R-PL5

Gabriel López Cuenca, Sergio Sanz Sacristán, Álvaro Zamorano Ortega

3 de diciembre de 2019

1. Ejercicio realizado en clase.

1.1. K-vecinos

A partir del siguiente conjunto de calificaciones académicas,formados por dos notas: teoría y laboratorio, que tendrán valores entre 0 y 5, realizar un análisis de detección de datos anómalos utilizando el algoritmo **K-vecinos**.

Alumno	Teoría	Laboratorio
A1	4	4
A2	4	3
A3	5	5
A4	1	1
A5	5	4

En primer lugar se introducirán los datos en forma de matriz y se hará la traspuesta de esta.

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

En segundo lugar, calculamos las distancias euclídeas entre todos los puntos y los almacenamos en una matriz. El cálculo de las distancias lo realizamos mediante la función **as.matrix**.

```
> distancias = as.matrix(dist(muestra))
> distancias = matrix(distancias,5,5)
```

En tercer lugar, ordenamos las columnas de la matriz de distancias por los valores de cada una de las filas de menor a mayor. Se tiene en cuenta el **tercer** vecino, por lo que debemos estudiar la cuarta fila de la matriz ya que se tiene en cuenta la distancia de un punto consigo mismo. Se obtienen las muestras cuyo suceso es anómalo o outlier, el grado de outlier es `2.5`.

```
> source("./Funciones/anomalosKVecinos.R")
> anomalosKVecinos
```

```
function (distancias, muestra, k, grado,dimensiones) {
    tmp<-""

    for(i in 1:(length(muestra)/dimensiones)){
        distancias[,i] = sort(distancias[,i])
        if(distancias[k+1,i] > grado) {
            valor<-""
            for (j in 1:dimensiones){
                valor<-paste(valor,muestra[k+1,j],sep=" ")
            }
            tmp<-paste(tmp,"La muestra ", i,
                " con valor (", valor, " ) es outlier - ",sep="")
        }
    }

    return(tmp)
}
```

Procedemos a su ejecución:

```
> (anomalosKVecinos(distancias,muestra,3,2.5,2))

[1] "La muestra 4 con valor ( 1 1 ) es outlier - "
```

Es importante indicar que la función necesita conocer el número de **dimensiones** que tiene la muestra.

1.2. Caja y bigotes.

A partir del siguiente conjunto de valores de resistencia y densidad para diferentes tipos de hormigón, se hará un análisis para la detección de outliers utilizando medidas de ordenación sobre la resistencia con el método de **Caja y Bigotes**.

Hormigón	Resistencia	Densidad
H1	3	2
H2	3.5	12
H3	4.7	4.1
H4	5.2	4.9
H5	7.1	6.1
H6	6.2	5.2
H7	14	5.3

En primer lugar, introducimos los datos en una matriz y lo convertimos a dataframe para que el trabajo con las columnas sea más cómodo.

```
> 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"))))
> muestra = data.frame(muestra)
```

En segundo lugar:

1. Se determina el grado de outlier de forma arbitraria. El valor dado es `1.5`.
2. Se ordenan los datos y se obtienen los cuartiles.

   ```
   > (cuar1 <- quantile(muestra$r,0.25))

   25%
   4.1

   > (cuar3 <- quantile(muestra$r,0.75))

    75%
   6.65
   ```

3. Se calculan los límites del intervalo.

   ```
   > (int = c(cuar1 - 1.5 * (cuar3 - cuar1),cuar3 + 1.5*(cuar3-cuar1)))

      25%    75%
    0.275 10.475
   ```

4. Se identifican como outliers los valores que quedan fuera del intervalo.
   Para ello hemos creado una función:

   ```
   > source("./Funciones/sucesosAnomalos.R")
   > sucesosAnomalos

   function (muestra, intervalo) {
       tmp<-""

       for(i in 1:length(muestra)){
           if(muestra[i] < intervalo[1] || muestra[i] > intervalo[2]){
               tmp<-paste(tmp, "El suceso ", i, " con valor ", muestra[i],
               " es un outlier - ", sep="")
           }
       }

       return(tmp)
   }
   ```

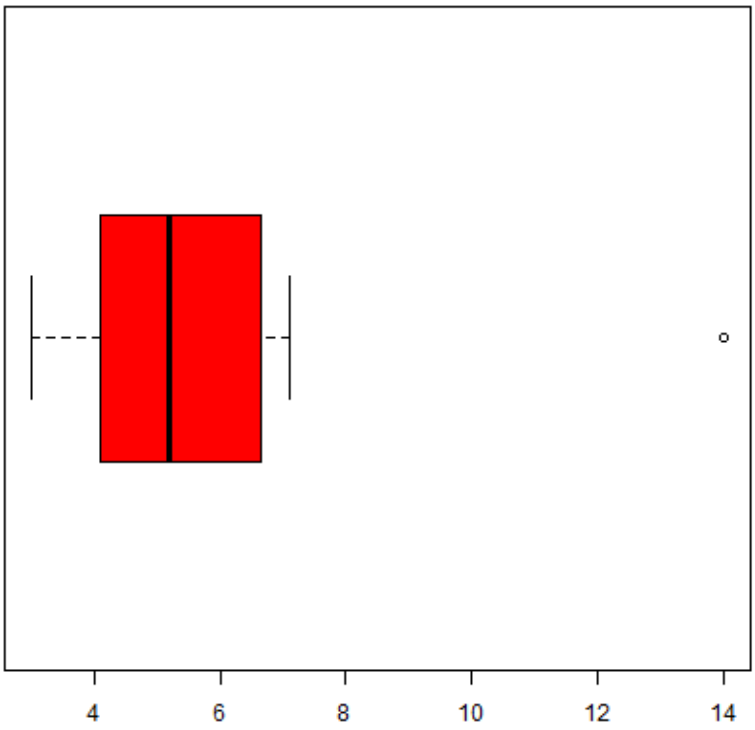
   Procedemos a su ejecución:

   ```
   > (sucesosAnomalos(muestra$r,int))

   [1] "El suceso 7 con valor 14 es un outlier - "
   ```

Por último, mostramos el diagrama de caja y bigotes donde se identifica el valor anómalo.

```
> source("./Funciones/Bigotes.R")
> bigotes(muestra$r,"bigotes.png",1.5)
```



1.3. Desviación Típica.

Ahora con los mismos datos del apartado anterior deseamos realizar un análisis de detección de datos anómalos utilizando medidas de dispersión sobre la densidad con el método de **Desviación Típica**.

1. Se determina el grado de outlier de forma arbitraria. El valor dado es 2.
2. Se obtiene la media aritmética.

   ```
   > (media<-mean(muestra$d))

   [1] 5.657143
   ```

3. Se obtiene la desviación típica.

```
> sdd<-sd(muestra$d)
> (desviacion<-sqrt((sdd^2)*((length(muestra$d)-1)/length(muestra$d))))

[1] 2.857
```

4. Y por último, se calculan los límites del intervalo para los valores atípicos.

```
> (intdes<-c(media-2*desviacion,media+2*desviacion))

[1] -0.05685714 11.37114285
```

5. Se identifican como outliers los valores que quedan fuera del intervalo. Para ello hemos creado una función:

```
> source("./Funciones/sucesosAnomalos.R")
> sucesosAnomalos

function (muestra, intervalo) {
    tmp<-""

    for(i in 1:length(muestra)){
        if(muestra[i] < intervalo[1] || muestra[i] > intervalo[2]){
            tmp<-paste(tmp, "El suceso ", i, " con valor ", muestra[i],
            " es un outlier - ", sep="")
        }
    }

    return(tmp)
}
```

Procedemos a su ejecución:

```
> (sucesosAnomalos(muestra$d,intdes))

[1] "El suceso 2 con valor 12 es un outlier - "
```

1.4. Regresión.

Con los mismos datos de los apartados anteriores deseamos realizar un análisis de detección de datos anómalos sobre la regresión de las variables, densidad en función de la resistencia, utilizando el error estándar de los residuos con el método de **Regresión**.

1. Se determina el grado de outlier de forma arbitraria. El valor dado es 2.

2. Se obtiene la regresión lineal.

```
> (dfr<-lm(muestra$d~muestra$r))
```

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

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

3. Se obtiene el Error Estándar de los residuos.

```
> res<-summary(dfr)$residuals
> (sr=sqrt(sum(res^2)/7))

[1] 2.850242
```

4. Se calcula el límite para considerar atípico un valor siguiendo la ecuación `Grado outlier * Error estándar`.

```
> (intSr<-(2*sr))

[1] 5.700484
```

5. Se identifican los outliers como aquellos tales que `|yObservada - yCalculada| >limite`. El valor de |yObservada - yCalculada| es cada uno de los residuos.

```
> source("./Funciones/anomalosRegresion.R")
> anomalosRegresion

function (residuos, limite, muestra) {
    tmp<-""

    for(i in 1:length(residuos)){
        if(residuos[i]>limite){
            tmp<-paste(tmp,"El suceso ", i,
                " con valor (", muestra$r[i]," ",
                 muestra$d[i], ") es outlier - ",sep="")
        }
    }

    return(tmp)
}
```

Procedemos a su ejecución:

```
> (anomalosRegresion(res, intSr, muestra))

[1] "El suceso 2 con valor (3.5 12) es outlier - "
```

2. Desarrollo de métodos.

Como en el apartado anterior hemos desarrollado el método de K-vecinos, ahora hemos buscado otros paquetes capaces de encontrar los outliers de una muestra realizando éste mismo método.

2.1. Paquete <adamethods>

En primer lugar hemos encontrado el paquete **adamethods**, el cual necesita previamente **shapes**.

```
> install.packages("shapes")
> library(shapes)
> install.packages("adamethods")
> library(adamethods)
```

Vamos a utilizar la función **do_knno()** que forma parte del paquete adamethods, la cual nos devuelve los números de las muestras que son outliers. Para ello primero vamos a introducir la muestra con la que vamos a trabajar, la cual es la utilizada en el apartado 1.1.

```
> muestra = matrix(c(4,4,4,3,5,5,1,1,5,4),2,5)
> muestra = t(muestra)
> data = as.matrix(muestra)
```

Para usar la función debemos introducir la muestra, el número de vecino con el que queremos trabajar (en este caso 3) y el número de outliers que queremos obtener. Por esta última característica consideramos que el funcionamiento del paquete no es adecuado ya que se recomienda indicar el grado de outlier, y con ello el rango de valores permitidos.

Además aunque al calcular el número de outliers de la forma indicada en el apartado 1.1 nos salgan 2, si al realizar está función le introducimos que solo nos calcule 1 outlier, solo nos devolverá 1. Por lo tanto, la función no se corresponde con la realidad.

Procedemos a llamar a la función.

```
> out <- do_knno(data,3,1)
> (data[out,])

[1] 1 1
```

2.2. Paquete <outliers>

Ahora hemos estudiado el paquete **outliers**, el cual nos devuelve un outlier de la muestra.

```
> install.packages("outliers")
> library(outliers)
```

Una vez que inicializamos el paquete a utilizar introducimos la muestra de datos, la cual corresponde con la utilizada en el apartado anterior y en el apartado 1.1 para comparar los resultados.

```
> muestra = matrix(c(4,4,4,3,5,5,1,1,5,4),2,5)
> muestra = t(muestra)
> data = as.matrix(muestra)
```

Una vez introducidos los datos de la muestra procedemos a encontrar los outliers utilizando la función del paquete **outlier()**. A esta función hay que pasarle como parámetro la muestra de datos introducida, y esta nos devolverá un outlier, es decir, solo un outlier, si por ejemplo en el apartado 1.1 la función K-Vecino nos devuelve más de un outlier, esta función tan solo nos va a devolver uno de esos outliers. Incluso si no detecta ningúna outlier nuestra función, aqui nos dará uno de los datos de la muestra como outlier. Por esto consideramos que la funcionalidad de este paquete no es óptima.

Procedemos a llamar a la función.

```
> outlier(data)
```

```
[1] 1 1
```

2.3. Paquete <referenceIntervals>

Ahora estudiaremos el paquete **referenceIntervals**, entonces primero inicializamos este paquete.

```
> install.packages("referenceIntervals")
> library(referenceIntervals)
```

Posteriormente introducimos la muestra de datos, la cual corresponde con la utilizada en los apartados anteriores y en el 1.1 para comparar resultado.

```
> muestra = matrix(c(4,4,4,3,5,5,1,1,5,4),2,5)
> muestra = t(muestra)
> data = as.matrix(muestra)
```

Ahora buscaremos los outliers de la muestra con la función **cook.outliers()**. A esta función se le introduce una columna de la muestra de datos, por ello, la funcionalidad de este paquete no es muy útil para muestras que cuenten con varios datos. Porque puede darse que con una de las columnas de datos identifique unos outliers, y con otras, otros outliers distintos.

Por lo tanto, procedemos a llamar a la función con los datos correspondientes a las calificaciones de teoría.

```
> cook.outliers(data[,1])

$outliers
[1] 1

$subset
[1] 4 4 5 5
```

Sin embargo, si en vez de introducir la columna de datos de las calificaciones de la teoría, introducimos las calificaciones de la práctica, vemos como los outliers son distintos.

```
> cook.outliers(data[,2])

$outliers
numeric(0)

$subset
[1] 4 3 5 1 4
```