidad de incendios es baja o nula.

Comentar también como aquí para un mismo $\mathbf{d = 2}$ ha vuelto a cambiar la frontera, siendo esta ahora 565. Una vez más tendremos un intervalo diferente y, por tanto, un conjunto diferente de sucesos marcados como outliers. Una vez más, dependerá de nuestro problema el considerar modificar o no ese grado de outlier.

Por último, y gracias a la recta de regresión, hay una conclusión más que podemos extraer con este método:

```
> plot(normales$month,normales$data,xlab="",ylab="Número de incendios",
+ main="Número de incendios en el Amazonas",xaxt='n',col=1,ylim=c(0,max(incendios.data)))
> points(anomalos$month,anomalos$data,col=2)
> axis(1,at = 1:length(incendios.month),labels = as.character(incendios.month),las=2,cex.axis=0.4)
> axis(1,at=((1:length(incendios.year))-1)*12+6,labels = as.character(incendios.year),mgp=c(3,3,0))
> grid(NA,NULL)
> abline(h=max(normales$data),col=3,lty='dashed')
> abline(a=a,b=b,col=2)
```



Aunque es pequeña, la pendiente de la recta es positiva. Esto se traduce en una tendencia creciente en el número de incendios en Amazonas desde 1998 hasta 2017. Esto es bastante preocupante ya que, o bien cada vez hay más meses con un número alto de incendios o bien en los meses que típicamente hay un número excesivamente alto de incendios esta cifra ha crecido con el paso de los años. Ninguno de los dos escenarios es deseable y este y otros estudios pueden ayudar a encontrar una tendencia, determinar si realmente hay un problema y, de ser así intentar ponerle solución. De momento con nuestro estudio de valores anómalos solo podríamos ver si se cumple el segundo escenario: analizar si las cantidades anómalas o extraordinarias de incendios son cada vez más altas con el paso de los años.

2.2. Otros algoritmos

2.2.1. Algoritmo LOF para detección de *outliers*

Con el objetivo de detectar si un valor se encuentra **alejado con respecto al resto de puntos de un *dataset*, utilizaremos el algoritmo *LOF* (*Local Outlier Factor*), basado en el algoritmo de los K-Vecinos**. Se trata de un algoritmo **basado en la densidad de los vecinos**, es decir, **detectar en qué áreas existe una mayor concentración de puntos y en qué zonas existen escasos puntos o una mayor separación entre ellos**.
El algoritmo *LOF* se basa en el concepto de **densidad local**, es decir, la **densidad de los *k* vecinos** o **la distancia típica a la que un punto es alcanzable desde sus *k* vecinos**. De este modo, comparando las densidades locales entre cada uno de sus vecinos, podremos detectar **regiones con valores de densidad similares, así como puntos con densidades significativamente pequeñas con respecto al resto de vecinos**. En este último caso, los clasificaremos como *outliers*.

Estructura del algoritmo : Sea $k\_distancia(A)$ **la distancia de un punto A a su *k* vecino más cercano**. Por otro lado, denotaremos como $N_k(A)$ al **conjunto de los *k* vecinos más cercanos al punto A**.
En primer lugar, debemos calcular la **distancia de alcance** de nuestro punto A a cada uno de sus k vecinos más cercanos. La distancia de alcance de un punto A a un punto B (k vecino de A) es la distancia **verdadera** entre ambos puntos:

$$distancia\_alcance_k(A, B) = max\{k\_distancia(B), d(A, B)\}$$

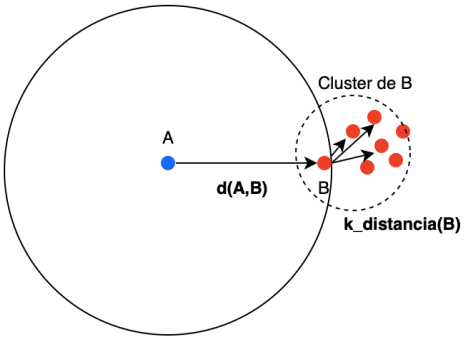


Figura 1: Ejemplo del calculo de la distancia de alcance entre un punto A y un punto B

Es decir, supongamos que B pertenece a un *cluster*, luego el objetivo es detectar la distancia máxima entre cualquier punto del *cluster de B* con respecto al punto A, detectando de este modo a qué distancia (como máximo) se puede alcanzar el punto A.
A continuación, debemos calcular la **densidad local**. Este valor permite obtener una estimación de la distancia a partir de la cual un punto puede ser encontrado por sus vecinos [2]. Para ello, calculamos en primer lugar la **distancia de alcance media entre el punto A y sus *k* vecinos**:

$$\frac{\sum_{B \in N_k(A)} distancia\_alcance_k(A, B)}{|N_k(A)|}$$

[2] `https://medium.com/@doedotdev/local-outlier-factor-example-by-hand-b57cedb10bd1`

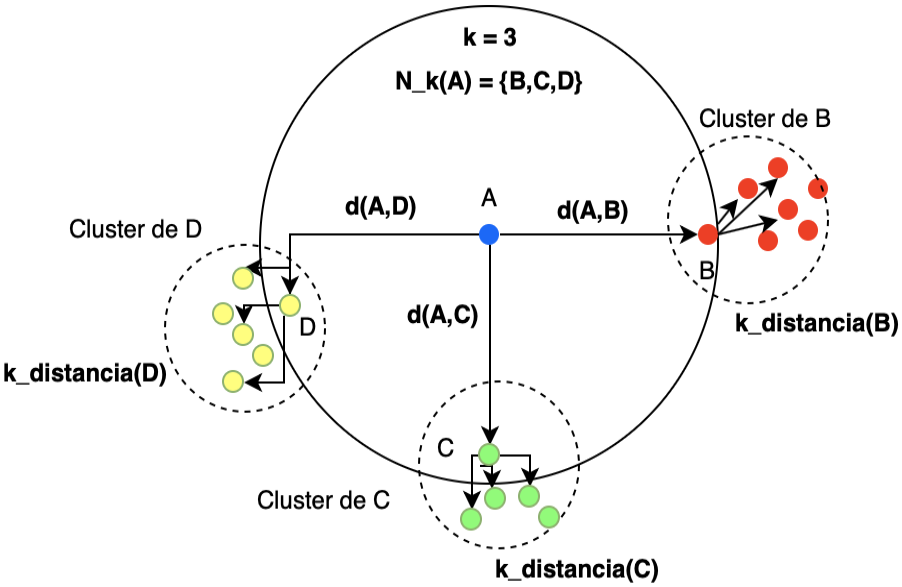


Figura 2: Ejemplo del calculo de la densidad local entre un punto A y sus k vecinos: {B,C,D}

Lo que acabamos de calcular es la **distancia media con la cual el punto A puede alcanzar cualquiera de sus vecinos**. Sin embargo, lo que queremos calcular es justo lo contrario, es decir, la **distancia desde la cual el punto A pueda ser alcanzado por sus vecinos**. Por lo tanto, la densidad local será la inversa de la media calculada:

$$densidad\_local_k(A) = \frac{1}{\frac{\sum_{B \in N_k(A)} distancia\_alcance_k(A,B)}{|N_k(A)|}}$$

Ya tenemos la distancia media con la cual un punto A puede ser alcanzado por sus vecinos. No obstante, cada uno de los k vecinos de A tendrá un conjunto de puntos vecinos, luego debemos calcular la **la media de densidad de cada uno de sus *k* vecinos en relación con la densidad local de A**, obteniendo de este modo el *Local Outliter Factor* (LOF).

$$LOF_k(A) = \frac{\frac{\sum_{B \in N_k(A)} densidad\_local_k(B)}{|N_k(A)|}}{densidad\_local_k(A)}$$

- LOF(k) $\sim$ 1 Implica que todos los puntos presentan la misma densidad, por lo que no se trata de un *outlier*.
- LOF(k) <1 implica que la densidad media local del punto A es mayor que la media de densidad local de sus k vecinos. Al igual que en el caso anterior, no se trata de un *outlier*.
- LOF(k) >1 implica que la densidad media local del punto A es menor que la media de densidad local de sus k vecinos. En este último caso, se trata de un *outlier*.

Caso práctico: detección de *outliers* con las estadísticas de tendencias en YouTube : En este apartado analizaremos la detección posibles *outliers* sobre vídeos etiquetados como **tendencias** en **YouTube**, concretamente en la región de Estados Unidos. [3]

[3] `https://www.kaggle.com/datasnaek/youtube-new`

En primer lugar, importamos el fichero *csv*:

```
> listado.videos <- read.csv("USvideos.csv")
> colnames(listado.videos)
```

```
 [1] "video_id"               "trending_date"          "title"                  "channel_title"
[10] "dislikes"               "comment_count"          "thumbnail_link"         "comments_disabled"
```

El *dataset* contiene un total de 16 columnas, entre las que destacan:

1. Número de visualizaciones (*views*)
2. Número de *me gusta* (*likes*)
3. Número de *no me gusta* (*dislikes*)
4. Número de comentarios (*comment_count*)

Para esta práctica, **analizaremos los outliers de las columnas anteriores**, realizando todas las posibles combinaciones fila-columna:

- **Número de visualizaciones-número de *likes***
- **Número de visualizaciones-número de *dislikes***
- **Número de visualizaciones-número de comentarios**

Por tanto, eliminaremos el resto de columnas del *dataset*:

```
> listado.videos2 <- listado.videos[,c(8,9,10,11)]
> head(listado.videos2)
```

```
    views  likes dislikes comment_count
1  748374  57527     2966         15954
2 2418783  97185     6146         12703
3 3191434 146033     5339          8181
4  343168  10172      666          2146
5 2095731 132235     1989         17518
6  119180   9763      511          1434
```

```
> # Numero de columnas
> nrow(listado.videos2)
```

```
[1] 40949
```

Dado el elevado número de columnas que presenta el *dataframe*, lo reducimos a 10.000:

```
> # Reducimos el numero de columnas a 10000
> listado.videos2 <- head(listado.videos2,10000)
```

Una vez reducido el tamaño del *dataframe*, comenzamos con el **análisis de datos anómalos**. Para ello, `R` dispone del paquete *DMwR*, el cual contiene la función *lofactor*, que devuelve un vector **con el factor de *outlier* de cada fila**, dado un *dataframe* de entrada:

```
> par(mfrow=c(2,2))
> plot(listado.videos2$views, main = "Numero de visitas", xlabel = "Fila",
+ ylabel = "Visitas")
> plot(listado.videos2$likes, main = "Numero de \"me gusta\"", xlabel = "Fila",
+ ylabel = "Nº \"Me gusta\"")
> plot(listado.videos2$dislikes, main = "Numero de \"no me gusta\"", xlabel = "Fila",
+ ylabel = "Nº \"No me gusta\"")
> plot(listado.videos2$comment_count, main = "Numero de comentarios", xlabel = "Fila",
+ ylabel = "Comentarios")
```

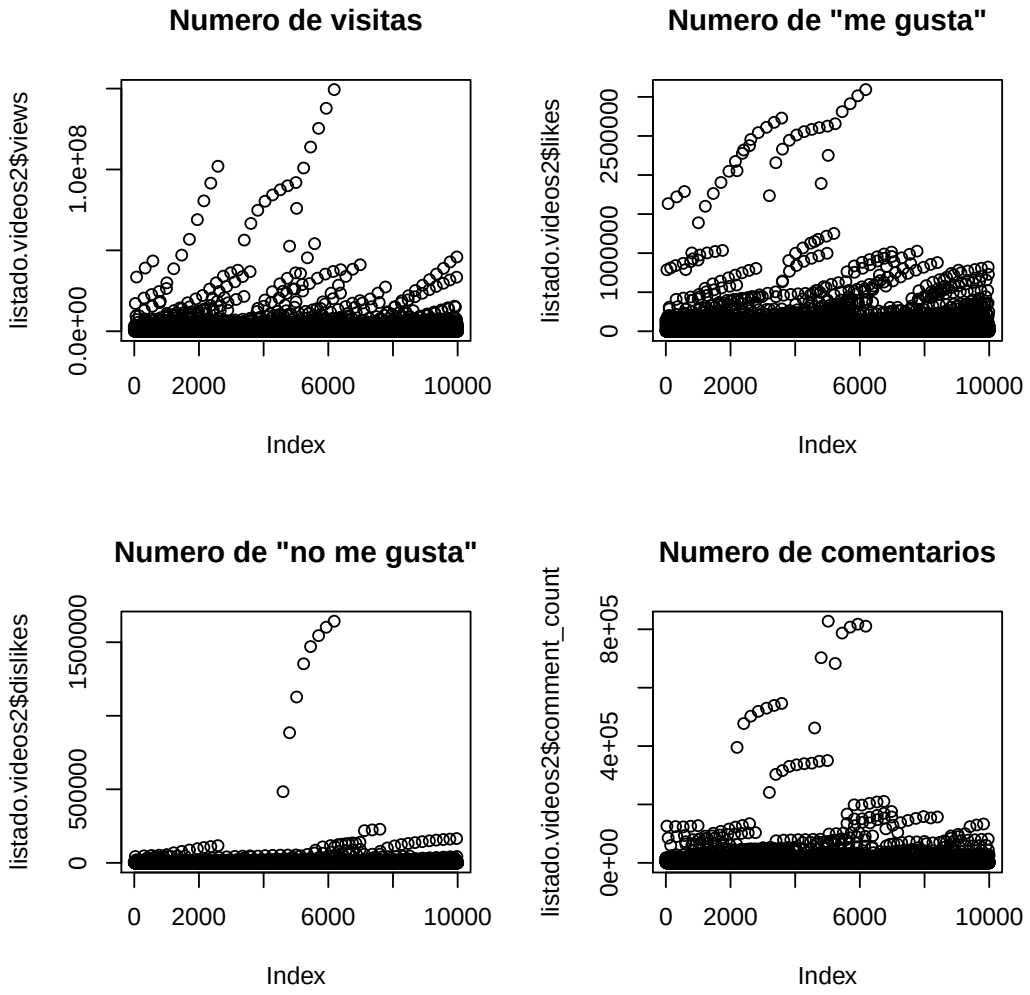


Figura 3: Columnas del dataset

```
> # Importamos el paquete
> if(!require(DMwR)){
+   install.packages("DMwR")
```

```
+   require(DMwR)
+ }
```

Una vez importado, aplicamos el algoritmo *LOF*. Para ello, utilizaremos la función *lofactor*, pasando como parámetro el *dataframe* anterior, así como el número de vecinos utilizado para el cálculo del *Local Outlier Factor* (en nuestro caso elegiremos k = 5):

```
> # Numero de vecinos
> k = 5
> outlier.scores <- lofactor(listado.videos2, k)
> # Analicemos la salida de uno de los vectores (outliers.scores.views, por ejemplo)
> outlier.scores[1:12]
```

```
 [1] 1.429558 1.065041 1.002741 1.031322 1.020761 1.019975 0.957584 1.021816 1.329381 1.072529 1.102719 1.388098
```

Como podemos observar, la función devuelve el grado de *outlier* de cada elemento en forma de vector. Analizando el resultado, vemos que existen **datos anómalos** a lo largo del vector (aquellos cuyo factor *LOF* sean mayores que 1).
A continuación, analizaremos la distribución del factor de *outlier* a lo largo del *dataframe*, mediante un **gráfico de densidad**:

```
> plot(density(outlier.scores), main = "Grafica de densidad",
+ xlabel = "Fila", ylabel = "Densidad")
```

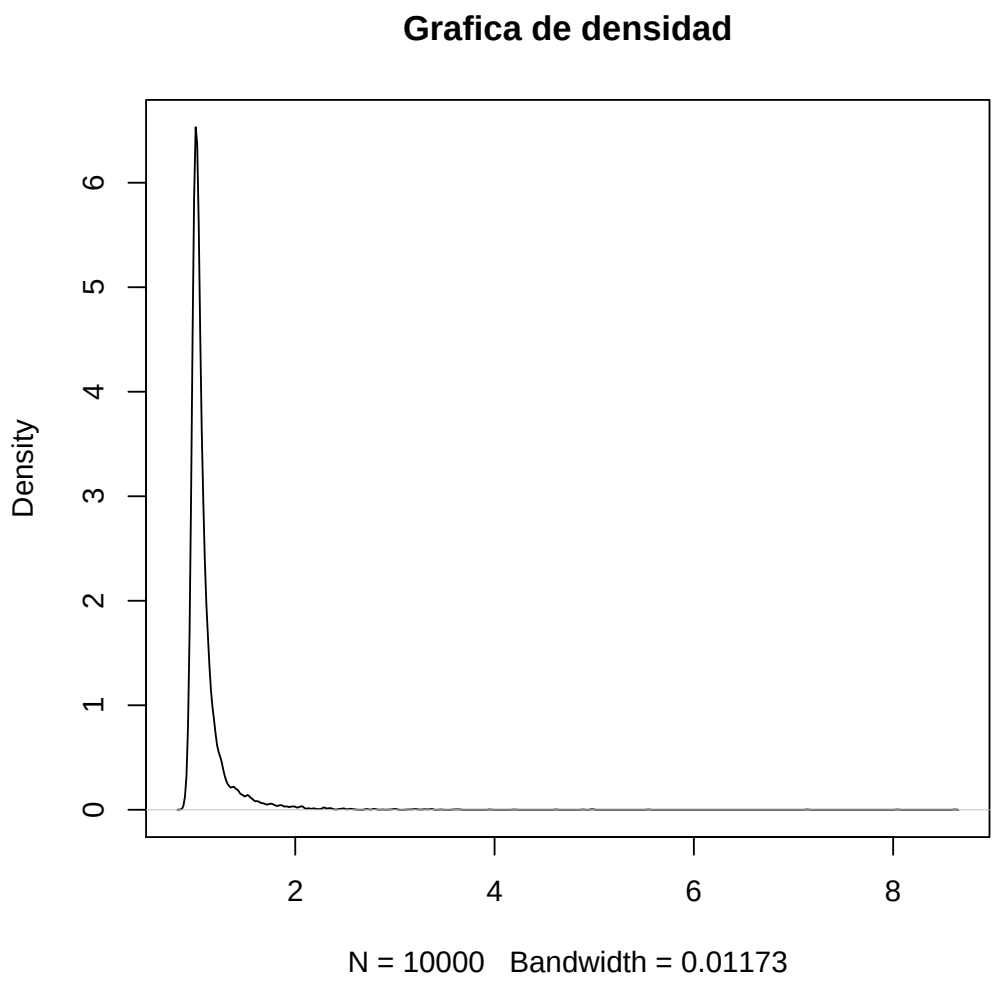


Figura 4: Distribucion del factor

En conclusión, **la mayoría de los valores de cada una de las filas se concentran en torno a un valor *LOF* de 1**. A continuación **reordenamos** el vector *outlier* en orden ascendente, con el fin de seleccionar aquellos puntos cuyo factor LOF sea mayor que 1 (*outliers*). Sin embargo, los factores obtenidos en el algoritmo pueden llegar a ser muy difíciles de interpretar. Por ejemplo, en un determinado *dataset*, un valor de 1.1 puede ser considerado un *outlier*, mientras que un factor de 2 no se considera un dato anómalo, dado que no existe un criterio específico para clasificar un punto como *outlier* o no. Por ello, definimos un **factor de *outlier***, a partir del cual consideraremos a un punto como **dato anómalo** (por ejemplo, 1.2):

```
> # Definimos un factor de outlier
> factor_outlier <- 1.2
> # Utilizaremos la funcion order
> # que reordena los indices del vector
> index.outliers <- order(outlier.scores[outlier.scores >= factor_outlier], decreasing = T)
> # Ejemplo
> head(index.outliers)

[1]  803  774  835  804   46 1139
```

Finalmente, **representamos gráficamente los *outliers***. Para ello, crearemos un vector cuya longitud sea el número de filas, etiquetando con un **.** aquellos valores que no sean *outliers*. Por otro lado, marcamos con un + aquellos puntos etiquetados como *outliers*. A continuación, mediante la función *pairs* representamos una matriz con las diferentes correlaciones, mostrando los *outliers* en cada uno de ellos:

```
> n <- nrow(listado.videos2)
> # Outliers
> pch <- rep(".", n)
> # Inliers
> pch[index.outliers] <- "+"
> col <- rep("black",n)
> col[index.outliers] <- "red"
> pairs(listado.videos2, pch = pch, col = col)
```

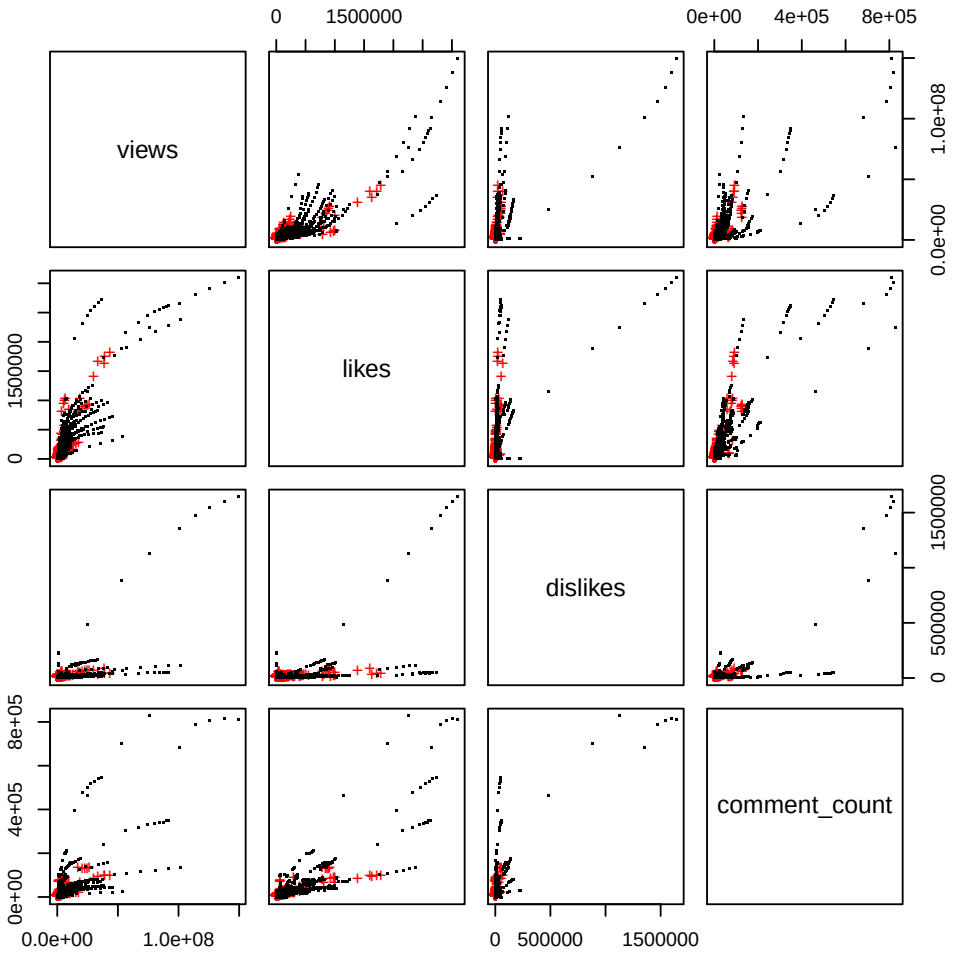


Figura 5: Distribucion de los outliers con el algoritmo LOF

2.2.2. Extensión del algoritmo *LOF*: *LoOP*

Como hemos podido comprobar, el algoritmo *LOF* no proporciona la suficiente información como para clasificar un dato como *outlier*. En contraposición, existe una variante del algoritmo, denominado *LoOP* (*Local Outlier Probabilities*), desarrollado en la universidad de Munich [4]. Mientras que el algoritmo *LOF* se basa en **comparar cada una de las distancias entre un punto y sus *k* vecinos**, el algoritmo *LoOP* introduce un nuevo concepto para el cálculo de las distancias, **la distancia probabilística**.
Para empezar, tomamos un punto o como valor a clasificar, así como un conjunto de puntos situados alrededor. Dicho conjunto de puntos lo denotaremos como S (siendo S la **distribución normal** de o). Asumimos que o está situado en el centro del conjunto S.
En primer lugar, calculamos la **distancia estándar** del punto o al resto de puntos del conjunto S (similar al cálculo de la **desviación estándar**):

$$\sigma(o, S) = \sqrt{\frac{\sum_{s \in S} d(o, s)^2}{|S|}}$$

Para el cálculo de la distancia, partimos del supuesto de que los puntos del conjunto S se **distribuyen normalmente** alerededor de o, por lo que debemos tener esto muy en cuenta a la hora de determinar el conjunto S. Por ello, se pretende obtener el conjunto S por medio del k vecino más cercano situado alrededor de o. De esta manera, el hecho de asumir que S se encuentra centrado alrededor de o es algo razonable.
Como consecuencia, definimos la **distancia probabilística** como el producto de la distancia estándar por un factor λ, obteniendo de este modo la **densidad aproximada del punto *o***:

$$pdist(\lambda, o, S) = \lambda\sigma(o, S)$$

Donde la λ es un **factor de normalización** (comprendido entre 1 y 3). A continuación, calculamos el valor *PLOF* (*Probabilistic Local Outlier Factor*), dado el punto o y el conjunto de los k vecinos más cercanos a o (N_k):

$$PLOF_{\lambda,k}(o) = \frac{pdist(\lambda, o, N_k(o))}{E_{s \in N_k(o)}[pdist(\lambda, s, N_k(s))]} - 1$$

Donde E es la **estimación de la densidad de todos los puntos del conjunto *S* con respecto a los *k* vecinos más cercanos a *o***. Finalmente, obtenemos el valor *LoOP*:

$$LoOP_S(o) = max(0, erf(\frac{PLOF_{\lambda,S(o)}}{\lambda\sqrt{E[(PLOF)^2]}\sqrt{2}}))$$

Donde erf es la **función error de Gauss**. El resultado final es un valor probabilístico comprendido entre 0 y 1: un valor cercano a 0 implica que el punto se sitúa en una región densa (con un gran número de puntos alrededor), por lo que hay menos probabilidades de que se trate de un *outlier*. Por el contrario, si el valor es más cercano a 1, es mucho más probable de que se trate de un *outlier*.
A continuación, vamos a analizar la distribución de *outliers* utilizando el mismo *dataset* que en el apartado anterior. Para ello, `R` dispone del paquete **DDoutlier** [5] el cual contiene la función *LOOP*, que tendrá como parámetros:

- *Dataframe* a analizar
- Número k de vecinos (Al igual que en el ejemplo anterior, lo establecemos a 5)
- Factor λ de normalización (por defecto está establecido a 3)

[4] `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.439.2035&rep=rep1&type=pdf`
[5] `https://cran.r-project.org/web/packages/DDoutlier/DDoutlier.pdf`

```
> # Instalamos el paquete
> if(!require(DDoutlier)){
+   install.packages("DDoutlier")
+   require(DDoutlier)
+ }
> k = 5
> outlier.scores.loop <- LOOP(listado.videos2, k = 5)
> # Analicemos el resultado
> head(outlier.scores.loop, 12)
 [1] 0.345584318 0.000000000 0.000000000 0.257515866 0.010776137 0.117353546 0.130665546 0.038521712 0.431026768 0.154571492 0.0
```

Como podemos comprobar, el algoritmo devuelve un vector de valores comprendidos entre 0 y 1 (cuanto más cercano esté de 1, **más probabilidades habrá de que se trate de un *outlier***). Una vez ejecutado el algoritmo, comparamos el número de *outliers* obtenidos utilizando el algoritmo *LOF* (para valores mayores que 1) con el número de datos anómalos obtenidos con *LOOP* (valores mayores o iguales a 0.5, ya que estos tienen más posibilidades de tratarse de *outliers*):

```
> # Total de outliers del algoritmo LOF
> length(outlier.scores[outlier.scores > 1])

[1] 7024

> # Total de outliers del algoritmo LoOP
> length(outlier.scores.loop[outlier.scores.loop >= 0.5])

[1] 363
```

En conclusión, el número de *outliers* obtenidos con el algoritmo *LoOP* es **significativamente menor con respecto al número de *outliers* obtenidos con el algoritmo LOF**:

```
> index.outliers.loop <- order(outlier.scores.loop[outlier.scores.loop >= 0.5], decreasing = T)
> n <- nrow(listado.videos2)
> pch <- rep(".", n)
> pch[index.outliers.loop] <- "+"
> col <- rep("black",n)
> col[index.outliers.loop] <- "red"
> pairs(listado.videos2, pch = pch, col = col)
```

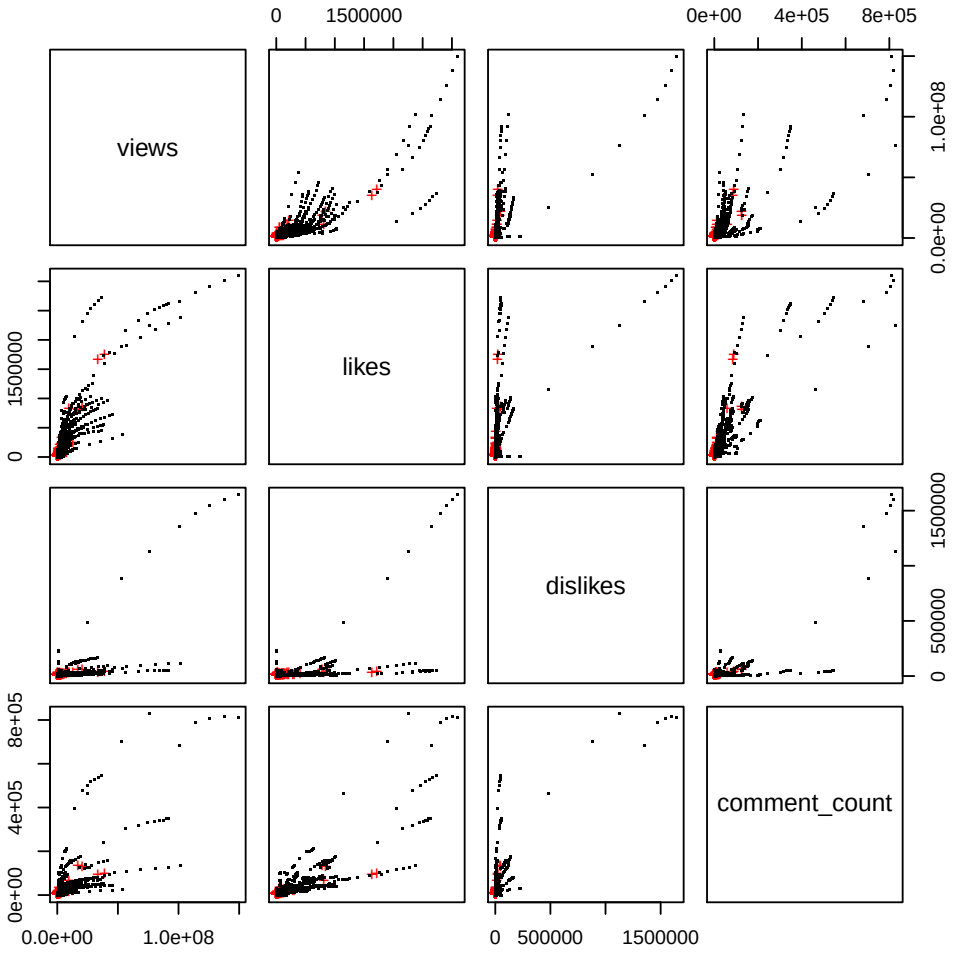


Figura 6: Distribucion de los outliers con el algoritmo LoOP