

Ciencia de datos, práctica 1

Juan Casado Ballesteros

October 13, 2019

Abstract

En esta práctica vamos a realizar tres análisis estadísticos, en los dos primeros utilizaremos las funciones propias de R sobre los datos proporcionados por el profesor en los formatos .txt para el primero y .sav para el segundo. En el tercer análisis hemos programado nosotros mismos nuestras propias funciones. Como datos hemos elegido un .csv que contiene información sobre el alquiler en la Ciudad de Nueva York. Intentaremos para este tercer análisis realizar un estudio crítico que nos permita llegar a conocer los datos con los que estamos trabajando.

Contents

1	Primer análisis satelites menores de urano.txt	3
2	Segundo análisis cardata .sav	7
3	Tercer análisis del alquiler en Nueva York con AirBNB durante 2019 .csv	9
4	Guia en R para el análisis estadístico	17
4.1	Frecuencias	17
4.2	Medidas representativas	17
4.3	Medidas de ordenación	17
5	Funciones creadas	18
5.1	Frecuencias	18
5.1.1	Frecuencia Absoluta	18
5.1.2	Frecuencia Absoluta Acumulada	18
5.1.3	Frecuencia Relativa	19
5.1.4	Frecuencia Relativa Acumulada	20
5.2	Medidas representativas	20
5.2.1	Media Aritmetica	20
5.2.2	Media Geométrica	21
5.2.3	Media Armónica	21
5.2.4	Desviacion Típica	21
5.2.5	Desviacion Media	21
5.2.6	Varianza	22
5.2.7	Tchebychev	22
5.3	Medidas de ordenación	23

5.3.1	Mediana	23
5.3.2	Cuartiles	23
5.3.3	Cuantil54	24

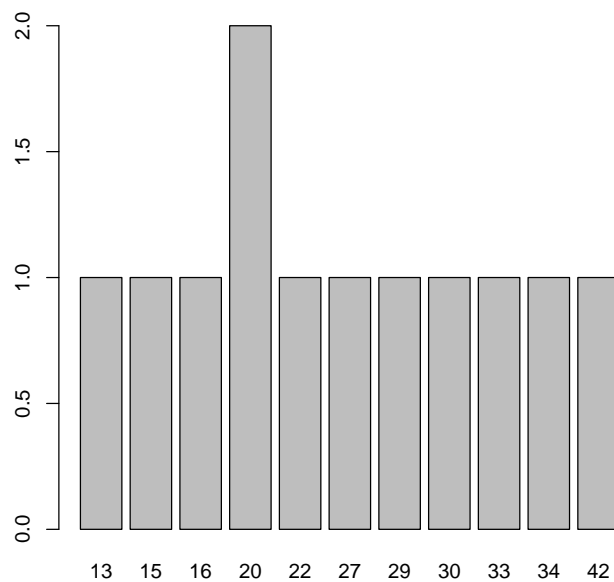
1 Primer análisis satelites menores de urano.txt

Comenzamos leyendo los datos del archivo .txt ya que dicho archivo lo hemos escrito con la sintaxis que `textbfread.table` espera por lo que no tendremos que utilizar ningún parámetro adicional para configurar la lectura de los datos.

```
> source("init.R")  
> satelites <- read.table("./satelites.txt")
```

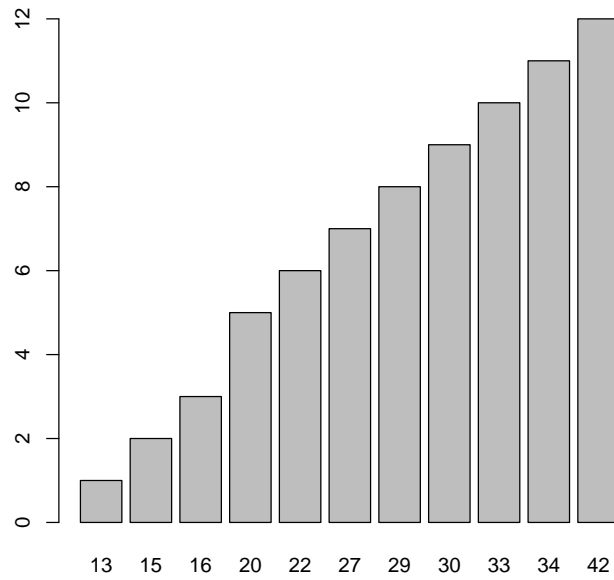
Primero calcularemos la frecuencia absoluta de los datos, que es el número de apariciones de cada uno de ellos.

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



Ahora calcularemos la frecuencia absoluta acumulada.

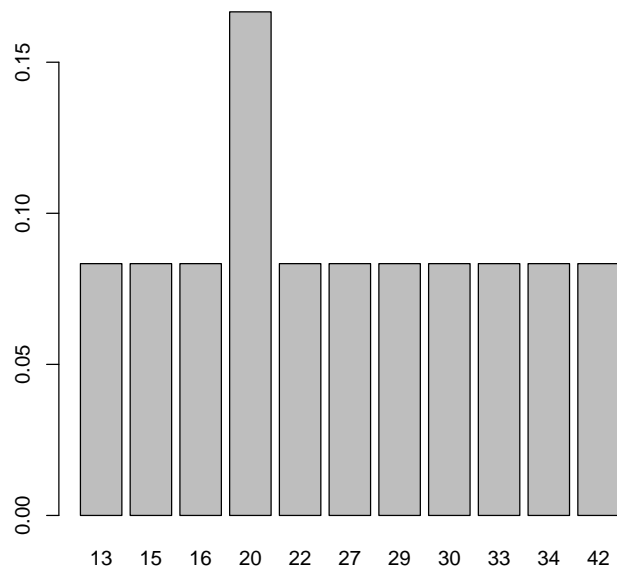
```
> frecuencia_absoluta_acumulada<-cumsum(frecuencia_absoluta)
```



Ahora calcularemos la frecuencia relativa.

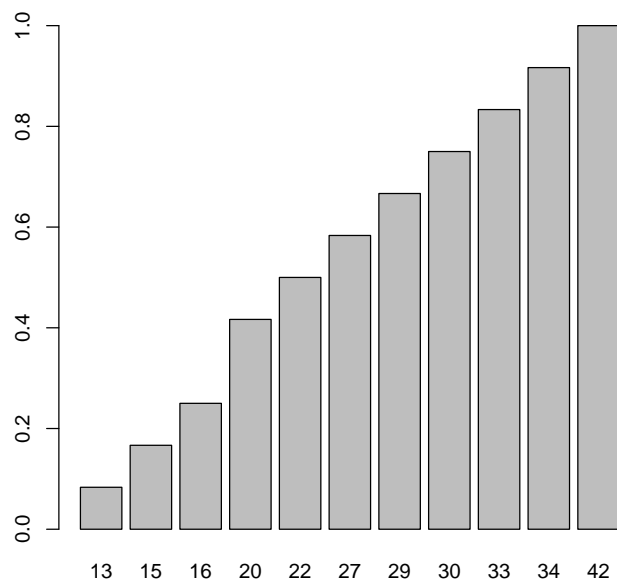
```
> frecuencia_relativa <- (function(data) table(data)/length(data))(satelites$radio)
> sum(frecuencia_relativa)
```

```
[1] 1
```



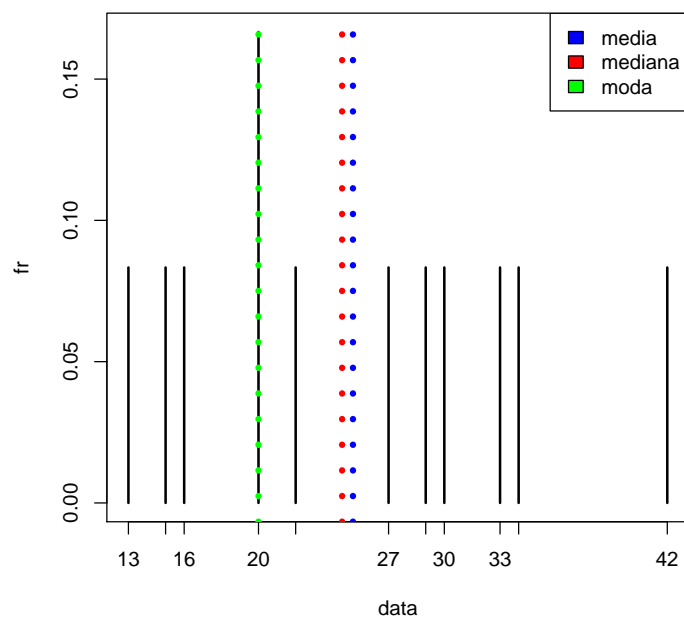
Ahora calcularemos la frecuencia relativa acumulada.

```
> frecuencia_relativa_acumulada <- cumsum(frecuencia_relativa)
```



Calcularemos a continuación estadísticos cuya función es resumir los datos de los que disponemos. Estos estadísticos son la media, la moda y la mediana.

```
> media <- mean(satelites$radio)
> v_mediana <- median(satelites$radio)
> v_moda <- moda(satelites$radio)
> desviacion_tipica <- sd(satelites$radio)
> v_varianza <- var(satelites$radio)
```



```
> desviacion_tipica
```

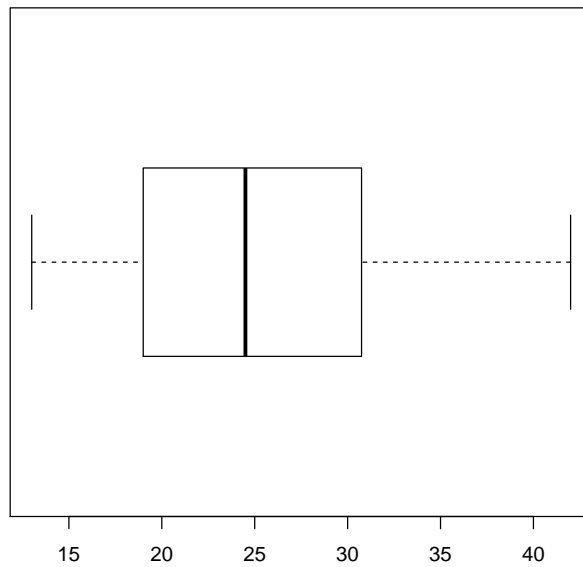
```
[1] 8.857029
```

```
> v_varianza
```

```
[1] 78.44697
```

Cuantiles y cuartiles ...

```
> v_cuartiles <- quantile(satelites$radio, prob=c(0, .25, .5, .75, 1))
> v_cuantil54 <- quantile(satelites$radio, prob=(.54))
```



```
> v_cuartiles
      0%   25%   50%   75%  100%
13.00 19.00 24.50 30.75 42.00

> v_cuantil54
      54%
26.7
```

2 Segundo análisis cardata .sav

```
> source("init.R")
> cardata <- read.spss("./cardata.sav")$mpg
> cardata <- cardata[!is.na(cardata)]

> mean(cardata)

[1] 28.79351

> median(cardata)

[1] 28.9

> moda(cardata)

[1] 36
```

```

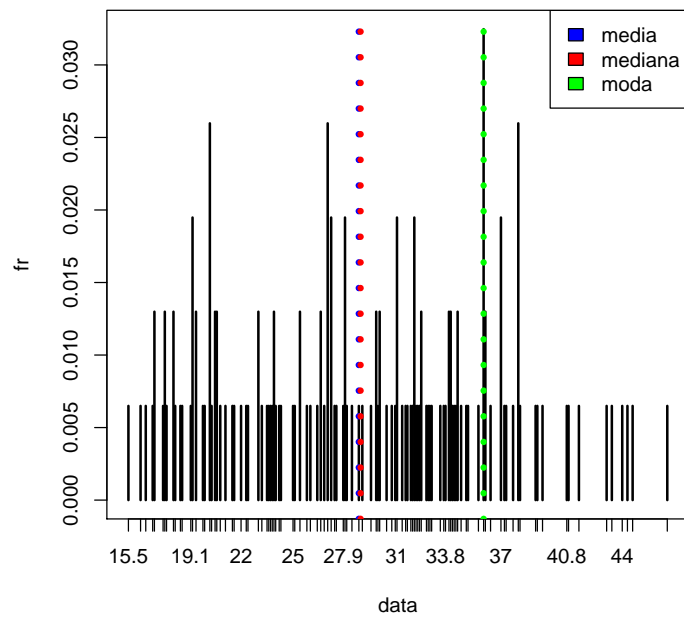
> sd(cardata)

[1] 7.37721

> var(cardata)

[1] 54.42323

```

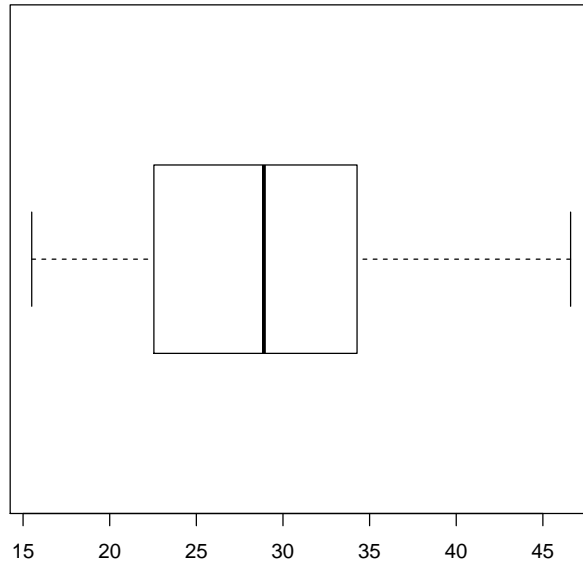


```

> v_cuartiles <- quantile(cardata, prob=c(0, .25, .5, .75, 1))
> v_cuartiles

      0%      25%      50%      75%     100%
15.500 22.550 28.900 34.275 46.600

```

3 Tercer análisis del alquiler en Nueva York con AirBNB durante 2019 .csv

Haremos ahora un análisis de los datos del alquiler en la ciudad de Nueva York durante el año 2019 con la compañía AirBNB. Los datos contienen las siguientes categorías:

```
> getInfo(data)
```

	unlist.res.
id	integer
name	factor
host_id	integer
host_name	factor
neighbourhood_group	factor
neighbourhood	factor
latitude	numeric
longitude	numeric
room_type	factor
price	integer
minimum_nights	integer
number_of_reviews	integer
last_review	factor
reviews_per_month	numeric
calculated_host_listings_count	integer

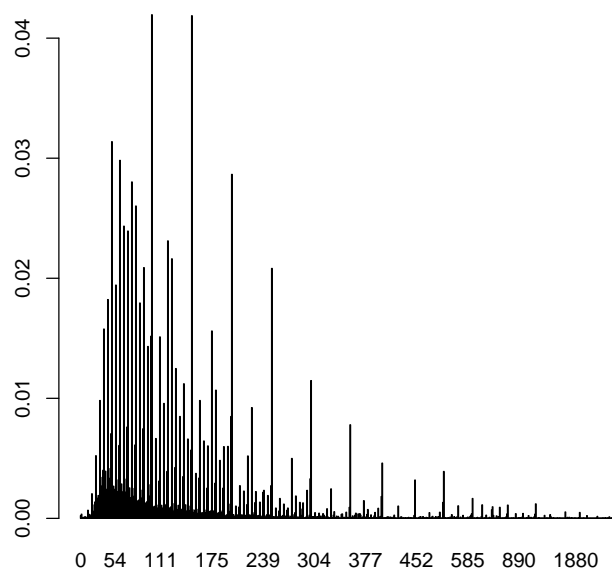
```

availability_365          integer
res_frame
  factor integer numeric
        6       7       3

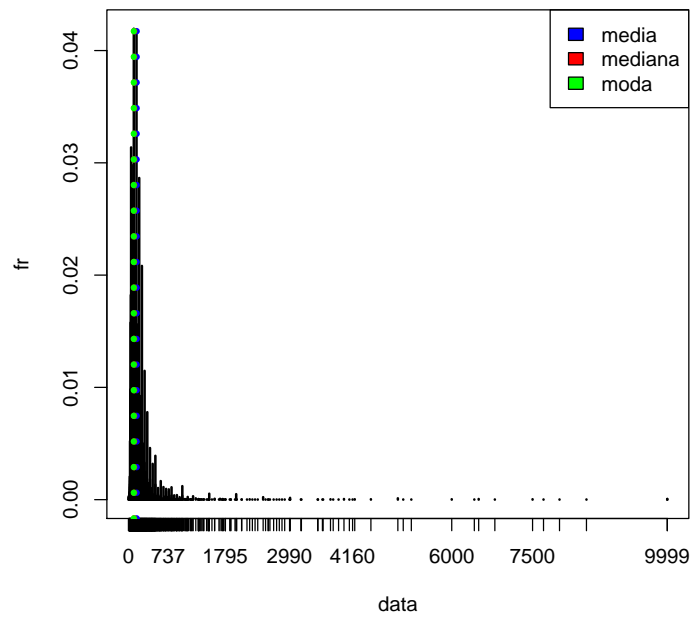
```

Comenzaremos por analizar el precio de los alquileres. Para ello calcularemos la frecuencia relativa de los precios.

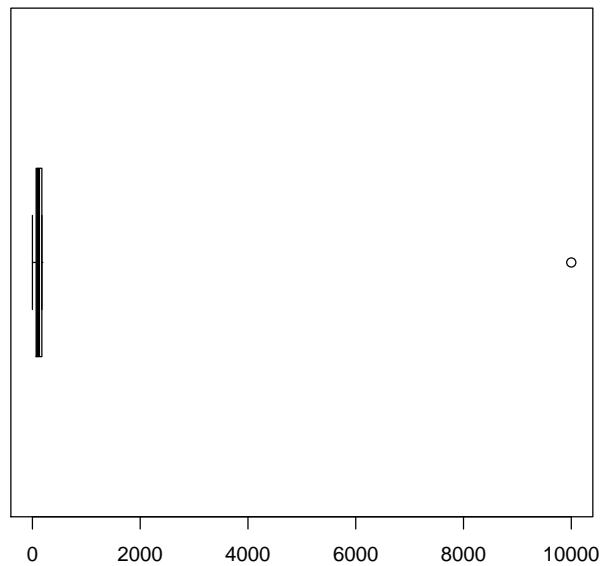
```
> frecuencia_relativa<-frecuenciaRelativa(data$price)
```



Podemos observar que los precios están muy agrupados en la parte izquierda de la gráfica (precios bajos) con altos porcentajes de aparición, no obstante aunque con una densidad mucho menor estos se alejan en gran medida llegando a alcanzar precios muy elevados pero con pocos porcentajes de aparición.



```
> v_cuartiles <- cuartiles(data$price)
```



Vemos que la media no es un valor representativo, su desviación típica es

muy alta. Podemos observar como la moda y la mediana en este caso parecen representar mejor a la mayoría de los datos. Debido a esto decidimos que hacer el análisis de los precios para toda la ciudad no iba a aportarnos una visión representativa de los datos. Decidimos ahora calcular el precio medio por cada barrio en vez de hacerlo sobre toda la ciudad a la vez.

	Group.1	x.media	x.desviacion_tipica	x.mediana	x.moda
1	Bronx	87.49679	106.66043	65.00000	60.00000
2	Brooklyn	124.38321	186.86889	90.00000	100.00000
3	Manhattan	196.87581	291.37646	150.00000	150.00000
4	Queens	99.51765	167.08741	75.00000	50.00000
5	Staten Island	114.81233	277.24801	75.00000	75.00000

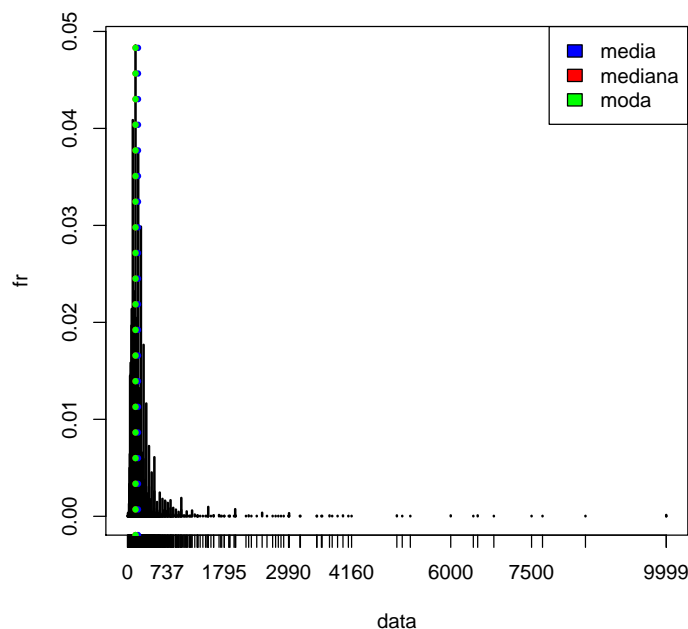
Valores sobre el total de los datos sin haber separado por barrios

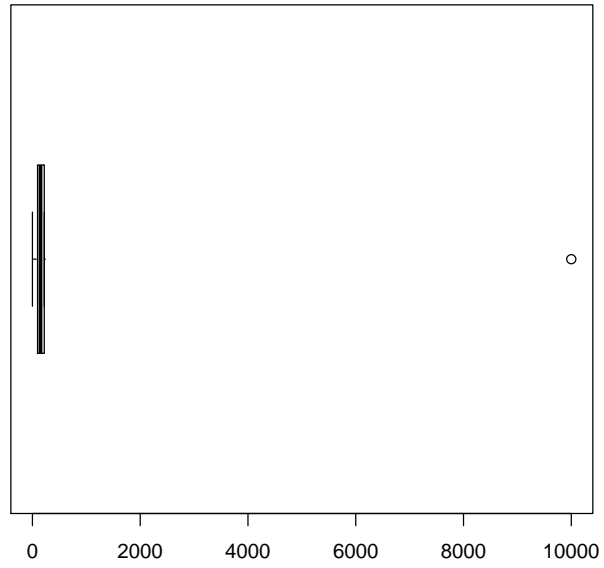
media	desviacion_tipica	v_mediana	v_moda
152.7207	240.1517	106.0000	100.0000

Podemos ver que la varianza se ha reducido para más barrios de en los que ha aumentado. En Bronx es en el barrio en el que tanto la varianza como la media es menor y Manhattan es en el que ambas son mayores.

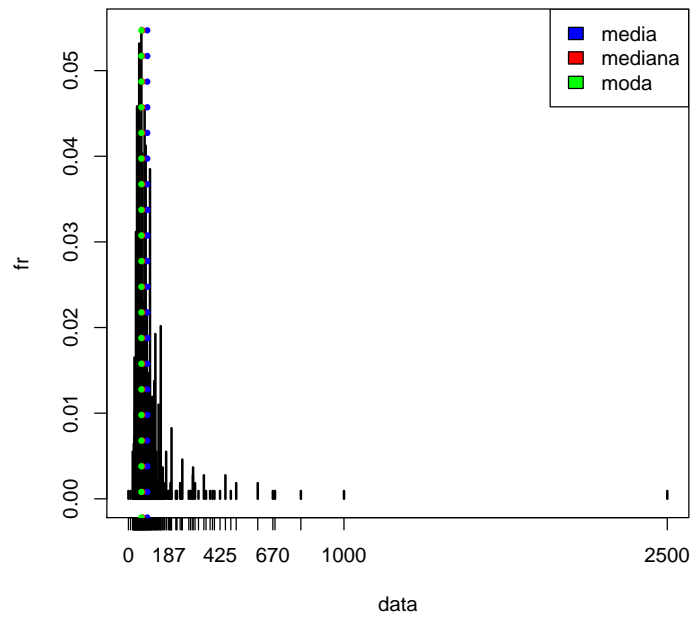
Datos para Manhattan

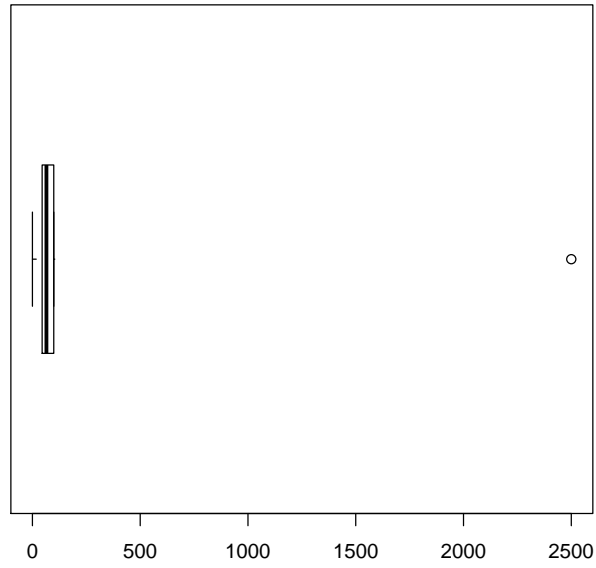
```
> manhattan_data = data$price[data$neighbourhood_group == "Manhattan"]
> plotFrequencyData(manhattan_data)
```



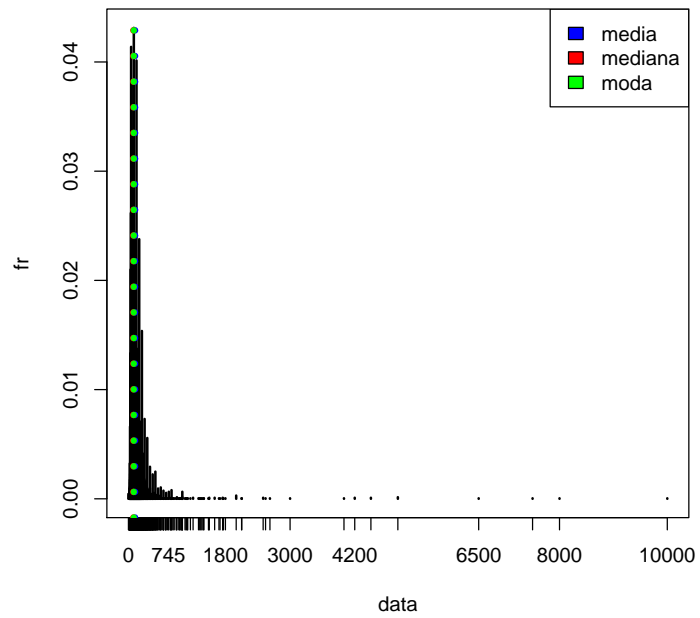


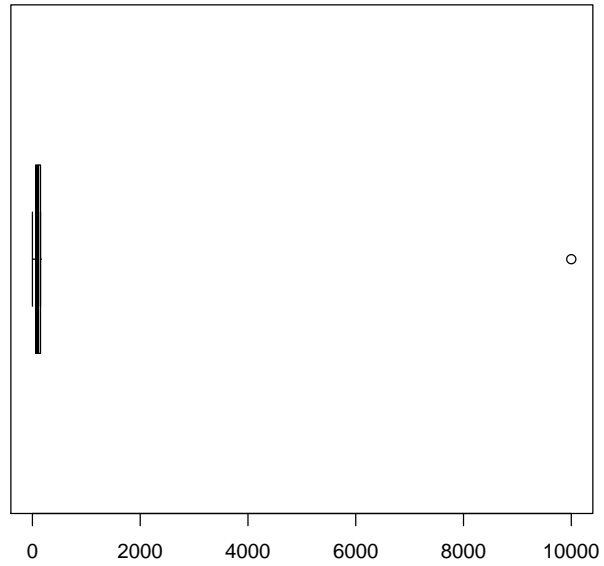
Datos para Bronx



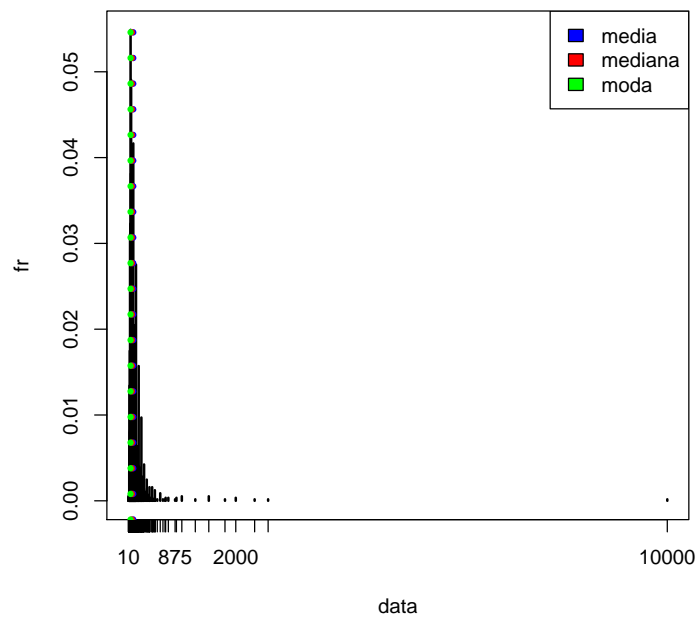


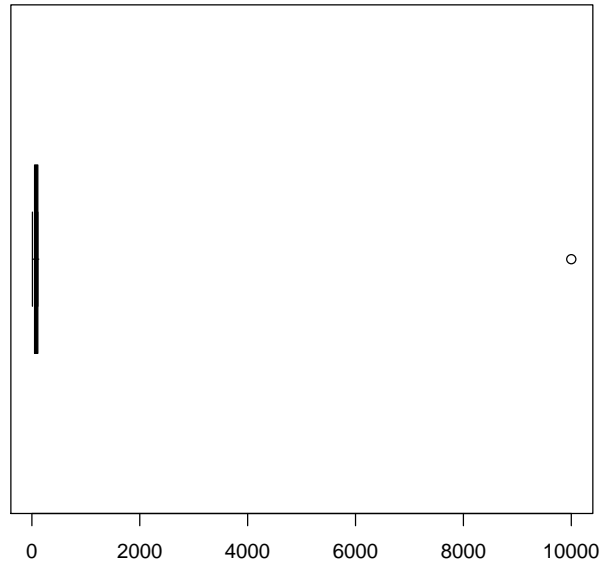
Datos para Brooklyn



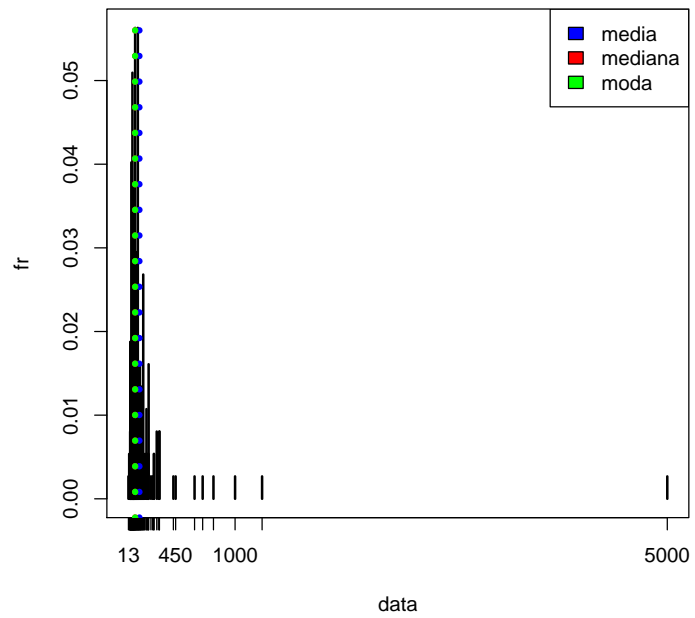


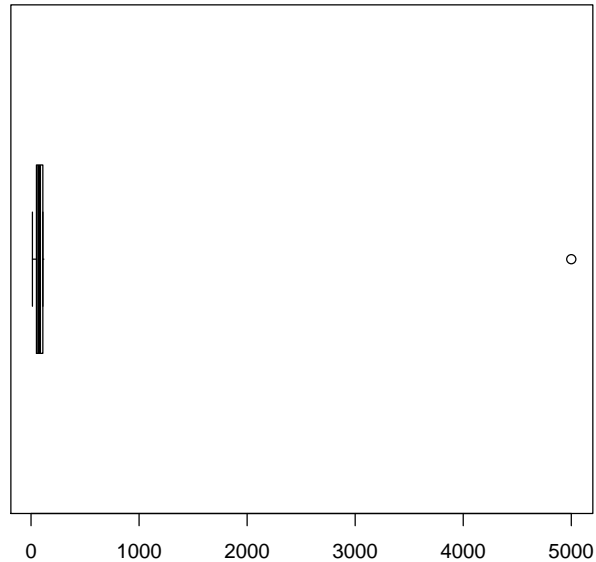
Datos para Queens





Datos para Staten Island





4 Guia en R para el análisis estadístico

4.1 Frecuencias

```
> Frecuencia_absoluta <- table(data)
> Frecuencia_absoluta_acumulada <- cumsum(frecuencia_absoluta)
> Frecuencia_relativa <- (function(data) table(data)/length(data))(data)
> Frecuencia_relativa_acumulada <- cumsum(frecuencia_relativa)
```

4.2 Medidas representativas

```
> Media <- mean(data)
> Desviacion_tipica <- sd(data)
> Varianza <- var(data)
```

4.3 Medidas de ordenación

```
> Mediana <- median(data)
> Cuartiles <- quantile(data, prob=c(0, .25, .5, .75, 1))
> Cuantil54 <- quantile(data, prob=(.54))
```

5 Funciones creadas

5.1 Frecuencias

5.1.1 Frecuencia Absoluta

```
> frecuenciaAbsoluta
```

```
function(data){
  # creamos un vector donde solo estas los elementos sin repetir
  uniquedata<-unique(data)
  # ordenamos el vector para aparezcno correcto
  uniquedata<- sort(uniquedata)
  # creamos un vector vacio numerico
  newdata<- vector(mode="numeric", length=0)
  # recorremos el vector elementos no repetidos
  for (value in uniquedata) {
    # calculamos la cantidad de veces que se repite un numero
    total<-length(data[data==value])
    # añadimos el total al vector con veces que se repitio
    newdata<-c(newdata, total)
  }
  # hacemos una matriz de tamaño 1xlongitud datos, añadimos los elementos la frecuencia
  matrix<-matrix(nrow=1,ncol=length(uniquedata), newdata, byrow=T)
  # en las filas ponemos el nombre que representa
  rownames(matrix)<-c("frecuencia")
  #en las columnas ponemos los datos
  colnames(matrix)<-c(uniquedata)
  #convertimos la matriz en una tabla
  matrix<-as.table(matrix)
}
<bytecode: 0x7fdf56451dc8>
```

5.1.2 Frecuencia Absoluta Acumulada

```
> frecuenciaAbsolutaAcumulada
```

```
function(data){

  # calculamos la frecuencia
  frecuencia <-frecuenciaAbsoluta(data)
  # recuperamos el nombre de las columnas
  uniquedata<-colnames(frecuencia)
  # cogemos el vector de frecuencia
  onlydata<-frecuencia[1,]
  # creamos un vector vacio numerico
  newdata<- vector(mode="numeric", length=0)
  # recorremos desde la posicion 1 hasta lo maximo el vector
  for (value in 1:length(onlydata)) {
    # si el valor primero
    if(value==1){
```

```

        # el acumulado es el mismo valor
        accumulated<-onlydata[value]
    # en otros casos
    }else{
        # el valor es anterior mas el valor suyo
        accumulated<-onlydata[value]+accumulated
    }
    # añadimos al vector el nuevo valor
    newdata<-c(newdata, accumulated)
}
# hacemos una matriz de tamaño 1xlongitud datos, añadimos los elementos la frecuencia
matrix<-matrix(nrow=1,ncol=length(uniquedata), newdata, byrow=T)
# en las filas ponemos el nombre que representa
rownames(matrix)<-c("frecuencia acumulada")
#en las columnas ponemos los datos
colnames(matrix)<-c(uniquedata)
#convertimos la matriz en una tabla
matrix<-as.table(matrix)
}

```

5.1.3 Frecuencia Relativa

> frecuenciaRelativa

```

function(data){
    #conseguimos frecuencia
    frecuencia <-frecuenciaAbsoluta(data)
    # recuperamos el nombre de las columnas
    uniquedata<-colnames(frecuencia)
    #cogemos solo el vector de frecuencia
    onlydata<-frecuencia[1,]
    #calculamos la suma total de frecuencia
    total<-sum(onlydata)
    #creo vector vacio numerico
    newdata<- vector(mode="numeric", length=0)
    #recorro todo el vector de datos
    for (value in 1:length(onlydata)) {
        # realizamos la division entre los datos y el total
        accumulated<-onlydata[value]/total
        # lo añadido al nuevo array
        newdata<-c(newdata, accumulated)
    }
    # hacemos una matriz de tamaño 1xlongitud datos, añadimos los elementos la frecuencia
    matrix<-matrix(nrow=1,ncol=length(uniquedata), newdata, byrow=T)
    # en las filas ponemos el nombre que representa
    rownames(matrix)<-c("frecuencia relativa")
    #en las columnas ponemos los datos
    colnames(matrix)<-c(uniquedata)
    #convertimos la matriz en una tabla
    matrix<-as.table(matrix)
}

```

```
}
<bytecode: 0x7fdf55ac3998>
```

5.1.4 Frecuencia Relativa Acumulada

```
> frecuenciaRelativaAcumulada
```

```
function(data){
  # calcumos la frecuencia acumulada
  frecuencia <-frecuenciaAcumulada(data)
  # recuperamos el nombre de las columnas
  unquedata<-colnames(frecuencia)
  # cogemos el vector de frecuencia acumulada
  onlydata<-frecuencia[1,]
  # calculamos el total de datos
  total<-onlydata[length(onlydata)]
  # creamos un vector vacio numerico
  newdata<- vector(mode="numeric", length=0)
  # recorremos desde la posicion 1 hasta lo maximo el vector
  for (value in 1:length(onlydata)) {
    #variable que calcula relativa entre la esa posicion entre total
    relative<-onlydata[value]/total
    # añadimos al vector el nuevo valor
    newdata<-c(newdata, relative)
  }
  # hacemos una matriz de tamaño 1xlongitud datos, añadimos los elementos la frecuencia
  matrix<-matrix(nrow=1,ncol=length(unquedata), newdata, byrow=T)
  # en las filas ponemos el nombre que representa
  rownames(matrix)<-c("frecuencia acumulada relativa")
  #en las columnas ponemos los datos
  colnames(matrix)<-c(unquedata)
  #convertimos la matriz en una tabla
  matrix<-as.table(matrix)
}
```

5.2 Medidas representativas

Los siguientes valores pretenden ser representates de todos los datos que se estén analizando. No tenemos garantías de que realmente lo sean así que para ellos necesitamos de erramientas que nos lo confiermen o desmientan.

5.2.1 Media Aritmetica

Suma ponderada por la cantidad de valores sumados. Se podría haber utilizado la función sum que suma todos los valores de un vector en vez de haber iterado sobre ellos.

```
> mediaAritmetica
```

```
function(data){
  acc <- 0
```

```

    for (value in data) {
      acc <- acc + value
    }
    acc / length(data)
  }
<bytecode: 0x7fdf59920b98>

```

5.2.2 Media Geométrica

Raíz de orden igual a la cantidad de valores de la multiplicación de todos ellos. El logaritmo de la media geométrica también puede expresarse como la media aritmética de los logaritmos de cada dato.

```

> mediaGeometrica

function(data){
  prod(data)^(1/length(data))
}

```

5.2.3 Media Armónica

Inverso de la media de los inversos.

```

> mediaArmonica

function(data){
  1/mediaAritmetica(1/data)
}

```

5.2.4 Desviacion Típica

Nos indica como de buena es la media aritmética. Cuanto menor sea la desviación mejor será la media. Raíz cuadrada de la varianza.

```

> desviacionTipica

function (data) {
  varianza(data)^(1/2)
}
<bytecode: 0x7fdf56781c10>

```

5.2.5 Desviacion Media

Nos indica como de buena es la media aritmética. Cuanto menor sea la desviación mejor será la media. Se suele utilizar la desviacion típica en vez de la desviacion media. Suma ponderada por la cantidad de valores del valor absoluto de los errores. Se define error como la resta entre un valor y la media aritmética de todos ellos.

```

> desviacionMedia

```

```
function (data){
  v_media <- media(data)
  acc = 0
  for (value in data){
    acc <- acc + abs(value - v_media)
  }
  acc/length(data)
}
```

5.2.6 Varianza

Nos indica como de buena es la media aritmética. Cuanto menor sea la varianza mejor será la media. Suma ponderada por la cantidad de valores del cuadrado de los errores. Se define error como la resta entre un valor y la media aritmética de todos ellos.

```
> varianza

function(data){
  v_media <- mediaAritmetica(data)
  acc = 0
  for (value in data){
    acc <- acc + (value - v_media)^2
  }
  acc/length(data)
}
<bytecode: 0x7fdf54650560>
```

5.2.7 Tchebychev

Decir que la desviación típica es mejor cuanto menor sea no es demasiado precios. El teorema de tchebychev nos proporciona una representación muy visual del significado de la media. Se puede entender la media como un valor entorno al que se ubican los datos y la desviación típica como una circunferencia con centro en la media y radio el valor de la desviación multiplicado por un factor K. Tchebychev nos proporciona el porcentaje mínimo de valores que estarán dentro de cada valor de radio. Esto nos ayuda a comparar la desviación obtenida con los datos que tratamos de una forma más significativa.

```
> tchebychev

function(data, limit=10){
  desviacion_tipica = desviacionTipica(data)
  for (k in 2:limit){
    data_percentage <- (1-(1/k^2))*100
    radius = desviacion_tipica*k
    message(paste("En un radio de",radius,"se encuentran el",data_percentage,"de los dato
  })
}
```

5.3 Medidas de ordenación

5.3.1 Mediana

La mediana la hemos calculado ordenando los datos en orden ascendente y obteniendo el punto que se encuentra en la mitad de ese vector o la media entre los 2 puntos que se encuentran en este lugar.

```
> mediana

function(data){
  x = sort(data)
  size = length(x)
  if (size %% 2 == 0){
    (x[size / 2] + x[(size / 2) + 1]) / 2.0
  }
  else {
    x[size / 2]
  }
}
<bytecode: 0x7fdf55c62168>
```

5.3.2 Cuartiles

Los cuartiles los hemos calculado ordenando los datos en orden ascendente y obteniendo los 3 puntos que dividen el vector en 4 partes iguales, es decir, el cuartil 25, el 50 y el 75.

```
> cuartiles

function(data){
  x = sort(data)
  size = length(x)
  if (size %% 2 == 0){
    c(
      x[1],
      (x[size / 4] + x[(size / 4) + 1]) / 2.0 ,
      (x[size / 2] + x[(size / 2) + 1]) / 2.0 ,
      (x[size * 0.75] + x[size * 0.75 + 1]) / 2.0,
      x[size]
    )
  }
  else {
    c(
      x[1],
      x[size / 4],
      x[size / 2],
      (x[size * 0.75] + x[size * 0.75 + 1]) / 2.0,
      x[size]
    )
  }
}
```

```
}  
<bytecode: 0x7fdf54c2bbc0>
```

5.3.3 Cuantil54

El cuantil 54 los hemos calculado ordenando los datos en orden ascendente y obteniendo el punto en el que detrás de él se encuentran el 54

```
> cuantil54  
  
function(x){  
  size = length(x)  
  x[ceiling(size*0.54)]  
}
```