Tema 1. Estadística Descriptiva con R (1ª parte)

Estadística

Ángel Serrano Sánchez de León

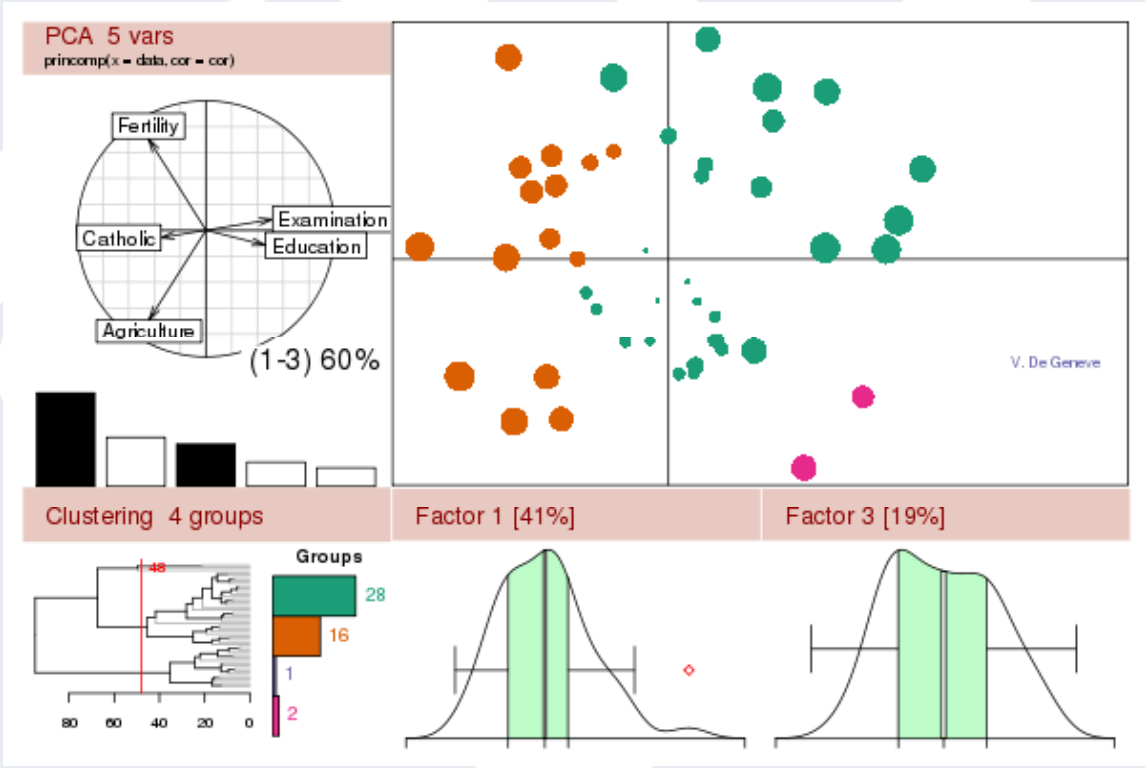
Índice

- Introducción a R
- Paquetes instalados
- Introducción de datos por teclado o concatenación
- Variables cuantitativas
- Variables categóricas (nominales, ordinales)
- Distribuciones de frecuencias
- Gráficos básicos (barras, histogramas, tartas)

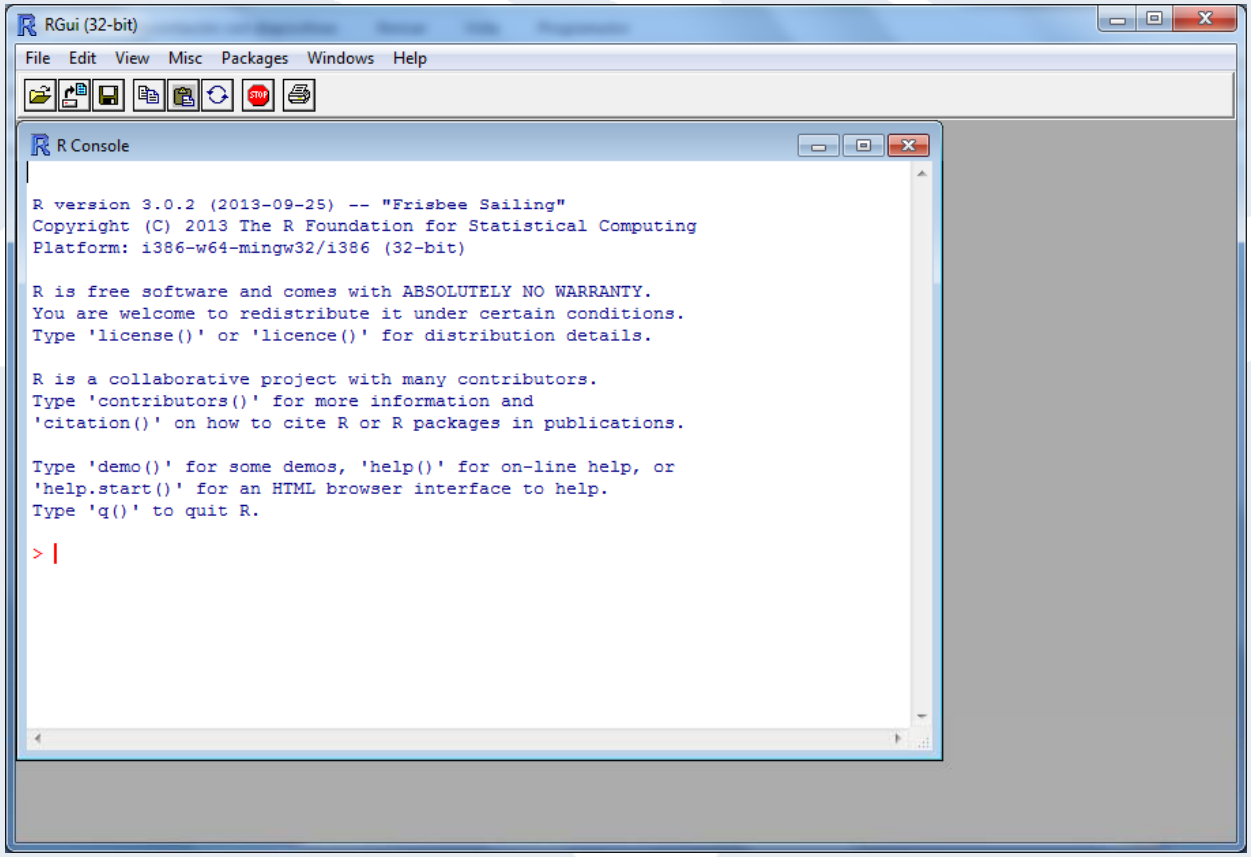
Introducción

- **R:** Lenguaje y entorno de programación para cálculos estadísticos y visualización de datos.
- Creado por Ross Ihaka y Robert Gentleman de la the Universidad de Auckland (Nueva Zelanda) en 1993.
- Basado en el lenguaje de programación estadístico S.
- Software libre (licencia GNU).
- Multiplataforma.
- Basado en consola de comandos.
- http://www.r-project.org/

Introducción



Introducción

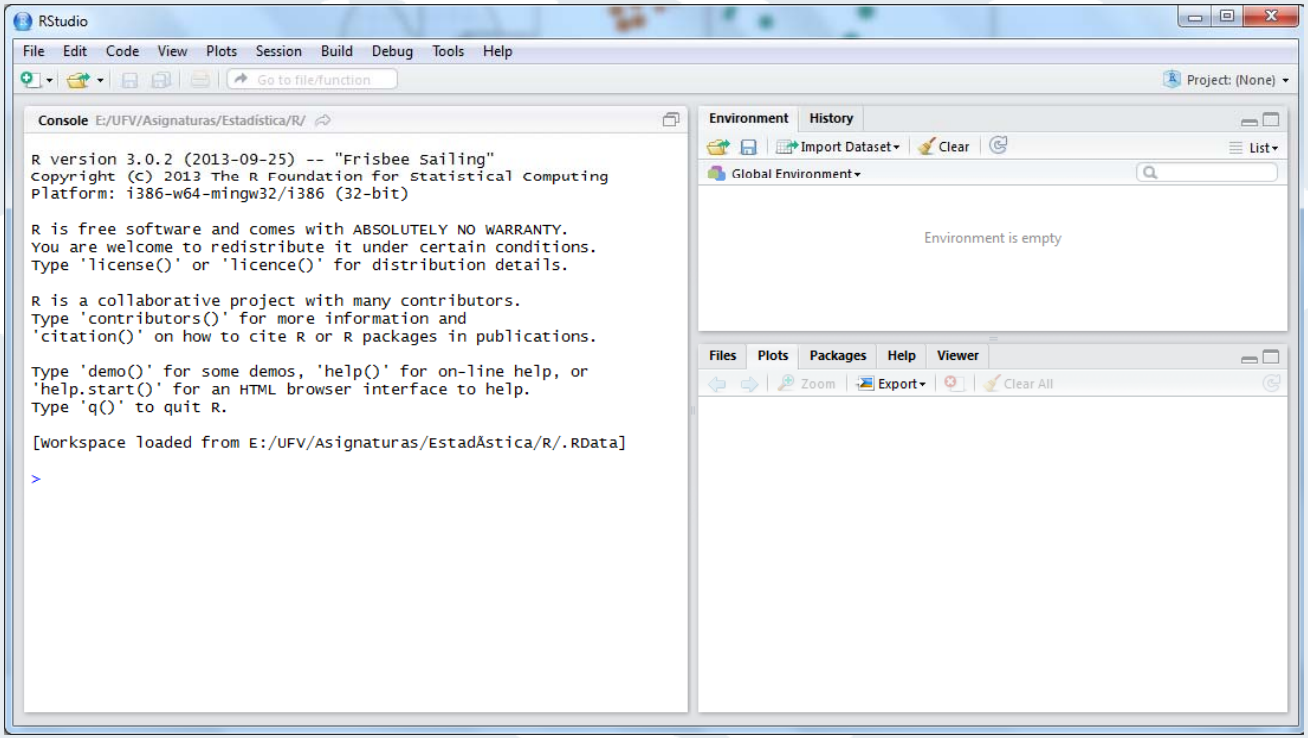


5

Introducción

- **RStudio:** IDE (entorno gráfico) para R, también software libre y multiplataforma (incluidos navegadores web).
- Consola de comandos y ventanas informativas (historial de comandos, variables, gráficos, ayuda, etc.).
- Para trabajar con RStudio necesitamos tener instalado previamente R.
- http://www.rstudio.com

Introducción



Directorio de trabajo

- R guarda en el directorio de trabajo dos ficheros:
  - **.Rhistory** → Historial de comandos ejecutados previamente.
  - **.RData** → Variables y funciones definidas en el entorno.

- Saber el directorio de trabajo actual:

```
> getwd()
[1] "E:/UFV/Asignaturas/Estadística/R"
```

- Cambiar el directorio de trabajo (usando barras hacia la derecha /):

```
> setwd("C:/Temp")
```

- Establecer el directorio de trabajo por defecto: menú Tools, Global Options, Default working directory.

Ayuda

- Iniciar la ayuda (navegador web integrado en RStudio):

```
> help.start()
```

- Iniciar ayuda sobre determinada función (ejemplo: plot):

```
> help(plot)
> ?plot
```

- Buscar todas las menciones a una palabra en la ayuda (ejemplo: mean):

```
> ??mean
```

- Ejecutar los ejemplos de la ayuda de una función (ejemplo: curve):

```
> example(curve)
```

- Ver demos de una función (ejemplo: graphics):

```
> demo(graphics)
```

Paquetes instalados

- R está organizado en diversos paquetes y bibliotecas de funciones.
- Saber qué paquetes están instalados:

```
> library()
```

- Para instalar un nuevo paquete (ejemplo: “moments”):

```
> install.packages(moments)
```

- Saber qué paquetes están cargados en memoria (listos para ejecutar):

```
> search()
```

- Cargar un paquete a memoria (ejemplo: “stats”):

```
> require(stats)
```

Paquetes instalados

```
> library()
```

Packages in library 'C:/Program Files/R/R-3.0.2/library':

base The R Base Package
boot Bootstrap Functions (originally by Angelo Canty for S)
class Functions for Classification
cluster Cluster Analysis Extended Rousseeuw et al.
codetools Code Analysis Tools for R
compiler The R Compiler Package
datasets The R Datasets Package
foreign Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, dBase, ...
graphics The R Graphics Package
grDevices The R Graphics Devices and Support for Colours and Fonts
grid The Grid Graphics Package
KernSmooth Functions for kernel smoothing for Wand & Jones (1995)
lattice Lattice Graphics
manipulate Interactive Plots for RStudio
MASS Support Functions and Datasets for Venables and Ripley's MASS
Matrix Sparse and Dense Matrix Classes and Methods
methods Formal Methods and Classes
mgcv Mixed GAM Computation Vehicle with GCV/AIC/REML smoothness estimation
moments Moments, cumulants, skewness, kurtosis and related tests
nlme Linear and Nonlinear Mixed Effects Models
nnet Feed-forward Neural Networks and Multinomial Log-Linear Models
parallel Support for Parallel computation in R
rpart Recursive Partitioning
rstudio Tools and Utilities for RStudio
spatial Functions for Kriging and Point Pattern Analysis
splines Regression Spline Functions and Classes
stats The R Stats Package
stats4 Statistical Functions using S4 Classes
survival Survival Analysis
tcltk Tcl/Tk Interface
tools Tools for Package Development
utils The R Utils Package

Paquetes instalados

```
> search()
[1] ".GlobalEnv"        "tools:rstudio"
[3] "package:stats"     "package:graphics"
[5] "package:grDevices" "package:utils"
[7] "package:datasets"  "package:methods"
[9] "Autoloads"         "package:base"
```

Clasificación de datos en R

- Según su "modo" (=tipo de valor), los datos en R pueden ser:
  - **Logical** (booleano): con valores posibles TRUE o FALSE.
  - **Numeric** (numérico): para variables cuantitativas. A su vez pueden ser:
    - **Integer** (valores enteros).
    - **Double**(valores reales), por defecto.
  - **Complex** (complejos): para variables complejas (parte real e imaginaria).
  - **Character** (carácter): donde los valores posibles son caracteres separados por comillas.

Clasificación de datos en R

- Según su "tipo" (=estructura), los datos en R pueden ser:
  - **Vector** (vector): conjunto de elementos del mismo modo.
  - **Matrix** (matriz): disposición bidimensional de elementos del mismo modo.
    - Si tiene más de 2 dimensiones, se llama **array**.
  - **Factor** (factor): para variables categóricas (nominales u ordinales).
  - **Data Frame** (estructura de datos): disposición bidimensional de elementos cuyas columnas pueden estar formadas por elementos de distinto modo.
  - **List** (lista): colección arbitraria de datos.

Introducción de datos

- Para asignar un valor a una variable, se usa el operador flecha <- (por compatibilidad con S):

```
> a <- 5
> "Hola" -> b
```

- También se puede usar el operador = (el valor a su derecha se asigna a la variable a la izquierda), pero por defecto usaremos <-:

```
> c = 2.3
> d <- FALSE
> e <- 6+7i
```

- Ver el valor de una variable:

```
> a
[1] 5
> print(a) #Esto es un comentario
[1] 5
```

Tipos de datos

```
> typeof(a) # Un número por defecto es double
[1] "double"
> a <- as.integer(5)
> typeof(a)
[1] "integer"
> typeof(b)
[1] "character"
> typeof(c)
[1] "double"
> typeof(d)
[1] "logical"
> typeof(e)
[1] "complex"
```

Tipos de datos

```
> mode(a)
[1] "numeric"
> mode(b)
[1] "character"
> mode(c)
[1] "numeric"
> mode(d)
[1] "logical"
> mode(e)
[1] "complex"
```

Tipos de datos

- Para cambiar el tipo de dato a entero:

```
> c <- as.integer(c) #Se pierde el decimal
> c
[1] 2
> typeof(c)
[1] "integer"
```

- Para cambiar a otros tipos:

```
> c <- as.double(c) #c vale 2 doble
> f <- "TRUE" #Cadena de caracteres
> typeof(f)
[1] "character"
> f <- as.logical(f) #Lo convertimos a valor lógico
> typeof(f)
[1] "logical"
```

18

Variables cuantitativas

- Creación de variables cuantitativas (vectores numéricos) por teclado:

```
> x <- scan()  #Terminamos con dos intros
1: 3
2: 6
3: 4
4: 0
5: 1
6: 2
7:
Read 6 items
> x
[1] 3 6 4 0 1 2
```

Variables cuantitativas

- Creación de variables cuantitativas (vectores numéricos) por concatenación:

```
> x <- c(3,6,4,0,1,2)
> x
[1] 3 6 4 0 1 2
> x[1] #Los índices empiezan en 1!!!
[1] 3
> length(x) #Número elementos del vector
[1] 6
> x[length(x)] #Último elemento
[1] 2
```

Acceso a elementos de un vector

```
> x
[1] 3 6 4 0 1 2
```

- Por índice numérico (empezando por 1):

```
> x[1] #Devuelve el primer elemento de x
[1] 3
```

- Por índice lógico:

```
> g <- c(FALSE,TRUE,FALSE,TRUE,TRUE,FALSE)
> x[g] #Devuelve los elementos de x donde g es TRUE
[1] 6 0 1
```

- Por índice condicional:

```
> x[x>3] #Devuelve los elementos de x mayores que 3
[1] 6 4
> which(x>3) #Devuelve los índices
[1] 2 3
```

21

Acceso a elementos de un vector

```
> x[2:4] #Desde el elemento 2 al 4
[1] 6 4 0
> x[c(1,2,6)] #Devuelve los elementos
                    1, 2 y 6
[1] 3 6 2
> x[-4] #Devuelve todos menos el 4
[1] 3 6 4 1 2
```

Operaciones con vectores numéricos

- Las operaciones con vectores numéricos se hacen elemento a elemento:

```
> x
[1] 3 6 4 0 1 2
> z <- c(1,-2,0,5,2,3)
> x+z
[1] 4 4 4 5 3 5
> x-z
[1] 2 8 4 -5 -1 -1
> x*z
[1] 3 -12 0 0 2 6
> x/z
[1] 3.00000 -3.00000 Inf 0.00000 0.50000
[6] 0.66667
```

Operaciones con vectores numéricos

- Si uno de los vectores es de menor tamaño, se replica (reutiliza) tantas veces como sea necesario.

```
> x
[1] 3 6 4 0 1 2
> z <- c(1,-2)
> x+z #Es como si z fuera [1 -2 1 -2 1 -2]
[1] 4 4 5 -2 2 0
> x-z
[1] 2 8 3 2 0 4
> x*z
[1] 3 -12 4 0 1 -4
> x/z
[1] 3 -3 4 0 1 -1
```

Operaciones con vectores numéricos

```
> x
[1] 3 6 4 0 1 2
> mean(x) # Media aritmética
[1] 2.666667
> sort(x) # Ordenación
[1] 0 1 2 3 4 6
> median(x) # Mediana (valor central o media de los dos centrales):
[1] 2.5
> max(x) # Máximo
[1] 6
> min(x) # Mínimo
[1] 0
> range(x) #Devuelve un vector con el mínimo y el máximo
[1] 0 6
> range(x)[2]-range(x)[1] #Diferencia entre el máximo y el mínimo
[1] 6
```

Vectores de caracteres

- Creación de vectores de caracteres por concatenación:

```
> y<-c("Madrid","Barcelona","Valencia",
  "Sevilla")
> y
[1] "Madrid" "Barcelona" "Valencia"
  "Sevilla"
> length(y) #Número elementos del vector
[1] 4
> nchar(y)  #Caracteres de cada elemento
[1] 6 9 8 7
```

Estructura de una variable

- Para visualizar de manera compacta la estructura de un objeto de R:

```
> str(x)
num [1:6] 3 6 4 0 1 2
> str(y)
chr [1:4] "Madrid" "Barcelona"
  "Valencia" ...
> str(g)
logi [1:6] FALSE TRUE FALSE TRUE TRUE
  FALSE
```

27

Variables categóricas nominales

- Cuando queremos forzar que una variable sea categórica nominal, la creamos del tipo **factor**.
- Los diferentes valores del factor se denominan **niveles**.

```
> respuesta <- c("S","N","S","S","S","S","N")
> respuesta
[1] "S" "N" "S" "S" "S" "S" "N"
> respuesta <- factor(c("S","N","S","S","S","S","N"))
> respuesta
[1] S N S S S S N
Levels: N S
> str(respuesta)
Factor w/ 2 levels "N","S": 2 1 2 2 2 2 1
```

- **OJO:** Los niveles se ordenan automáticamente de manera alfabética. Por eso a "N" le corresponde el nivel 1 y a "S" le corresponde el nivel 2.
- La conversión de una variable no categórica al tipo factor se realiza com **as.factor**.

Variables categóricas ordinales

- Este tipo de variables categóricas son factores con la propiedad **ordered** igual a TRUE.
- Hay que indicar el orden correcto de los niveles.

```
> notas <- c("MH","SB","AP","AP","AP","NT","AP","AP","AP","AP", "SS","SS",
   "SS","NT","AP","SS")   # Mal!! Se interpreta como un array de caracteres
> notas
 [1] "MH" "SB" "AP" "AP" "AP" "NT" "AP" "AP" "AP" "AP"
[11] "SS" "SS" "SS" "NT" "AP" "SS"
> notas <- factor(c("MH","SB","AP","AP","AP","NT","AP","AP","AP",
   "AP","SS","SS","SS","NT","AP","SS"))
   # Mal!! Factores ordenados alfabéticamente
> notas
[1] MH SB AP AP AP NT AP AP AP AP SS SS SS NT AP SS
Levels: AP MH NT SB SS
> notas <- factor(c("MH","SB","AP","AP","AP","NT","AP","AP","AP","AP","SS",
   "SS","SS","NT","AP","SS"),ordered=TRUE,levels=c("SS","AP","NT","SB","MH"))
  # Bien!!
> notas
[1] MH SB AP AP AP NT AP AP AP AP SS SS SS NT AP SS
Levels: SS < AP < NT < SB < MH
```

Operaciones con variables categóricas ordinales

- Como existe el orden, tiene sentido la noción de mínimo, máximo, rango.

```
> max(notas)
[1] MH
Levels: SS < AP < NT < SB < MH
> min(notas)
[1] SS
Levels: SS < AP < NT < SB < MH
> range(notas)
[1] SS MH
Levels: SS < AP < NT < SB < MH
```

Conversión de tipo de variable

- De variable categórica a cuantitativa (**as.integer**):

```
> respuesta  # Variable categórica nominal
[1] S N S S S S N
Levels: N S
> as.integer(respuesta) # Conversión a variable cuantitativa
[1] 2 1 2 2 2 2 1

> notas  # Variable categórica ordinal
[1] MH SB AP AP AP NT AP AP AP AP SS SS SS NT AP SS
Levels: SS < AP < NT < SB < MH
> as.integer(notas) # Conversión a variable cuantitativa
[1] 5 4 2 2 2 3 2 2 2 2 1 1 1 3 2 1
```

Conversión de tipo de variable

- De variable cuantitativa a categórica (**cut**):

```
> x <- runif(20,0,10)  # Generamos 20 números aleatorios
  entre 0 y 10 (distribución de probabilidad uniforme)
> x
 [1] 3.6372178 9.2116284 9.9038358 0.9522409 2.3131724
 [6] 7.0095696 7.5791517 7.8860561 0.6972082 9.9543938
[11] 8.3144410 4.0522823 6.6169572 6.6007821 3.9583144
[16] 5.7719143 9.5096915 8.1517746 3.0302474 2.8090360
> x2 <- cut(x,breaks=c(0,5,7,9,10,10.1),
  labels=c("SS","AP","NT","SB","MH"),right=FALSE,
  ordered_result=TRUE)
> x2
 [1] SS SB SB SS SS NT NT NT SS SB NT SS AP AP SS AP SB
[18] NT SS SS
Levels: SS < AP < NT < SB < MH
```

Distribuciones de frecuencias

- Para variables cuantitativas:

```
> notas2 <- c(10,9,5,5,6,8,5,6,6,5,3,2,0,8,5,3)
> table(notas2) #Frecuencias absolutas
notas2
0 2 3 5 6 8 9 10
1 1 2 5 3 2 1  1
> table(notas2)/length(notas2) #Relativas
notas2
     0      2      3      5      6      8      9     10
0.0625 0.0625 0.1250 0.3125 0.1875 0.1250 0.0625 0.0625
```

Distribuciones de frecuencias

- Frecuencias absolutas y relativas acumuladas:

```
> cumsum(table(notas2))
0 2 3 5  6  8  9 10
1 2 4 9 12 14 15 16
> cumsum(table(notas2)/length(notas2))
     0      2      3      5      6      8      9     10
0.0625 0.1250 0.2500 0.5625 0.7500 0.8750 0.9375 1.0000
```

- Obviamente para no estar repitiendo todos los cálculos, los resultados intermedios pueden asignarse a variables reutilizables. Ejemplo:

```
> frecAbs <- table(notas2)
> cumsum(frecAbs)
```

Distribuciones de frecuencias

- Para variables categóricas nominales:

```
> respuesta #Definido previamente como factores
[1] S N S S S S N
Levels: N S
> table(respuesta) #Frecuencias absolutas
respuesta
N S
2 5
> table(respuesta)/length(respuesta) #Frecuencias relativas
respuesta
        N         S
0.2857143 0.7142857
> cumsum(table(respuesta)) #Absolutas acumuladas
N S
2 7
> cumsum(table(respuesta)/length(respuesta)) #Relativas
  acumuladas
        N         S
0.2857143 1.0000000
```

Distribuciones de frecuencias

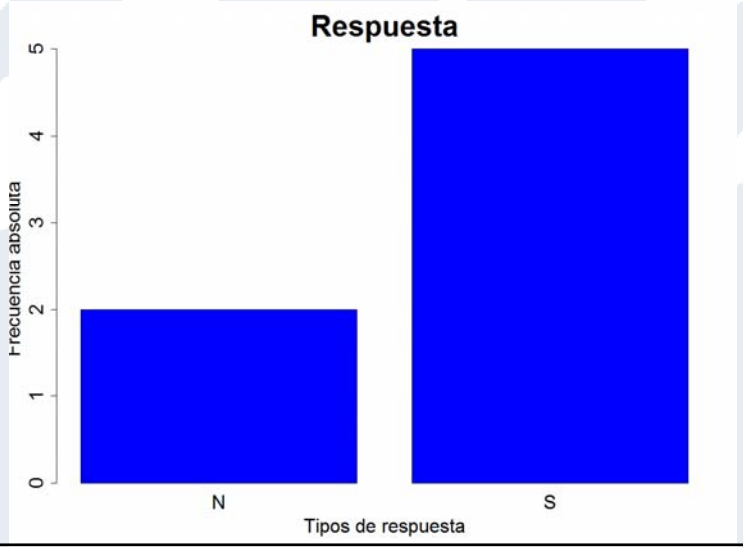
- Para variables categóricas ordinales:

```
> notas #Definido previamente como factores ordinales
[1] MH SB AP AP AP NT AP AP AP AP SS SS SS NT AP SS
Levels: SS < AP < NT < SB < MH
> table(notas) #Frecuencias absolutas
notas
SS AP NT SB MH
 4  8  2  1  1
> table(notas)/length(notas) #Frecuencias relativas
notas
    SS     AP     NT     SB     MH
0.2500 0.5000 0.1250 0.0625 0.0625
> cumsum(table(notas)) #Absolutas acumuladas
SS AP NT SB MH
 4 12 14 15 16
> cumsum(table(notas)/length(notas)) #Relativas acumuladas
    SS     AP     NT     SB     MH
0.2500 0.7500 0.8750 0.9375 1.0000
```

Gráficos básicos: diagrama de barras

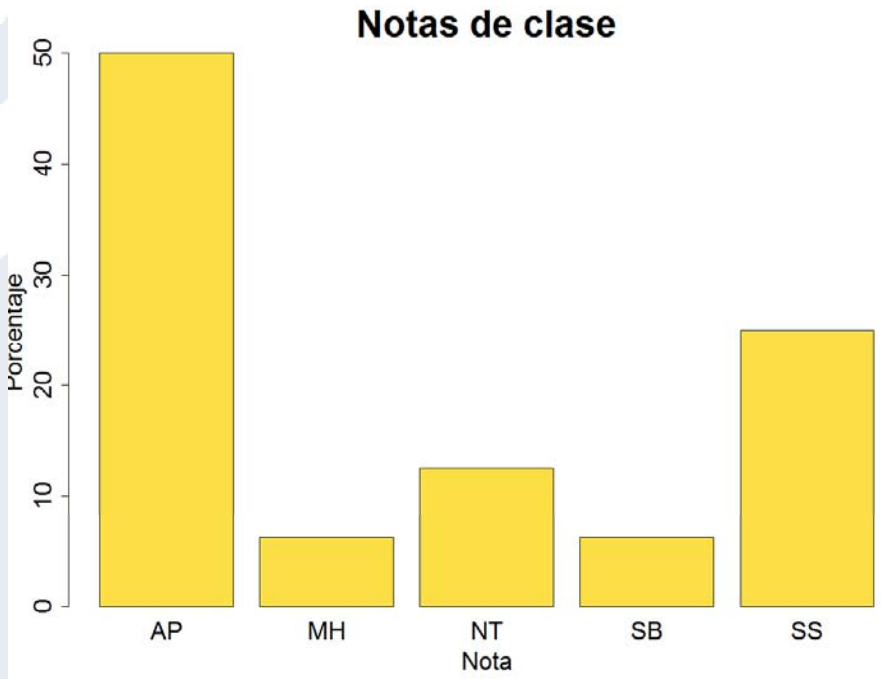
- Diagrama de barras para datos categóricos:

```
> barplot(table(respuesta)) #Diagrama de frecuencias absolutas
> barplot(table(respuesta),col="blue",xlab="Tipos de
  respuesta",ylab="Frecuencia absoluta",main="Respuesta")
  #En color azul, con etiquetas en los ejes X e Y, y título de
  figura
```



Gráficos básicos: diagrama de barras

```
> barplot(table(notas)*100/length(notas),col="#fcde45",
  main="Notas de clase",xlab="Nota",ylab="Porcentaje")
  # Frecuencia relativa en % y color en notación RGB
```

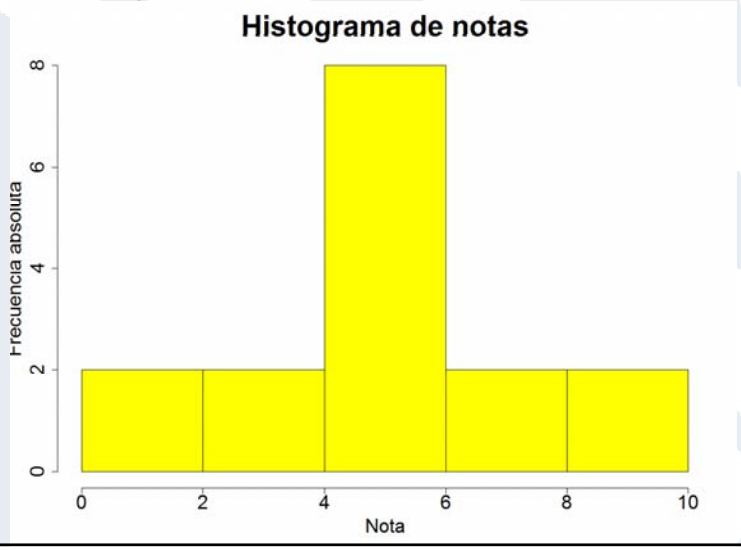


38

Gráficos básicos: histograma

- Histograma para variables numéricas:

```
> hist(notas2) #El nº de intervalos es automático
> hist(notas2,col="yellow",xlab="Nota", ylab="Frecuencia
  absoluta",main="Histograma de notas")
  # En color amarillo, con etiquetas en los ejes X e Y, y título
  de figura
```



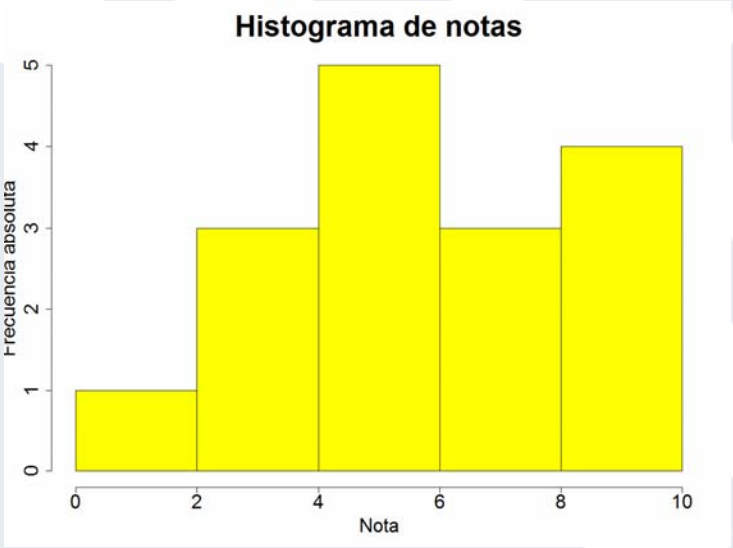
Fronteras de los intervalos (por defecto: incluida la frontera derecha):

[0,2]
(2,4]
(4,6]
(6,8]
(8,10]

Gráficos básicos: histograma

- Para no incluir la frontera derecha de cada intervalo:

```
> hist(notas2,col="yellow",xlab="Nota", ylab="Frecuencia
  absoluta",main="Histograma de notas",right=FALSE)
```



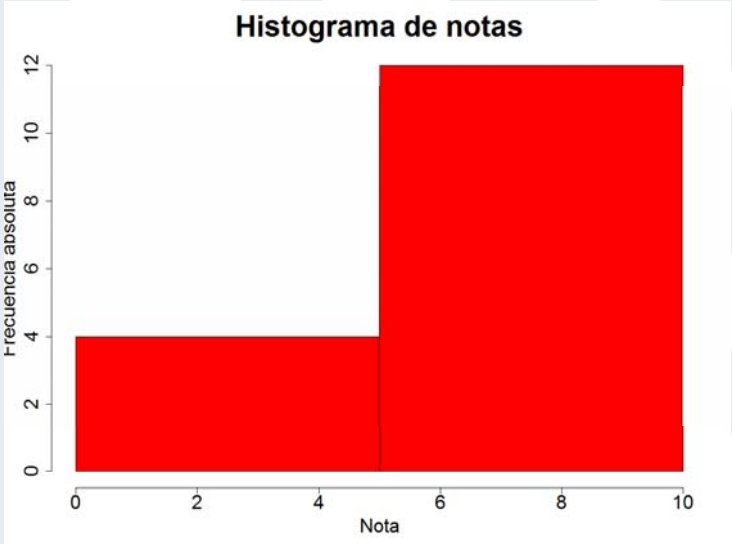
Fronteras de los intervalos:

[0,2)
[2,4)
[4,6)
[6,8)
[8,10]

Gráficos básicos: histograma

- Establecer a mano las fronteras de los intervalos:

```
> hist(notas2,breaks=c(seq(0,10,5)),col="red",xlab="Nota",
  ylab="Frecuencia absoluta",main="Histograma de
  notas",right=FALSE)
```



`seq(0,10,5)` genera la secuencia numérica:
`0 5 10`

Notación: `seq(inicio,fin,salto)`

Si el salto es 1, se usa ":"

`0:10` genera la secuencia numérica:
`0 1 2 3 4 5 6 7 8 9 10`

Notación: `inicio:fin`

41

Gráficos básicos: histograma

- Intervalos de anchura variable: la altura de las barras se ajusta automáticamente para que el área sea proporcional a la frecuencia relativa.

```
> h <- hist(notas2,breaks=c(0,5,7,9,10),col="green",
  xlab="Nota",ylab="Frecuencia relativa",main="Histograma de
  notas",right=FALSE) #Lo asignamos a variable h
```

Intervalo [0,5)
Anchura = 5
Frec.abs. = 4
Frec.rel. = 4/16=0.25
Altura de la barra = 0.25/5 = 0.05

Gráficos básicos: histograma

```
> str(h) #Estructura compacta del objeto devuelto por el
  histograma
List of 6
 $ breaks : num [1:5] 0 5 7 9 10    #Fronteras de intervalos
 $ counts : int [1:4] 4 8 2 2       #Frec.absolutas
 $ density : num [1:4] 0.05 0.25 0.0625 0.125  #Frec. relativas
 $ mids : num [1:4] 2.5 6 8 9.5     #Marcas de clase
 $ xname : chr "notas2"
 $ equidist: logi FALSE             #Igual anchura falso
 - attr(*, "class")= chr "histogram“
> h$mids # Accedo a un campo del objeto h con el operador $
  (notación de Programación Orientada a Objetos)
[1] 2.5 6.0 8.0 9.5
```

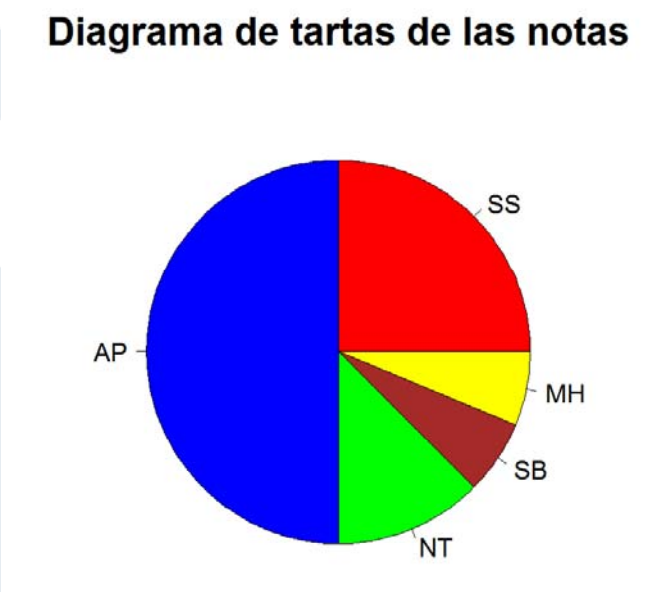
Gráficos básicos: diagrama de tallo y hojas

- Variante del histograma para variables cuantitativas:
  - Útil cuando tenemos pocos datos.
  - Vemos los datos agrupados según las primeras cifras (tallo), seguidas de una barra |.
  - Las hojas son la última cifra y se colocan ordenadas de menor a mayor después de la barra.

```
> notas
[1] 8.0 6.5 2.0 5.0 6.6 7.1 5.3 9.5 6.3 4.5 6.1
> stem(notas,scale=2) # Probar a quitar el parámetro scale
The decimal point is at the |
 2 | 0
 3 |
 4 | 5
 5 | 03
 6 | 1356
 7 | 1
 8 | 0
 9 | 5
```

Gráficos básicos: diagrama de tartas

```
> pie(table(notas)) #Los colores elegidos son horrorosos
> pie(table(notas),main="Diagrama de tartas de las
  notas",col=c("red","blue","green","brown","yellow"))
  # Con título de figura y vector de colores
```



Parámetros básicos de los gráficos

`col`	Color (puede ser un valor constante o un vector)
`xlab`	Etiqueta del eje X
`ylab`	Etiqueta del eje Y
`main`	Título del gráfico
`cex`	Valor numérico que da el factor de amplificación de los símbolos usados en el gráfico (defecto = 1)
`cex.lab`	Factor de amplificación de las etiquetas de ejes X e Y
`cex.main`	Factor de amplificación del título del gráfico
`cex.axis`	Factor de amplificación de los valores en los ejes