

# **Fundamentos de la Ciencia de Datos**

## **Práctica 1**

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

15 de octubre del 2019

# Índice

<b>1. Apartado 1</b>	<b>3</b>
1.1. Fichero <i>.txt</i>	3
1.1.1. Frecuencia Absoluta y Acumulada	4
1.1.2. Frecuencia Relativa y Acumulada	5
1.1.3. Media Aritmética	6
1.1.4. Desviación Típica	6
1.1.5. Varianza	7
1.1.6. Mediana	7
1.1.7. Cuantiles	7
1.2. Fichero <i>.sav</i> (SPSS)	8
1.2.1. Frecuencia Absoluta y Acumulada	9
1.2.2. Frecuencia Relativa y Acumulada	9
1.2.3. Media Aritmética	10
1.2.4. Desviación Típica	10
1.2.5. Varianza	10
1.2.6. Mediana	10
1.2.7. Cuartiles	11
<b>2. Apartado 2</b>	<b>11</b>
2.1. Lectura de ficheros <i>.csv</i>	16
2.2. Moda, Frecuencia Absoluta y Frecuencia Absoluta Acumulada	17
2.2.1. Frecuencia absoluta y las funciones recursivas en R	17
2.3. Frecuencia relativa y frecuencia relativa acumulada	22
2.3.1. Versión iterativa	23
2.3.2. Ejemplos de cálculo de frecuencias relativas	23
2.4. Media aritmética	26
2.4.1. Versión iterativa	27
2.4.2. Ejemplos	27
2.5. Varianza y Desviación Típica	29
2.5.1. Versión iterativa	29
2.5.2. Ejemplos	30
2.6. Mediana	31
2.7. Medidas de dispersión: Cuartiles	32

## 1. Apartado 1

Análisis estadístico de descripción de Datos en R. Para realizar este análisis utilizaremos dos ficheros de datos:

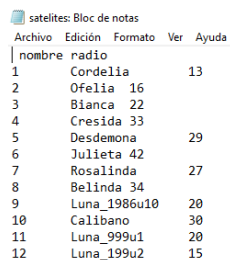
1. **Fichero .txt** con los datos de los satélites menores de Urano: nombre del satélite y su radio en km.
2. **Fichero .sav** (SPSS) formado por datos de automóviles, tales como su consumo en mpg (millas por galón), cilindrada, aceleración, año de fabricación, modelo etc.

### 1.1. Fichero .txt

Para comenzar a leer ficheros .txt deberemos seguir una serie de reglas para generar el formato correcto:

- Debe haber una tabulación entre dato y dato.
- Debe haber una primera columna que enumere las filas excepto la primera fila que tendrá un espacio en blanco. Además, en la primera fila irá el nombre de las variables.
- Hay que introducir un *enter* al final de la última fila.
- Los decimales se introducen con punto.
- En las variables tipo caracter no se puede dejar un espacio entre caracteres.

Siguiendo estas reglas generaremos un fichero con los datos de los satélites menores de Urano:



	nombre	radio	
1	Cordelia	13	
2	Ofelia	16	
3	Bianca	22	
4	Cresida	33	
5	Desdemona	29	
6	Julietta	42	
7	Rosalinda	27	
8	Bellinda	34	
9	Luna_1986u10	20	
10	Calibano	30	
11	Luna_999u1	20	
12	Luna_199u2	15	

Figura 1: Fichero .txt.

Una vez creado el fichero nos disponemos a leer los datos que contiene. Para ello utilizaremos el comando ***read.table***:

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

	satelites	radio
1	Cordelia	13
2	Ofelia	16
3	Bianca	22
4	Cresida	33
5	Desdemona	29
6	Julietta	42
7	Rosalinda	27
8	Belinda	34
9	Luna-1986U10	20
10	Calibano	30
11	Luna-999U1	20
12	Luna-199U2	15

Para llevar a cabo el análisis de los datos anteriores calcularemos las siguientes magnitudes:

#### 1.1.1.1. Frecuencia Absoluta y Acumulada

Para calcular la frecuencia absoluta utilizaremos el comando **table**:

```
> frecabsradio <- table(satelites$radio)
> frecabsradio
```

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

Por otro lado, para calcular la frecuencia absoluta acumulada utilizaremos la frecuencia absoluta anterior y, con el comando **cumsum**, realizaremos la suma acumulativa de las frecuencias absolutas:

```
> frecabsacmradio <- cumsum(frecabsradio)
> frecabsacmradio
```

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

### 1.1.2. Frecuencia Relativa y Acumulada

Para calcular la frecuencia relativa crearemos la siguiente función a través del comando ***function***. Esta función dividirá la frecuencia absoluta de cada dato entre el número total de datos:

```
> frecrel <- function(x) {table(x)/length(x)}
```

Una vez creada la función la guardamos en un fichero *.R* a través del comando ***dump***, lo que nos permitirá cargarla en cualquier script que hagamos:

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

Para cargar scripts en *.R* y poder utilizar sus funciones utilizaremos el comando ***source***:

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

Una vez cargado calcularemos la frecuencia relativa:

```
> frecrelradio <- frecrel(satelites$radio)
> #Lo convertimos a dataframe para mostrarlo por pantalla de una forma más limpia
> df = data.frame(frecrelradio)
> print(df, row.names = FALSE)
```

x	Freq
13	0.08333333
15	0.08333333
16	0.08333333
20	0.16666667
22	0.08333333
27	0.08333333
29	0.08333333
30	0.08333333
33	0.08333333
34	0.08333333
42	0.08333333

Por otro lado, para calcular la frecuencia relativa acumulada utilizaremos la frecuencia relativa anterior y con el comando **cumsum** realizaremos la suma acumulativa de las frecuencias relativas:

```
> frecrelacmradio <- cumsum(frecrelradio)
> #Lo convertimos a dataframe para mostrarlo por pantalla de una forma más limpia
> df = data.frame(frecrelacmradio)
> print(df)
```

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

### 1.1.3. Media Aritmética

Para calcular la media aritmética utilizaremos el comando **mean**:

```
> mr <- mean(satelites$radio)
> mr

[1] 25.08333
```

### 1.1.4. Desviación Típica

Para calcular la desviación típica utilizaremos el comando **sd**:

```
> sdr <- sd(satelites$radio)
> sdr

[1] 8.857029
```

El problema de esta función **sd** es que está pensada para poblaciones haciendo uso de la siguiente fórmula matemática:

$$\sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}} \quad (1)$$

Mientras que la fórmula de la desviación aplicable a nuestra muestra es la siguiente:

$$\sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}} \quad (2)$$

Si nos fijamos la única diferencia la tenemos en el denominador por lo que debemos realizar la siguiente modificación para calcular la desviación típica aplicable a nuestra muestra:

```
> #Elevando al cuadrado la desviación típica, quitamos la raíz,  
> #multiplicamos por n-1/n (n = numero de elementos) y, a continuacion,  
> #creamos de nuevo la raiz cuadrada  
> sdr <- sqrt((sdr^2)*(11/12))  
> sdr  
  
[1] 8.47996
```

#### 1.1.5. Varianza

Para calcular la varianza utilizaremos el comando *var*. Como la varianza es el cuadrado de la desviación debemos realizar la misma modificación que antes para calcular la varianza sobre nuestra muestra y no sobre una población:

```
> varr <- var(satelites$radio)  
> varr <- varr*(11/12)  
> varr  
  
[1] 71.90972
```

#### 1.1.6. Mediana

Para calcular la mediana utilizaremos el comando *median*:

```
> medianr <- median(satelites$radio)  
> medianr  
  
[1] 24.5
```

#### 1.1.7. Cuantiles

Para calcular los cuantiles utilizaremos el comando *quantile*:

```
> #Cuartil 1: 1/4  
> cuar1 <- quantile(satelites$radio,0.25)  
> cuar1  
  
25%  
19  
  
> #Cuartil 2: 2/4 (coincide con la mediana)  
> cuar2 <- quantile(satelites$radio,0.50)  
> cuar2  
  
50%  
24.5  
  
> #Cuartil 3: 3/4  
> cuar3 <- quantile(satelites$radio,0.75)  
> cuar3
```

```
75%
30.75
```

```
> #Cuantil 54
> cuar54 <- quantile(satelites$radio,0.54)
> cuar54
```

```
54%
26.7
```

## 1.2. Fichero .sav (SPSS)

Para comenzar a leer ficheros .sav deberemos cargar el paquete *foreign*:

```
> library(foreign)
> #Vemos los paquetes cargados
> #Lo convertimos a dataframe para mostrarlo por pantalla de una forma más limpia
> df <- data.frame(search())
> df
```

```
      search..
1      .GlobalEnv
2  package:foreign
3  package:stats
4  package:graphics
5  package:grDevices
6  package:utils
7  package:datasets
8  package:methods
9      Autoloads
10 package:base
```

Una vez cargado el paquete nos disponemos a cargar el fichero .sav con el comando ***read.spss***:

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

A partir de hora trabajaremos con la variable mpg para realizar el análisis:

```
> #Veamos los datos de mpg
> mpg <- cardata$mpg
> #Lo convertimos a dataframe para mostrarlo por pantalla de una forma más limpia
> #Mostraremos solo del dato 100 al 110
> df <- data.frame(mpg)
> df <- df[100:110,]
> df
```

```
[1] 36.4 30.4 40.9 29.8 35.0 33.0 34.5 28.1 NA 30.7 36.0
```



Como podemos observar, tenemos valores a NA por lo que deberemos quitarlos (por el momento) para realizar el análisis:

```
> mpg <- mpg[!is.na(mpg)]
> # Lo convertimos a dataframe para mostrarlo por pantalla de una forma más limpia
> # Mostraremos solo del dato 100 al 110 para verificar que ha quitado los NA
> df <- data.frame(mpg)
> df <- df[100:110,]
> df

[1] 36.4 30.4 40.9 29.8 35.0 33.0 34.5 28.1 30.7 36.0 44.0
```

Una vez tenemos los datos listos nos disponemos a realizar el análisis a través del cálculo de las siguientes magnitudes:

### 1.2.1. Frecuencia Absoluta y Acumulada

Para calcular la frecuencia absoluta utilizaremos de nuevo el comando **table**:

```
> frecabsmpg <- table(mpg)
> # Mostramos los 10 primeros elem de la tabla generada
> frecabsmpg[1:10]
```

```
mpg
15.5 16.2 16.5 16.9    17 17.5 17.6 17.7 18.1 18.2
    1    1    1    1    2    1    2    1    2    1
```

Y para calcular la frecuencia absoluta acumulada realizaremos la suma acumulativa de las frecuencias absolutas:

```
> frecabsacmpg <- cumsum(frecabsmpg)
> # Mostramos los 10 primeros elem de la tabla generada
> frecabsacmpg[1:10]
```

```
15.5 16.2 16.5 16.9    17 17.5 17.6 17.7 18.1 18.2
    1    2    3    4    6    7    9   10   12   13
```

### 1.2.2. Frecuencia Relativa y Acumulada

Para calcular la frecuencia relativa utilizaremos de nuevo comando **frecrel**:

```
> frecrelmpg <- frecrel(mpg)
> # Lo convertimos a dataframe para mostrarlo por pantalla de una forma más limpia
> df = data.frame(frecrelmpg)
> # Mostramos los 5 primeros elem de la tabla generada
> print(df[1:5,], row.names = FALSE)
```

```
      x      Freq
15.5 0.006493506
16.2 0.006493506
16.5 0.006493506
16.9 0.006493506
17   0.012987013
```

Y para calcular la frecuencia relativa acumulada realizaremos la suma acumulativa de las frecuencias relativas:

```
> frecrelacmpg <- cumsum(frecrelmpg)
> # Lo convertimos a dataframe para mostrarlo por pantalla de una forma más limpia
> df = data.frame(frecrelacmpg)
> # Mostramos los 5 primeros elem de la tabla generada
> print(df[1:5,])

[1] 0.006493506 0.012987013 0.019480519 0.025974026 0.038961039
```

### 1.2.3. Media Aritmética

Calculamos la media aritmética de la columna mpg:

```
> mmpg <- mean(mpg)
> mmpg

[1] 28.79351
```

### 1.2.4. Desviación Típica

Calculamos la desviación típica de la columna mpg, haciendo de nuevo la corrección del  $n-1$  por el  $n$ :

```
> sdr <- sd(mpg)
> sdr <- sqrt((sdr^2)*((length(mpg)-1)/length(mpg)))
> sdr

[1] 7.353219
```

### 1.2.5. Varianza

Calculamos la varianza de la columna mpg con la misma corrección:

```
> varr <- var(mpg)
> varr <- varr*((length(mpg)-1)/length(mpg))
> varr

[1] 54.06983
```

### 1.2.6. Mediana

Calculamos la mediana de la columna mpg:

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

[1] 28.9
```

### 1.2.7. Cuartiles

Calculamos los cuartiles de la columna mpg, sin olvidar el cuantil 54:

```
> cuar1 <- quantile(mpg,0.25)
> cuar1

25%
22.55

> cuar2 <- quantile(mpg,0.50)
> cuar2

50%
28.9

> cuar3 <- quantile(mpg,0.75)
> cuar3

75%
34.275
```

## 2. Apartado 2

Análisis estadístico de descripción de Datos en R usando nuevos formatos de fichero, así como nuevas funciones y librerías. Para ello se han utilizado los siguientes archivos:

- Fichero **.txt** con los datos de los satélites de Urano: nombre del satélite y su radio en km.
- Fichero **.csv** con los datos de los satélites anteriores (ver anexo para lectura de ficheros en .csv).
- Fichero **.sav** (SPSS) formado por datos de automóviles, tales como su consumo en mpg (millas por galón), cilindrada, aceleración, año de fabricación, modelo etc.
- Fichero **.json** con los datos de contaminación registrados en el año 2019 en la ciudad de Alcobendas. <sup>1</sup>
- Fichero **.xlsx** con la información de diferentes especies de plantas. <sup>2</sup>

---

<sup>1</sup><https://datos.gob.es/es/catalogo/101280066-contaminacion-atmosferica-por-horas-ano-en-curso>

<sup>2</sup><https://www.kaggle.com/uciml/iris/download>

Para la realización de la práctica se han utilizado los siguientes paquetes:

- `package(foreign)` para la lectura de ficheros `.sav`.
- `package(rjson)` para la lectura de ficheros `.json`.
- `package(XLConnect)` para la lectura de ficheros Excel `.xlsx`.
- `package(dplyr)` , el cual proporciona una gramática para la manipulación de *data frames*.

Para la lectura de ficheros, utilizaremos una función denominada **leer.archivo** que se encargará de crear un *dataframe* a partir del archivo indicado como parámetro. Por otro lado, la función proporcionará una serie de argumentos adicionales en función del tipo de archivo.

```
> leer.archivo <- function(nombre, header = FALSE, sep = "", dec=".", skipNul=FALSE,
+ to.data.frame=TRUE, sheet=1, startRow=1, endCol=2){}
```

1. *header(para ficheros .txt y .csv)*: indica si el archivo presenta o no cabecera. Por defecto está establecido a **FALSE**.
2. *sep(para ficheros .txt y .csv)*: indica el carácter separador, establecido a **cadena vacía** por defecto.
3. *dec(para ficheros .txt y .csv)*: indica la separación de números decimales, por defecto a **'.'**
4. *skipNul(para ficheros .txt y .csv)*: indica si la carga debe saltarse valores a **NA**. Por defecto, está establecido a **FALSE**.
5. *to.data.frame(para ficheros .sav)*: si queremos que los datos estén almacenados en un *data frame*, por lo que está establecido a **TRUE** por defecto.
6. *sheet(para ficheros .xlsx)*: indica el número de hoja que deseamos importar. Por defecto, la lectura de los datos se realiza sobre la primera hoja.
7. *startRow(para ficheros .xlsx)*: indica la fila de inicio (**1**, por defecto).
8. *endCol(para ficheros .xlsx)*: indica la última columna (**2**, por defecto).

Para distinguir entre los diferentes tipos de archivo, utilizaremos el comando *strsplit* con el fin de separar el nombre del archivo de su extensión. Por otro lado, mediante el comando **unlist** convertimos la lista obtenida a vector. Ejemplo:

```
> aux = unlist(strsplit("fichero_entrada.txt", "."));
> aux;
```

```
[1] "fichero_entrada" "txt"
```

Una vez obtenida la extensión, mediante la función **switch** realizaremos la lectura de archivo, en función de su extensión. Para los archivos `.json` y `.xlsx` importaremos las librerías necesarias para su lectura.

Código:

```
> leer.archivo <- function(nombre, header = FALSE, sep = "", dec=".", skipNul=FALSE,
+                           to.data.frame=TRUE, sheet=1,startRow=1,endCol=2){
+   aux <- unlist(strsplit(nombre,"[.]"))
+   switch(aux[length(aux)],
+     "txt"={
+       read.table(nombre,header=header, sep=sep, dec=dec,
+                 skipNul=skipNul)
+     },
+     "csv"={
+       read.csv(nombre,header=header, sep=sep, dec=dec,
+               skipNul=skipNul)
+     },
+     "json"={
+       if(!require(rjson)){
+         install.packages("rjson")
+         require(rjson)
+       }
+       na.omit(as.data.frame(do.call(rbind,fromJSON(file=nombre))))
+     },
+     "sav"={
+       require(foreign)
+       read.spss(nombre,to.data.frame=to.data.frame)
+     },
+     "xlsx"={
+       if(!require(XLConnect)){
+         install.packages("XLConnect")
+         require(XLConnect)
+       }
+       readWorksheetFromFile(nombre,sheet=sheet,startRow=startRow,
+                             endCol=endCol)
+     }
+   )
+ }
```

### Ejemplo de ejecución con el fichero *satelites.txt*:

```
> satellites <- leer.archivo("satelites.txt", header=T)
> satellites
```

	satellites	radio
1	Cordelia	13
2	Ofelia	16
3	Bianca	22
4	Cresida	33
5	Desdemona	29
6	Julietta	42
7	Rosalinda	27
8	Belinda	34
9	Luna-1986U10	20
10	Calibano	30
11	Luna-999U1	20
12	Luna-199U2	15

### Ejemplo de ejecución con el fichero *datosDecontaminacion.json*

```
> datos_contaminacion <- leer.archivo("datos_de_contaminacion.json")
> nrow(datos_contaminacion)
```

Dado el elevado número de filas obtenidas, vamos a utilizar el paquete *dplyr* para mostrar un subconjunto del data frame. Esta librería permite realizar consultas al dataframe, similar a una consulta **SQL**<sup>3</sup>. Esta librería proporciona los siguientes comandos para realizar consultas sobre un data frame:

- **select**: permite seleccionar un conjunto de columnas.
- **filter**: devuelve un conjunto de filas que cumplan una condición dada.
- **arrange**: permite reordenar las filas de un data frame.
- **rename**: permite renombrar variables en un data frame.
- **mutate**: permite añadir nuevas columnas o modificar columnas existentes.
- **head**: para obtener las primeras n filas.
- **summarise**: para calcular resúmenes estadísticos.
- **pipe**: se emplea para concatenar varias acciones.

---

<sup>3</sup><https://rsanchezs.gitbooks.io/rprogramming/content/chapter9/dplyr.html>

Si queremos obtener las 10 primeras filas del data frame anterior:

```
> library(dplyr)
> #pipe = %>%
> #Equivalente a SELECT(fecha_medicion, tipo_contaminante,
> #contaminante, porcentaje) * FROM datos_contaminacion LIMIT 10
> datos_contaminacion %>% select(fecha_medicion, tipo_contaminante,
+ contaminante, porcentaje) %>% head(10)
```

	fecha_medicion	tipo_contaminante	contaminante
1	2019-01-01T00:00	Hidrocarburo	Hidrocarburos no metano
2	2019-01-01T00:00	Hidrocarburo	Hidrocarburos totales
3	2019-01-01T00:00	No hidrocarburo	Benceno
4	2019-01-01T00:00	No hidrocarburo	Dioxido de nitrogeno
5	2019-01-01T00:00	No hidrocarburo	Meta-para-xileno
6	2019-01-01T00:00	No hidrocarburo	Monoxido de nitrogeno
7	2019-01-01T00:00	No hidrocarburo	Ozono
8	2019-01-01T00:00	No hidrocarburo	Particulas en suspension < PM10
9	2019-01-01T00:00	No hidrocarburo	Tolueno
10	2019-01-01T01:00	Hidrocarburo	Hidrocarburos no metano

	porcentaje
1	9.52380952380952
2	90.4761904761905
3	0.72793448589627
4	30.9372156505914
5	3.41219290263876
6	44.1310282074613
7	1.81983621474067
8	13.6487716105551
9	5.32302092811647
10	9.52380952380952

#### Ejemplo de ejecución con el fichero *satelites.csv*

```
> satelites_csv <- leer.archivo("satelites.csv", header=T, sep=",")
> satelites_csv
```

	satelites	radio
1	Cordelia	13
2	Ofelia	16
3	Bianca	22
4	Cresida	33
5	Desdemona	29
6	Julietta	42
7	Rosalinda	27
8	Belinda	34
9	Luna-1986U10	20
10	Calibano	30
11	Luna-999U1	20
12	Luna-199U2	15

### Ejemplo de ejecucion con el fichero *cardata.sav*

```
> cardata <- leer.archivo("cardata.sav")
> #Equivalente a SELECT(mpg, cylinders, accel, weight) FROM cardata
> #LIMIT 10
> cardata %>% select(mpg, cylinders, accel, weight) %>% head(10)
```

	mpg	cylinders	accel	weight
1	36.1	4	14.4	1800
2	19.9	8	15.5	3365
3	19.4	8	13.2	3735
4	20.2	8	12.8	3570
5	19.2	6	19.2	3535
6	20.5	6	18.2	3155
7	20.2	6	15.8	2965
8	25.1	4	15.4	2720
9	20.5	6	17.2	3430
10	19.4	6	17.2	3210

### Ejemplo de ejecucion con el fichero *iris.xlsx*

```
> #Fila de inicio: 1
> #Numero de columnas: 6
> iris <- leer.archivo("iris.xlsx", startRow = 1, endCol = 6)
> #Equivalente a SELECT * FROM iris LIMIT 10
> iris %>% head(10)
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	1	5.1	3.5	1.4	0.2	Iris-setosa
2	2	4.9	3.0	1.4	0.2	Iris-setosa
3	3	4.7	3.2	1.3	0.2	Iris-setosa
4	4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	5.0	3.6	1.4	0.2	Iris-setosa
6	6	5.4	3.9	1.7	0.4	Iris-setosa
7	7	4.6	3.4	1.4	0.3	Iris-setosa
8	8	5.0	3.4	1.5	0.2	Iris-setosa
9	9	4.4	2.9	1.4	0.2	Iris-setosa
10	10	4.9	3.1	1.5	0.1	Iris-setosa

## 2.1. Lectura de ficheros .csv

Al igual que vimos en el apartado 1 para los ficheros .txt, para los ficheros .csv debemos seguir una serie de reglas a la hora de escribir nuestro archivo para poder generar el formato correcto una vez que importemos los datos a R:

- Debe haber una coma y un espacio entre dato y dato.
- Los datos se pueden separar con otro elemento. Por ejemplo: sep = ",".
- Cada grupo de datos (instancia) va en una sola fila.
- Hay que introducir un *enter* al final de cada fila.
- Los decimales se introducen con punto.
- En las variables tipo caracter no se puede dejar un espacio entre caracteres.



## 2.2. Moda, Frecuencia Absoluta y Frecuencia Absoluta Acumulada

### 2.2.1. Frecuencia absoluta y las funciones recursivas en R

Para calcular la frecuencia absoluta creamos en un primer momento una función recursiva. Para ello, utilizamos la función *match*, que analiza las apariciones de un elemento en una lista, devolviendo su posición. Por otro lado, calculamos la frecuencia absoluta acumulada con la suma progresiva de la absoluta.

```
> ## Frecuencia absoluta
> # Calcula la frecuencia absoluta recursivamente, analizando en cada llamada el primer
> # elemento del vector de datos (la condicion de parada es que quede solo un elemento
> # en el). Para cada elemento analiza si este ya forma parte de la lista. Si es asi
> # suma uno a su contador. Si no, lo añade a la lista y pone su contador a 1.
> freq.absoluta <- function(original,fi=NULL,valor=NULL){
+   #Analizamos en primer lugar la primera aparicion
+   #de nuestro primer elemento en la lista de valores
+   #(inicialmente a NULL)
+   num=match(original[1],valor)
+
+   #Como condicion de parada, comprobamos si la lista de
+   #elementos tiene longitud 1
+   if(length(original)==1){
+     if(is.na(num)){
+       valor=c(valor,original[1])
+       fi=c(fi,1)
+     } else{
+       fi[num]=fi[num]+1
+     }
+     aux=data.frame(valor,fi)
+     aux[order(aux$valor),]
+   } else{
+     if(is.na(num)){
+       valor=c(valor,original[1])
+       fi=c(fi,1)
+     } else{
+       fi[num]=fi[num]+1
+     }
+     freq.absoluta(original[2:length(original)],fi,valor)
+   }
+ }
> ## Frecuencia absoluta acumulada
> # Calcula la frecuencia absoluta acumulada llamando a freq.absoluta y realizando la
> # suma acumulativa sobre el vector resultante.
> freq.absoluta.acumulada <- function(vector){
+   aux=freq.absoluta(vector)
+   valor=aux$valor
+   fai=cumsum(aux$fi)
+   data.frame(valor,fai)
+ }
```

Estas funciones funcionan correctamente para un número pequeño de elementos, pero cuando el número de estos aumenta demasiado, la pila (R utiliza la pila de C, la cual no es muy adecuada para un gran número de llamadas anidadas) se desborda y no puede calcularse el resultado. Por ello, hemos acabado realizando una versión iterativa de esta y de todas las funciones recursivas que se expondrán a continuación a pesar de que su funcionamiento sea correcto dejando a parte las limitaciones de memoria.

Y si, se ha intentado solucionar dicho problema aumentando la memoria reservada para la pila al maximo posible y, aun así, se han obtenido errores en algunos de los archivos más grandes que veremos a continuación.

Aqui tenemos la versión iterativa de las frecuencias absoluta y absoluta acumulada:

```
> ## Frecuencia absoluta iterativa
> freq.absoluta.iterativa <- function(original){
+     fi=NULL
+     valor=NULL
+     for (i in 1:length(original)){
+         num=match(original[i],valor)
+         if(is.na(num)){
+             valor=c(valor,original[i])
+             fi=c(fi,1)
+         } else{
+             fi[num]=fi[num]+1
+         }
+     }
+     valor=unlist(valor)
+     aux=data.frame(valor,fi)
+     aux[order(aux$valor),]
+ }
> ## Frecuencia absoluta acumulada iterativa
> freq.absoluta.acumulada.iterativa <- function(vector){
+     aux=freq.absoluta.iterativa(vector)
+     valor=aux$valor
+     fai=cumsum(aux$fi)
+     data.frame(valor,fai)
+ }
```

Para realizar el cálculo de la Moda, creamos una función que obtenga la mayor frecuencia absoluta

```
> ## Moda
> #
> moda <- function(vector){
+     aux=freq.absoluta.iterativa(vector)
+     moda=aux[which.max(aux$fi),]$valor
+     data.frame(moda)
+ }
```

**Cálculo de moda, frecuencia absoluta y acumulada  
para *satelites.txt* y *satelites.csv*:**

```
> #Moda y frecuencias absolutas de satellites.txt
> moda(satelites$radio)
```

```
moda
1    20
```

```
> # Como son pocos datos, podemos utilizar las versiones recursivas
> freq.absoluta(satelites$radio)
```

```
valor fi
1      13  1
11     15  1
2      16  1
9      20  2
3      22  1
7      27  1
5      29  1
10     30  1
4      33  1
8      34  1
6      42  1
```

```
> freq.absoluta.acumulada(satelites$radio)
```

```
valor fai
1      13  1
2      15  2
3      16  3
4      20  5
5      22  6
6      27  7
7      29  8
8      30  9
9      33 10
10     34 11
11     42 12
```

```
> #Moda y frecuencias absolutas de satellites.csv
> moda(satelites_csv$radio)
```

```
moda
1    20
```

```
> freq.absoluta(satelites_csv$radio)
```

	valor	fi
1	13	1
11	15	1
2	16	1
9	20	2
3	22	1
7	27	1
5	29	1
10	30	1
4	33	1
8	34	1
6	42	1

```
> freq.absoluta.acumulada(satelites_csv$radio)
```

	valor	fai
1	13	1
2	15	2
3	16	3
4	20	5
5	22	6
6	27	7
7	29	8
8	30	9
9	33	10
10	34	11
11	42	12

**Cálculo de moda, frecuencia absoluta y acumulada  
para los valores mpg de *cardata.sav***

```
> moda(na.omit(cardata$mpg))
```

	moda
1	36

```
> #Debido a la elevado numero de valores, vamos a mostrar los 10 primeros datos con dplyr
> freq.absoluta(na.omit(cardata$mpg)) %>% head(10)
```

	valor	fi
26	15.5	1
68	16.2	1
23	16.5	1
25	16.9	1
21	17.0	2
13	17.5	1
22	17.6	2
12	17.7	1
11	18.1	2
24	18.2	1

```
> freq.absoluta.acumulada(na.omit(cardata$mpg)) %>% head(10)
```

	valor	fai
1	15.5	1
2	16.2	2
3	16.5	3
4	16.9	4
5	17.0	6
6	17.5	7
7	17.6	9
8	17.7	10
9	18.1	12
10	18.2	13

De forma adicional, podemos también calcular tanto la moda como las frecuencias absolutas agrupadas en clases, mediante *dplyr*. Dicha librería dispone de la función **do**, la cual permite ejecutar cualquier función sobre una o varias columnas de nuestro dataframe. Veamos un ejemplo: **Cálculo de la moda, frecuencias absoluta y acumulada de longitud de pétalo en función de la especie de planta en *iris.xlsx*:**

```
> freq.absoluta(iris$Species)
```

	valor	fi
1	Iris-setosa	50
2	Iris-versicolor	50
3	Iris-virginica	50

```
> freq.absoluta.acumulada(iris$Species)
```

	valor	fai
1	Iris-setosa	50
2	Iris-versicolor	100
3	Iris-virginica	150

```
> #Equivalente a:
```

```
> #SELECT freq.absoluta(iris$PetalLengthCm) FROM iris GROUP_BY "Species"
```

```
> iris %>% group_by(Species) %>% do(freq.absoluta(iris$PetalLengthCm)) %>% head(5)
```

```
# A tibble: 5 x 3
```

# Groups:	Species	[1]
	Species	valor fi
	<chr>	<dbl> <dbl>
1	Iris-setosa	1 1
2	Iris-setosa	1.1 1
3	Iris-setosa	1.2 2
4	Iris-setosa	1.3 7
5	Iris-setosa	1.4 12

### Cálculo de moda, frecuencia absoluta y acumulada para *datosDecontaminacion.json*

```
> moda(datos_contaminacion$contaminante)

      moda
1 Benceno

> freq.absoluta.iterativa(datos_contaminacion$contaminante)

      valor  fi
3          Benceno 929
4      Dioxido de nitrogeno 929
1  Hidrocarburos no metano 929
2  Hidrocarburos totales 929
5      Meta-para-xileno 928
6      Monoxido de nitrogeno 928
7              Ozono 928
8 Particulas en suspension < PM10 928
9          Tolueno 928

> freq.absoluta.acumulada.iterativa(datos_contaminacion$contaminante)

      valor  fai
1          Benceno 929
2      Dioxido de nitrogeno 1858
3  Hidrocarburos no metano 2787
4  Hidrocarburos totales 3716
5      Meta-para-xileno 4644
6      Monoxido de nitrogeno 5572
7              Ozono 6500
8 Particulas en suspension < PM10 7428
9          Tolueno 8356
```

### 2.3. Frecuencia relativa y frecuencia relativa acumulada

Para realizar el cálculo de la frecuencia relativa, calculamos las frecuencias absolutas y las dividimos entre el número total de elementos. Una vez obtenidos los valores de frecuencia relativa, mediante la función *freq.relaciva.acumulada* vamos sumando progresivamente los valores de

$$f_i \quad (3)$$

:

```
> ## Frecuencia relativa
> # Obtiene el vector de frecuencias absolutas y divide cada valor entre el numero
> # total de elementos.
> freq.relaciva <- function(vector){
+     aux=freq.absoluta(vector)
+     valor=aux$valor
+     fri=aux$fi/sum(aux$fi)
```

```

+         data.frame(valor,fri)
+ }
> ## Frecuencia relativa acumulada
> # Calcula la frecuencia relativa acumulada a partir de la suma acumulativa de
> # la frecuencia relativa.
> freq.relacumulada <- function(vector){
+     aux=freq.relacumulada(vector)
+     valor=aux$valor
+     frai=cumsum(aux$fri)
+     data.frame(valor,frai)
+ }

```

### 2.3.1. Versión iterativa

```

> ## Frecuencia relativa iterativa
> freq.relacumulada.iterativa <- function(vector){
+     aux=freq.absoluta.iterativa(vector)
+     valor=aux$valor
+     fri=aux$fri/sum(aux$fri)
+     data.frame(valor,fri)
+ }
> ## Frecuencia relativa acumulada iterativa
> freq.relacumulada.iterativa <- function(vector){
+     aux=freq.relacumulada.iterativa(vector)
+     valor=aux$valor
+     frai=cumsum(aux$fri)
+     data.frame(valor,frai)
+ }

```

### 2.3.2. Ejemplos de cálculo de frecuencias relativas

```

> #satelites.txt
> freq.relacumulada(satelites$radio)

```

	valor	fri
1	13	0.08333333
2	15	0.08333333
3	16	0.08333333
4	20	0.16666667
5	22	0.08333333
6	27	0.08333333
7	29	0.08333333
8	30	0.08333333
9	33	0.08333333
10	34	0.08333333
11	42	0.08333333

```
> freq.relativa.acumulada(satelites$radio)
```

	valor	frai
1	13	0.08333333
2	15	0.16666667
3	16	0.25000000
4	20	0.41666667
5	22	0.50000000
6	27	0.58333333
7	29	0.66666667
8	30	0.75000000
9	33	0.83333333
10	34	0.91666667
11	42	1.00000000

```
> #satelites.csv
```

```
> freq.relativa(satelites_csv$radio)
```

	valor	fri
1	13	0.08333333
2	15	0.08333333
3	16	0.08333333
4	20	0.16666667
5	22	0.08333333
6	27	0.08333333
7	29	0.08333333
8	30	0.08333333
9	33	0.08333333
10	34	0.08333333
11	42	0.08333333

```
> freq.relativa.acumulada(satelites_csv$radio)
```

	valor	frai
1	13	0.08333333
2	15	0.16666667
3	16	0.25000000
4	20	0.41666667
5	22	0.50000000
6	27	0.58333333
7	29	0.66666667
8	30	0.75000000
9	33	0.83333333
10	34	0.91666667
11	42	1.00000000



```

> #cardata.sav
> freq.relativa(na.omit(cardata$mpg)) %>% head(10)

  valor      fri
1  15.5 0.006493506
2  16.2 0.006493506
3  16.5 0.006493506
4  16.9 0.006493506
5  17.0 0.012987013
6  17.5 0.006493506
7  17.6 0.012987013
8  17.7 0.006493506
9  18.1 0.012987013
10 18.2 0.006493506

> freq.relativa.acumulada(na.omit(cardata$mpg)) %>% head(10)

  valor      frai
1  15.5 0.006493506
2  16.2 0.012987013
3  16.5 0.019480519
4  16.9 0.025974026
5  17.0 0.038961039
6  17.5 0.045454545
7  17.6 0.058441558
8  17.7 0.064935065
9  18.1 0.077922078
10 18.2 0.084415584

> #iris.xlsx
> freq.relativa(iris$Species)

  valor      fri
1  Iris-setosa 0.3333333
2 Iris-versicolor 0.3333333
3  Iris-virginica 0.3333333

```

```

> freq.relativa.acumulada(iris$Species)

      valor      frai
1  Iris-setosa 0.3333333
2 Iris-versicolor 0.6666667
3  Iris-virginica 1.0000000

> #datosDecontaminacion.json
> freq.relativa.iterativa(datos_contaminacion$contaminante)

      valor      fri
1      Benceno 0.1111776
2  Dioxido de nitrogeno 0.1111776
3  Hidrocarburos no metano 0.1111776
4  Hidrocarburos totales 0.1111776
5    Meta-para-xileno 0.1110579
6  Monoxido de nitrogeno 0.1110579
7          Ozono 0.1110579
8 Particulas en suspension < PM10 0.1110579
9          Tolueno 0.1110579

> freq.relativa.acumulada.iterativa(datos_contaminacion$contaminante)

      valor      frai
1      Benceno 0.1111776
2  Dioxido de nitrogeno 0.2223552
3  Hidrocarburos no metano 0.3335328
4  Hidrocarburos totales 0.4447104
5    Meta-para-xileno 0.5557683
6  Monoxido de nitrogeno 0.6668262
7          Ozono 0.7778842
8 Particulas en suspension < PM10 0.8889421
9          Tolueno 1.0000000

```

## 2.4. Media aritmética

Para realizar el cálculo de la media aritmética emplearemos una función que sumará recursivamente los elementos de la columna, hasta que la longitud de la lista sea 1 (condición de parada), dividiendo finalmente la suma resultante entre el número total de elementos.

```

> ## Media recursiva
> # Calcula la media aritmética de forma recursiva, sumando en cada llamada el primer
> # elemento del vector en ese momento y llevando la cuenta del numero de elementos
> # vistos hasta el momento. La condicion de parada es que el numero de elementos en
> # dicho vector sea 1.
> media.recursiva <- function(vector,n=0,sum=0){
+   if(length(vector)==1){
+     (sum+vector[1])/(n+1)
+   } else{
+     media.recursiva(vector[2:length(vector)],n+1,sum+vector[1])
+   }
+ }

```

### 2.4.1. Versión iterativa

```
> ## Media iterativa
> media.iterativa <- function(vector){
+   sum=0
+   for(i in 1:length(vector)){
+     sum=sum+vector[i]
+   }
+   sum/length(vector)
+ }
```

### 2.4.2. Ejemplos

```
[1] "Media de radios de satelites.txt: 25.0833333333333"
[1] "Media de radios de satelites.csv: 25.0833333333333"
[1] "Media de mpg de cardata.sav: 28.7935064935065"
[1] "Media de las longitudes de pétalo de iris.xlsx: 3.75866666666667"
```

Si queremos calcular la media agrupada en clases, utilizaremos la librería *dplyr*:

**Aceleración en función de la marca de automóvil en *cardata.sav*:**

```
> library(dplyr)
> #Eliminamos posibles filas a NA
> #Equivalente a:
> #SELECT mean(accel) FROM cardata GROUP_BY(make)
> cardata %>% group_by(cardata$make) %>% summarise(aceleracion = media.iterativa(na.omit(c

# A tibble: 26 x 2
  `cardata$make`      aceleracion
  <fct>             <dbl>
1 "AMC"              16.3
2 "Audi"             16.3
3 "Buick"            16.3
4 "Cadillac"         16.3
5 "Chevrolet"        16.3
6 "Chrysler"         16.3
7 "Datsun"           16.3
8 "Dodge"            16.3
9 "Fiat"             16.3
10 "Ford"            16.3
# ... with 16 more rows
```

### Longitud de pétalo en función de la especie en *iris.xlsx*:

```
> #Equivalente a:
> #SELECT PetalLengthCm FROM iris GROUP_BY "Species"
> iris %>% group_by(Species) %>% summarise(longitudPetal = media.iterativa(PetalLengthCm))

# A tibble: 3 x 2
  Species      longitudPetal
  <chr>          <dbl>
1 Iris-setosa      1.46
2 Iris-versicolor  4.26
3 Iris-virginica   5.55
```

### Niveles medios de concentracion por contaminante en función del tipo de contaminante en *datosDecontaminacion.json*:

```
> #Mediante el comando MUTATE creamos una nueva columna
> datos_contaminacion = datos_contaminacion %>%
+ mutate(aux=unlist(datos_contaminacion$contaminante))
> datos_contaminacion = datos_contaminacion %>%
+ mutate(aux_num=as.numeric(datos_contaminacion$concentracion))
> datos_contaminacion %>% group_by(aux) %>%
+ summarise(media_concentracion=media.iterativa(na.omit(aux_num)))

# A tibble: 9 x 2
  aux                      media_concentracion
  <chr>                      <dbl>
1 Benceno                    0.507
2 Dioxido de nitrógeno       44.4
3 Hidrocarburos no metano    0.142
4 Hidrocarburos totales      1.56
5 Meta-para-xileno          3.56
6 Monóxido de nitrógeno      41.2
7 Ozono                     31.2
8 Partículas en suspensión < PM10 12.3
9 Tolueno                    5.98
```

## 2.5. Varianza y Desviación Típica

Para el calculo de la Varianza utilizaremos un función recursiva que calcule el sumatorio de las diferencias entre cada valor y la media. Como condición de parada, cuando el vector tenga longitud 1 dividimos el sumatorio entre el número total de elementos  $n$ . Por otro lado, para el cálculo de la Desviación Típica aplicamos la raíz cuadrada sobre la Varianza.

```
> ## Varianza recursiva
> # Calcula la varianza aplicando la formula. En cada llamada, se suma al total el
> # resultado de restar al primer elemento del vector de valores la media (calculada
> # al inicio) y elevarla al cuadrado. Cuando la longitud del vector de elementos es
> # 1, se divide el sumatorio entre el numero total de elementos.
> varianza.recursiva <- function(vector, n = 0, suma = 0, media = media.recursiva(vector))
+   if(length(vector) == 1){
+     suma = suma + (vector[1]-media)^2
+     n = n + 1
+     suma/n
+   }
+   else{
+     suma = suma + (vector[1]-media)^2
+     n = n + 1
+     varianza.recursiva(vector[2:length(vector)], n, suma, media)
+   }
+ }
> ## Desviacion tipica recursiva
> # Realiza la raiz cuadrada de la varianza.
> desviacion.tipica.recursiva <- function(vector){
+   sqrt(varianza.recursiva(vector))
+ }
```

### 2.5.1. Versión iterativa

```
> ## Varianza iterativa
> varianza.iterativa <- function(vector){
+   media = media.iterativa(vector)
+   suma=0
+   for(i in 1:length(vector)){
+     suma = suma + (as.numeric(vector[i])-media)^2
+   }
+   suma/length(vector)
+ }
> ## Desviacion tipica iterativa
> desviacion.tipica.iterativa <- function(vector){
+   sqrt(varianza.iterativa(vector))
+ }
```

### 2.5.2. Ejemplos

```
> #satelites.txt
> varianza.rekursiva(satelites$radio)

[1] 71.90972

> desviacion.tipica.rekursiva(satelites$radio)

[1] 8.47996

> #satelites_csv
> varianza.rekursiva(satelites_csv$radio)

[1] 71.90972

> desviacion.tipica.rekursiva(satelites_csv$radio)

[1] 8.47996

> #cardata.sav
> varianza.rekursiva(na.omit(cardata$mpg))

[1] 54.06983

> desviacion.tipica.rekursiva(na.omit(cardata$mpg))

[1] 7.353219

> #iris.xlsx
> varianza.rekursiva(iris$PetalLengthCm)

[1] 3.092425

> desviacion.tipica.rekursiva(iris$PetalLengthCm)

[1] 1.758529

> #datosDecontaminacion.json
> #Eliminamos filas con cadenas vacias
> datos_contaminacion <- datos_contaminacion[datos_contaminacion$concentracion != "" & datos_contaminacion$porcentaje != "",]
> varianza.iterativa(datos_contaminacion$aux_num)

[1] 957.4109

> desviacion.tipica.iterativa(datos_contaminacion$aux_num)

[1] 30.94206
```

## 2.6. Mediana

Para el cálculo de la mediana utilizaremos una función que comprobará inicialmente la longitud del vector de entrada:

- Si el número de elementos es **impar**, la mediana es el elemento situado en la mitad del vector.
- Si el número de elementos es **par**, la mediana es la media resultante entre los dos elementos situados a la mitad del vector.

```
> #Calculo de la mediana
> #Funcion para comprobar si un numero es impar
> is.odd <- function(x) x %% 2 != 0
> mediana <- function(vector) {
+   if(is.odd(length(vector))) #impar
+   {
+       as.numeric(vector[(length(vector)+1)/2])
+   }else{ #par
+       (as.numeric(vector[(length(vector))/2]) +
+        as.numeric(vector[((length(vector))/2)+1]))/2
+   }
+ }
```

Ejemplos de ejecución:

```
> #satelites.txt
> mediana(satelites$radio)

[1] 34.5

> #satelites.csv
> mediana(satelites_csv$radio)

[1] 34.5

> #cardata.sav
> mediana(cardata$mpg)

[1] 26

> #iris.xlsx
> mediana(iris$PetalLengthCm)

[1] 4.35

> #datos_contaminacion.json
> mediana(datos_contaminacion$porcentaje)

[1] 0.5940594
```

## 2.7. Medidas de dispersión: Cuartiles

**IMPORTANTE:** para realizar el cálculo de los cuartiles los elementos han de estar ordenados. Para realizar el cálculo de los cuartiles, crearemos la siguiente función<sup>4</sup>:

1. Para calcular el primer cuartil, calculamos la siguiente expresión

$$\frac{N + 1}{4} \quad (4)$$

- a) Si no contiene parte decimal el primer cuartil será

$$x_{\frac{N+1}{4}} \quad (5)$$

- b) Si contiene parte decimal el primer cuartil será

$$x_i + d \times (x_{i+1} - x_i) \quad (6)$$

donde  $i$  es la parte entera y  $d$  la parte decimal

2. Para calcular el segundo cuartil calculamos la mediana.

3. Para calcular el tercer cuartil, calculamos la siguiente expresión

$$\frac{3 \times (N + 1)}{4} \quad (7)$$

- a) Si no contiene parte decimal el primer cuartil será

$$x_{\frac{3 \times (N+1)}{4}} \quad (8)$$

- b) Si contiene parte decimal el primer cuartil será

$$x_i + d \times (x_{i+1} - x_i) \quad (9)$$

donde  $i$  es la parte entera y  $d$  la parte decimal

```
> #Para el calculo de cuartiles, empleamos el algoritmo de Freund and Perles
> #1er cuartil: (n+3)/4
> #3er cuartil: (3*n+1)/4
> quartiles <- function(vector) {
+   #El vector ha de estar ordenado
+   vector <- sort(vector)
+   q1 <- (length(vector) + 3)/4
+   if(q1 %% 1 != 0)
+   {
+     cuartil1 <- vector[trunc(q1)] + (q1 %% 1)*(vector[trunc(q1)+1]
+       - vector[trunc(q1)])
+   }
+   else{
+
+     cuartil1 <- vector[trunc(q1)]
+   }
+ }
```

---

<sup>4</sup><https://www.universoformulas.com/estadistica/descriptiva/cuartiles/>



```

+   }
+
+   q2 <- mediana(vector)
+
+   q3 <- (3*length(vector) + 1)/4
+   if(q3 %% 1 != 0)
+   {
+       cuartil3 <- vector[trunc(q3)] + (q3 %% 1)*(vector[trunc(q3)+1]
+       - vector[trunc(q3)])
+   }
+   else{
+
+       cuartil3 <- vector[trunc(q3)]
+   }
+   print(c(cuartil1,q2,cuartil3))
+ }

```

Para calcular el percentil 54, utilizaremos la función *quantile*:

```

> #satelites.txt
> quartiles(satelites$radio)

[1] 19.00 24.50 30.75

> quantile(satelites$radio, probs=0.54)

54%
26.7

> #satelites.csv
> quartiles(satelites_csv$radio)

[1] 19.00 24.50 30.75

> quantile(satelites_csv$radio, probs=0.54)

54%
26.7

> #cardata.sav
> quartiles(cardata$mpg)

[1] 22.550 28.900 34.275

> #iris.xlsx
> quartiles(iris$PetalLengthCm)

[1] 1.60 4.35 5.10

> quantile(iris$PetalLengthCm, probs=0.54)

54%
4.5

```

```

> #datos_contaminacion.json
> quartiles(as.numeric(datos_contaminacion$concentracion))

[1] 0.8 2.1 16.0

> #Eliminamos filas que contengan cadenas vacias
> datos_contaminacion <- datos_contaminacion[datos_contaminacion$concentracion != ""
+ & datos_contaminacion$porcentaje != "",]
> quantile(as.numeric(datos_contaminacion$concentracion), probs=0.54)

54%
3

```