# Exercises

Estadística avanzada

Septiembre 2017

---

## Generar secuencias

---

Rellenar los vectores siguientes:

v1: 1 2 3

v2: 1 2 3 1 2 3 1 2 3

v3: 0 10 20 30 40 50 60 70 80 90 100

v4: 1000 900 800 700 600 500 400 300 200 100 0

v5: 1 1 1 1 1

```
v1 <- c(1,2,3)
v1
```

```
## [1] 1 2 3
```

```
v2 <- rep( v1, 3 )
v2
```

```
## [1] 1 2 3 1 2 3 1 2 3
```

```
v3<-seq(0, 100, 10)
v3
```

```
##  [1]   0  10  20  30  40  50  60  70  80  90 100
```

```
v4<-seq(1000, 0, -100)
v4
```

```
##  [1] 1000  900  800  700  600  500  400  300  200  100    0
```

```
v5<-rep( 1, 5)
v5
```

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

---

## Realizar cálculos a nivel de vector

---

Cacular la suma, media, rango, máximo y mínimo de un vector de números que contiene valores desde 1,500 hasta 500 step -2.

Ordenar el vector en orden creciente

Contar cuantos elementos hay en el vector.

```
v<-seq( 1500, 500, -2)
v
```

```
##   [1] 1500 1498 1496 1494 1492 1490 1488 1486 1484 1482 1480 1478 1476 1474 1472
##  [16] 1470 1468 1466 1464 1462 1460 1458 1456 1454 1452 1450 1448 1446 1444 1442
##  [31] 1440 1438 1436 1434 1432 1430 1428 1426 1424 1422 1420 1418 1416 1414 1412
##  [46] 1410 1408 1406 1404 1402 1400 1398 1396 1394 1392 1390 1388 1386 1384 1382
##  [61] 1380 1378 1376 1374 1372 1370 1368 1366 1364 1362 1360 1358 1356 1354 1352
##  [76] 1350 1348 1346 1344 1342 1340 1338 1336 1334 1332 1330 1328 1326 1324 1322
##  [91] 1320 1318 1316 1314 1312 1310 1308 1306 1304 1302 1300 1298 1296 1294 1292
## [106] 1290 1288 1286 1284 1282 1280 1278 1276 1274 1272 1270 1268 1266 1264 1262
## [121] 1260 1258 1256 1254 1252 1250 1248 1246 1244 1242 1240 1238 1236 1234 1232
## [136] 1230 1228 1226 1224 1222 1220 1218 1216 1214 1212 1210 1208 1206 1204 1202
## [151] 1200 1198 1196 1194 1192 1190 1188 1186 1184 1182 1180 1178 1176 1174 1172
## [166] 1170 1168 1166 1164 1162 1160 1158 1156 1154 1152 1150 1148 1146 1144 1142
## [181] 1140 1138 1136 1134 1132 1130 1128 1126 1124 1122 1120 1118 1116 1114 1112
## [196] 1110 1108 1106 1104 1102 1100 1098 1096 1094 1092 1090 1088 1086 1084 1082
## [211] 1080 1078 1076 1074 1072 1070 1068 1066 1064 1062 1060 1058 1056 1054 1052
## [226] 1050 1048 1046 1044 1042 1040 1038 1036 1034 1032 1030 1028 1026 1024 1022
## [241] 1020 1018 1016 1014 1012 1010 1008 1006 1004 1002 1000  998  996  994  992
## [256]  990  988  986  984  982  980  978  976  974  972  970  968  966  964  962
## [271]  960  958  956  954  952  950  948  946  944  942  940  938  936  934  932
## [286]  930  928  926  924  922  920  918  916  914  912  910  908  906  904  902
## [301]  900  898  896  894  892  890  888  886  884  882  880  878  876  874  872
## [316]  870  868  866  864  862  860  858  856  854  852  850  848  846  844  842
## [331]  840  838  836  834  832  830  828  826  824  822  820  818  816  814  812
## [346]  810  808  806  804  802  800  798  796  794  792  790  788  786  784  782
## [361]  780  778  776  774  772  770  768  766  764  762  760  758  756  754  752
## [376]  750  748  746  744  742  740  738  736  734  732  730  728  726  724  722
## [391]  720  718  716  714  712  710  708  706  704  702  700  698  696  694  692
## [406]  690  688  686  684  682  680  678  676  674  672  670  668  666  664  662
## [421]  660  658  656  654  652  650  648  646  644  642  640  638  636  634  632
## [436]  630  628  626  624  622  620  618  616  614  612  610  608  606  604  602
## [451]  600  598  596  594  592  590  588  586  584  582  580  578  576  574  572
## [466]  570  568  566  564  562  560  558  556  554  552  550  548  546  544  542
## [481]  540  538  536  534  532  530  528  526  524  522  520  518  516  514  512
## [496]  510  508  506  504  502  500
```

```
sum(v)
```

```
## [1] 501000
```

```
mean(v)
```

```
## [1] 1000
```

```
range(v)
```

```
## [1]  500 1500
```

```
max(v)
```

```
## [1] 1500
```

```
min(v)
```

```
## [1] 500
```

```
sort(v)
```

```
##   [1]  500  502  504  506  508  510  512  514  516  518  520  522  524  526  528
##  [16]  530  532  534  536  538  540  542  544  546  548  550  552  554  556  558
##  [31]  560  562  564  566  568  570  572  574  576  578  580  582  584  586  588
##  [46]  590  592  594  596  598  600  602  604  606  608  610  612  614  616  618
##  [61]  620  622  624  626  628  630  632  634  636  638  640  642  644  646  648
##  [76]  650  652  654  656  658  660  662  664  666  668  670  672  674  676  678
##  [91]  680  682  684  686  688  690  692  694  696  698  700  702  704  706  708
## [106]  710  712  714  716  718  720  722  724  726  728  730  732  734  736  738
## [121]  740  742  744  746  748  750  752  754  756  758  760  762  764  766  768
## [136]  770  772  774  776  778  780  782  784  786  788  790  792  794  796  798
## [151]  800  802  804  806  808  810  812  814  816  818  820  822  824  826  828
## [166]  830  832  834  836  838  840  842  844  846  848  850  852  854  856  858
## [181]  860  862  864  866  868  870  872  874  876  878  880  882  884  886  888
## [196]  890  892  894  896  898  900  902  904  906  908  910  912  914  916  918
## [211]  920  922  924  926  928  930  932  934  936  938  940  942  944  946  948
## [226]  950  952  954  956  958  960  962  964  966  968  970  972  974  976  978
## [241]  980  982  984  986  988  990  992  994  996  998 1000 1002 1004 1006 1008
## [256] 1010 1012 1014 1016 1018 1020 1022 1024 1026 1028 1030 1032 1034 1036 1038
## [271] 1040 1042 1044 1046 1048 1050 1052 1054 1056 1058 1060 1062 1064 1066 1068
## [286] 1070 1072 1074 1076 1078 1080 1082 1084 1086 1088 1090 1092 1094 1096 1098
## [301] 1100 1102 1104 1106 1108 1110 1112 1114 1116 1118 1120 1122 1124 1126 1128
## [316] 1130 1132 1134 1136 1138 1140 1142 1144 1146 1148 1150 1152 1154 1156 1158
## [331] 1160 1162 1164 1166 1168 1170 1172 1174 1176 1178 1180 1182 1184 1186 1188
## [346] 1190 1192 1194 1196 1198 1200 1202 1204 1206 1208 1210 1212 1214 1216 1218
## [361] 1220 1222 1224 1226 1228 1230 1232 1234 1236 1238 1240 1242 1244 1246 1248
## [376] 1250 1252 1254 1256 1258 1260 1262 1264 1266 1268 1270 1272 1274 1276 1278
## [391] 1280 1282 1284 1286 1288 1290 1292 1294 1296 1298 1300 1302 1304 1306 1308
## [406] 1310 1312 1314 1316 1318 1320 1322 1324 1326 1328 1330 1332 1334 1336 1338
## [421] 1340 1342 1344 1346 1348 1350 1352 1354 1356 1358 1360 1362 1364 1366 1368
## [436] 1370 1372 1374 1376 1378 1380 1382 1384 1386 1388 1390 1392 1394 1396 1398
## [451] 1400 1402 1404 1406 1408 1410 1412 1414 1416 1418 1420 1422 1424 1426 1428
## [466] 1430 1432 1434 1436 1438 1440 1442 1444 1446 1448 1450 1452 1454 1456 1458
## [481] 1460 1462 1464 1466 1468 1470 1472 1474 1476 1478 1480 1482 1484 1486 1488
## [496] 1490 1492 1494 1496 1498 1500
```

```
length(v)
```

```
## [1] 501
```

---

## Cálculos ariméticos con vectores numéricos

---

Crear un vector con valores de 1 a 20.

Restar 1 a todos los elementos del vector

Para cada elemento, calcular su cuadrado

```
v<-c(1:20)
v
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
v-1
```

```
##  [1]  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
```

```
v^2
```

```
##  [1]   1   4   9  16  25  36  49  64  81 100 121 144 169 196 225 256 289 324 361
## [20] 400
```

---

## Cálculos aritméticos con varios vectores numéricos

---

Crer un vector v1 con valores de 0 a 20 y otro vector v2 con valores de -10 to +10

Generar un vector vsum que contiene la suma de los elementos de los dos vectores, posición por posición.

```
v1<-c(0:20)
v2<-c(-10:10)
length(v1)
```

```
## [1] 21
```

```
length(v2)
```

```
## [1] 21
```

```
vsum <- v1 + v2
vsum
```

```
##  [1] -10  -8  -6  -4  -2   0   2   4   6   8  10  12  14  16  18  20  22  24  26
## [20]  28  30
```

---

## Vectores numéricos

---

Disponemos de un vector “quantity” y un vector de precios “price”:

quantity: 20 3 4 5 6 7 8 5 2 12

price: 2 4 5 63 40 21 12 6 7 4

Cada posición del vector quantity es la cantidad de productos vendidos. El precio está guardado en la misma posición del vector price. Calcular el precio total de todos los productos.

```
quantity <- c(20,3,4,5,6,7,8,5,2,12)
price <- c(2,4,5,63,40,21,12,6,7,4)
quantity
```

```
##  [1] 20  3  4  5  6  7  8  5  2 12
```

```
price
```

```
##  [1]  2  4  5 63 40 21 12  6  7  4
```

```
total_price<-quantity*price
total_price
```

```
##  [1]  40  12  20 315 240 147  96  30  14  48
```

```
total_amount<-sum( total_price )
total_amount
```

```
## [1] 962
```

---

## Vectores numéricos y lógicos

---

Dados los siguientes vectores:

x: 3 4 6 -5 1 3 9 10 -1 0 -4

y: -4 4 6 5 1 3 9 1 -10 5 14

Realizar los cálculos siguientes:

Cuántos elementos del vector x son positivos?

```
x<-c(3,4,6,-5,1,3,9,10,-1,0,-4)
y<-c(-4,4,6,5,1,3,9,1,-10,5,14)
#Elements of x greater than 0
x>0
```

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

```
sum(x>0)
```

```
## [1] 7
```

```
length( which(x>0) )
```

```
## [1] 7
```

Qué elementos del vector x son mayores que los elementos del vector y (considerando posición por posición).

```
#elements where x > y: index, elements, how many
which( x>y )
```

```
## [1] 1 8 9
```

```
x[ x>y ]
```

```
## [1]  3 10 -1
```

```
length( which(x>y) )
```

```
## [1] 3
```

Borrar del vector x los elementos que son iguales a los elementos que ocupan la misma posición en y.

```
#Visualization
paste(x, y)
```

```
##  [1] "3 -4"   "4 4"    "6 6"    "-5 5"   "1 1"    "3 3"    "9 9"    "10 1"
##  [9] "-1 -10" "0 5"    "-4 14"
```

```
#elements where x == y
x==y
```

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

```
which(x==y)
```

```
## [1] 2 3 5 6 7
```

```
x[ !(x==y) ]
```

```
## [1]  3 -5 10 -1  0 -4
```

```
#Modifying x so that only the elements that are different remain
x<-x[ !(x==y) ]
x
```

```
## [1]  3 -5 10 -1  0 -4
```

---

# Factors

---

El siguiente vector pertenece al nivel de estudios de 20 personas:

studies <- c("n", "p", "s", "p", "s", "t", "o", "s", "o" , "p", "p","s", "s","t","p","t","o","n","p","o")

Calcular cuántos tipos de educación hay.

Para hacerlo más comprensible, cambiar el valor de los niveles a los siguientes:

n : none

p : primary

s : secondary

t : tertiary

o : other

Mostrar cuantas personas hay para cada tipo de educación.

```
studies <- factor(c("n", "p", "s", "p", "s", "t", "o", "s", "o" , "p", "p","s", "s","t","p","t","o","n"
length( studies)
```

```
## [1] 20
```

```
#types of education
levels( studies )
```

```
## [1] "n" "o" "p" "s" "t"
```

```
#changing names
levels( studies ) <- c("none", "other", "primary", "secondary", "tertiary")
studies
```

```
##  [1] none      primary   secondary primary   secondary tertiary  other
##  [8] secondary other     primary   primary   secondary secondary tertiary
## [15] primary   tertiary  other     none      primary   other
```

```
## Levels: none other primary secondary tertiary
```

```
table( studies )
```

```
## studies
##      none     other   primary secondary  tertiary
##         2         4         6         5         3
```

---

# Matrices

---

Generar una matriz de 10 filas y 6 columnas, cuyos valores de cada columna son los múltiples del índice de la columna, tal como sigue:

[1,] 1 2 3 4 5 6

[2,] 2 4 6 8 10 12

[3,] 3 6 9 12 15 18

[4,] 4 8 12 16 20 24

[5,] 5 10 15 20 25 30

[6,] 6 12 18 24 30 36

[7,] 7 14 21 28 35 42

[8,] 8 16 24 32 40 48

[9,] 9 18 27 36 45 54

[10,] 10 20 30 40 50 60

Etiquetar cada columna como: M1, M2, ... M6

Borrar las dos últimas filas.

Cambiar el signo de los valores de la columna 3.

Mostrar el contenido de la columna 2

```
#Creating
v1<- c(1:10)
v2<-v1*2
v3<-v1*3
v4<-v1*4
v5<-v1*5
v6<-v1*6
mat <- cbind( v1, v2, v3, v4, v5, v6 )
#Structure and content ok?
str(mat)
```

```
##  num [1:10, 1:6] 1 2 3 4 5 6 7 8 9 10 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:6] "v1" "v2" "v3" "v4" ...
```

```
mat
```

```
##       v1 v2 v3 v4 v5 v6
##  [1,]  1  2  3  4  5  6
##  [2,]  2  4  6  8 10 12
##  [3,]  3  6  9 12 15 18
##  [4,]  4  8 12 16 20 24
##  [5,]  5 10 15 20 25 30
##  [6,]  6 12 18 24 30 36
##  [7,]  7 14 21 28 35 42
##  [8,]  8 16 24 32 40 48
##  [9,]  9 18 27 36 45 54
## [10,] 10 20 30 40 50 60
```

```
#Column names
colnames( mat ) <- c("M1", "M2", "M3", "M4", "M5", "M6")
mat
```

```
##       M1 M2 M3 M4 M5 M6
##  [1,]  1  2  3  4  5  6
##  [2,]  2  4  6  8 10 12
##  [3,]  3  6  9 12 15 18
##  [4,]  4  8 12 16 20 24
##  [5,]  5 10 15 20 25 30
##  [6,]  6 12 18 24 30 36
##  [7,]  7 14 21 28 35 42
##  [8,]  8 16 24 32 40 48
##  [9,]  9 18 27 36 45 54
## [10,] 10 20 30 40 50 60
```

```
mat <- mat[1:8,]
mat
```

```
##      M1 M2 M3 M4 M5 M6
## [1,]  1  2  3  4  5  6
## [2,]  2  4  6  8 10 12
## [3,]  3  6  9 12 15 18
## [4,]  4  8 12 16 20 24
## [5,]  5 10 15 20 25 30
## [6,]  6 12 18 24 30 36
## [7,]  7 14 21 28 35 42
## [8,]  8 16 24 32 40 48
```

```
#Sign
mat[,3] <- -mat[,3]
mat
```

```
##      M1 M2  M3 M4 M5 M6
## [1,]  1  2  -3  4  5  6
## [2,]  2  4  -6  8 10 12
## [3,]  3  6  -9 12 15 18
## [4,]  4  8 -12 16 20 24
## [5,]  5 10 -15 20 25 30
## [6,]  6 12 -18 24 30 36
## [7,]  7 14 -21 28 35 42
## [8,]  8 16 -24 32 40 48
```

```
#Show
mat[,"M2"]
```

```
## [1]  2  4  6  8 10 12 14 16
```

---

## Data Frames ("ejercicio avanzado")

---

Leer el fichero "bank.csv" (http://archive.ics.uci.edu/ml/datasets/Bank+Marketing) y guardar los datos en un data frame. Verificar que la información se ha leído correctamente en el data.frame.

Calcular el promedio de edad.

Calcular cuantos trabajos diferentes hay en la muestra.

Calcular cuantas personas hay de cada tipo de trabajo.

Calcular cuantos "entprepreneurs" hay.

Seleccionar los individuos que son emprendedores y calcular su promedio de edad.

Seleccionar los emprenedores cuya edad es inferior al promedio de edad de los emprenedeores. ¿Cuántos son?

¿Cuál es el nivel de estudios de estos emprendedores jóvenes?

Borrar del data frame todos los emprendedores jóvenes.

```
bank<-read.csv("bank.csv", sep=";",na.strings = "NA")
head( bank ) #shows the first rows
```

```
##   age         job marital education default balance housing loan  contact day
## 1  30  unemployed married   primary      no    1787      no   no cellular  19
## 2  33    services married secondary      no    4789     yes  yes cellular  11
## 3  35  management  single  tertiary      no    1350     yes   no cellular  16
## 4  30  management married  tertiary      no    1476     yes  yes  unknown   3
## 5  59 blue-collar married secondary      no       0     yes   no  unknown   5
## 6  35  management  single  tertiary      no     747      no   no cellular  23
##   month duration campaign pdays previous poutcome  y
## 1   oct       79        1    -1        0  unknown no
## 2   may      220        1   339        4  failure no
## 3   apr      185        1   330        1  failure no
## 4   jun      199        4    -1        0  unknown no
## 5   may      226        1    -1        0  unknown no
## 6   feb      141        2   176        3  failure no
```

```
str(bank)
```

```
## 'data.frame':    4521 obs. of  17 variables:
##  $ age      : int  30 33 35 30 59 35 36 39 41 43 ...
##  $ job      : chr  "unemployed" "services" "management" "management" ...
##  $ marital  : chr  "married" "married" "single" "married" ...
##  $ education: chr  "primary" "secondary" "tertiary" "tertiary" ...
##  $ default  : chr  "no" "no" "no" "no" ...
##  $ balance  : int  1787 4789 1350 1476 0 747 307 147 221 -88 ...
##  $ housing  : chr  "no" "yes" "yes" "yes" ...
##  $ loan     : chr  "no" "yes" "no" "yes" ...
##  $ contact  : chr  "cellular" "cellular" "cellular" "unknown" ...
##  $ day      : int  19 11 16 3 5 23 14 6 14 17 ...
##  $ month    : chr  "oct" "may" "apr" "jun" ...
##  $ duration : int  79 220 185 199 226 141 341 151 57 313 ...
```

```
##  $ campaign : int  1 1 1 4 1 2 1 2 2 1 ...
##  $ pdays    : int  -1 339 330 -1 -1 176 330 -1 -1 147 ...
##  $ previous : int  0 4 1 0 0 3 2 0 0 2 ...
##  $ poutcome : chr  "unknown" "failure" "failure" "unknown" ...
##  $ y        : chr  "no" "no" "no" "no" ...
```

```
nrow( bank )
```

```
## [1] 4521
```

```
#Promedio de edad
mean( bank$age )
```

```
## [1] 41.1701
```

```
#Calcular cuantos trabajos diferentes hay en la muestra.
levels( bank$job )
```

```
## NULL
```

```
#Número de personas para cada tipo de trabajo y para "entrepreneur"
table( bank$job )
```

```
##
##        admin.   blue-collar  entrepreneur     housemaid    management
##           478           946           168           112           969
##       retired self-employed      services       student    technician
##           230           183           417            84           768
##    unemployed       unknown
##           128            38
```

```
table( bank$job )['entrepreneur']
```

```
## entrepreneur
##          168
```

```
#Edad promedio de entrepreneurs
#bank$job == "entrepreneur"
entrep <- bank[ bank$job=='entrepreneur', ]
head( entrep )
```

```
##     age          job  marital education default balance housing loan  contact
## 9    41 entrepreneur  married  tertiary      no     221     yes   no  unknown
## 24   44 entrepreneur  married secondary      no      93      no   no cellular
## 49   32 entrepreneur   single   primary     yes    -849     yes  yes cellular
## 72   42 entrepreneur divorced  tertiary     yes       2     yes   no  unknown
## 151  41 entrepreneur  married   unknown      no      89     yes   no  unknown
## 154  55 entrepreneur  married secondary      no    8104      no   no cellular
##     day month duration campaign pdays previous poutcome  y
## 9    14   may       57        2    -1        0  unknown no
## 24    7   jul      125        2    -1        0  unknown no
## 49    4   feb      204        1    -1        0  unknown no
## 72    5   may      380        1    -1        0  unknown no
## 151   6   may      333        2    -1        0  unknown no
## 154   6   feb      213        2    -1        0  unknown no
```

```
nrow( entrep )
```

```
## [1] 168
```

```r
meanAgeEntre <- mean( entrep$age )

#jóvenes entrepreneurs
ye <- entrep[ entrep$age<meanAgeEntre, ]
nrow( ye )
```

```
## [1] 94
```

```r
# nivel de estudios
table( ye$education )
```

```
##
##   primary secondary  tertiary   unknown
##        12        31        47         4
```

```r
#borrar jóvenes entrepreneurs
length( which(  bank$job=='entrepreneur' ) )
```

```
## [1] 168
```

```r
indexYE <- which(  bank$job=='entrepreneur' & bank$age<meanAgeEntre )   #indexes
indexYE
```

```
##  [1]    9   49   72  151  232  299  319  328  370  450  473  596  613  645  649
## [16]  652  673  700  749  760  826  840  877  910  979 1004 1012 1193 1208 1245
## [31] 1253 1282 1305 1325 1427 1437 1488 1507 1523 1556 1640 1646 1829 1921 1992
## [46] 2106 2140 2170 2196 2211 2227 2291 2294 2299 2347 2394 2424 2611 2751 2809
## [61] 2823 2951 2978 2989 2990 3075 3238 3277 3366 3391 3396 3486 3488 3542 3581
## [76] 3586 3674 3730 3806 3886 3917 3922 3946 4008 4009 4097 4106 4157 4335 4386
## [91] 4400 4402 4403 4498
```

```r
length( indexYE )    #how many
```

```
## [1] 94
```

```r
bank <- bank[ -indexYE, ]
nrow(bank)
```

```
## [1] 4427
```

```r
nrow( bank[ bank$job=='entrepreneur',  ] )
```

```
## [1] 74
```

---

## Ejercicio resumen

---

Dado el siguiente vector generado aleatoriamente:

x<-round( rnorm(300, mean=25, sd=9) )

x<-x[x>10]

Supongamos que pertenece a las edades de 300 individuos. Calcular:

Copiar la información en un fichero “ages.csv”

Leer el fichero y cargar los datos de las edades de nuevo.

Calcular cuantos individuos hay en la muestra.

Calcular el máximo, mínimo, promedio y mediana.

¿Cuántas personas de cada edad hay en la muestra?

Calcular la fecha de nacimiento de cada persona.

Selecionar las personas que nacieron entre 1980 and 2000, ambos incluidos.

Calcular cuantos jugadores hay mayores de 30 años y cuántos son menores de 20.

Convertir los valores en un nuevo vector, donde se clasifican los individuos en:

VY (very young): <20 años

Y (young), 20-34 años

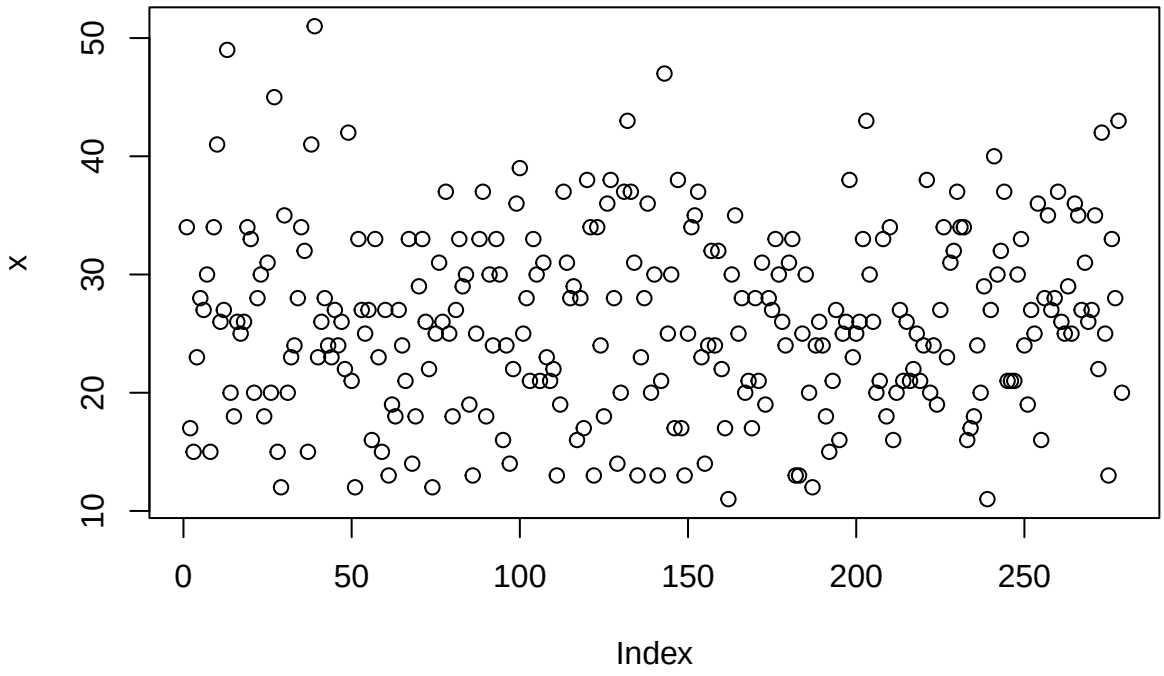
A (adult): 35-49 años

OA (older adult): 50-65 años

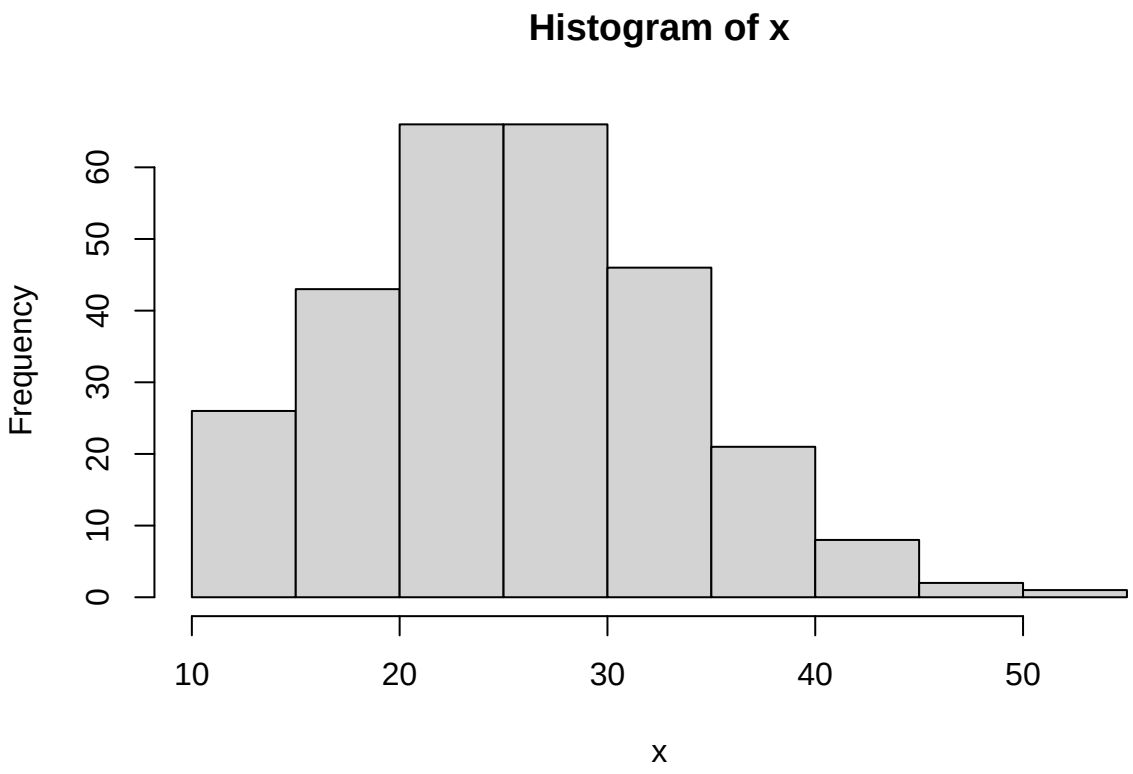
O (old): >65 años

Calculas cuantos individuos hay de cada categoría de edad.

```
x<-round( rnorm(300, mean=25, sd=9) )
x<-x[x>10]
plot(x)
```




```
hist(x)
```




```
str(x)
```

```
##  num [1:279] 34 17 15 23 28 27 30 15 34 41 ...
```

```
x<-as.data.frame(x)

write.csv(x, file="ages.csv",row.names=F)

#Lee solo la columna x del data.frame
ages<-read.csv("ages.csv")$x
str(ages)
```

```
##  int [1:279] 34 17 15 23 28 27 30 15 34 41 ...
```

```
#Número de personas
length(ages)
```

```
## [1] 279
```

```
#max,min, mean, median
max(ages)
```

```
## [1] 51
```

```
min(ages)
```

```
## [1] 11
```

```
mean(ages)
```

```
## [1] 26.09677
```

```
median(ages)
```

```
## [1] 26
```

```
#Número de personas por edad
table(ages)
```

```
## ages
## 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
##  2  4 10  4  6  7  7 10  6 13 16  7 10 16 17 15 17 15  5 14  9  5 15 11  6  5
## 37 38 39 40 41 42 43 45 47 49 51
##  9  5  1  1  2  2  3  1  1  1  1
```

```
#Fecha nacimiento
birthDate <- 2017 - ages
birthDate
```

```
##   [1] 1983 2000 2002 1994 1989 1990 1987 2002 1983 1976 1991 1990 1968 1997 1999
##  [16] 1991 1992 1991 1983 1984 1997 1989 1987 1999 1986 1997 1972 2002 2005 1982
##  [31] 1997 1994 1993 1989 1983 1985 2002 1976 1966 1994 1991 1989 1993 1994 1990
##  [46] 1993 1991 1995 1975 1996 2005 1984 1990 1992 1990 2001 1984 1994 2002 1990
##  [61] 2004 1998 1999 1990 1993 1996 1984 2003 1999 1988 1984 1991 1995 2005 1992
##  [76] 1986 1991 1980 1992 1999 1990 1984 1988 1987 1998 2004 1992 1984 1980 1999
##  [91] 1987 1993 1984 1987 2001 1993 2003 1995 1981 1978 1992 1989 1996 1984 1987
## [106] 1996 1986 1994 1996 1995 2004 1998 1980 1986 1989 1988 2001 1989 2000 1979
## [121] 1983 2004 1983 1993 1999 1981 1979 1989 2003 1997 1980 1974 1980 1986 2004
## [136] 1994 1989 1981 1997 1987 2004 1996 1970 1992 1987 2000 1979 2000 2004 1992
## [151] 1983 1982 1980 1994 2003 1993 1985 1993 1985 1995 2000 2006 1987 1982 1992
## [166] 1989 1997 1996 2000 1989 1996 1986 1998 1989 1990 1984 1987 1991 1993 1986
## [181] 1984 2004 2004 1992 1987 1997 2005 1993 1991 1993 1999 2002 1996 1990 2001
## [196] 1992 1991 1979 1994 1992 1991 1984 1974 1987 1991 1997 1996 1984 1999 1983
## [211] 2001 1997 1990 1996 1991 1996 1995 1992 1996 1993 1979 1997 1993 1998 1990
## [226] 1983 1994 1986 1985 1980 1983 1983 2001 2000 1999 1993 1997 1988 2006 1990
## [241] 1977 1987 1985 1980 1996 1996 1996 1987 1984 1993 1998 1990 1992 1981 2001
## [256] 1989 1982 1990 1989 1980 1991 1992 1988 1992 1981 1982 1990 1986 1991 1990
## [271] 1982 1995 1975 1992 2004 1984 1989 1974 1997
```

```
#Personas nacidas entre 1980 y 2000
birthDate[ birthDate >=1980 & birthDate <= 2000 ]
```

```
##   [1] 1983 2000 1994 1989 1990 1987 1983 1991 1990 1997 1999 1991 1992 1991 1983
##  [16] 1984 1997 1989 1987 1999 1986 1997 1982 1997 1994 1993 1989 1983 1985 1994
##  [31] 1991 1989 1993 1994 1990 1993 1991 1995 1996 1984 1990 1992 1990 1984 1994
##  [46] 1990 1998 1999 1990 1993 1996 1984 1999 1988 1984 1991 1995 1992 1986 1991
##  [61] 1980 1992 1999 1990 1984 1988 1987 1998 1992 1984 1980 1999 1987 1993 1984
##  [76] 1987 1993 1995 1981 1992 1989 1996 1984 1987 1996 1986 1994 1996 1995 1998
##  [91] 1980 1986 1989 1988 1989 2000 1983 1983 1993 1999 1981 1989 1997 1980 1980
## [106] 1986 1994 1989 1981 1997 1987 1996 1992 1987 2000 2000 1992 1983 1982 1980
## [121] 1994 1993 1985 1993 1985 1995 2000 1987 1982 1992 1989 1997 1996 2000 1989
## [136] 1996 1986 1998 1989 1990 1984 1987 1991 1993 1986 1984 1992 1987 1997 1993
## [151] 1991 1993 1999 1996 1990 1992 1991 1994 1992 1991 1984 1987 1991 1997 1996
## [166] 1984 1999 1983 1997 1990 1996 1991 1996 1995 1992 1996 1993 1997 1993 1998
## [181] 1990 1983 1994 1986 1985 1980 1983 1983 2000 1999 1993 1997 1988 1990 1987
## [196] 1985 1980 1996 1996 1996 1987 1984 1993 1998 1990 1992 1981 1989 1982 1990
## [211] 1989 1980 1991 1992 1988 1992 1981 1982 1990 1986 1991 1990 1982 1995 1992
```

```
## [226] 1984 1989 1997
```

```
#Mayores de 30 años
#Menores de 20
length( ages[ ages>30 ] )
```

```
## [1] 78
```

```
length( ages[ ages<20 ] )
```

```
## [1] 56
```

```
#Conversión
ageCategory <- ages
ages<20
```

```
##   [1] FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
##  [13] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
##  [25] FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [37]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [49] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE
##  [61]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE
##  [73] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
##  [85]  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE
##  [97]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [109] FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE
## [121] FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
## [133] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
## [145] FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
## [157] FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [169]  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [181] FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE  TRUE
## [193] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [205] FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
## [217] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [229] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE
## [241] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
## [253] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [265] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
## [277] FALSE FALSE FALSE
```

```
ageCategory[ ages<20 ] <- "VY"
ageCategory[ ages>=20 & ages<35] <-"Y"
ageCategory[ ages>=35 & ages<50] <-"A"
ageCategory[ ages>=50 & ages<65] <-"OA"
ageCategory[ ages>=65 ] <-"O"
ageCategory
```

```
##   [1] "Y"  "VY" "VY" "Y"  "Y"  "Y"  "Y"  "VY" "Y"  "A"  "Y"  "Y"  "A"  "Y"  "VY"
##  [16] "Y"  "Y"  "Y"  "Y"  "Y"  "Y"  "Y"  "Y"  "VY" "Y"  "Y"  "A"  "VY" "VY" "A"
##  [31] "Y"  "Y"  "Y"  "Y"  "Y"  "Y"  "VY" "A"  "OA" "Y"  "Y"  "Y"  "Y"  "Y"  "Y"
##  [46] "Y"  "Y"  "Y"  "A"  "Y"  "VY" "Y"  "Y"  "Y"  "Y"  "VY" "Y"  "Y"  "VY" "Y"
##  [61] "VY" "VY" "VY" "Y"  "Y"  "Y"  "Y"  "VY" "VY" "Y"  "Y"  "Y"  "Y"  "VY" "Y"
##  [76] "Y"  "Y"  "A"  "Y"  "VY" "Y"  "Y"  "Y"  "Y"  "VY" "VY" "Y"  "Y"  "A"  "VY"
##  [91] "Y"  "Y"  "Y"  "Y"  "VY" "Y"  "VY" "Y"  "A"  "A"  "Y"  "Y"  "Y"  "Y"  "Y"
## [106] "Y"  "Y"  "Y"  "Y"  "Y"  "VY" "VY" "A"  "Y"  "Y"  "Y"  "VY" "Y"  "VY" "A"
## [121] "Y"  "VY" "Y"  "Y"  "VY" "A"  "A"  "Y"  "VY" "Y"  "A"  "A"  "A"  "Y"  "VY"
```

```
## [136] "Y"  "Y"  "A"  "Y"  "Y"  "VY" "Y"  "A"  "Y"  "Y"  "VY" "A"  "VY" "VY" "Y"
## [151] "Y"  "A"  "A"  "Y"  "VY" "Y"  "Y"  "Y"  "Y"  "Y"  "VY" "VY" "Y"  "A"  "Y"
## [166] "Y"  "Y"  "Y"  "VY" "Y"  "Y"  "Y"  "VY" "Y"  "Y"  "Y"  "Y"  "Y"  "Y"  "Y"
## [181] "Y"  "VY" "VY" "Y"  "Y"  "Y"  "VY" "Y"  "Y"  "Y"  "VY" "VY" "Y"  "Y"  "VY"
## [196] "Y"  "Y"  "A"  "Y"  "Y"  "Y"  "Y"  "A"  "Y"  "Y"  "Y"  "Y"  "Y"  "VY" "Y"
## [211] "VY" "Y"  "Y"  "Y"  "Y"  "Y"  "Y"  "Y"  "Y"  "Y"  "A"  "Y"  "Y"  "VY" "Y"
## [226] "Y"  "Y"  "Y"  "Y"  "A"  "Y"  "Y"  "VY" "VY" "VY" "Y"  "Y"  "Y"  "VY" "Y"
## [241] "A"  "Y"  "Y"  "A"  "Y"  "Y"  "Y"  "Y"  "Y"  "Y"  "VY" "Y"  "Y"  "A"  "VY"
## [256] "Y"  "A"  "Y"  "Y"  "A"  "Y"  "Y"  "Y"  "Y"  "A"  "A"  "Y"  "Y"  "Y"  "Y"
## [271] "A"  "Y"  "A"  "Y"  "VY" "Y"  "Y"  "A"  "Y"
```

```
str(ageCategory)
```

```
##  chr [1:279] "Y" "VY" "VY" "Y" "Y" "Y" "Y" "VY" "Y" "A" "Y" "Y" "A" "Y" ...
```

```
#Cuantas personas de cada categoría de edad
table(ageCategory)
```

```
## ageCategory
##   A  OA  VY   Y
##  37   1  56 185
```