Practica 1: Fundamentos de la Ciencia de Datos

Daniel Lopez Moreno Alejandro Fernandez Maceira
Alvaro Maestre Santa

October 14, 2019

1 Análisis de datos con R

En esta parte de la practica se nos pide que apliquemos los conceptos aprendidos en el tema.

1.1 Analisis de un archivo .txt

1.1.1 Comandos basicos

Primero comenzaremos con los comandos basicos para comenzar a utilizar R.

- **help()**: Muestra la pagina de ayuda de R.
- **help("comando()")**: Muestra la ayuda relativa al comando que se pasa como argumento
- **help.start()**: Abre la pagina de documentación online de R
- **setwd("directorio")**: Cambia el directorio de trabajo al directorio seleccionado
- **list.files()**: Muestra una lista de los archivos del directorio actual de trabajo. Otra función equivalente sería **dir()**

A continuación mostraremos algunos comandos basicos para asignar valor a las variables que creemos:

- **x<-2**: De esta forma asignamos un valor numérico a la variables x. También podríamos utilizar = como sustituto de <-.
- **num<-c(2.3,4,5.6)**: Utilizamos la función c para generar vectores de valores y asignárselos a num. Estos vectores pueden ser cuantitativos (c(2.3,4,5.6)), cualitativos(c("verde","azul","rojo")) y lógicos (c(TRUE,FALSE,TRUE) o c(T,F,T))
- **mat1<-matrix(nrow=2,ncol=2,c(69.7,71.3,168,178))**: Utilizando el comando matrix() podemos crear una matriz de dimensiones nrowXncol y llenarla con los datos del vector c. Si usamos el modificador byrow=T, la matriz se llenará por filas y no por columnas.

1.1.2 Tratamiento de un archivo .txt

Una vez hemos aprendido algunos comando basicos de R vamos a leer un archivo de datos llamado *"satelites.txt"*, que contiene información sobre los satelites menores de Urano, y vamos a operar con su información.

El archivo *"satelites.txt"* tendrá la siguiente estructura:

Nombre Radio
1 Cordelia 13
2 Ofelia 16
3 Bianca 22
4 Cresida 33
5 Desdemona 29
6 Julieta 42
7 Rosalinda 27
8 Belinda 34
9 Luna-198U10 20
10 Calibano 30
11 Luna-999U1 20
12 "Luna 199U2" 15

Para leer el archivo y asignarlo a una variable **sat** ejecutaremos la siguiente función:

```
> sat<-read.table("satelites.txt")
```

1.1.3 Cálculo de frecuencias, media, desviación típica y varianza

Una vez hemos cargado el archivo ya podemos operar con su información.

Longitud . Para empezar vamos a calcular la longitud de la columna *Radio* de la matriz generada, utilizaremos la función **length()** sobre la columna Radio:

```
> length(sat$Radio)

[1] 12
```

Media aritmetica . Ahora vamos a calcular la media aritmética de los radios, para ello utilizaremos la función **mean()** sobre la columna Radio y asignaremos el valor a la variable **mr**:

```
> mr<-mean(sat$Radio)
> mr

[1] 25.08333
```

Frecuencia absoluta . Utilizando la función **table()** calcularemos las frecuencias absolutas de una columna. Haremos la prueba calculando las frecuencias de los radios:

```
> frecabsradio<-table(sat$Radio)
> frecabsradio

13 15 16 20 22 27 29 30 33 34 42
 1  1  1  2  1  1  1  1  1  1  1
```

Frecuencia acumulada . También podemos calcular la frecuencia acumulada de la columna Radio. Para ello utilizaremos la función **cumsum()**:

```
> frecabsacumradio<-cumsum(table(sat$Radio))
> frecabsacumradio

13 15 16 20 22 27 29 30 33 34 42
 1  2  3  5  6  7  8  9 10 11 12
```

Frecuencia relativa . En R no existe una función específica que nos ayude a calcular la frecuencia relativa, pero sí existe la opción de crear nuestras propias funciones. Por lo tanto, crearemos nuestra propia función para calcular la frecuencia relativa dividiendo la **frecabsradio** entre **length()** de la columna Radio:

```
> frecrel<-function(Radio){table(Radio)/length(Radio)}
> frecrel(sat$Radio)

Radio
        13         15         16         20         22         27         29
0.08333333 0.08333333 0.08333333 0.16666667 0.08333333 0.08333333 0.08333333
        30         33         34         42
0.08333333 0.08333333 0.08333333 0.08333333
```

Una vez hemos creado nuestra función, podemos guardarla en un archivo .R utilizando la función **dump()**, indicando el nombre del archivo destino:

```
> dump("frecrel",file="FrecRelativa.R")
```

También podemos cargar funciones que hayamos generado anteriormnete utilizando **source()** y el nombre del archivo .R:

```
> source("FrecRelativa.R")
```

Frecuencia Rel. Acumulada . Ahora que ya sabemos como crear nuestras propias funciones, vamos a hacer los mismo para calcular la frecuencia relativa acumulada, creando la función **frecrelacum()**:

```
> frecrelacum <-function(Radio){cumsum(table(Radio))/length(Radio)}
> dump("frecrelacum",file="FrecRelativaAcum.R")
> source("FrecRelativaAcum.R")
> frecrelacum(sat$Radio)
```

```
        13         15         16         20         22         27         29
0.08333333 0.16666667 0.25000000 0.41666667 0.50000000 0.58333333 0.66666667
        30         33         34         42
0.75000000 0.83333333 0.91666667 1.00000000
```

Desviación típica y varianza . La siguiente magnitud que vamos a calcular será la **desviacion tipica** y la **varianza**.
La desviación típica y la varianza se pueden calcular utilizando las funciones **sd()** y **var()**, pero se calculan con un denominador N-1, por lo que para lograr los mismos resultados que hemos obtenido a mano, crearemos nuestras propias funciones que cambien ese denominador N-1 por un denominador N:

```
> desvtipica <-function(Radio){sqrt(sd(Radio)*sd(Radio)*(length(Radio)-1)/length(Radio))}
> varian<- function(Radio){var(Radio)*(length(Radio)-1)/length(Radio)}
> dump("desvtipica",file="DesviacionTipica.R")
> dump("varian",file="Varianza.R")
> source("DesviacionTipica.R")
> source("Varianza.R")
> desviaciontipica<-desvtipica(sat$Radio)
> desviaciontipica

[1] 8.47996

> varianza<-varian(sat$Radio)
> varianza

[1] 71.90972
```

1.1.4 Cálculo de medidas de ordenación, mediana y cuartiles

En este apartado vamos a proceder a calcular las **medidas de ordenacion** sobre nuestra columna Radio.

Ordenación . Lo primero que vamos a ver es como ordenar nuestra columna de manera **ascendente** utilizando el comando **order()**:

```
> so= sat[order(sat$Radio), ]
> so

        Nombre Radio
1     Cordelia    13
12  Luna 199U2    15
2       Ofelia    16
9  Luna-198U10    20
11  Luna-999U1    20
3       Bianca    22
7    Rosalinda    27
5    Desdemona    29
10    Calibano    30
4      Cresida    33
8      Belinda    34
6      Julieta    42
```

Si por el contrario queremos ordenarlo de manera **descendente** simplemente debemos invertir la ordenación con **rev()**:

```
> so= sat[rev(order(sat$Radio)), ]
> so
```

```
        Nombre Radio
6      Julieta    42
8      Belinda    34
4      Cresida    33
10    Calibano    30
5    Desdemona    29
7    Rosalinda    27
3       Bianca    22
11  Luna-999U1    20
9  Luna-198U10    20
2       Ofelia    16
12  Luna 199U2    15
1     Cordelia    13
```

Mediana . Lo siguiente que vamos a calcular es la **mediana**. Para ello es suficiente con ejecutar la función **median()** sobre nuestra columna:

```
> mediant<-median(sat$Radio)
> mediant
```

```
[1] 24.5
```

Cuantiles . Por último vamos a calcular los cuantiles. Para calcular cualquier cuantil entre 0 y 100 debemos utilizar la función **quantile()** indicando el vector sobre el que lo queremos calcular y el cuantil exacto que deseamos obtener.
Por ejemplo, para calcular el cuantil 54 debemos ejecutar lo siguiente:

```
> cuant54<-quantile(sat$Radio,0.54)
> cuant54
```

```
 54%
26.7
```

Como se puede imaginar, si se quiere calcular cualquiera de los **4 cuartiles**, basta con calcular los **cuantiles 25, 50, 75 y 100**. Así como si queremos calcular la **mediana** también lo podemos hacer calculando el **segundo cuartil**, equivalente al **cuantil 50**.

```
> cuartil1<-quantile(sat$Radio,0.25)
> cuartil1
```

```
25%
 19
```

```
> cuartil2<-quantile(sat$Radio,0.50)
> cuartil2
```

```
 50%
24.5

> mediant

[1] 24.5

> cuartil3<-quantile(sat$Radio,0.75)
> cuartil3

  75%
30.75

> cuartil4<-quantile(sat$Radio,1)
> cuartil4

100%
  42
```

1.2 Pruebas sobre archivo .sav (SPSS)

En este apartado vamos a ejecutar las mismas funciones que ya hemos explicado, pero esta vez será sobre un archivo de SPPS llamado **"cardata.sav"**. Este archivo contiene datos de automóviles como su consumo, cilindrada, aceleración, etc.

1.2.1 Añadir paquetes y leer archivos SPSS

Para leer un archivo .txt era suficiente con ejecutar el comando **read.table()**, sin embargo, para los archivos SPSS esta función no funcionará. Por ello, para poder leer un archivo .sav debemos importar primero una librería que nos permita hacerlo, la librería **"foreign"**. En este caso la librería se encuentra dentro de los archivos de R, por lo que lo único que hay que hacer es instalarla:

```
> library(foreign)
```

Una vez instalada, podemos comprobar que se ha añadido utilizando la función **search()**:

```
> search()

 [1] ".GlobalEnv"        "package:foreign"   "package:stats"
 [4] "package:graphics"  "package:grDevices" "package:utils"
 [7] "package:datasets"  "package:methods"   "Autoloads"
[10] "package:base"
```

Una vez estamos listos para leer el archivo, lo cargaremos en la variable **coches** utilizando la función **read.spss()**:

```
> coches<-read.spss("cardata.sav")
```

1.2.2 Cálculo de media, desviación típica y varianza

Una vez tenemos cargado el archivo, vamos a operar con la columna referente al **consumo(mpg)**. Para simplificar la tarea guardaremos la columna en la variable **mpg**.
Para empezar, vamos a calcular la media aritmetica:

```
> mpg<-coches$mpg
> med<-mean(mpg)
> med

[1] NA
```

Como se puede comprobar, el resultado de la **media es NA**. Esto se debe a que algunos datos de la columna "mpg" tienen como valor NA y no puede sumarlos, por lo que debemos ignorar estos datos a la hora de calcular la media:

```
> mpg<-mpg[!is.na(mpg)]
> med<-mean(mpg)
> med

[1] 28.79351
```

Lo siguiente que vamos a calcular sobre esta columna son la **desviacion típica** y la **varianza** utilizando las funciones que hemos creado en el apartado anterior:

```
> desviacioncoches<-desvtipica(mpg)
> varianzacoches<-varian(mpg)
> desviacioncoches

[1] 7.353219

> varianzacoches

[1] 54.06983
```

1.2.3 Cálculo de medidas de ordenación, mediana y cuartiles

Por último, vamos a calcular las medidas de ordenación (mediana y cuartiles) sobre la columna mpg.
El cálculo sería similar al realizado anteriormente sobre los satélites:

Mediana:

```
> mediantmpg<-median(mpg)
> mediantmpg

[1] 28.9
```

Cuartiles:

```
> cuartil1mpg<-quantile(mpg,0.25)
> cuartil1mpg
```

```
  25%
22.55

> cuartil2mpg<-quantile(mpg,0.50)
> cuartil2mpg

 50%
28.9

> mediantmpg

[1] 28.9

> cuartil3mpg<-quantile(mpg,0.75)
> cuartil3mpg

   75%
34.275

> cuartil4mpg<-quantile(mpg,1)
> cuartil4mpg

100%
46.6
```

2 Análisis estadístico de proyectos en Kickstarter con R

En esta sección se muestra el proceso seguido a la hora de obtener estadísticas sobre los datos de una base de datos de Kaggle, en este caso de proyectos en la plataforma Kickstarter.

2.1 Instalación de paquetes utilizados

Primero comenzaremos con la instalación de los paquetes que se han utilizado en este apartado. Estos paquetes son **descr** para la frecuencia, **modeest** para la moda y **FSA** para la media geométrica.
Para instalarlos, se siguen los siguientes pasos:
Paquete descr:

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

Paquete modeest:

```
> install.packages("BiocManager")
> library("BiocManager")
> BiocManager::install("genefilter")
> install.packages("modeest")
> library(modeest)
```

Paquete FSA:

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

El paquete modeest, como se puede observar, requiere de la función **genefilter** del paquete **BiocManager**. Después de tener instalados los paquetes, se puede proceder al análisis estadístico de los datos.

2.2 Preparación del entorno

Lo primero que hacemos es preparar el entorno de trabajo. Para ello, se lee el archivo de datos, el cual viene dado en formato .csv. Para leer este tipo de archivo, se hace uso de la instrucción del paquete estándar de R **read.csv**. Antes de esto, a modo opcional, se puede comprobar si el archivo a leer está en el directorio actual con la instrucción **dir()**.

Después de esto, ya podemos trabajar con el fichero en cuestión. Ahora utilizamos la función **colnames** para obtener el nombre de las filas cargadas y poder aplicar las funciones estadísticas.

```
> colnames(ksp)
```

```
 [1] "ID"              "name"             "category"         "main_category"
 [5] "currency"        "deadline"         "goal"             "launched"
 [9] "pledged"         "state"            "backers"          "country"
[13] "usd.pledged"     "usd_pledged_real" "usd_goal_real"
```

Una vez elegidas las columnas a tratar, podemos crear un atajo para no tener que llamar a esa columna por su nombre completo dentro del fichero. Esto se puede lograr con la siguiente instrucción, en este caso aplicada a la columna **goal**, aunque el proceso es el mismo para cualquier otra columna.

```
> goals = ksp$goal
```

Por último, se utilizarán las funciones de varianza y desviación típica del anterior ejercicio de la práctica, previamente importadas.

2.3 Análisis de datos

En esta sección se analizarán los datos elegidos mediante técnicas estadísticas. El primer bloque de datos a analizar pertenece a los objetivos de apoyo monetario de los proyectos de Kickstarter.

2.3.1 Calculo de media, desviacion tipica y varianza

Media aritmética:

```
> mean(goals)
```

```
[1] 49080.79
```

Media geométrica, para la cual utilaremos el paquete FSA:

```
> geomean(goals)
```

```
[1] 5687.223
```

Desviacion tipica:

```
> desvGoals<-desvtipica(goals)
> desvGoals
```

```
[1] 1183390
```

Varianza:

```
> varianGoals<-varian(goals)
> varianGoals
```

```
[1] 1.400411e+12
```

Moda, utilizando el paquete **modeest**:

```
> mlv(goals, method = "mfv")
```

```
[1] 5000
```

En el método el valor es mfv porque es el valor más frecuente (most frequent value).

Mínimo absoluto:

```
> min(goals)
```

```
[1] 0.01
```

Máximo absoluto:

```
> max(goals)
```

```
[1] 1e+08
```

Rango:

```
> range(goals)
```

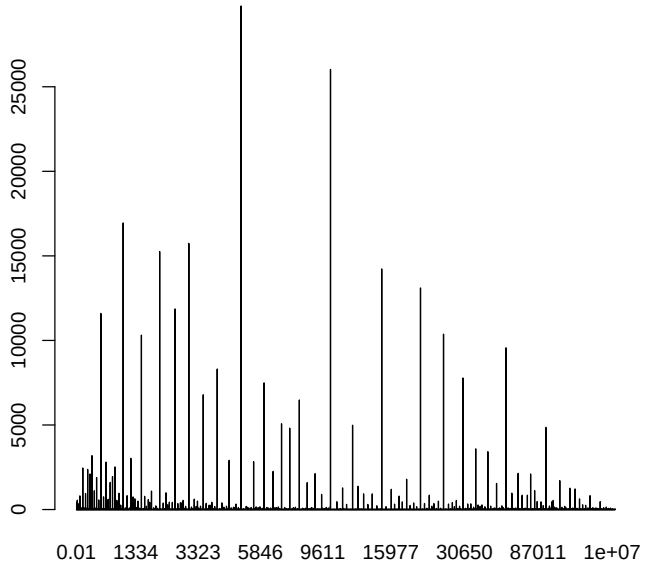
```
[1] 1e-02 1e+08
```

Las frecuencias las vamos a mostrar sobre las 10 primeras filas de datos, ya que si mostrasemos todos los datos el documento quedaría demasiado extenso.
Frecuencia absoluta y relativa:

```
> frecuencias<-freq(goals)[1:10,]
> frecuencias
```

```
     Frequency      Percent
0.01         2 0.0005281769
0.15         1 0.0002640885
0.5          1 0.0002640885
1          430 0.1135580374
1.85         1 0.0002640885
2           24 0.0063381230
3           20 0.0052817692
4            9 0.0023767961
5          142 0.0375005612
6            9 0.0023767961
```

```
>
```



Frecuencia absoluta acumulada:

```
> cumsum(table(goals))[1:10]
```

```
0.01 0.15  0.5    1 1.85    2    3    4    5    6
   2    3    4  434  435  459  479  488  630  639
```

Frecuencia relativa acumulada:

```
> frecrel(goals)[1:10]
```

```
Radio
        0.01         0.15          0.5            1         1.85            2
5.281769e-06 2.640885e-06 2.640885e-06 1.135580e-03 2.640885e-06 6.338123e-05
           3            4            5            6
5.281769e-05 2.376796e-05 3.750056e-04 2.376796e-05
```

2.3.2 Cálculo de medidas de ordenación, mediana y cuartiles

Mediana (equivalente al segundo cuartil):

```
> median(goals)

[1] 5200
```

Primer y tercer cuartil:

```
> cuartil1<-median (goals [which (goals <= median (goals))])
> cuartil1

[1] 2000

> cuartil3<-median (goals [which (goals >= median (goals))])
> cuartil3

[1] 16000
```

A la vista de estas estadísticas, se puede llegar a la conclusión de que los datos están muy dispersos entre sí, ya que la desviación típica es muy grande. Por lo tanto, la media no da un valor significativo de la población.
Estas mismas estadísticas se pueden aplicar a otras columnas, como la de **backers** o la de **pledged**. En otras columnas como **category** o **country**, no se pueden aplicar todas las medidas, ya que los datos no son numéricos, pero se podrían aplicar las funciones de moda o frecuencia absoluta y relativa.