Practica 5: Fundamentos de la Ciencia de Datos

Daniel Lopez Moreno Alejandro Fernandez Maceira
Alvaro Maestre Santa

December 8, 2019

1 Análisis de detección de datos anómalos

En esta parte vamos a realizar cuatro ejercicios con ayuda del profesor en el que realizaremos un análisis de detección de datos anómalos.

1.1 Ejercicio 1: K-vecinos

En este apartado vamos a utilizar el algoritmo K-vecinos, utilizando la muestra dada en clase, para así obtener los outliers. Para ello cargamos los datos, en una matriz:

```
> (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
```

Como podemos observar, son 5 pares de datos a analizar. Seguidamente realizamos la transpuesta de la matriz:

```
> (muestra = t(muestra))
```

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

Calculamos la distancia entre todos los puntos, este calculo, se hace con dist y posteriormente lo convertimos en una matriz con as.matrix

```
> (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
```

Como tenemos 5 pares de datos, y hemos calculado las distancias de cada par de datos, tenemos un total de 25 operaciones de distancia, por lo tanto, vamos a crear una matriz 5x5 para tener una mejor visualización de los datos calculados:

```
> (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
```

Ahora ordenamos la matriz, que hemos creado con las distancias en el paso anterior, según los valores de las columnas. Esto se consigue con la función sort y con un bucle, que iremos recorriendo todas las columnas para así ordenarlas con la función mencionada anteriormente:

```
> for(i in 1:5){
+         distancias[,i] = sort(distancias[,i])
+ };
> (distanciasordenadas = 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
```

Ahora lo que hacemos es que para los valores de la 4 fila mayores de 2.5, imprimiremos que es un outlier o un suceso anómalo. Para ello creamos un bucle que ira recorriendo todas las columnas de la matriz y accediendo al valor de la fila 4 para ver si es mayor que 2.5:

```
> for(i in 1:5){
+         if(distanciasordenadas[4,i]>2.5){
+                 print(i);print("es un suceso anómalo o outlier")
+         }
+ }

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

Como podemos observar, podemos ver como muestra que el par 4 es un outlier.

1.2 Ejercicio 2: Método caja y bigotes

En este apartado vamos a realizar un análisis de detección de datos anómalos utilizando medidas de ordenación sobre la resistencia, empleando el algoritmo

caja y bigotes sobre la muestra 2. Por lo tanto lo primero de todo es cargar los datos, hacemos la transpuesta y añadimos nombres a las columnas con dimnames, en este caso será **"r" y "d"**, que es resistencia y densidad respectivamente:

```
> 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 =
> 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
```

Una vez cargado los datos, creamos una estructura de datos de dos dimensiones con **frame**:

```
> (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
```

Ahora vamos a proceder a realizar el algoritmo **caja y bigotes**, para ello utilizaremos la función boxplot, cuyos parámetros son los siguientes:

- **Primer parámetro**: introducimos un vector númerico, el cuál nos dará un diagrama de caja sobre esa variable.
- **Segundo parámetro**: introducimos el rando, es decir, el valor del grado de outlier, en este caso, se elige arbitrariamente.
- **Tercer parámetro**: introducimos un booleano para que nos muestre la gráfica o no, en este caso, lo ponemos a false, debido a que lo que nos interesa son los cálculos.

```
> (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"
```

Ahora calculamos los cuartiles, con la función quantile, que el primer parámetro es el vector de datos y el segundo parámetro es el porcentaje, es decir, 0.25 es el primer cuartil y 0.75 el tercero. El resultado del primer cuartil es:

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

25%
4.1
```

El del tercer cuartil es:

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

 75%
6.65
```

Ahora aplicamos la fórmula del rango, que como su propio nombre indica, nos indicará que todo lo que sea mayor a este rango, será un outlier o suceso anómalo. El resultado es el siguiente:

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

   25%    75%
 0.275 10.475
```

El rango obtenido es: **0.275-10.475.**

Ahora recorremos el vector de datos y vamos mostrando aquello datos que sean mayores que el rango calculado previamente, mostrando así, un mensaje informativo, diciendo si es un outlier o un suceso anómalo. El resultado obtenido es el siguiente:

```
> for(i in 1:length(muestra$r)){
+         if(muestra$r[i]<int[1] || muestra$r[i]>int[2]){
```

```
+                 print("el suceso"); print(i); print(muestra$r[i]);
+                 print("es un suceso anómalo o outlier")
+         }
+ }
```

```
[1] "el suceso"
[1] 7
[1] 14
[1] "es un suceso anómalo o outlier"
```

Los sucesos anómalos son: **7 y 14**

1.3 Ejercicio 3: Desviación Típica.

En este apartado, nos piden realizar un análisis de detección de datos anómalos utilizando medidas de dispersión sobre la densidad, desviación típica sobre la misma muestra del apartado anterior.

En primer lugar, calculamos lso vectores sobre los datos, para ello se hace mediante la siguiente formula:

- **c**: concatenamos los vectores.
- **mean**: función que calcula la media.
- **sd**: función que calcula la desviación típica.

```
> (des = c(mean(muestra$d)-2*sd(muestra$d),mean(muestra$d)+2*sd(muestra$d)))
```

```
[1] -0.5146825 11.8289682
```

Seguidamente, una vez obtenido el rango, vemos que valores son outliers o sucesos anómalos, para aquellos valores que no estén en ese rango. El resultado es el siguiente:

```
> for(i in 1:length(muestra$r)){
+         if(muestra$r[i]<des[1] || muestra$r[i]>des[2]){
+                 print("el suceso"); print(i); print(muestra$r[i]);
+                 print("es un suceso anómalo o outlier")
+         }
+ }
```

```
[1] "el suceso"
[1] 7
[1] 14
[1] "es un suceso anómalo o outlier"
```

Como podemos observar los outliers o sucesos anómalos obtenidos son **7 y 14**

1.4 Ejercicio 4: Error estándar

En este apartado nos piden 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 sobre la misma muestra del ejercicio 2.

En primer lugar, vamos a corregir el resultado, hay que mencionar, que hay muchas maneras de corregir los errores de resultado, pero nosotros vamos a utilizar la que nos ha dicho el profesor, usando la siguiente fórmula:

- **sqrt**: función que calcula la raíz cuadrada.
- **var**: función que calcula la varianza muestral o cuasi-varianza.
- **length**: función que te devuelve un entero cuyo valor es la longitud total del vector de datos.

```
> (sdd = sqrt(var(muestra$d)*((length(muestra$d)-1/length(muestra$d)))))
[1] 8.080816
```

Ahora calculamos los residuos, para ello usamos la función **lm()**, la cual, genera un modelo de regresión lineal por mínimos cuadrados en el que la variable es **muestra$d** y el predictor es **muestra$r**.

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

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

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

Ahora para visualizar los principales parámetros generados, utilizamos la función **summary()**:

```
> (summary (dfr))

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

Residuals:
       1        2        3        4        5        6        7
-3.84275  6.18587 -1.64545 -0.81683  0.49192 -0.45960  0.08684

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.01445    2.64632   2.273   0.0722 .
muestra$r   -0.05723    0.37148  -0.154   0.8836
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.372 on 5 degrees of freedom
Multiple R-squared:  0.004725,        Adjusted R-squared:  -0.1943
F-statistic: 0.02374 on 1 and 5 DF,  p-value: 0.8836
```

Como nos muestra muchisima información y solo nos interesa el valor de los residuos calculados, ponemos el siguiente comando:

```
> (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
```

Como podemos observar, nos sale un total de siete residuos. Posteriormente, calculamos la desviación estándar, cuyo valor se utilizará para clasificar si es un outlier o suceso anómalo. La formula es la raiz cuadrada del sumatorio de cada residuo elevado al cuadrado entre el total de residuos que haya:

```
> (sr = sqrt (sum(res^2)/7))

[1] 2.850242
```

Ahora calculamos los outliers, recorriendo cada uno de los residuos, y viendo si es mayor que el doble de la desviación estándar (sr), calculado previamente, será un outlier o suceso anómalo:

```
> for(i in 1:length(res)){
+         if(res[i]>2*sr){
+                 print("el suceso");print(i);print(muestra$r[i]);
+                 print("es un suceso anómalo o outlier")
+         }
+ }

[1] "el suceso"
[1] 2
[1] 3.5
[1] "es un suceso anómalo o outlier"
```

Como podemos ver los sucesos **2 y 3.5** son sucesos anómalos o outlier.

2 Desarrollo por parte del grupo

En este apartado realizaremos detección de datos anómalos de una manera distinta a la estudiada en la clase de laboratorio. Esta vez, realizaremos el análisis sobre una base de datos que contiene información relativa a las estadísticas de los **videojuegos más vendidos** hasta 2016.

Lo primero que debemos hacer es instalar las librerias que consideramos necesarias para el desarrollo de este apartado:

```
> install.packages("dbscan")

--- Please select a CRAN mirror for use in this session ---
package 'dbscan' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\alex1\AppData\Local\Temp\RtmpcP851A\downloaded_packages
```

```
> install.packages("fpc")

package 'fpc' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\alex1\AppData\Local\Temp\RtmpcP851A\downloaded_packages

> library("dbscan")
> library("fpc")
```

Ahora, leeremos el archivo **"videogames.csv"** y lo adaptaremos para poder trabajar con él.
Para empezar con la adaptación eliminaremos las filas con valor NA para evitar problemas.
A continuación, como contamos con más de **16000 filas**, reduciremos la cantidad de datos al **Top 50 Ventas** para facilitar la tarea de lectura, para ello utilizaremos la función **head**:

```
> videogames<-read.csv("videogames.csv")
> videogames<-videogames[complete.cases(videogames),]
> videogames<-head(videogames,50)
```

Una vez tenemos los datos adaptados, procedemos a realizar el análisis por distintos métodos:

2.1 Análisis con k-Vecinos

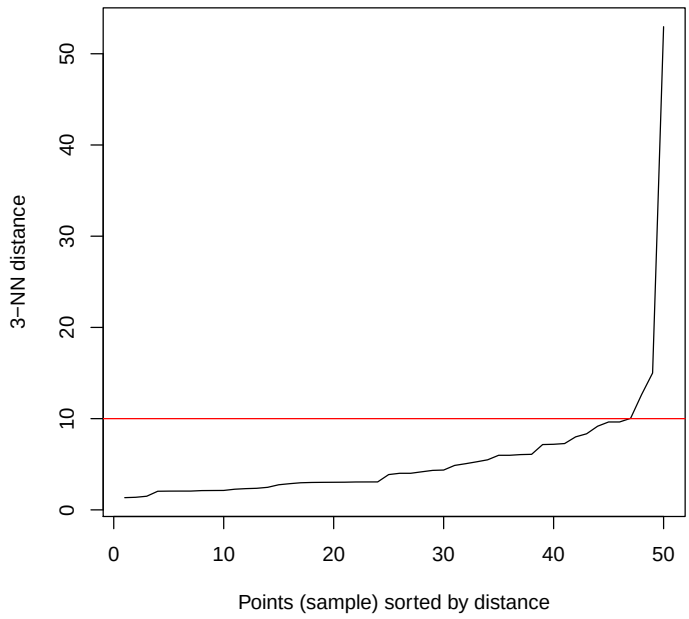
Para realizar el análisis con k-Vecinos utilizaremos algunas funciones de los paquetes **"dbscan"** y **"fpc"**, como **kNNdisplot** y la clasificación **dbscan** que ya hemos utilizado en la práctica 4 realizada con anterioridad.

Lo primero que debemos hacer es aislar las 2 variables sobre las que vamos a calcular las distancias. En el caso de nuestra base de datos usaremos el número de ventas total (**Global_Sales**) y las valoraciones de los usuarios (**User_Score**):

```
> sales<-videogames$Global_Sales
> score<-videogames$User_Score
> data<-cbind(sales,score)
```

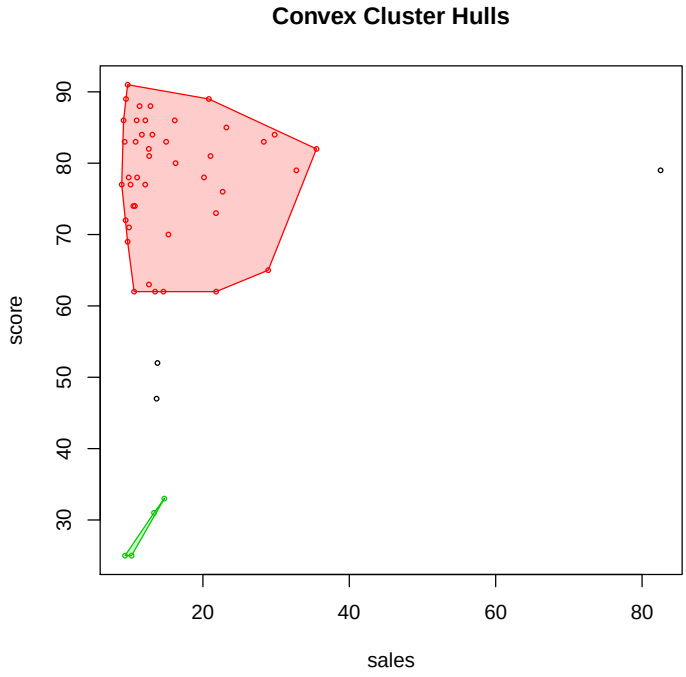
Ya tenemos los datos aislados, ahora debemos realizar el análisis de k-Vecinos y comprobar a partir de cuantos puntos aumenta drásticamente la distancia. Utilizaremos la distancia a partir del **vecino 3**:

```
> kNNdistplot(data, k=3)
> abline(h=10, col="red")
```



Tras hacer el análisis, podemos ver que a partir del **valor 10** la distancia aumenta, por lo que debemos hacer la **dbscan** con ese valor y comprobaremos como se clusterizan finalmente los valores y cuales de ellos quedan excluidos, los **outliers**:

```
> clasDBSCAN <- fpc::dbscan(data, eps = 10, MinPts = 4)
> hullplot(data, clasDBSCAN$cluster)
```



Claramente se han definido 2 clusters, un gran cluster **rojo** y otro pequeño cluster **verde**. Además, podemos ver **3 puntos negros**, correspondientes a **outliers**. Estos datos corresponden a los siguientes videojuegos:

```
> videogames<-cbind(clasDBSCAN$cluster, videogames)
> (outliers = subset(videogames,videogames[,1] == 0))
```

```
   clasDBSCAN$cluster                      Name Platform Year_of_Release
1                   0                 Wii Sports      Wii            2006
35                  0 Call of Duty: Black Ops II      PS3            2012
36                  0 Call of Duty: Black Ops II     X360            2012
     Genre  Publisher NA_Sales EU_Sales JP_Sales Other_Sales Global_Sales
1   Sports   Nintendo    41.36    28.96     3.77        8.45        82.53
35 Shooter Activision     4.99     5.73     0.65        2.42        13.79
36 Shooter Activision     8.25     4.24     0.07        1.12        13.67
   Critic_Score Critic_Count User_Score User_Count Developer Rating
1            76           51          8        322  Nintendo      E
35           83           21        5.3        922  Treyarch      M
36           83           73        4.8       2256  Treyarch      M
```

- **Wii Sports**: Este videojuego tiene aproximadamente las mismas valoraciones por parte de los usuarios que el resto de videojuegos del cluster **rojo**, sin embargo se considera outlier por su excesiva diferencia en el número de ventas globales.
- **CoD Black Ops II (PS3)**: Este videojuego tiene un número de ventas parecido a los existentes en los 2 cluster que hemos identificado. Sin embargo, se considera outlier ya que no tiene suficiente valoración como para

pertenecer al **cluster rojo**, pero tiene demasiada como para pertenecer al **cluster verde**

- **CoD Black Ops II (Xbox 360)**: Con este videojuego ocurre exactamente lo mismo que con el anterior, ya que se trata del mismo juego pero para una consola distinta.

Un método para comprobar el dato más extremo es la función **car::outlierTest**, la cual devuelve el valor con más probabilidades de ser un outlier de forma estadística empleando la regresión. Esta función viene por defecto en R. Lo primero que hay que hacer para poder obtener el valor outlier es convertir la matriz de datos a tipo **frame**, para después hacer una regresión de esos datos. Esa regresión es la que necesita la función **car::outlierTest**.

```
> datosFrame=data.frame(data)
> carOut=lm(datosFrame)
> car::outlierTest(carOut)

  rstudent unadjusted p-value Bonferroni p
1 9.824788         5.6481e-13   2.8241e-11
```

Como se puede observar, el dato que es un **outlier** es el videojuego **Wii Sports**, ya que el número que aparece es el 1, el cual coincide con el primer valor del data frame. Este resultado concuerda con el método **k-vecinos**, tal y como se había obtenido previamente.

Además de la función anterior, existe otro paquete llamado **outliers**, el cual contiene funciones para comprobar los outliers en un **data frame**. Este método emplea la media de los datos para ver cuál es el dato anómalo.

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

package 'outliers' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\alex1\AppData\Local\Temp\RtmpcP851A\downloaded_packages

> library(outliers)
> outlier(datosFrame)

sales score
82.53 25.00
```

En este caso, el valor obtenido aparece dividido en outlier de **ventas** y outlier de **puntuación**, que son las dos columnas que forman el **data frame**. Analizando el valor de la columna de ventas, podemos ver que el valor obtenido corresponde al juego **Wii Sports**, como hemos obtenido en los previos métodos de análisis de outliers. En la columna de puntuación el valor obtenido es **25.00**, que corresponde, si nos fijamos en el **data frame**, a los videojuegos **Call of Duty Ghosts** en PS3 y en Xbox, resultado que también coincide con el análisis de **k-vecinos** realizado previamente.

2.2 Problemas que hemos encontrado y soluciones

2.2.1 Cambio de variables a analizar

Al comenzar el desarrollo por parte del grupo las distancias para el algoritmo k-vecinos iban a utilizar las variables **Global_Sales y Critic_Score**. Sin embargo, tras probarlo pudimos ver que las notas de la crítica siempre eran altas y nunca bajaban del valor 50, por lo que todos los juegos quedaban acumulados en el mismo cluster y el único **outlier** que podiamos detectar era Wii Sports debido a su elevado número de ventas.
Por ello, optamos por eliminar dicho videojuego de la lista para ver si el resto de videojuegos se separaban en distintos cluster y quedaba algún otro outlier expuesto, pero no dio resultado.
Por lo tanto, finalmente cambiamos la variable **Critic_Score** por **User_Score** ya que los usuarios eran más críticos y si existian algunas notas por debajo de 50, dejando descubiertos más clusters y outliers.

2.2.2 Cambio de Top 100 a Top 50

Inicialmente el análisis se iba a realizar sobre el **Top 100** videojuegos en número de ventas, pero nos dimos cuenta de que **entre el Top 51 y el Top 100** la mayoría de videojuegos se estancaban alrededor del mismo valor de ventas, simplemente variando sus puntuaciones, **creando una linea recta** en nuestro gráfico que no aportaba nada a la clasificación.
Por lo tanto, decidimos reducir el espectro y analizar solamente los **primeros 50 videojuegos más vendidos** en vez de los 100 primeros.