Introducción al análisis de datos

Carlos Daniel Jiménez

September 5, 2018

Por qué trabajar con R?

R es el lenguaje más popular del mundo para el trabajo de análisis de datos, es un open - source y cuenta con soporte por microsoft

- R es libre;
- Es el software más usado para BI y marketing Analytics
- Tiene una visualización de datos impresionante

Para que R funciones plenamente debe hacerse el llamado de algunas librerias utiles , por ejemplo ggplot para hacer gráficas de manera espectacular y condicional, la forma de invocar una de estas es:

```
library(ggplot2)
library(dplyr)
```

R clasifica las variables según su tipo, esto es necesario para saber estructurar el análisis de los datos

```
z<-12

class(z)
```

```
## [1] "numeric"
```

```
h<-'Palabra'
class(h)
```

```
## [1] "character"
```

```r
a = TRUE; b = FALSE
a & b # Función y
```

```
## [1] FALSE
```

```r
a | b  #Función o
```

```
## [1] TRUE
```

```r
!a # Negación
```

```
## [1] FALSE
```

Vectores

```
c(1:100)
```

```
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  1
##  [18]  18  19  20  21  22  23  24  25  26  27  28  29  3
##  [35]  35  36  37  38  39  40  41  42  43  44  45  46  4
##  [52]  52  53  54  55  56  57  58  59  60  61  62  63  6
##  [69]  69  70  71  72  73  74  75  76  77  78  79  80  8
##  [86]  86  87  88  89  90  91  92  93  94  95  96  97  9
```

Combinación de vectores

```
a<-c(1:10)
b<-c('Maria', 'Alicia')
c(a,b)
```

```
##  [1] "1"      "2"      "3"      "4"      "5"      "6"
##  [8] "8"      "9"      "10"     "Maria"  "Alicia"
```

Vectores con funcionalidad matemática

```
a<-c(1:20)
b<-2
a/b
```

```
##  [1]  0.5  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0
## [15]  7.5  8.0  8.5  9.0  9.5 10.0
```

```
a*b
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34
```

Reciclaje de los vectores

```
## Warning in a + b: longer object length is not a multiple
## length
```

```
##  [1]  3  6  9  6  9 12  9 12 15 12
```

Extraer un dato

```
## [1] 5
```

Crear matrices (Ojo esto es importante para BBDD)

```
a<-c(1:10)
b<-matrix(a,nrow = 2, ncol = 5)
b
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    3    5    7    9
## [2,]    2    4    6    8   10
```

Colocar nombres a la matriz

```
colnames(b)<-c('a','b','c','d','e')
b
```

```
##      a b c d  e
## [1,] 1 3 5 7  9
## [2,] 2 4 6 8 10
```

Otra forma de de diseñar matrices

```
a<-c(1,2,3)
b<-c(4,5,6)
cbind(a,b)
```

```
##      a b
## [1,] 1 4
## [2,] 2 5
## [3,] 3 6
```

trasnponer una matriz en R

```
t(b)
```

```
##      [,1] [,2] [,3]
## [1,]    4    5    6
```

Construir una base de datos

```r
a<-c(1,2,3)
b<-c('a','b','c')
df<-data.frame(a,b)
df
```

```
##   a b
## 1 1 a
## 2 2 b
## 3 3 c
```

Ejemplo de base de datos en R

```
data(iris)
attach(iris)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Spec
## 1          5.1         3.5          1.4         0.2  set
## 2          4.9         3.0          1.4         0.2  set
## 3          4.7         3.2          1.3         0.2  set
## 4          4.6         3.1          1.5         0.2  set
## 5          5.0         3.6          1.4         0.2  set
## 6          5.4         3.9          1.7         0.4  set
```

Extraer valores de una base de datos

```
iris[,'Species']
```

```
##   [1] setosa     setosa     setosa     setosa     setosa
##   [7] setosa     setosa     setosa     setosa     setosa
##  [13] setosa     setosa     setosa     setosa     setosa
##  [19] setosa     setosa     setosa     setosa     setosa
##  [25] setosa     setosa     setosa     setosa     setosa
##  [31] setosa     setosa     setosa     setosa     setosa
##  [37] setosa     setosa     setosa     setosa     setosa
##  [43] setosa     setosa     setosa     setosa     setosa
##  [49] setosa     setosa     versicolor versicolor versic
##  [55] versicolor versicolor versicolor versicolor versic
##  [61] versicolor versicolor versicolor versicolor versic
##  [67] versicolor versicolor versicolor versicolor versic
##  [73] versicolor versicolor versicolor versicolor versic
##  [79] versicolor versicolor versicolor versicolor versic
##  [85] versicolor versicolor versicolor versicolor versic
##  [91] versicolor versicolor versicolor versicolor versic
```

```r
iris[1:10,]
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Spe
## 1           5.1         3.5          1.4         0.2  se
## 2           4.9         3.0          1.4         0.2  se
## 3           4.7         3.2          1.3         0.2  se
## 4           4.6         3.1          1.5         0.2  se
## 5           5.0         3.6          1.4         0.2  se
## 6           5.4         3.9          1.7         0.4  se
## 7           4.6         3.4          1.4         0.3  se
## 8           5.0         3.4          1.5         0.2  se
## 9           4.4         2.9          1.4         0.2  se
## 10          4.9         3.1          1.5         0.1  se
```

Para importar datos

- readr: libreria para importar datos csv
- readxl: Libreria para importar datos xls
- read_sas: Importar datos desde SAS
- read_stat: Importar datos desde Stata

Funciones de R

R como calculadora

En R se puede calcular directamente, sin necesidad de tener funciones asistidas como en Stata o matlab.

```r
1+12 # Adicción
```

```
## [1] 13
```

```r
100-12 # Resta
```

```
## [1] 88
```

```r
12*12 # Multiplicación
```

```
## [1] 144
```

```r
84/3 # División
```

```
## [1] 28
```

Funciones Lógicas

Las funciones lógicas son aquellas que cumplen unas condiciones especificas, donde los resultados se pueden expresar en true o false

```
3>100
```

```
## [1] FALSE
```

```
100>5
```

```
## [1] TRUE
```

```
3==4
```

```
## [1] FALSE
```

Funciones útiles

- Log ;
- exp;
- ^(Elevar);
- help
- rep()

```r
log(88)
```

```
## [1] 4.477337
```

```r
exp(2)
```

```
## [1] 7.389056
```

```r
help(mean)
rep(2,6)
```

```
## [1] 2 2 2 2 2 2
```

Crear objetos

Para crear objetos hay que declararlos como en cualquier lenguaje de programación, en este caso se puede usar alguna de las siguientes altenrativas:

- <-
- =

Ejemplo de declaraciones

```
a<-1
b<-3
c<-a+b
c
```

```
## [1] 4
```

Otra forma de hacerlo (más sofisticada)

```
c(1:10)->d
d
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

No cambia el resultado , simplemente la estructura de como se llama a las declaraciones.

Funciones a través de objetos

```
4->a
b<-3
a/b->c
c
```

```
## [1] 1.333333
```

Para su inquietud , hay una manera de colocar este resultado en fraccionario, su misión será averiguarlo

```
##  [1] 4 4 4 4 4 4 4 4 4 4 4
```

```
## [1]  0  2  4  6  8 10
```

```r
c(9,7,5,2,-11)->d
sort(d) # Función para organizar
```

```
## [1] -11   2   5   7   9
```

Atributos

Con la función **names** se puede extraer el nombre de las variables

```
names(mtcars)
```

```
##  [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "v
## [11] "carb"
```

```
mtcars$mpg
```

```
##  [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 1
## [15] 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 1
## [29] 15.8 19.7 15.0 21.4
```

Funciones Estádisticas

- mean() : Promedio
- sd() : Desviación Estandar
- var(): Varianza
- median() : Mediana
- quantile(): Percentiles
- summary(): Algunas medidas de centralidad
- cor(): Correlación lineal
- cov(): Covarianza

```
summary(mtcars)
```

```
##       mpg             cyl             disp             h
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.
##  Median :19.20   Median :6.000   Median :196.3   Median
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.
##       drat             wt             qsec             v
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.
##  Median :3.695   Median :3.325   Median :17.71   Median
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.
##        am              gear            carb
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
##  Median :0.0000   Median :4.000   Median :2.000
```

Programación en R

Comando if

Permite desarrollar determinadas acciones siempre y cuando estás cumplan unas condiciones lógicas, si la condición no se cumple no se imprimira nada

```
5->x
if(x>0){
  print('El número es positivo y mayor que ceo')
}
```

```
## [1] "El número es positivo y mayor que ceo"
```

```
-5->x
if(x>0){
  print('El número es positivo y mayor que ceo')
}
```

Ejemplos practicos

Suponga que desea saber en que rrss es más conocido , por lo tanto como buen geek descarga sus visitas desde una API(application programming interface)[1] donde obtiene los siguientes resultados

```
Instagram <- c(16, 19, 23, 15, 2, 7, 4)
facebook <- c(7, 17, 25, 6, 18, 3, 14)
```

[1] Para mayor información visite el siguiente linkn https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones

Usando las operaciones que se dieron en la parte anterior tenemos que determinar que uD es popular si recibe más de 15 views en un día

```
facebook>15
```

```
## [1] FALSE  TRUE  TRUE FALSE  TRUE FALSE FALSE
```

```
Instagram>15
```

```
## [1]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
```

Ahora desea saber en que red social es más popular

```
facebook>Instagram
```

```
## [1] FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE
```

```
Instagram>facebook
```

```
## [1]  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE
```

Para desarrollar mejor el ejercicio se transforman los datos anteriores en matrices

```
views<-matrix(c(facebook, Instagram), ncol = 2, byrow = TRU
colnames(views)<-c('Facebook', 'Instagram')
views
```

```
##      Facebook Instagram
## [1,]        7        17
## [2,]       25         6
## [3,]       18         3
## [4,]       14        16
## [5,]       19        23
## [6,]       15         2
## [7,]        7         4
```

Ahora veamos views menores a 12 y otras mayores a 15

```
views<12
```

```
##       Facebook Instagram
## [1,]      TRUE     FALSE
## [2,]     FALSE      TRUE
## [3,]     FALSE      TRUE
## [4,]     FALSE     FALSE
## [5,]     FALSE     FALSE
## [6,]     FALSE      TRUE
## [7,]      TRUE      TRUE
```

```
views>15
```

```
##       Facebook Instagram
## [1,]     FALSE      TRUE
## [2,]      TRUE     FALSE
## [3,]      TRUE     FALSE
## [4,]     FALSE      TRUE
## [5,]      TRUE      TRUE
## [6,]     FALSE     FALSE
```

Operaciones lógicas desde los vectores construidos

La función tail sirve para visualizar los últimos datos de un dataframe

```
tail(facebook,1)
```

```
## [1] 14
```

```
tail(facebook,1)>8
```

```
## [1] TRUE
```

```
tail(facebook,1)>8|tail(facebook,1)<4
```

```
## [1] TRUE
```

```
tail(facebook,1)>8 & tail(facebook,1)<4
```

```
## [1] FALSE
```

Else

Cuando una función lógica no se cumple, entonces se desarrolla la función else que trabaja como un entonces y secuencia una segunda condición cuando la primera no se cumple

```
x<-1
if(x>1){
  print('El número es mayor que uno')
}else{
  print('El número es menor o igual  a uno')
}
```

```
## [1] "El número es menor o igual  a uno"
```

Ejemplo de condicionales

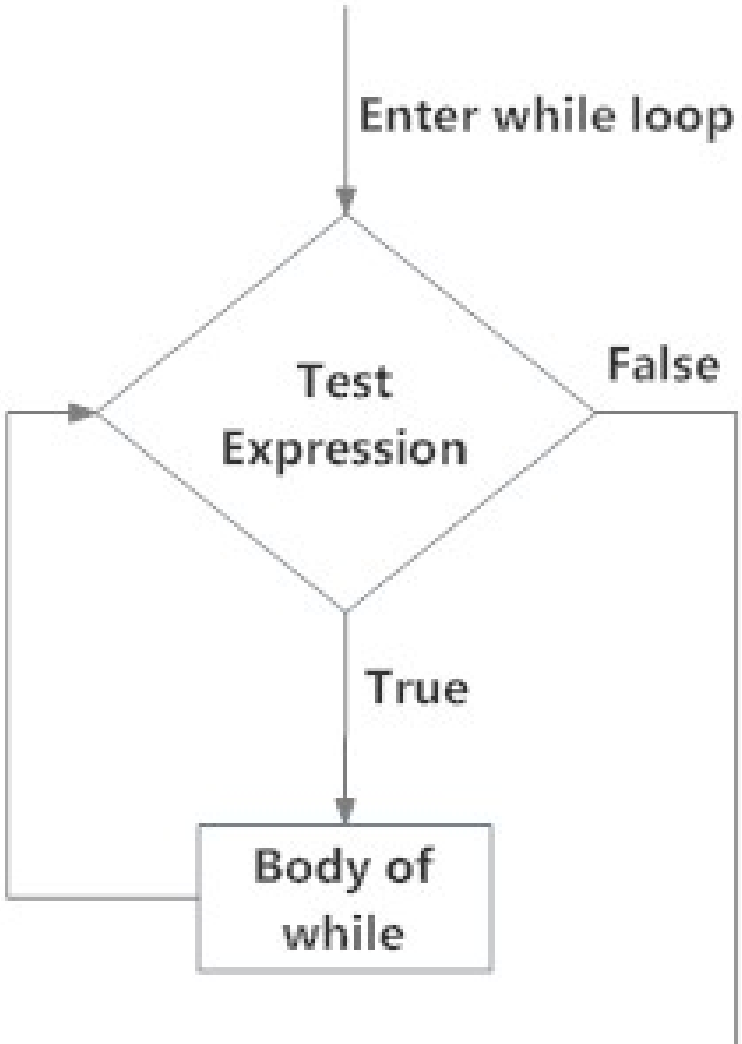
Suponga que desea programar respuestas, reciclando el ejemplo de las redes sociales, se dira que si se tiene una mediana de views de 12, usted es popular, si es por debajo la medida entonces a usted solo lo conocen en la casa, imprima dichas respuestas

```
mediana<-12
if(facebook==mediana){
  print('usted es popular!!!')
} else if (facebook>=mediana){
  print('Usted es muy Popular!!')
} else if(facebook<=mediana){
  print('A usted lo conocen solo en su casa!')
}
```

```
## [1] "A usted lo conocen solo en su casa!"
```

Función While loop

In R programming, while loops are used to loop until a specific condition is met.[2]



```r
i <- 1
while (i < 6) {
print(i)
i = i+1
}
```

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

Un ejemplo de datacamp

```
speed <- 64



while (speed > 30) {
  print(paste("Your speed is",speed))
  if ( speed>48) {
    print("Slow down big time!")
    speed=speed-11
  } else {
    print("Slow down!")
    speed=speed-6
  }
}
```

```
## [1] "Your speed is 64"
## [1] "Slow down big time!"
## [1] "Your speed is 53"
## [1] "Slow down big time!"
## [1] "Your speed is 42"
```

Función for

Es una función que sirve para repetir un número de iteracciones grandes que cumplan una función

```
y <- rep(NA, 100)
y[1] <- 1
for (i in 2:100) {
y[i] <- y[i - 1] + 4
}


y
```

```
##   [1]   1   5   9  13  17  21  25  29  33  37  41  45  4
##  [18]  69  73  77  81  85  89  93  97 101 105 109 113 11
##  [35] 137 141 145 149 153 157 161 165 169 173 177 181 18
##  [52] 205 209 213 217 221 225 229 233 237 241 245 249 25
##  [69] 273 277 281 285 289 293 297 301 305 309 313 317 32
##  [86] 341 345 349 353 357 361 365 369 373 377 381 385 38
```

Introducción al análisis de datos

Datos Cualitativos

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
print(painters)
```

```
##                 Composition Drawing Colour Expression Sc
## Da Udine                 10       8     16          3
## Da Vinci                 15      16      4         14
## Del Piombo                8      13     16          7
## Del Sarto                12      16      9          8
## Fr. Penni                 0      15      8          0
## Guilio Romano            15      16      4         14
## Michelangelo              8      17      4          8
## Perino del Vaga          15      16      7          6
```

Extraer una sola variable y convertirlo en una tabla de frecuencia

Una tabla de frecuencia es el resumen de ocurrencias de una observación

```
painters$School
```

```
##  [1] A A A A A A A A A A B B B B B B C C C C C C D D D D
## [36] E E E E F F F F G G G G G G G H H H H
## Levels: A B C D E F G H
```

```
cbind(table(painters$School))
```

```
##   [,1]
## A   10
## B    6
## C    6
## D   10
## E    7
## F    4
```

Frecuencia relativa

Número de observaciones sobre el total de las mismas

```
cbind(table(painters$School)/dim(painters)[1]*100)
```

```
##        [,1]
## A 18.518519
## B 11.111111
## C 11.111111
## D 18.518519
## E 12.962963
## F  7.407407
## G 12.962963
## H  7.407407
```

Para que se vea de mejor manera

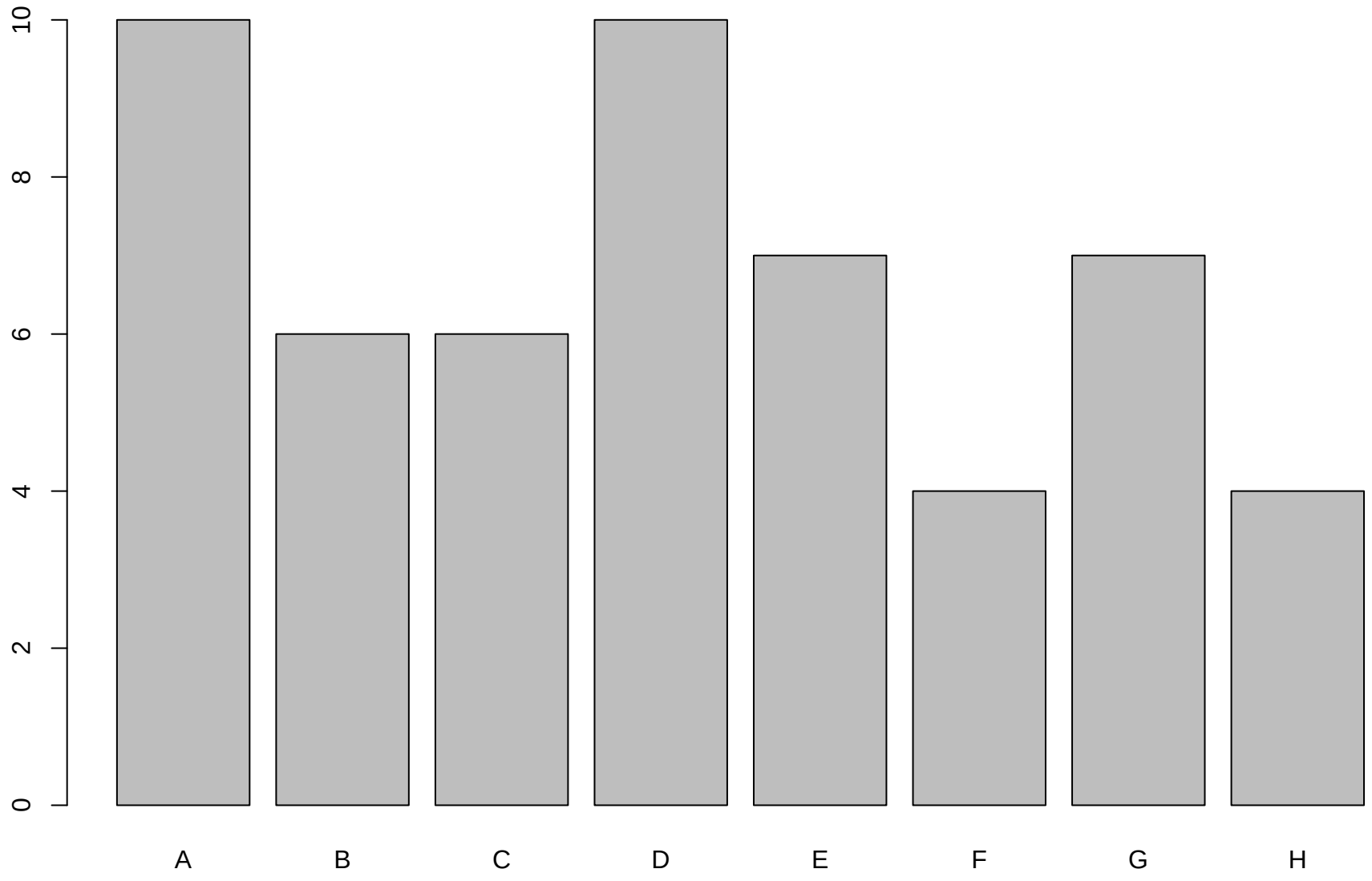
```
round(cbind(table(painters$School)/dim(painters)[1]*100),2)
```

```
##    [,1]
## A 18.52
## B 11.11
## C 11.11
## D 18.52
## E 12.96
## F  7.41
## G 12.96
## H  7.41
```

Gráficar el anterior dato

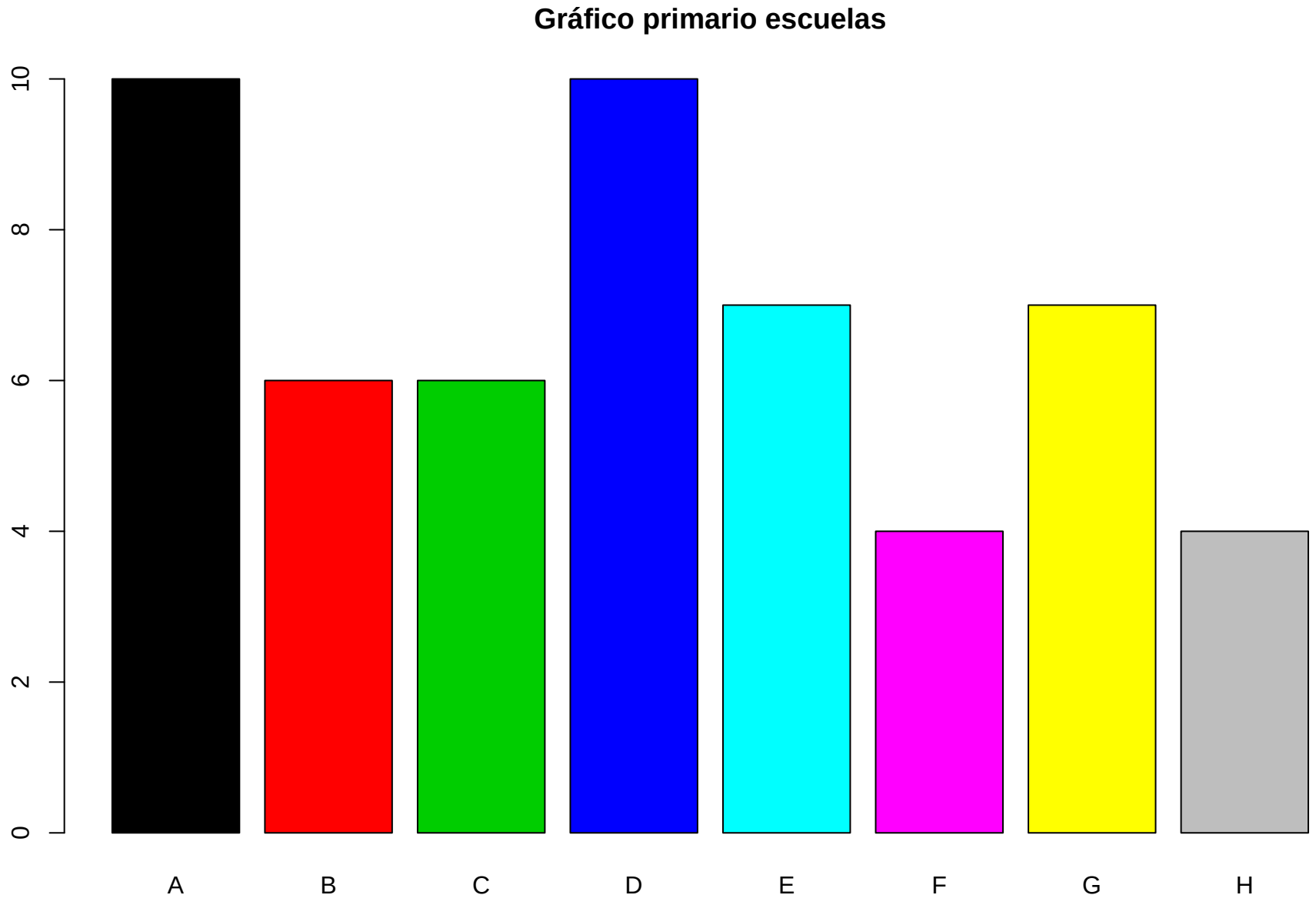
Está es una manera base de gráficar de manera *base*

```
barplot(table(painters$School))
```



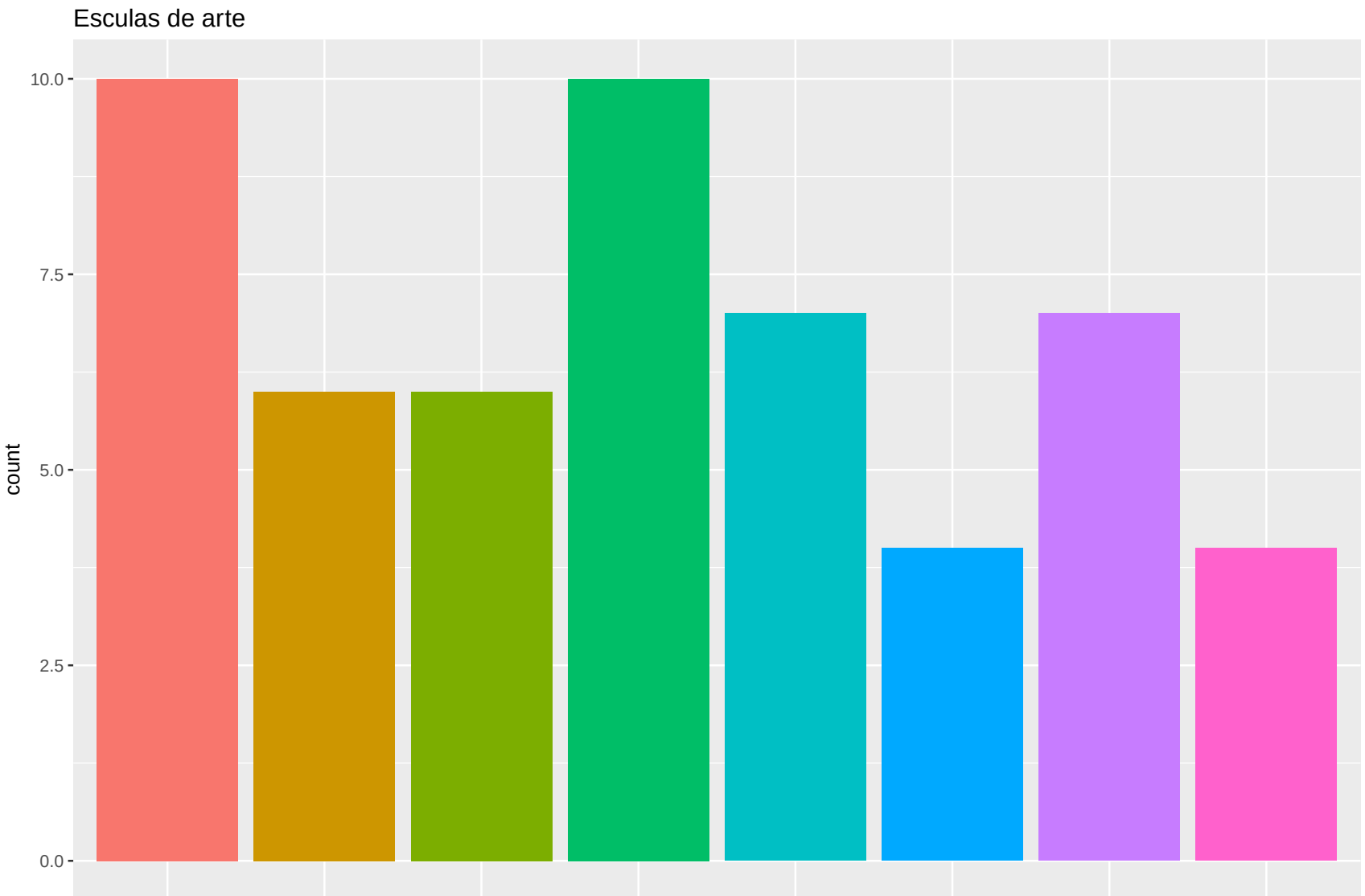
Si desea puede ponerle colores y titulos

```
barplot(table(painters$School), col=1:10)
title('Gráfico primario escuelas')
```



Está es la forma como graficaremos

```
library(ggplot2)
ggplot(painters, aes(painters$School, fill=painters$School)
  geom_bar(aes(y=..count..))+guides(fill=FALSE)+labs(title=
```



Filtrar datos

La presente es una opción basada en la creación de objetos

```
painters$School->ps
ds<-ps=="D"
ds
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FAI
## [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FAI
## [23]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TR
## [34] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FAI
## [45] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FAI
```

La forma de verlo es

```
dp<-painters[ds,]
dp
```

```
##                Composition Drawing Colour Expression Scho
## Bassano                  6       8     17          0
## Bellini                  4       6     14          0
## Giorgione                8       9     18          4
## Murillo                  6       8     15          4
## Palma Giovane           12       9     14          6
## Palma Vecchio            5       6     16          0
## Pordenone                8      14     17          5
## Tintoretto              15      14     16          4
## Titian                  12      15     18          6
## Veronese                15      10     16          3
```

Otra forma de hacerlo es

```
subset(painters, painters$School=="D")
```

```
##                 Composition Drawing Colour Expression Scho
## Bassano                   6       8     17          0
## Bellini                   4       6     14          0
## Giorgione                 8       9     18          4
## Murillo                   6       8     15          4
## Palma Giovane            12       9     14          6
## Palma Vecchio             5       6     16          0
## Pordenone                 8      14     17          5
## Tintoretto               15      14     16          4
## Titian                   12      15     18          6
## Veronese                 15      10     16          3
```

Una forma más elegante y sofisticada de hacerlo es

```
library(dplyr)
painters%>%
  filter(.$School=="D")
```

```
##    Composition Drawing Colour Expression School
## 1            6       8     17          0      D
## 2            4       6     14          0      D
## 3            8       9     18          4      D
## 4            6       8     15          4      D
## 5           12       9     14          6      D
## 6            5       6     16          0      D
## 7            8      14     17          5      D
## 8           15      14     16          4      D
## 9           12      15     18          6      D
## 10          15      10     16          3      D
```

Encuentre la composición de la escuela D

```
mean(dp$Composition)
```

```
## [1] 9.1
```

Una forma masiva de hacerlo, que a la par es eficiente es

```
tapply(painters$Composition,painters$School, mean)
```

```
##         A         B         C         D         E         F
## 10.40000 12.16667 13.16667  9.10000 13.57143  7.25000 13
```

Datos Cuantitativos

```
##   eruptions waiting
## 1     3.600      79
## 2     1.800      54
## 3     3.333      74
## 4     2.283      62
## 5     4.533      85
## 6     2.883      55
```

Frecuencia

```
duración<-faithful$eruptions
range(duración)
```

```
## [1] 1.6 5.1
```

En base a lo anterior se establecen los rangos para crear la frecuencia

```
breaks<-seq(1.5,5.5, by=0.5)
breaks
```

```
## [1] 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5
```

Se le pega a la tabla anterior la variable creada con la cunción cut

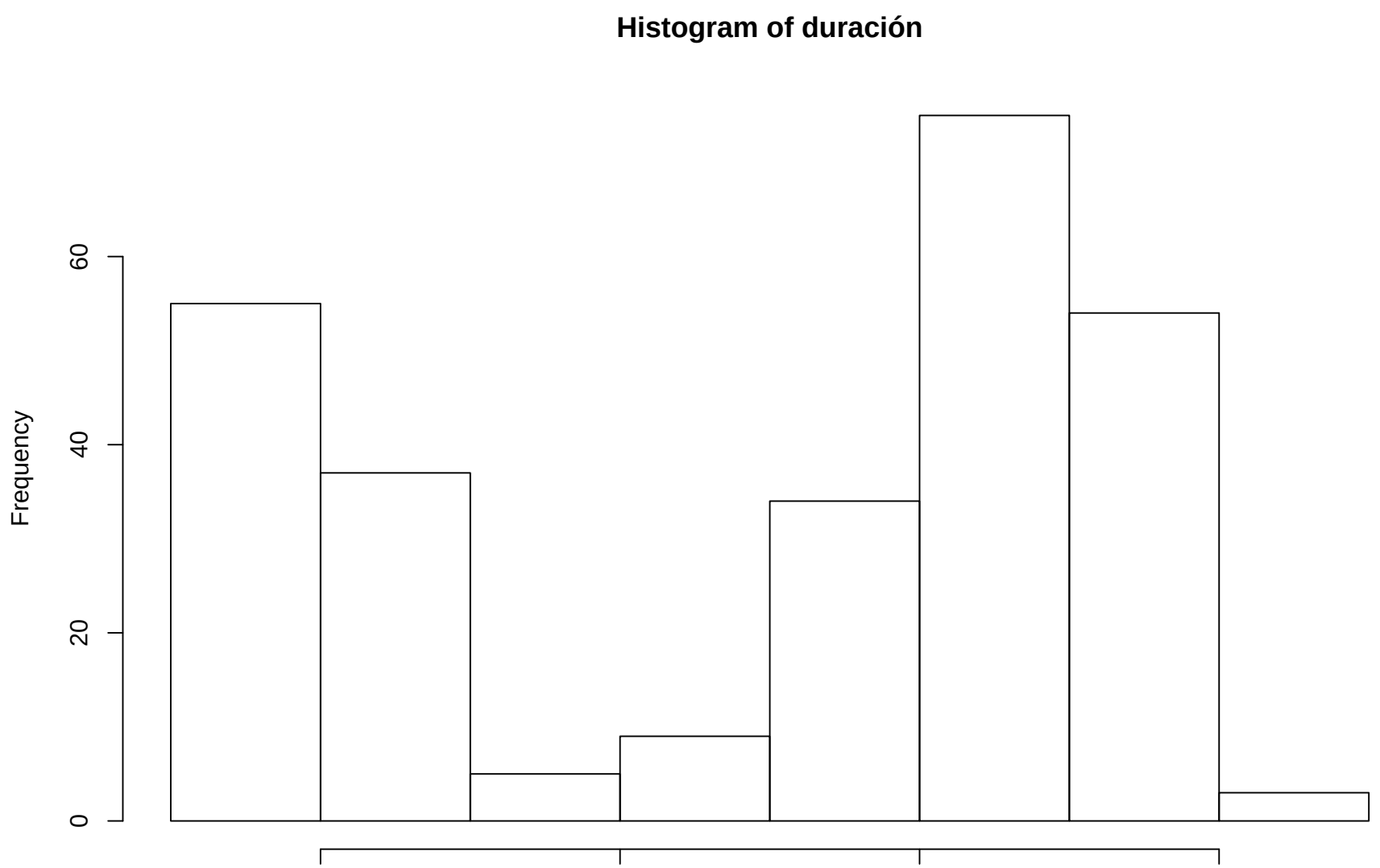
```
duración_freq<-cut(duración, breaks)
cbind(table(duración_freq))
```

```
##           [,1]
## (1.5,2]     55
## (2,2.5]     37
## (2.5,3]      5
## (3,3.5]      9
## (3.5,4]     34
## (4,4.5]     75
## (4.5,5]     54
## (5,5.5]      3
```

Histograma

Cuando se tiene una sola variable se usa histograma para ver la frecuencia de los datos

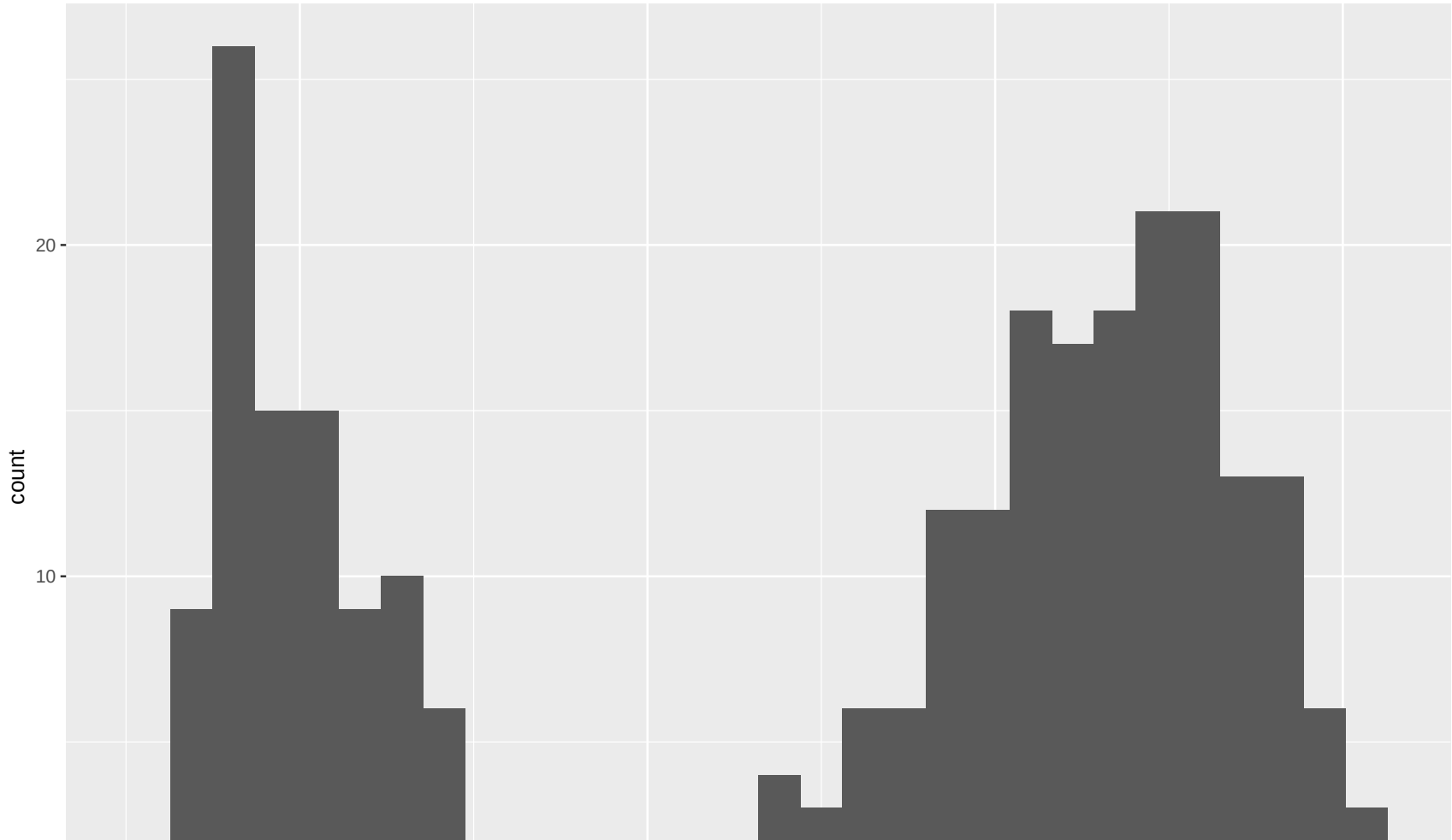
```
hist(duración)
```



Recuerde que lo vamos a trabajar de la siguiente manera

```
faithful%>%
  ggplot(aes(.$eruptions))+
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `
```



Frecuencia relativa

```
table(duración_freq)/nrow(faithful)
```

```
## duración_freq
##    (1.5,2]    (2,2.5]    (2.5,3]    (3,3.5]    (3.5,4]
## 0.20220588 0.13602941 0.01838235 0.03308824 0.12500000
##    (4.5,5]    (5,5.5]
## 0.19852941 0.01102941
```

Distribución de la frecuencia acumulativa

Resumen la frecuencia de los datos , bajo un nivel dado

```
cumsum(table(duración_freq))
```

```
## (1.5,2] (2,2.5] (2.5,3] (3,3.5] (3.5,4] (4,4.5] (4.5,5]
##      55      92      97     106     140     215     269
```

Gráfica

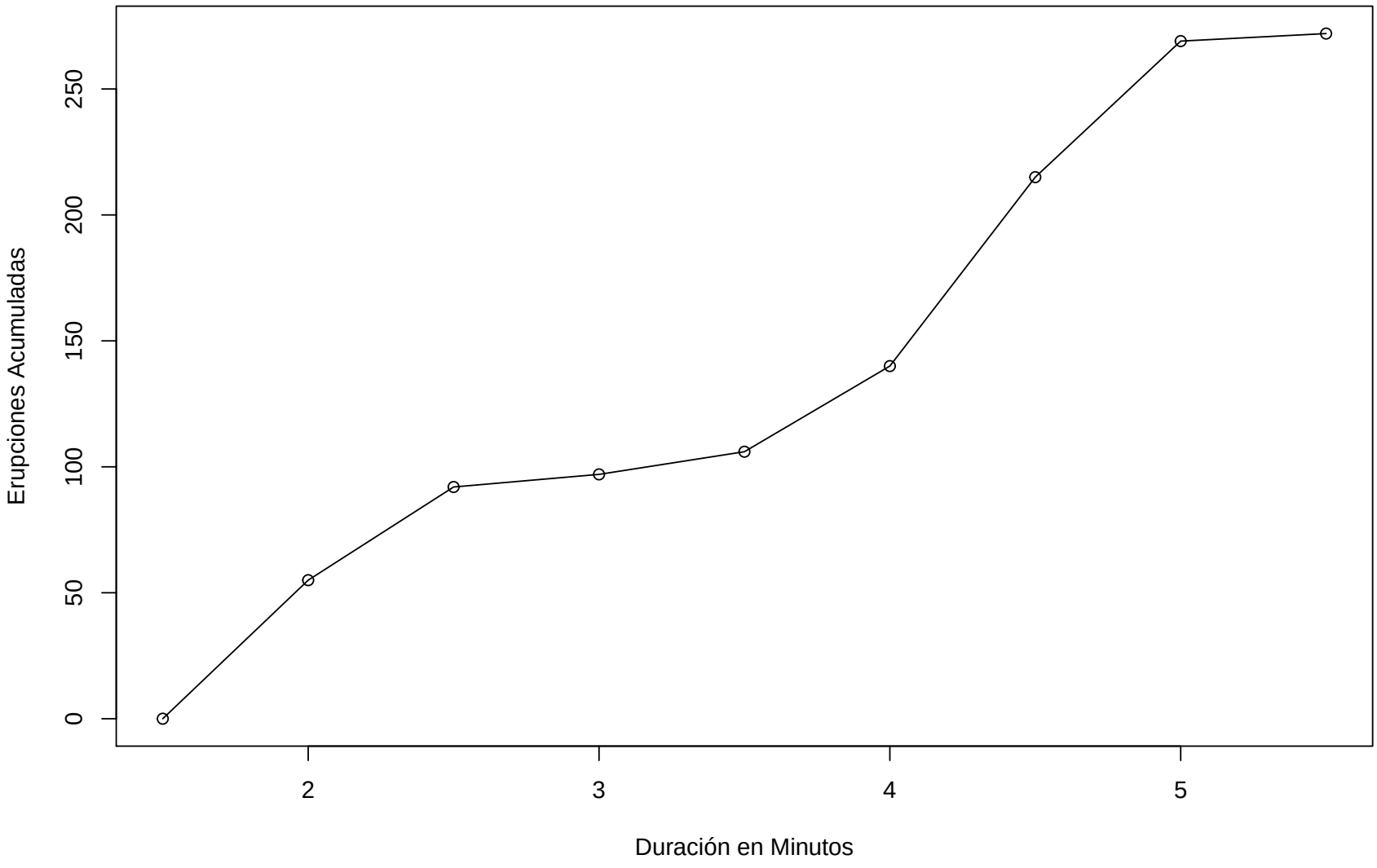
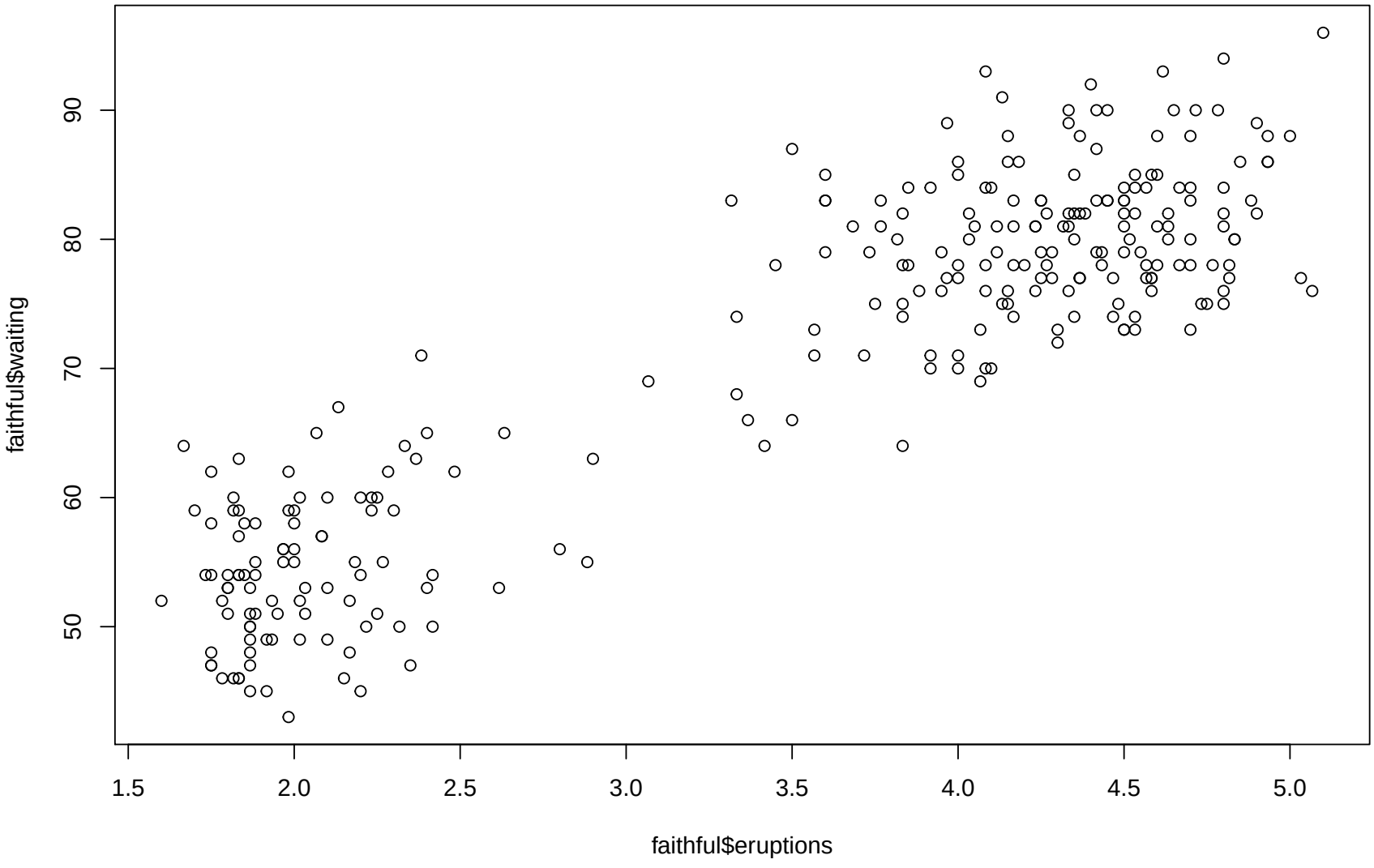

Erupciones Acumuladas
0
50
100
150
200
250
2
3
4
5
Duración en Minutos

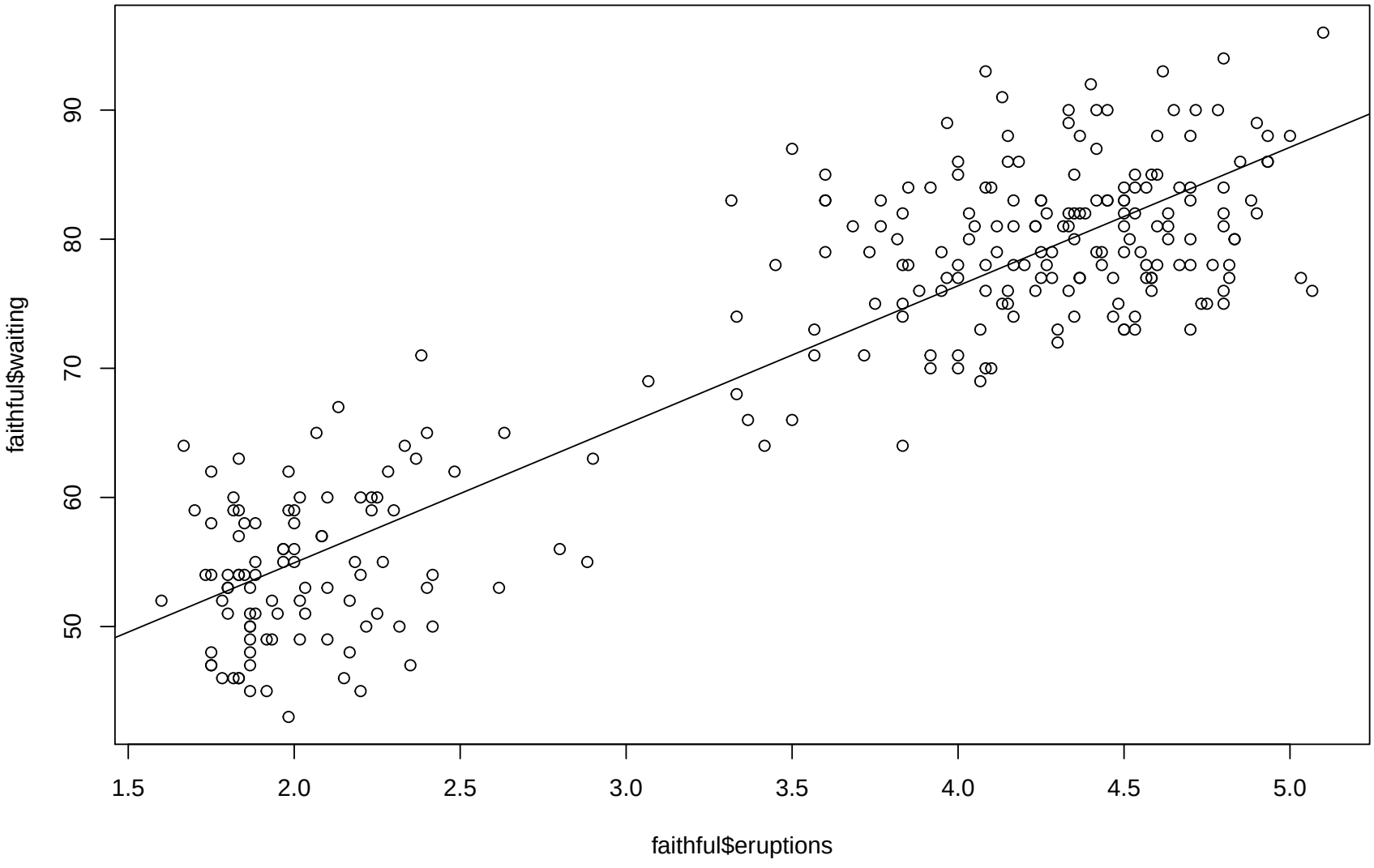
Gráfico de dispersión

```
plot(faithful$eruptions,faithful$waiting)
```



Regresión

```
plot(faithful$eruptions,faithful$waiting)
abline(lm(faithful$waiting~faithful$eruptions))
```



Probabilidad

Distribución	Descripción	Función en R	Parámetros
Binomial	Número de éxitos en *n* ensayos	`rbinom`	`size`, `prob`
Poisson	Número de eventos ocurridos en un lapso	`rpois`	`lambda`
Geométrica	Número de ensayos antes del primer éxito	`rgeom`	`prob`
Binomial negativa	Número de éxitos antes del fracaso *r*	`rnbinom`	`size`, `prob`
Hipergeométrica	*k* éxitos en *m* ensayos en una población de tamaño *n*	`rhyper`	`m`, `n`, `k`

Binomial

Es una distribución de probabilidad discreta, describe éxitos o fracasos

Suponga que está en una rifa, hay siete cajas y dentro de cada caja hay cuatro preguntas de tres respuestas multiples. Encuentre la probabilidad de acertar en dos cajas

```
pbinom(4, size=12, prob=0.333)
```

```
## [1] 0.6324743
```

Poisson

Es la probabilidad de ocurrencia de eventos en un lapso de tiempo

Doce individuos compran Iphones cada hora en Colombia , encuentre la probabilidad de que 30 personas compren en una hora este producto

```
ppois(30,lambda = 12,lower.tail = FALSE)
```

```
## [1] 3.371648e-06
```

Exponencial

Es el tiempo de ocurrencia de eventos Poisson

Suponga que el tiempo promedio de pasar la Septima con 43 es de dos minutos, encuentre la probabilidad de poder pasar en uno

```
pexp(1, rate=1/2)
```

```
## [1] 0.3934693
```