Estadística descriptiva con R

000R Team

Métodos estadísticos de investigación: introducción a R y Rstudio

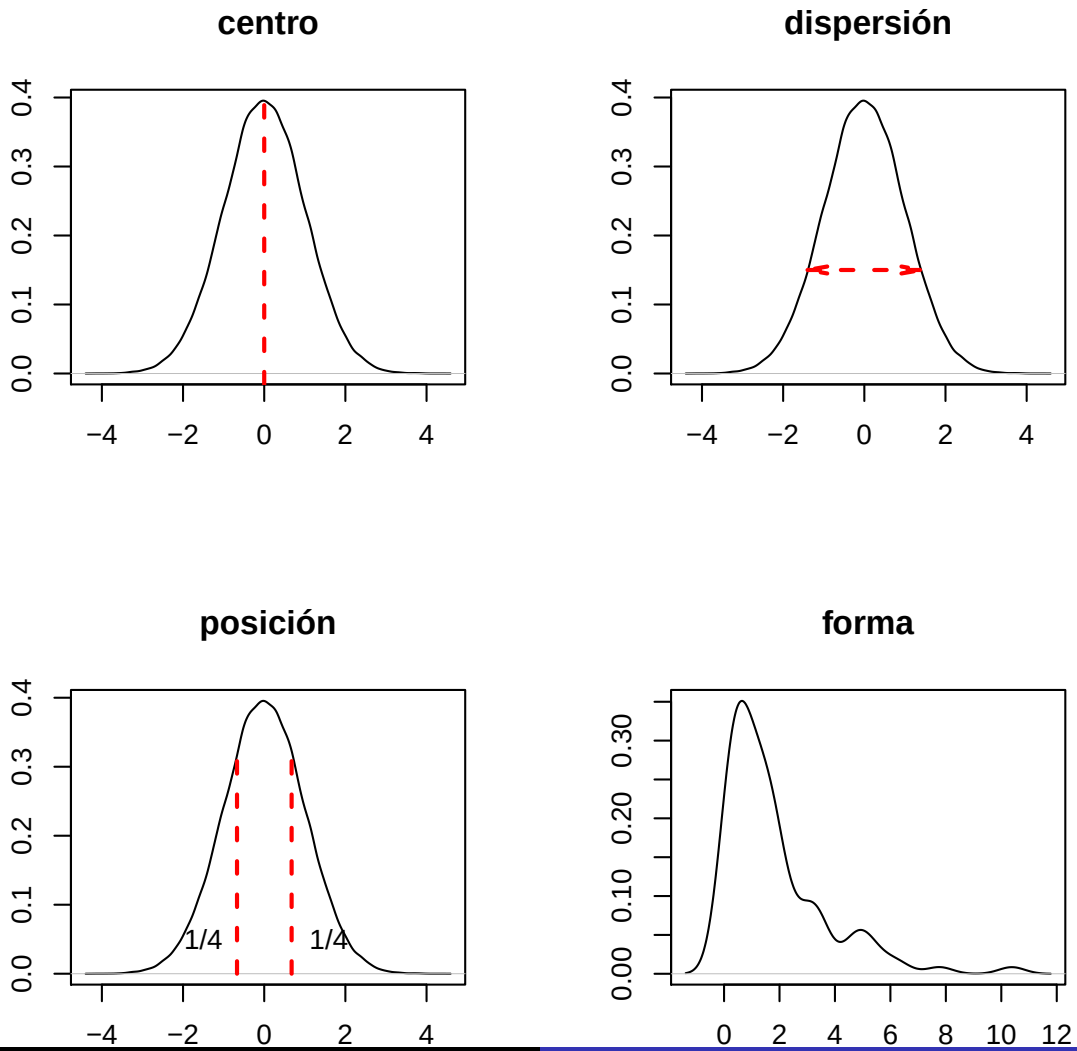
Objetivos

- repasar asignaciones
- `install.packages()`
- repasar aplicación de funciones
- algunos gráficos clásicos
- funciones útiles

Estadísticos

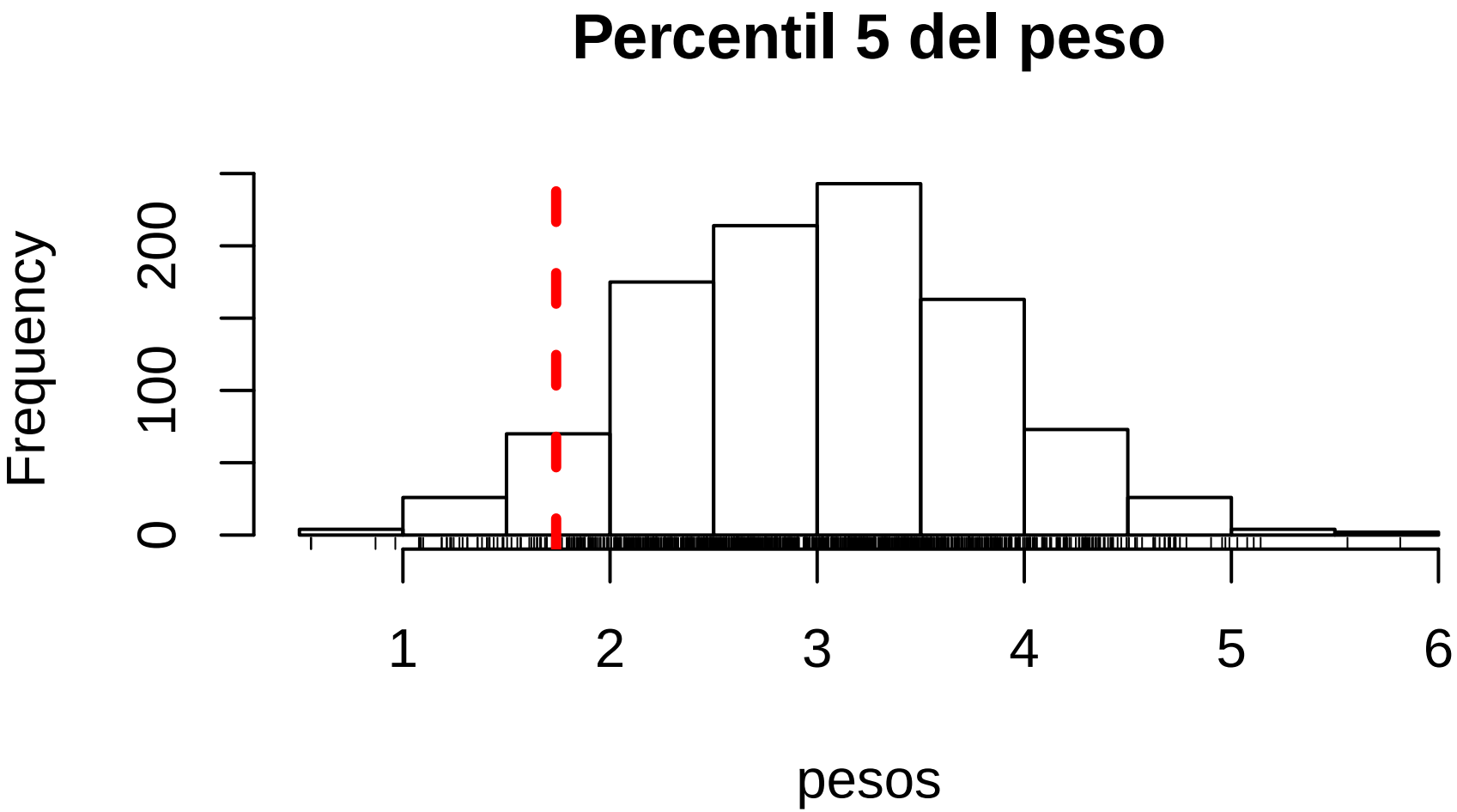
Tipos de estadísticos

Los estadísticos se calculan, y estos estiman parámetros



Percentil 5

Ejemplo: El 5 % de los recién nacidos tiene un peso demasiado bajo. ¿Qué peso se considera "demasiado bajo"? Percentil 5 o cuantil 0.05.



Percentil 75

Ejemplo: *¿Qué peso es superado sólo por el 25 % de los individuos? Percentil 75, tercer cuartil o cuantil 0.75.*

```
quantile( pesos, 0.75 )
```

```
##       75%
## 3.504159
```

Media

Media (*mean*) Es la media aritmética (promedio) de los valores de una variable. Suma de los valores dividido por el tamaño muestral

```
mean( pesos )
```

```
## [1] 2.99708
```

```
x <- c( 1, 2, 3, 4, 5 )
mean(x)
```

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

```
media <- ( 1 + 2 + 3 + 4 + 5 ) / 5
media
```

Mediana

Mediana ('median') Es un valor que divide a las observaciones en dos grupos con el mismo número de individuos (percentil 50). Si el número de datos es par, se elige la media de los dos datos centrales

- Mediana de 1, 2, 4, **5**, 6, 6, 8 es 5
- Mediana de 1, 2, 4, **5, 6**, 6, 8, 9 es $\frac{5+6}{2} = 5.5$
- Es conveniente cuando los datos son asimétricos. No es sensible a valores extremos.
- Mediana de 1, 2, 4, 5, 6, 6, 800 es 5. ¡La media es 117, 7!

Mediana (ii)

```
median( pesos )
```

```
## [1] 3.000515
```

```
x <- c( 1, 2, 4, 5, 6, 6, 8 )
median( x )
```

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

```
y <- c( 1, 2, 4, 5, 6, 6, 8, 9 )
z <- c( 1, 2, 4, 5, 6, 6, 800 )
median( z )
```

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

Moda

Moda : Es el valor donde la distribución de frecuencia alcanza un máximo

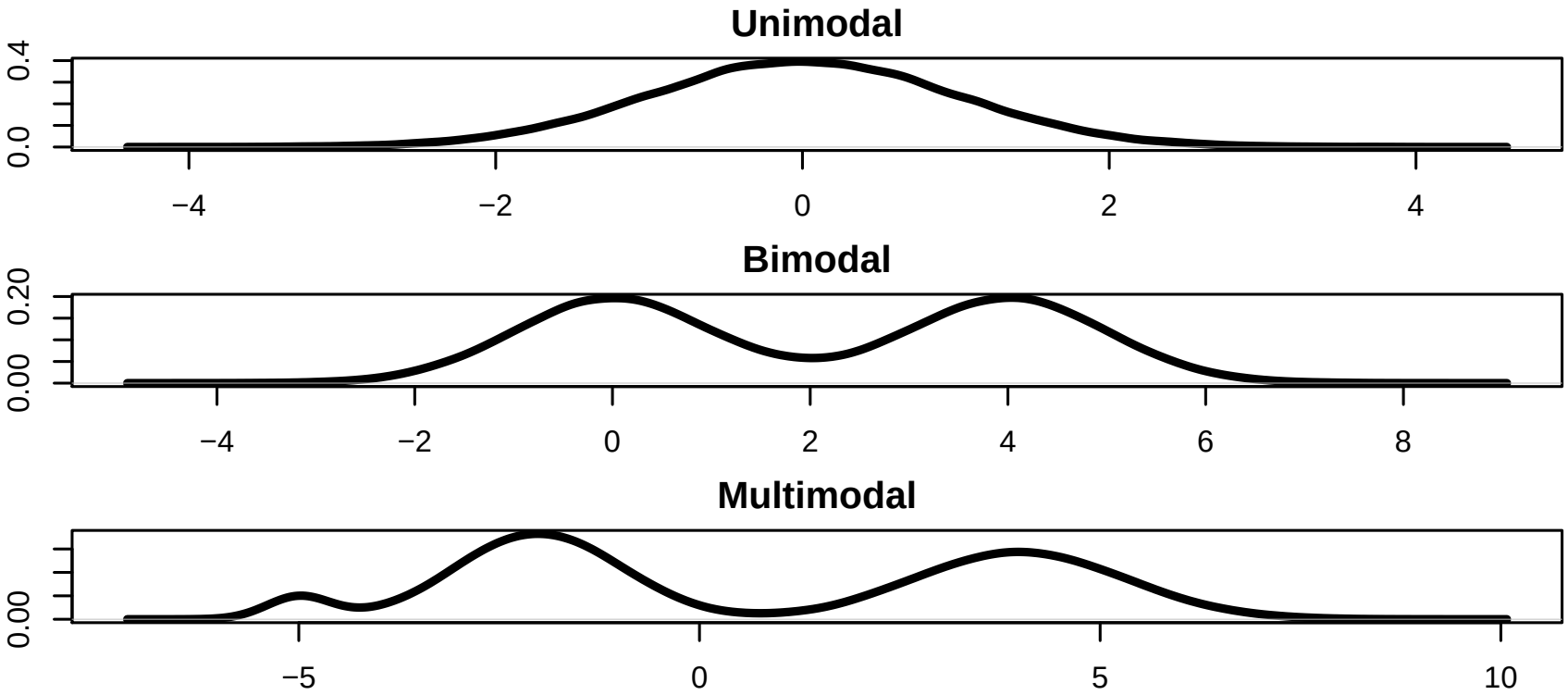


Figura 2: Ejempo de distrbución unimodal, la moda está en el intervalo 4–4.5.

Moda (ii). install.packages()

```
# install.packages( 'prettyR', dependencies = T )
library( "prettyR" )
help( package = "prettyR" )
```

Dispersión

```
pesos <- rnorm( 1000, 3, 0.8 )
range( pesos )
```

```
## [1] 0.1401877 5.9202955
```

```
IQR <- ( quantile( pesos, 0.75, names = F ) -
         quantile( pesos, 0.25, names = F ) )
IQR
```

```
## [1] 1.083211
```

```
var( pesos )
```

```
## [1] 0.6666681
```

```
sd( pesos )
```

```
## [1] 0.8164975
```

```
CV <- sd( pesos ) / mean( pesos )
```

Medidas de forma

```
library(e1071)
kurtosis(pesos)
```

```
## [1] 0.404083
```

```
skewness(pesos)
```

```
## [1] 0.1662267
```

Algunos gráficos útiles

histograma

es una representación gráfica de una variable en forma de barras, donde la superficie de cada barra es proporcional a la frecuencia de los valores representados.

Representar histogramas en R es tan sencillo como crear un objeto histograma, con la función `hist()`.

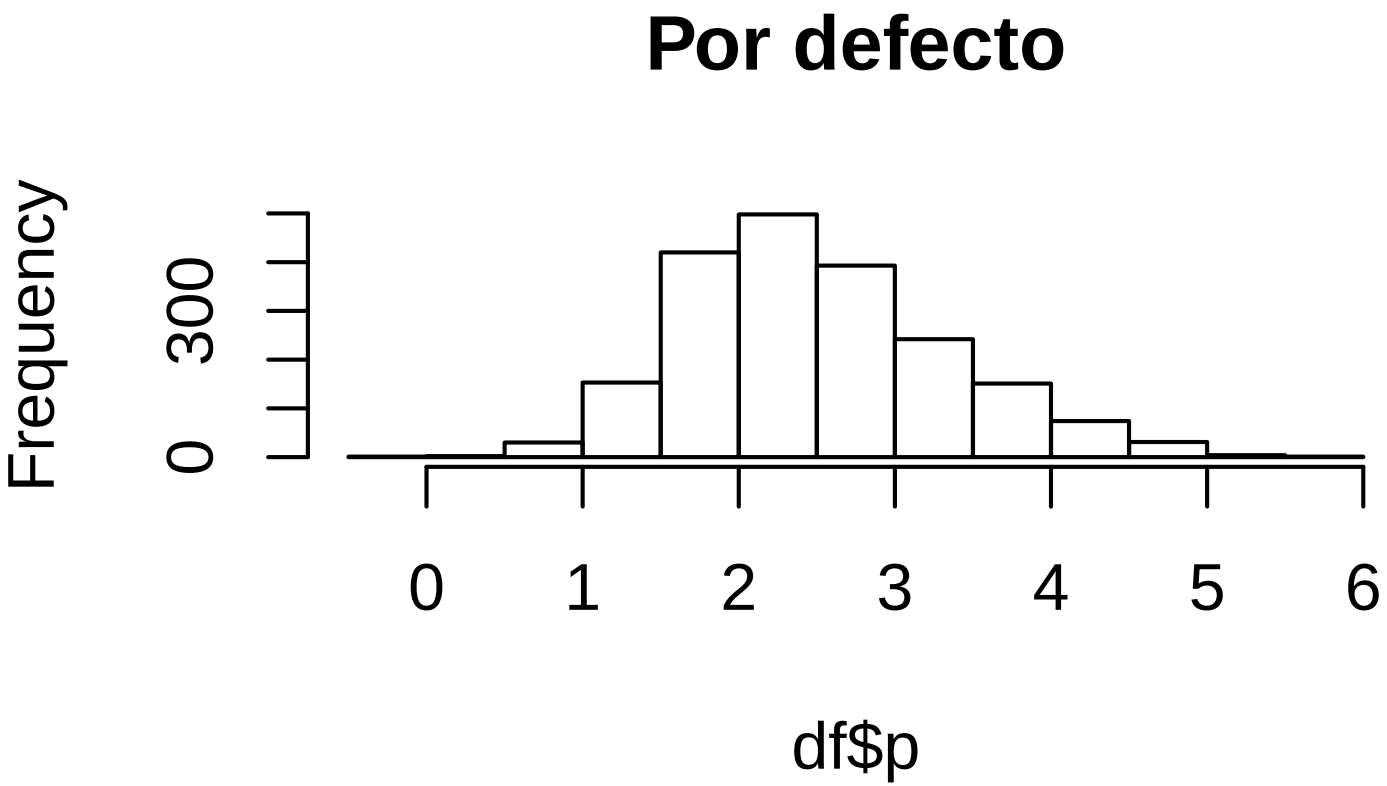
Prueba el siguiente código y observa qué ocurre.

histogramas en R (i)

```
##          p grupo
## 1 2.631145     M
## 2 3.428693     M
## 3 2.152377     M
## 4 2.507022     M
## 5 1.908969     M
## 6 1.559964     M
```

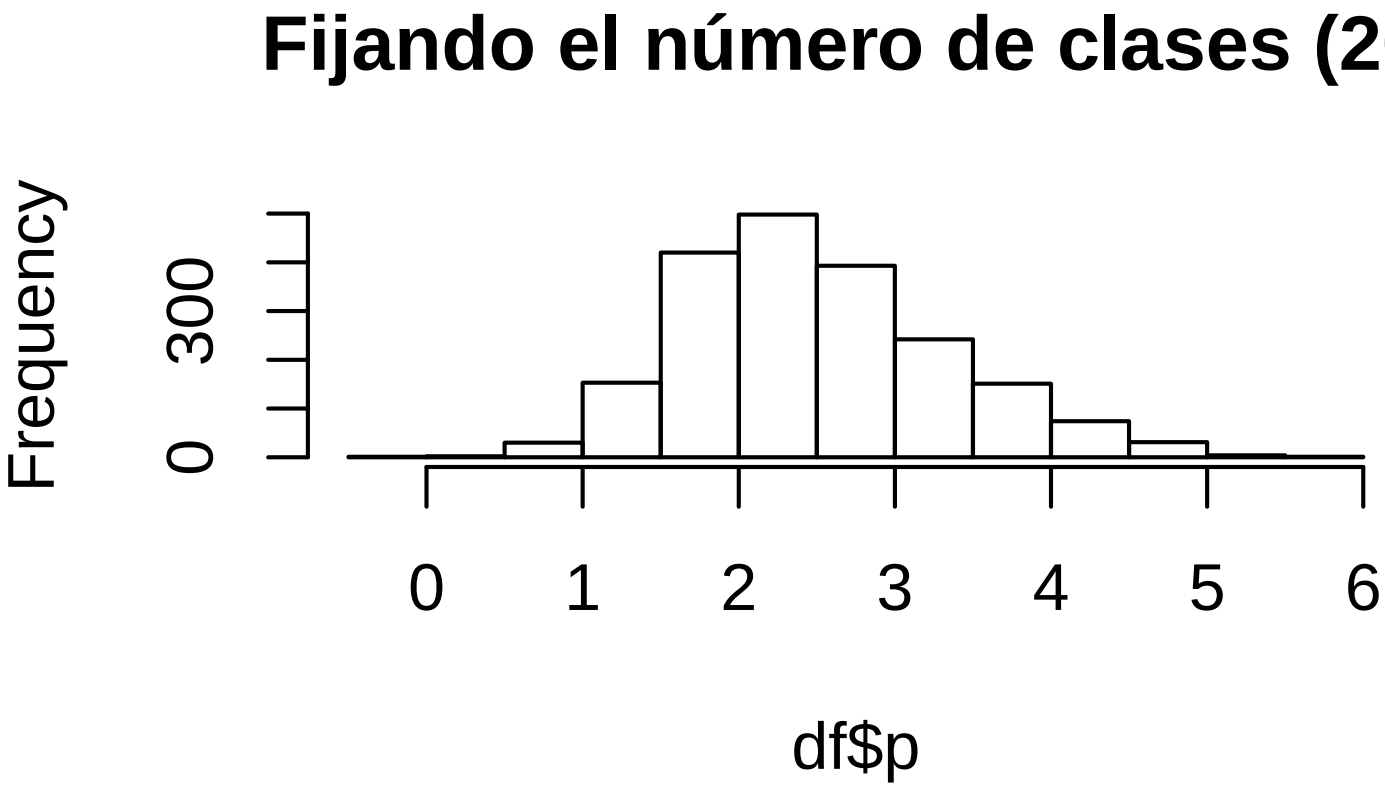
histogramas en R (ii)

```
hist( df$p, main = "Por defecto" )
```



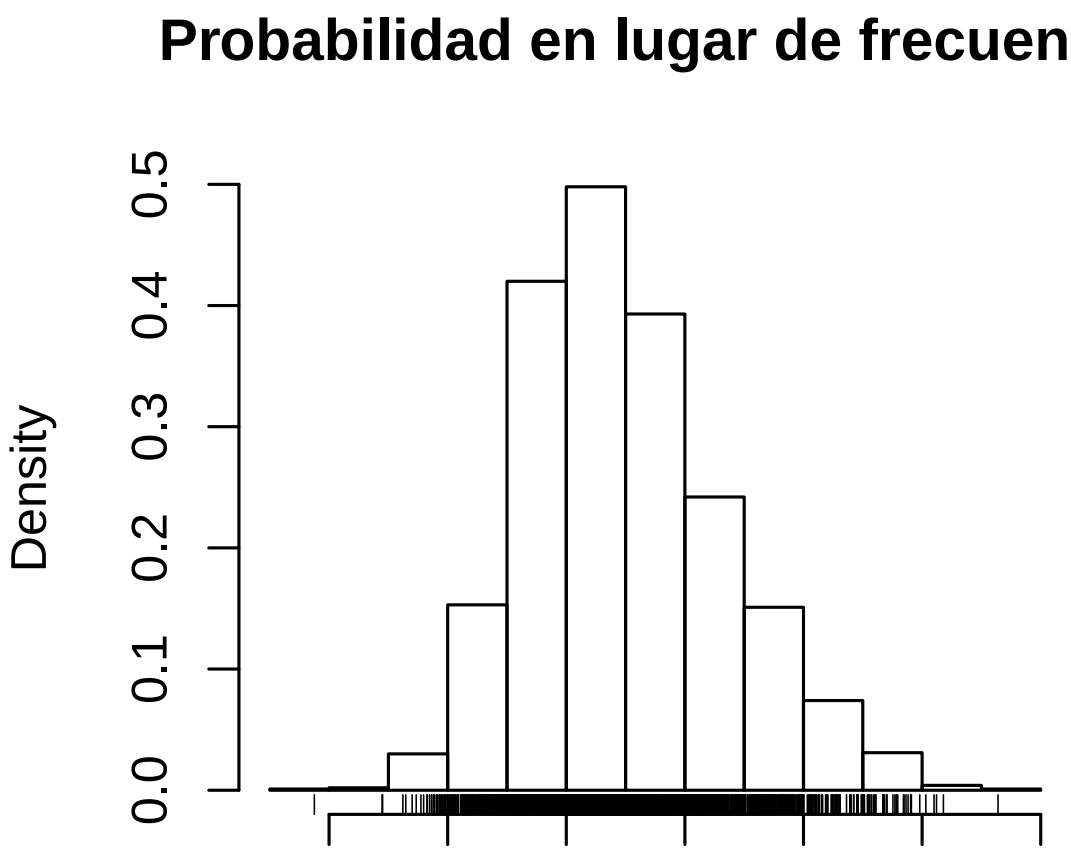
histogramas en R (iii)

```
hist( df$p, breaks = 20
          , main = "Fijando el número de clases (20)" )
```



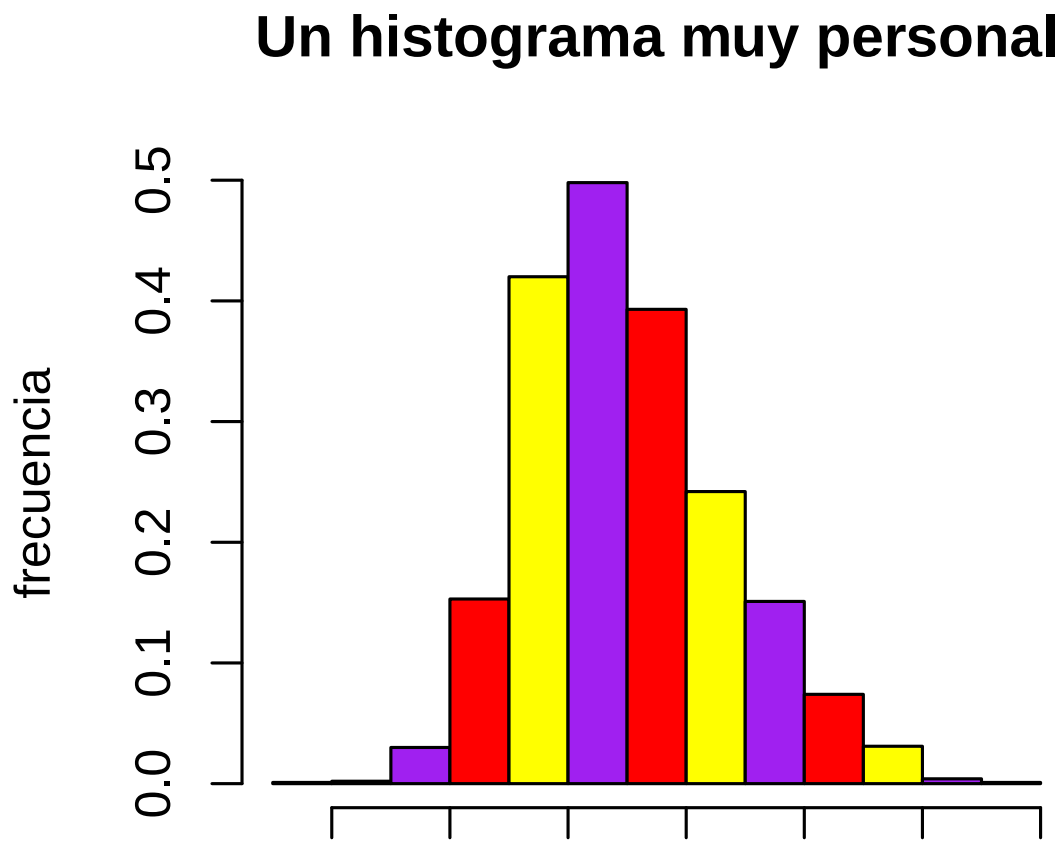
histogramas en R (iv)

```
hist( df$p, breaks = 20, probability = TRUE,
      main = "Probabilidad en lugar de frecuencia" )
rug( jitter( p ) )  # aspecto: añadimos marcas en el
```



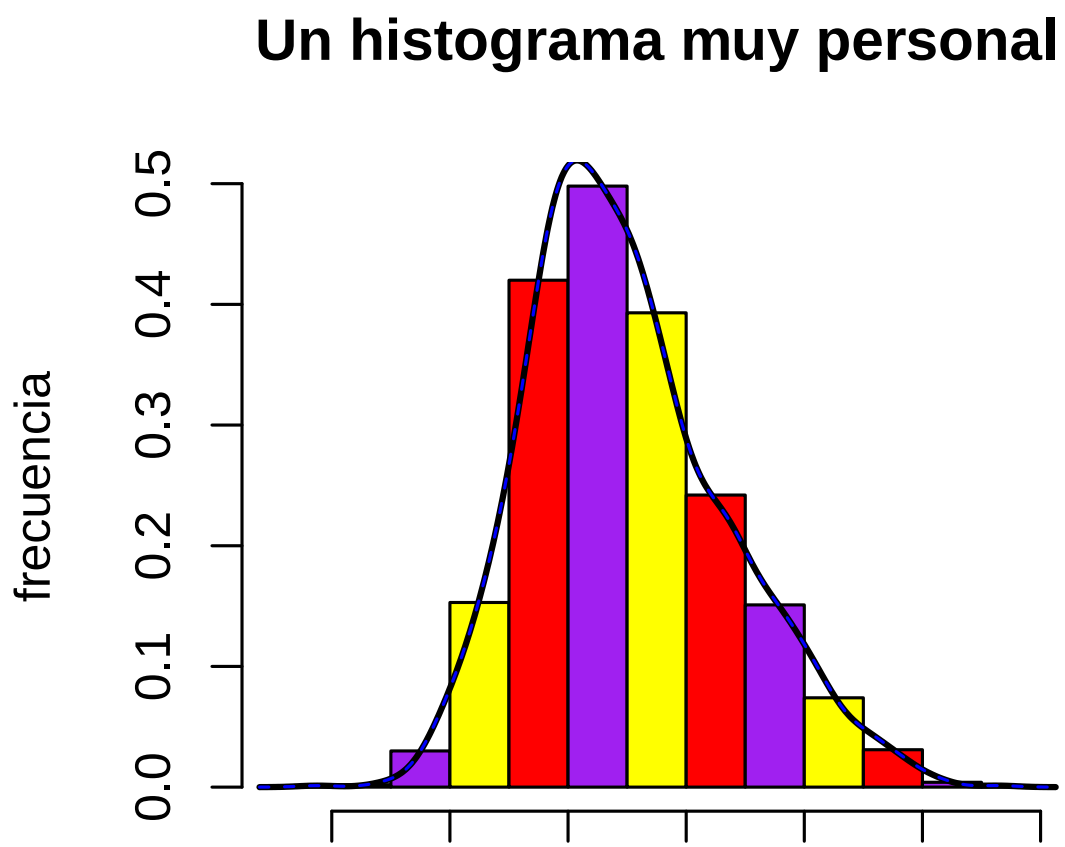
histogramas en R (v)

```
hist( df$p, breaks = 20,
      probability = TRUE,
      xlab = "peso en kg",ylab = "frecuencia",
      col = c( "red", "yellow", "purple" ),
      main = "Un histograma muy personal")
```



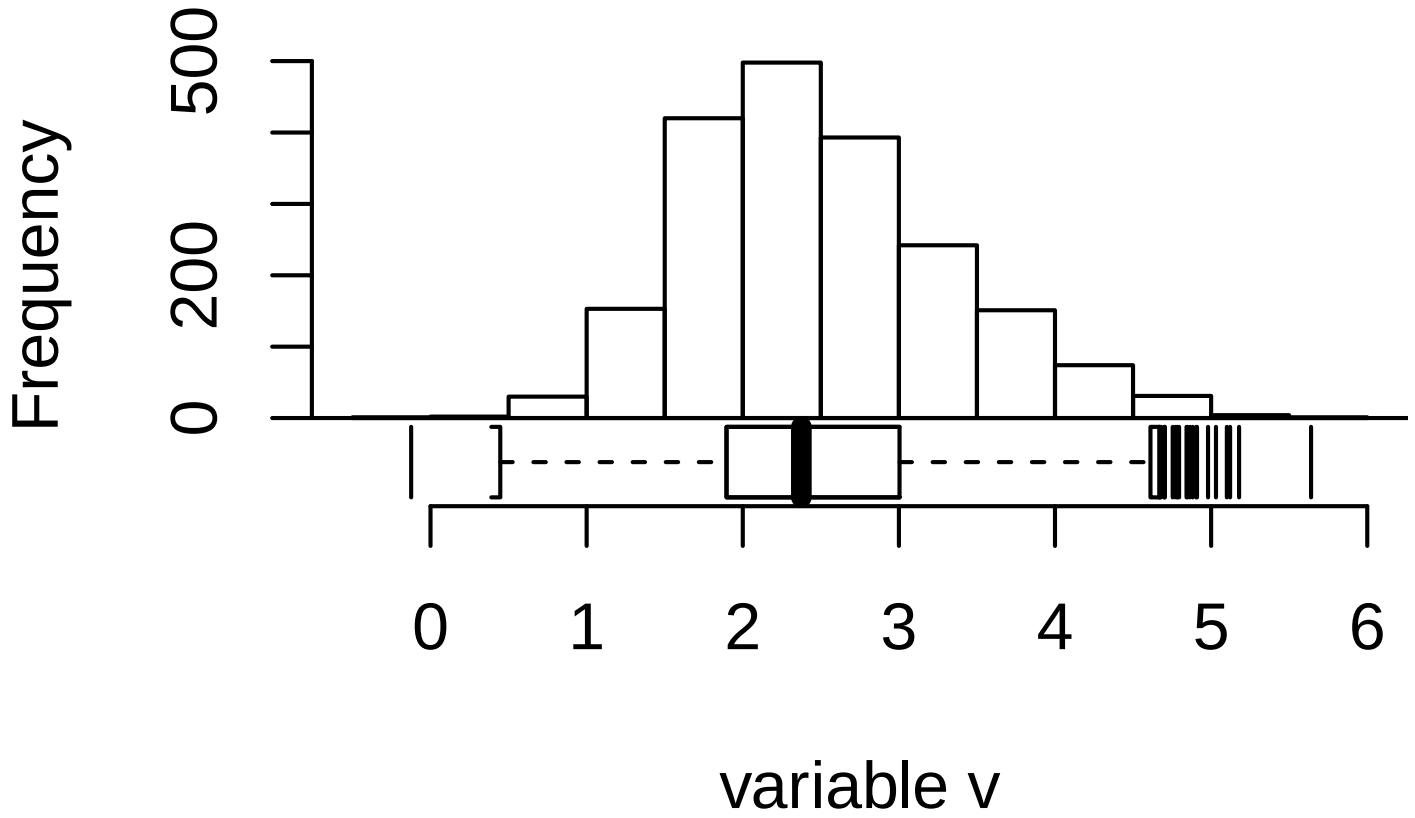
más detalles

```
lines( density( df$p ), lwd = 2 ) # + su curva de densidad
lines( density(df$p),
       col = "blue", lty = 2, ps = 20 )
```



histBxp()

```
library( sfsmisc )
histBxp( df$p, main = "",  xlab = "variable v",
         probability = F,
         boxcol = 0,  medcol = 1 )
```



Algunas funciones útiles

summary()

```
p1  <- rnorm( 1000, 3, 0.8 ); p2 <- rnorm( 1000, 2, 0.5 )
p <- c( p1, p2 )
altura <- c( rnorm( 1000, 87, 7 ), rnorm( 1000, 97, 6 ) )
grupo <- c( rep( "M", 1000 ), rep( "H", 1000 ) )
df <- data.frame( p, altura, grupo )
head( df )
```

```
##          p   altura grupo
## 1 1.229983 86.39785     M
## 2 2.510176 94.10382     M
## 3 2.576331 96.86273     M
## 4 1.095173 87.70147     M
## 5 2.879924 84.53924     M
## 6 3.456218 89.28255     M
```

summary() (ii)

```
summary( df )
```

```
##        p                altura          grupo
##  Min.   :0.5462   Min.   : 65.12   H:1000
##  1st Qu.:1.9030   1st Qu.: 86.52   M:1000
##  Median :2.3786   Median : 92.27
##  Mean   :2.4866   Mean   : 92.10
##  3rd Qu.:2.9668   3rd Qu.: 97.70
##  Max.   :5.3927   Max.   :117.08
```

summary con filtros

```
summary( df[ which( df$grupo == "M" ), ] )
```

```
##        p                altura          grupo
##  Min.   :0.5462   Min.   : 65.12   H:   0
##  1st Qu.:2.4177   1st Qu.: 82.41   M:1000
##  Median :2.9224   Median : 87.27
##  Mean   :2.9591   Mean   : 87.13
##  3rd Qu.:3.5287   3rd Qu.: 91.76
##  Max.   :5.3927   Max.   :111.03
```

crear data frames auxiliares

```
df.M <- df[ which( df$grupo == "M" ),  ]
summary( df.M )
```

```
##        p                altura          grupo
##  Min.   :0.5462   Min.   : 65.12   H:   0
##  1st Qu.:2.4177   1st Qu.: 82.41   M:1000
##  Median :2.9224   Median : 87.27
##  Mean   :2.9591   Mean   : 87.13
##  3rd Qu.:3.5287   3rd Qu.: 91.76
##  Max.   :5.3927   Max.   :111.03
```

stat.desc() del paquete pastecs

La función `stat.desc()` del paquete `pastecs`, tiene varias opciones muy interesantes:

stat.desc(x, basic = TRUE, desc = TRUE, norm = FALSE, p=0.95)

- `basic`: nº de valores, nº de nulls, nº de missing, min, max, rango, suma...
- `desc`: mediana, media, SEM, IC(95 %), var, sd, CV,
- `norm`: (OJO: no fijado en TRUE por defecto), curtosis, asimetría, Shapiro-Wilk

ejemplo stat.desc()

```
library( "pastecs" )
stat.desc( df )
```

```
##                          p       altura grupo
## nbr.val      2.000000e+03 2.000000e+03    NA
## nbr.null     0.000000e+00 0.000000e+00    NA
## nbr.na       0.000000e+00 0.000000e+00    NA
## min          5.462417e-01 6.511867e+01    NA
## max          5.392736e+00 1.170839e+02    NA
## range        4.846494e+00 5.196528e+01    NA
## sum          4.973162e+03 1.841934e+05    NA
## median       2.378609e+00 9.227171e+01    NA
## mean         2.486581e+00 9.209670e+01    NA
## SE.mean      1.842837e-02 1.831909e-01    NA
## CI.mean.0.95 3.614083e-02 3.592650e-01    NA
## var          6.792099e-01 6.711779e+01    NA
## std.dev      8.241419e-01 8.192545e+00    NA
## coef.var     3.314358e-01 8.895590e-02    NA
```

mas refinado

Sin incluir la variable *'grupo'* y con la opción `norm = T`.

```
stat.desc( df[ -3 ],  norm = TRUE )
```

```
##                         p        altura
## nbr.val      2.000000e+03  2.000000e+03
## nbr.null     0.000000e+00  0.000000e+00
## nbr.na       0.000000e+00  0.000000e+00
## min          5.462417e-01  6.511867e+01
## max          5.392736e+00  1.170839e+02
## range        4.846494e+00  5.196528e+01
## sum          4.973162e+03  1.841934e+05
## median       2.378609e+00  9.227171e+01
## mean         2.486581e+00  9.209670e+01
## SE.mean      1.842837e-02  1.831909e-01
## CI.mean.0.95 3.614083e-02  3.592650e-01
## var          6.792099e-01  6.711779e+01
## std.dev      8.241419e-01  8.192545e+00
## coef.var     3.314358e-01  8.895590e-02
## skewness     5.769014e-01 -6.429631e-02
## skew.2SE     5.270312e+00 -5.873822e-01
## kurtosis     1.176069e-01 -2.375780e-01
```

Función tapply()

tapply()

Con la función `tapply()` nos podemos construir fácilmente nuestras tablas de descriptivos de una forma muy elegante.

```
tapply( df$p, df$g, mean )
```

```
##        H        M
## 2.014093 2.959069
```

tapply() (ii)

```
m  <- tapply ( df$p, df$g, mean   )
s  <- tapply ( df$p, df$g, sd     )
m2 <- tapply ( df$p, df$g, median )
n  <- tapply ( df$p, df$g, length )
cbind( media = m,
       sd = s,
       mediana = m2,
       n )
```

```
##      media        sd  mediana    n
## H 2.014093 0.4915292 2.008546 1000
## M 2.959069 0.8188786 2.922375 1000
```

funcion *ad hoc* con tapply()

No es difícil construir una función para estadísticos descriptivos *ad hoc* haciendo uso de la función tapply().

```
fdescriptivos <- function ( variable, factor ) {
    df <- na.omit( data.frame(variable, factor )   )
    m  <- tapply ( df$variable , df$factor, mean   )
    s  <- tapply ( df$variable , df$factor, sd     )
    me <- tapply ( df$variable , df$factor, median )
    n  <- tapply ( df$variable , df$factor, length )
    cbind( media = m, sd = s, mediana = me, n )
}
```

Tablas de frecuencias

Tabla sencilla

Esto lo veremos más extensamente cuando hablemos de contrastes no paramétricos

```
pais <- c( "ES", "ES", "ES", "US", "US" )
sexo <- c( "F",  "F",   "M", "F",  "M"  )

t <- table( pais, sexo )  # tabla de frecuencias absolutas
t
```

```
##      sexo
## pais F M
##   ES 2 1
##   US 1 1
```

frec. relativas

```
# frec relativas
prop.table( t )  # porcentajes totales
```

```
##      sexo
## pais   F   M
##   ES 0.4 0.2
##   US 0.2 0.2
```

Porcentajes

```
prop.table( t ) * 100
```

```
##      sexo
## pais  F  M
##   ES 40 20
##   US 20 20
```

Porcentajes poor filas y columnas

```
# porcentajes por filas
prop.table( t, 1 )
```

```
##      sexo
## pais         F         M
##   ES 0.6666667 0.3333333
##   US 0.5000000 0.5000000
```

```
# porcentajes por columnas
prop.table( t, 2 )
```

```
##      sexo
## pais         F         M
##   ES 0.6666667 0.5000000
##   US 0.3333333 0.5000000
```

Fin

Nos vemos en gauss

`http://gauss.inf.um.es/`