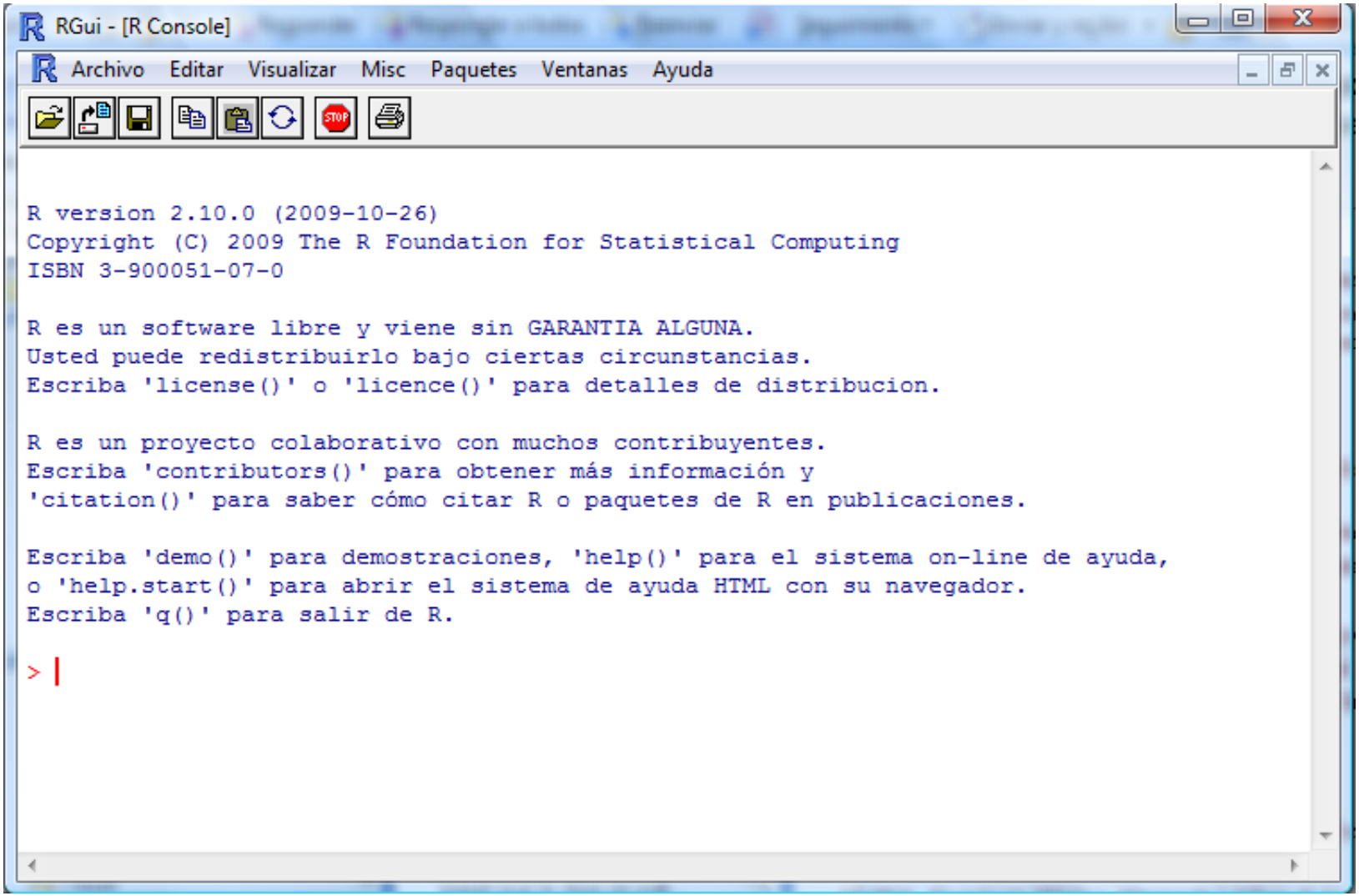
Tutorial de 

Javier Ramírez Pérez de Inestrosa
Departamento de Teoría de la Señal, Telemática y Comunicaciones
Universidad de Granada

Instalación de R

- Descarga de: http://www.r-project.org/
- Manuales: http://cran.r-project.org/manuals.html
- Paquetes: http://cran.r-project.org/web/packages/
  - Rwave — Time-Frequency analysis of 1-D signals
  - signalextraction — Real-Time Signal Extraction (Direct Filter Approach)
  - wavelets — A package of funtions for computing wavelet filters, wavelet transforms and multiresolution analyses
  - waveslim — Basic wavelet routines for one-, two- and three-dimensional signal processing
  - FKF — Fast Kalman Filter
  - KFAS — Multivariate Kalman filter and smoother, simulation smoother and forecasting of state space models. State smoothing and approximate likelihood of exponential family state space models
  - robfilter — Robust Time Series Filters
  - sapa — Insightful Spectral Analysis for Physical Applications
  - biOps — Image processing and analysis
  - biOpsGUI — GUI for Basic image operations
  - PET — Simulation and Reconstruction of PET Images
  - ReadImages — Image Reading Module for R
  - rimage — Image Processing Module for R
  - ripa — R Image Processing and Analysis
  - class — Functions for Classification

Entorno de trabajo



Demostraciones y Ayuda

- demo()
- Para ver todas las demostraciones incluidas en los paquetes:
  - demo(package = .packages(all.available = TRUE))
- Demostraciones interesantes:
  - demo(graphics)
  - demo(image)
- Obtención de ayuda:
  - help()
  - help(qnorm)

Vectores y asignaciones

- x <- c(10.4, 5.6, 3.1, 6.4, 21.7) % Asignación de vectores.
- x[1] % Indexado de vectores

  [1] 10.4

Alternativas:

- x = c(10.4, 5.6, 3.1, 6.4, 21.7) % =
- **assign**("x", c(10.4, 5.6, 3.1, 6.4, 21.7)) % Comando assign
- c(10.4, 5.6, 3.1, 6.4, 21.7) -> x % Asignación en dirección opuesta.

- y <- c(x, 0, x)

[1] 10.4 5.6 3.1 6.4 21.7 0.0 10.4 5.6 3.1 6.4 21.7

Operaciones con vectores

- Operadores usuales: +, -, *, / y ^
- Funciones aritméticas: log, exp, sin, cos, tan, sqrt, etc.
- Máximo, mínimo y rango: max, min, range equiv. a c(min(x), max(x))
- Longitud: length
- Producto y suma prod, sum
- Media y varianza mean, var
  - Si la entrada de var es una matriz nxp var calcula la matriz de covarianza (pxp)
- Ordenación sort
- Números complejos

Generación de secuencias regulares

- 1:10 Secuencia creciente
- Prioridades: 2*1:15 [1] 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
- Ejercicio: n<-10 1:n-1 1:(n-1)
- 30:1 Secuencia decreciente
- **Función seq**(from=value,to=value,by=value,length=value,along=*vector*)

  Ejemplos: seq(1,30), seq(from=1, to=30), seq(to=30, from=1)

  seq(-5, 5, by=.2) -> s3

  s4 <- seq(length=51, from=-5, by=.2)
- **Función rep**()

  Ejemplos: s5 <- rep(x, times=5), s6 <- rep(x, each=5)

Operaciones lógicas

- temp <- x > 13 %Toma valores FALSE and TRUE
- Operaciones lógicas: <, <=, >, >=, == y !=
- AND(&), OR (|), NOT (!)

Redimensionado e indexación de matrices

- Redimensionado de vectores:
  - Si z es un vector de 1500 elementos
    - dim(z) <- c(3,5,100) Redimensiona z a un vector de 3x5x100.
- Indexación z[, ,]
- Ejemplo:
  - z= 1:1500
  - dim(z) <- c(3,5,100)
  - dim(z) % Salida: [1] 3 5 100
  - z[2,4,2] % Salida: [1] 26
  - z[2,,] % Submatriz de 5x100 elementos de z.

Indexación de matrices

- Ejemplo: X es una matriz 4x5 y se desea:
  - 1) Extraer X[1,3], X[2,2] y X[3,1] y
  - 2) reemplazar estos valores por 0
- x <- array(1:20, dim=c(4,5))
- x
- i <- array(c(1:3,3:1), dim=c(3,2))
- i
- x[i] <- 0
- x

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

```
     [,1] [,2]
[1,]    1    3
[2,]    2    2
[3,]    3    1
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    0   13   17
[2,]    2    0   10   14   18
[3,]    0    7   11   15   19
[4,]    4    8   12   16   20
```

Construcción de matrices

- Función matrix()
  - A <- matrix(0, 10, 5) % Crea una matriz A de ceros de 10x5.
- Creación de matrices a través de vectores: Z <- array(data_vector, dim_vector)
  - h= 1:30
  - Z <- array(h, dim=c(3,4,2))
  - Z
  - Equivalente a
  - Z <- h ; dim(Z) <- c(3,4,2)
  - Si la longitud de h es inferior a 24 se reutilizarían sus valores
  - h= 1:10
  - Z <- array(h, dim=c(3,4,2))

```
, , 1

     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

, , 2

     [,1] [,2] [,3] [,4]
[1,]   13   16   19   22
[2,]   14   17   20   23
[3,]   15   18   21   24
```

```
, , 1

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

, , 2

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

cbind y rbind

- **cbind** Crea matrices por concatenación horizontal
- X <- cbind(*arg_1*, *arg_2*, *arg_3*, ...)
- Los argumentos pueden ser vectores, o matrices del mismo número de cols.
  - A <- matrix(0, 3, 3)
  - B <- matrix(1, 3, 3)
  - X= cbind(A,B)
  - v= 1:3
  - cbind(v,v,v)
- **rbind** Crea matrices por concatenación vertical
  - Y= rbind(A,B)
  - rbind(v,v,v)

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    0    0    0    1    1    1
[2,]    0    0    0    1    1    1
[3,]    0    0    0    1    1    1
```

```
     v v v
[1,] 1 1 1
[2,] 2 2 2
[3,] 3 3 3
```

```
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
[3,]    0    0    0
[4,]    1    1    1
[5,]    1    1    1
[6,]    1    1    1
```

```
  [,1] [,2] [,3]
v    1    2    3
v    1    2    3
v    1    2    3
```

Operaciones con matrices

- Las operaciones se realizan componente a componente:
- A= matrix(1,3,3)
- C= matrix(2,3,3)
- C= matrix(3,3,3)
- Z= A*B+2*C+1

```
> A
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1
[3,]    1    1    1
> B
     [,1] [,2] [,3]
[1,]    2    2    2
[2,]    2    2    2
[3,]    2    2    2
> C
     [,1] [,2] [,3]
[1,]    3    3    3
[2,]    3    3    3
[3,]    3    3    3
> Z= A*B+2*C+1
>
> Z
     [,1] [,2] [,3]
[1,]    9    9    9
[2,]    9    9    9
[3,]    9    9    9
```

Operaciones con matrices

- Outer product: Operador %o%
  - Contiene todos los posibles productos de los dos vectores.
  - x= 1:10 y= 10:1
  - xy <- x %o% y
  - xy

    Equivalente a:
  - xy <- outer(x, y, "*")

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]   10    9    8    7    6    5    4    3    2     1
 [2,]   20   18   16   14   12   10    8    6    4     2
 [3,]   30   27   24   21   18   15   12    9    6     3
 [4,]   40   36   32   28   24   20   16   12    8     4
 [5,]   50   45   40   35   30   25   20   15   10     5
 [6,]   60   54   48   42   36   30   24   18   12     6
 [7,]   70   63   56   49   42   35   28   21   14     7
 [8,]   80   72   64   56   48   40   32   24   16     8
 [9,]   90   81   72   63   54   45   36   27   18     9
[10,]  100   90   80   70   60   50   40   30   20    10
```

  - Es útil en la evaluación de funciones bidimensionales en 1 grid 2D:
  - f <- function(x, y) cos(y)/(1 + x^2)
  - z <- outer(x, y, f)

Producto de matrices y sistemas de ecuaciones

- A*B Calcula el producto componente a componente.
- Producto de matrices: A %*% B
- Elementos de la diagonal:
  - diag(v) v vector Construye una matriz diagonal
    - x=1:10 diag(x)
  - diag(M) M matriz Extrae los elementos de la diagonal de M.
    - x= 1:10 y=1:10 xy <- outer(x, y, "*") diag(xy)
- Resolución de sistemas de ecuaciones lineales: Ax= b
  - x= solve(A,b)

Autovalores y autovectores de una matriz

- ev <- eigen(Sm)
  - Sm= c(1, sqrt(2), sqrt(2), 1)
  - dim(Sm)<-c(2,2)

```
> ev<- eigen(Sm)
> ev$val
[1]  2.4142136 -0.4142136
> ev$vec
           [,1]       [,2]
[1,] 0.7071068 -0.7071068
[2,] 0.7071068  0.7071068
```

- Singular value decomposition: $M= UDV^T$
  - Si M es cuadrada, el determinante:

    absdetM <- prod(svd(M)$d)
  - Como función:
    - absdet <- function(M) prod(svd(M)$d)

```
> svd(Sm)
$d
[1] 2.4142136 0.4142136

$u
           [,1]       [,2]
[1,] -0.7071068 -0.7071068
[2,] -0.7071068  0.7071068

$v
           [,1]       [,2]
[1,] -0.7071068  0.7071068
[2,] -0.7071068 -0.7071068
```

Funciones de probabilidad

- R evalúa:
  - Función de distribución acumulada: Función de x

    P(X <= x)

    - Prefijo: p Ej:. pnorm(…)
  - Función densidad de probabilidad:
    - Prefijo d Ej.: dnorm(…)
  - Función cuantil: Función de q (valor de probabilidad)

    El valor más pequeño de x tal que
    P(X <= x) > q para cada q.

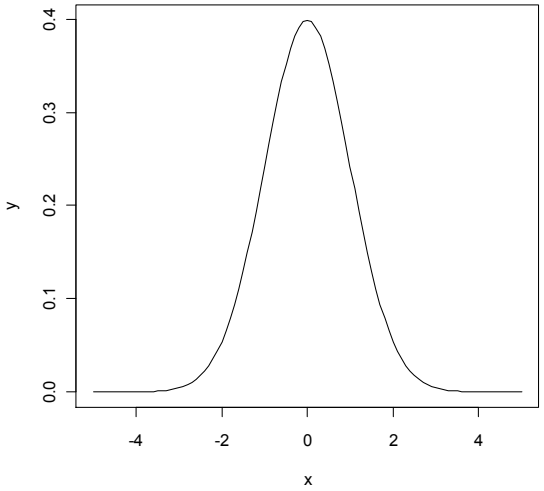
    - Prefijo: q Ej,: qnorm(…)

Distribution	R name	additional arguments
beta	beta	shape1, shape2, ncp
binomial	binom	size, prob
Cauchy	cauchy	location, scale
chi-squared	chisq	df, ncp
exponential	exp	rate
F	f	df1, df2, ncp
gamma	gamma	shape, scale
geometric	geom	prob
hypergeometric	hyper	m, n, k
log-normal	lnorm	meanlog, sdlog
logistic	logis	location, scale
negative binomial	nbinom	size, prob
normal	norm	mean, sd
Poisson	pois	lambda
Student's t	t	df, ncp
uniform	unif	min, max
Weibull	weibull	shape, scale
Wilcoxon	wilcox	m, n

Funciones de probabilidad (ejemplos)

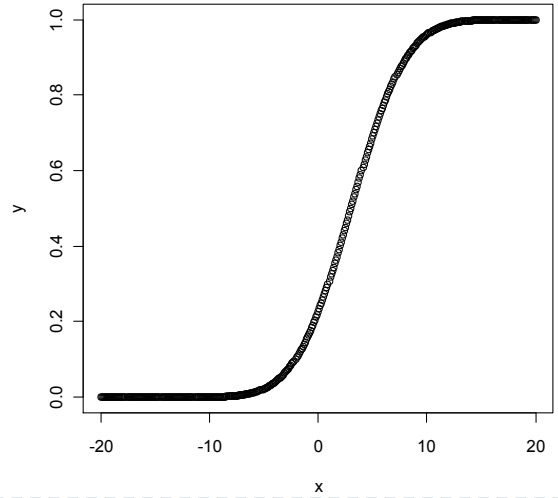
- Función densidad (distr. Normal):
  - x <- seq(-10,10,by=.1)
  - y <- dnorm(x)
  - plot(x,y,type = "l")

  - y <- dnorm(x,mean=2.5,sd=0.1)
  - plot(x,y,type = "l")



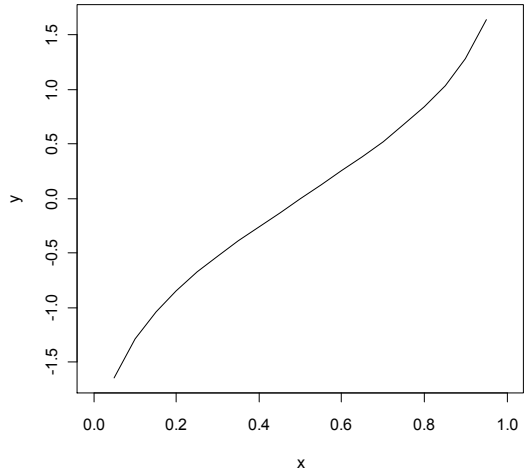
- Función distribución acumulada:
  - x <- seq(-20,20,by=.1)
  - y <- pnorm(x)
  - plot(x,y)

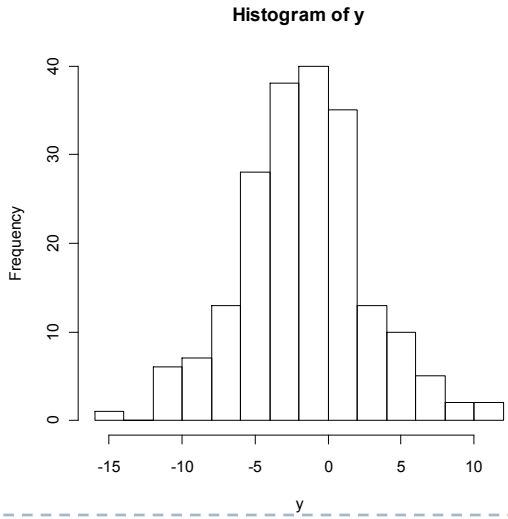
  - y <- pnorm(x,mean=3,sd=4)
  - plot(x,y)

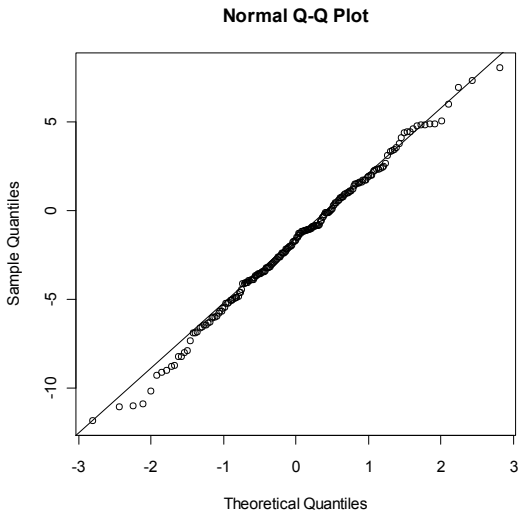


Funciones de probabilidad (ejemplos)

- Función cuantil qnorm()
  - x <- seq(0,1,by=.05)
  - y <- qnorm(x)
  - plot(x,y)
  - y <- qnorm(x,mean=3,sd=2)
  - plot(x,y)
  - y <- qnorm(x,mean=3,sd=0.1)
  - plot(x,y)
- Generación de números aleatorios:
  - y <- rnorm(200)
  - hist(y)
  - y <- rnorm(200,mean=-2)
  - hist(y)
  - y <- rnorm(200,mean=-2,sd=4)
  - hist(y)
  - qqnorm(y)
  - qqline(y)

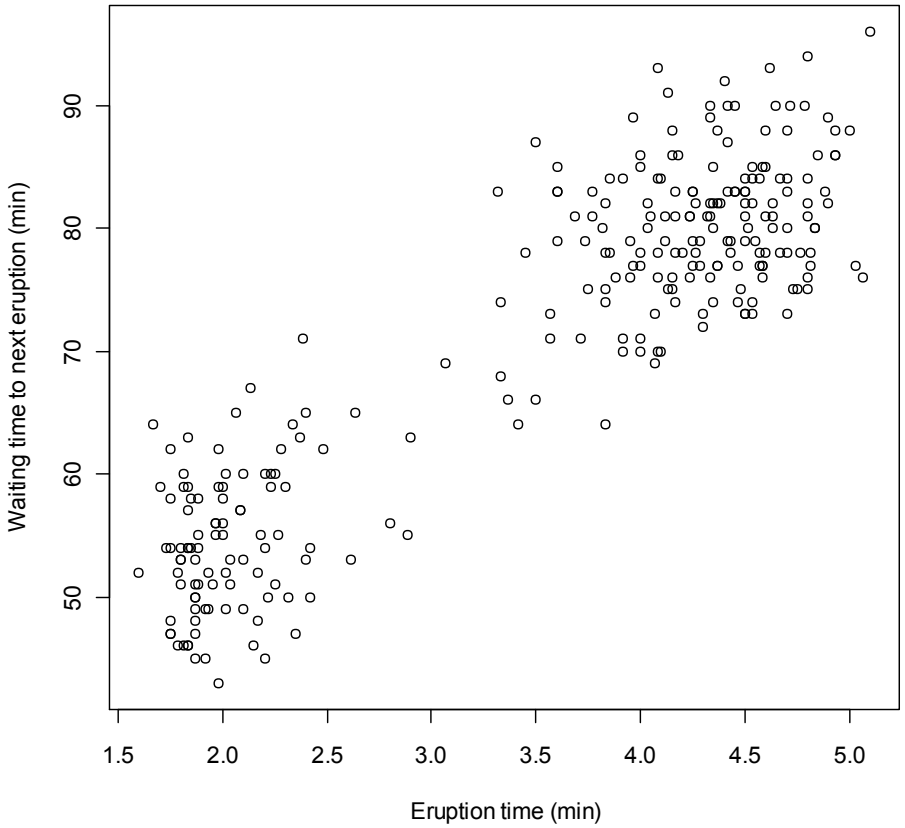






Ejemplo: análisis de datos

- faithful dataset:
  - **Waiting time** between eruptions and the **duration** of the eruption for the Old Faithful geyser in Yellowstone National Park, Wyoming, USA.
  - A data frame with 272 observations on 2 variables.
    - [,1] eruptions numeric Eruption time in mins
    - [,2] waiting numeric Waiting time to next eruption (in mins)



Ejemplo: análisis de datos

```
A data frame with 272 observations on 2 variables.

      [,1]  eruptions  numeric  Eruption time in mins
      [,2]  waiting    numeric  Waiting time to next
                                eruption (in mins)
```

- require(stats); require(graphics)
- f.tit <- "faithful data: Eruptions of Old Faithful"
- ne60 <- round(e60 <- 60 * faithful$eruptions)
- all.equal(e60, ne60) # relative diff. ~ 1/10000
- table(zapsmall(abs(e60 - ne60))) # 0, 0.02 or 0.04
- faithful$better.eruptions <- ne60 / 60
- te <- table(ne60)
- te[te >= 4] # (too) many multiples of 5 !
- plot(names(te), te, type="h", main = f.tit, xlab = "Eruption time (sec)")
- plot(faithful[, -3], main = f.tit, xlab = "Eruption time (min)", ylab = "Waiting time to next eruption (min)")
- lines(lowess(faithful$eruptions, faithful$waiting, f = 2/3, iter = 3), col = "red")

```
> faithful
    eruptions waiting
1       3.600      79
2       1.800      54
3       3.333      74
4   2.283      62

    .....


270     4.417      90
271     1.817      46
272     4.467      74
```

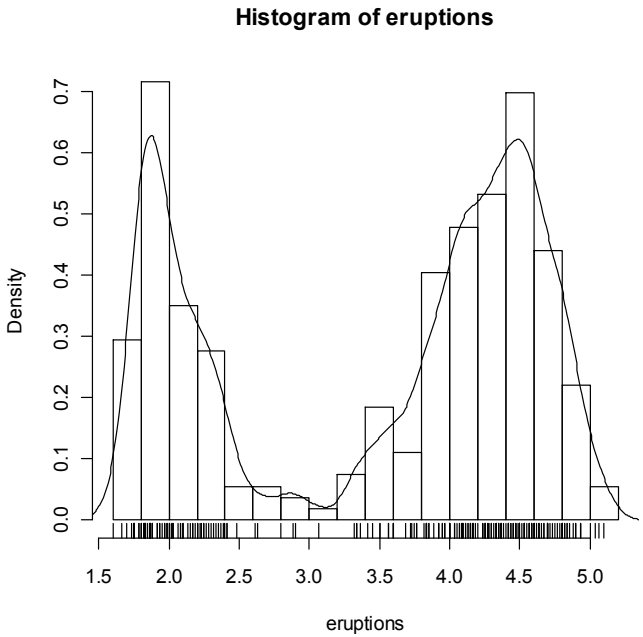
Ejemplo: análisis de datos

- attach(faithful)
- summary(eruptions)

```
Min. 1st Qu. Median Mean 3rd Qu. Max.
1.600 2.163 4.000 3.488 4.454 5.100
```

- fivenum(eruptions)

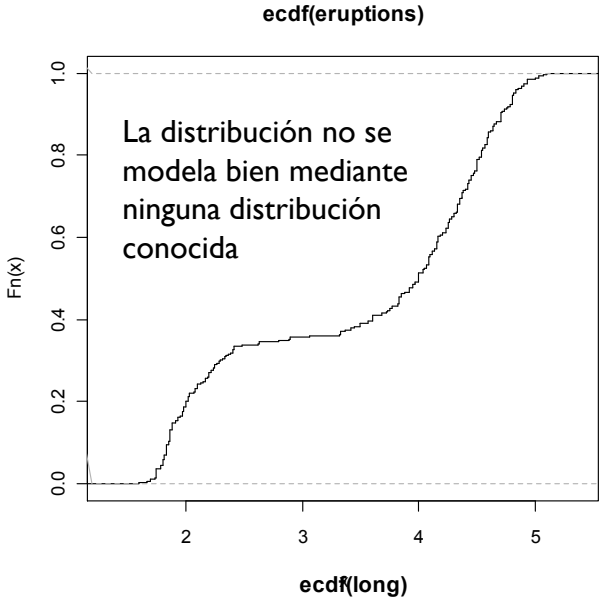
```
[1] 1.6000 2.1585 4.0000 4.4585 5.1000
```



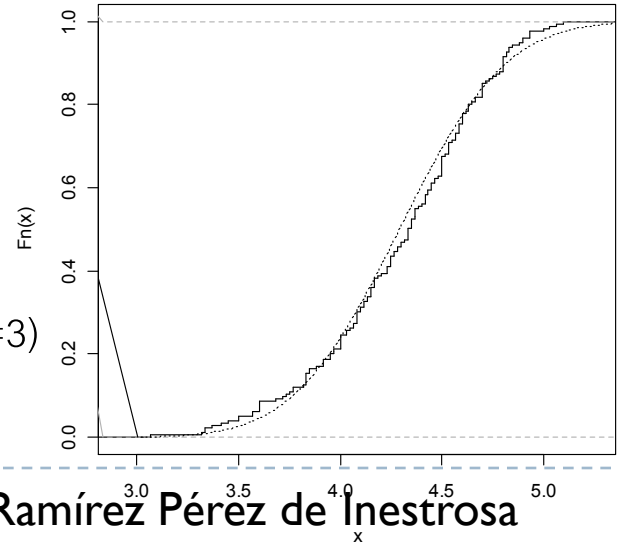
- stem(eruptions)
- hist(eruptions) ## make the bins smaller, make a plot of density
- hist(eruptions, seq(1.6, 5.2, 0.2), prob=TRUE)
- lines(density(eruptions, bw=0.1))
- rug(eruptions) # show the actual data points

Ejemplo: análisis de datos

- plot(ecdf(eruptions), do.points=FALSE, verticals=TRUE)

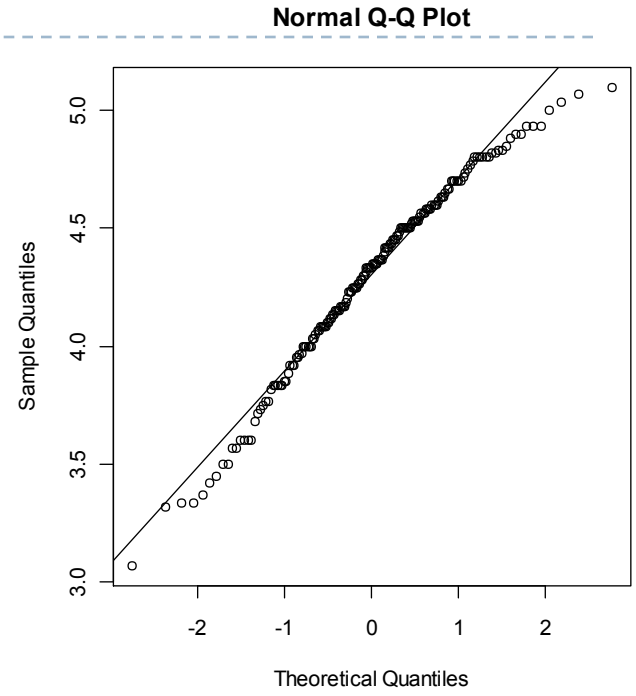


- long <- eruptions[eruptions > 3]
- plot(ecdf(long), do.points=FALSE, verticals=TRUE)
- x <- seq(3, 5.4, 0.01)
- lines(x, pnorm(x, mean=mean(long), sd=sqrt(var(long))), lty=3)

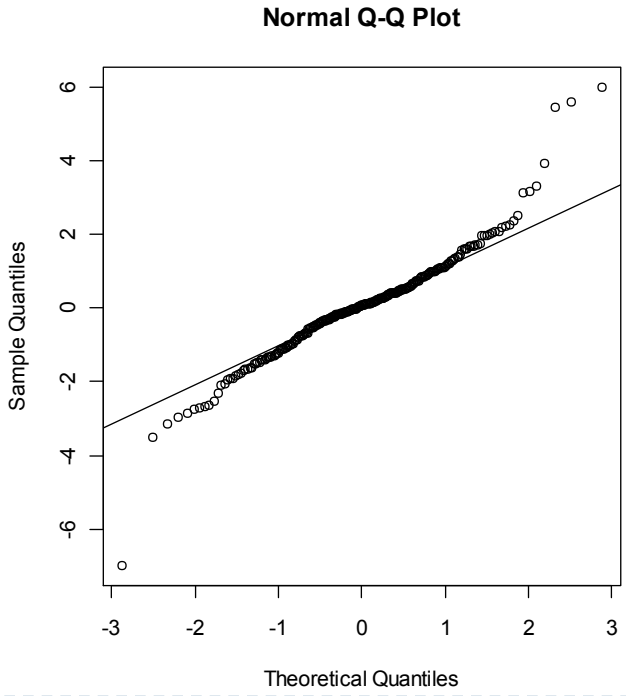


Ejemplo: análisis de datos

- par(pty="s")
- qqnorm(long); qqline(long)



- x <- rt(250, df = 5)
- qqnorm(x)
- qqline(x)



Programación (bucles)

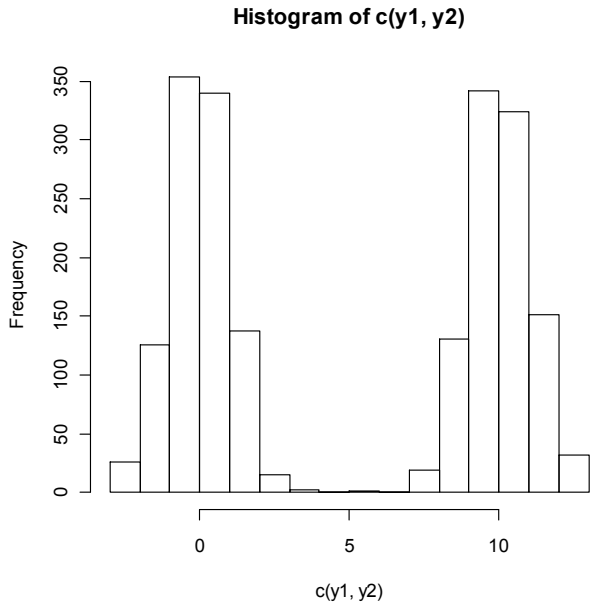
- Agrupación de comandos:
  - { *expr_1*; … ; *expr_m* }
- Ejecución condicional:
  - if (*expr_1*) *expr_2* else *expr_3*
- Bucles: for, repeat, while
  - for (*name* in *expr_1*) *expr_2*
  - repeat *expr* % Se finaliza con break
  - while (*condition*) *expr*

Programación (funciones)

- *name* <- function(*arg_1*, *arg_2*, ...) *expression*

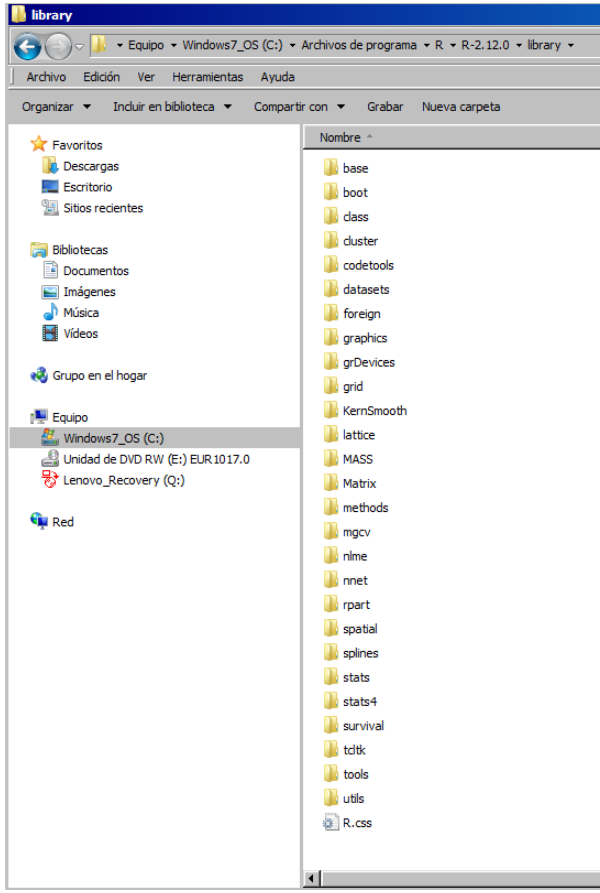
- Ejemplo: *t*-test de dos muestras.
  - Definición:
  - twosam <- function(y1, y2) {
  - n1 <- length(y1); n2 <- length(y2)
  - yb1 <- mean(y1); yb2 <- mean(y2)
  - s1 <- var(y1); s2 <- var(y2)
  - s <- ((n1-1)*s1 + (n2-1)*s2)/(n1+n2-2)
  - tst <- (yb1 - yb2)/sqrt(s*(1/n1 + 1/n2))
  - tst
  - }

  - Llamada:
  - y1= rnorm(1000,0,1)
  - y2= rnorm(1000,10,1)
  - tstat <- twosam(y1,y2); tstat
  - hist(c(y1,y2))

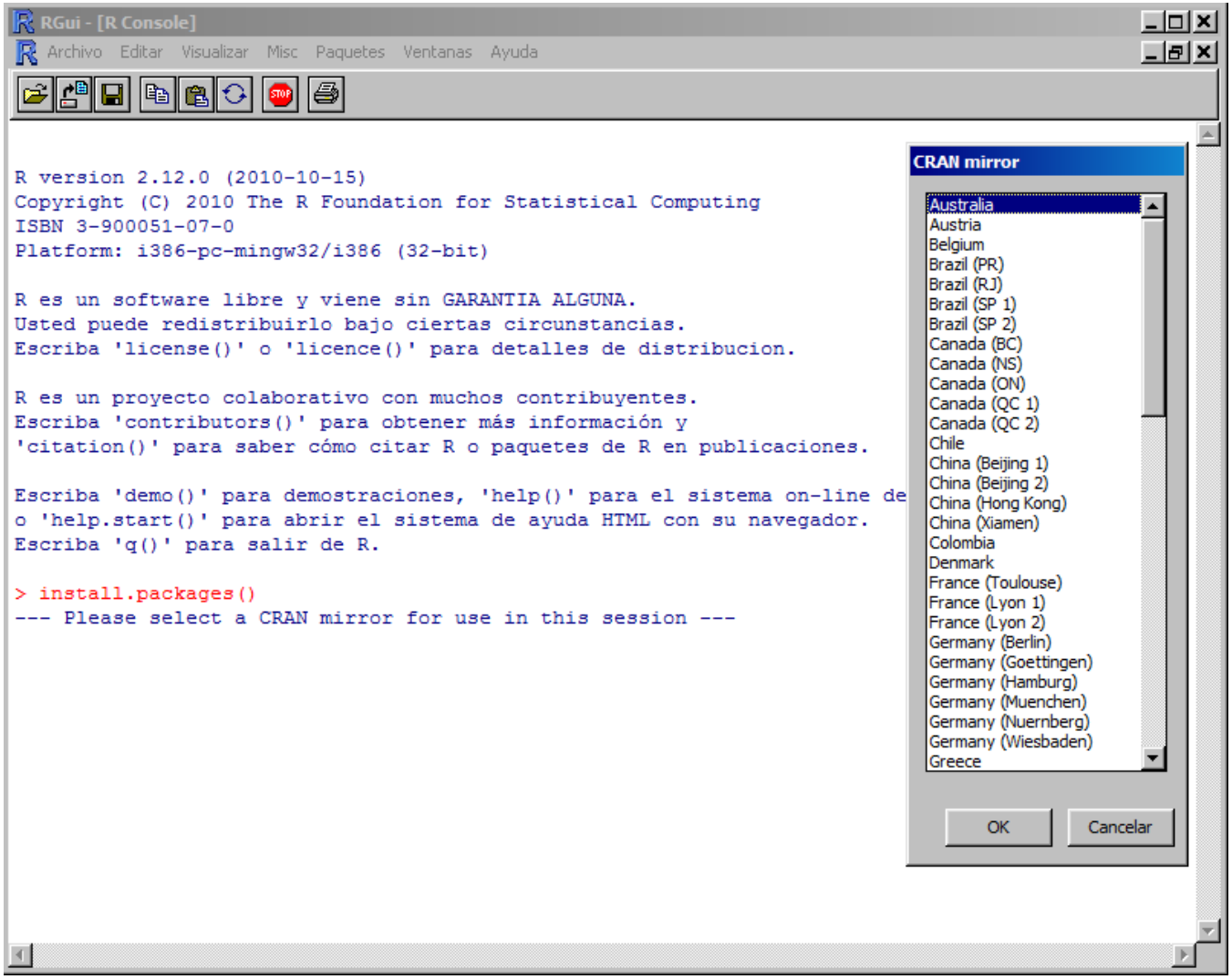


Instalación de Paquetes

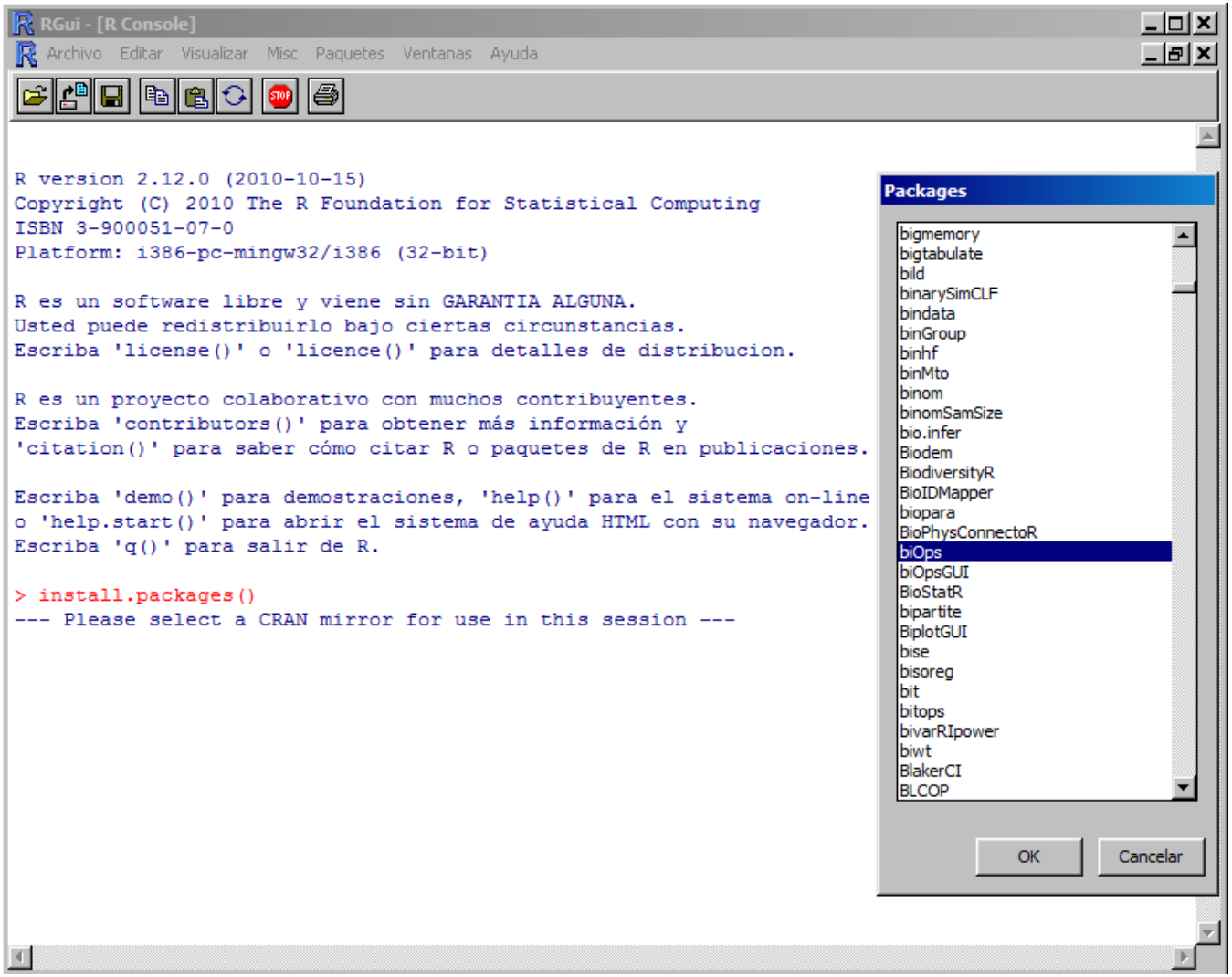
- La librería contiene los paquetes instalados.
  - Librería principal: *R_HOME*/library
- Paquetes cargados por defecto:
  - getOption("defaultPackages")
    - [1] "datasets" "utils" "grDevices" "graphics" "stats" "methods"
- Instalación de paquetes:
  - install.packages()



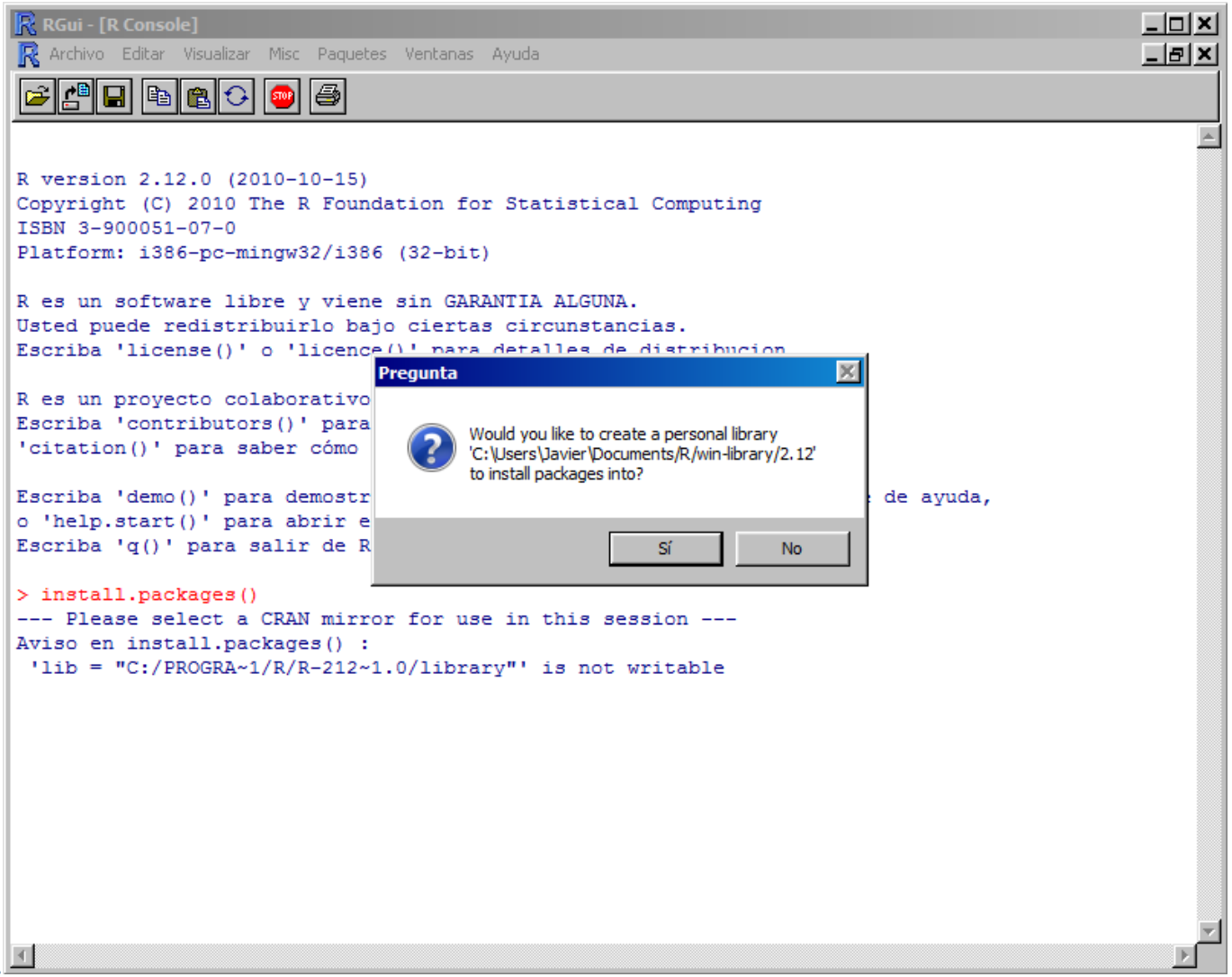
Instalación de Paquetes



Instalación de Paquetes



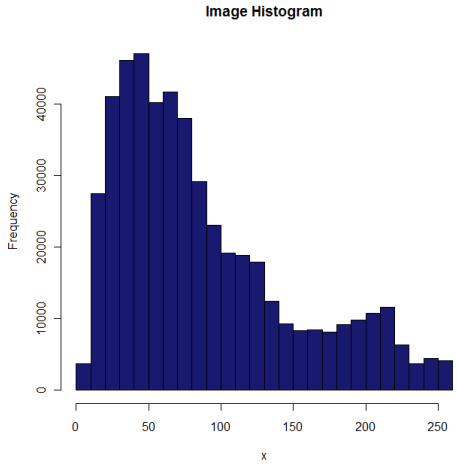
Instalación de Paquetes

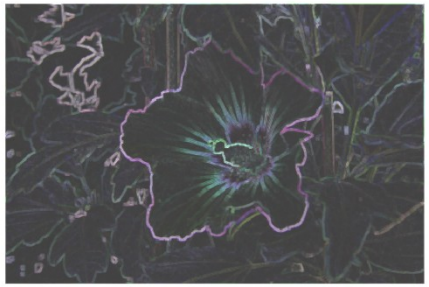


Ejemplos: Paquete biOps

- En R, seleccionamos del menú Paquetes -> Cargar Paquete -> biOps.
  - > x <- readJpeg(system.file("samples", "violet.jpg", package="biOps"))
  - > t <- imgFFT(x)
  - > i <- imgFFTSpectrum(t)
  - > plot(x)
  - > plot(i)

  - h <- imgHistogram(x)





  - y <- imgHomogeneityEdgeDetection(x, bias=64)
  - plot(y)

Trabajos de exposición (15 minutos)

- Paquetes de expansión de R:
  - Rwave — Time-Frequency analysis of 1-D signals
  - signalextraction — Real-Time Signal Extraction (Direct Filter Approach)
  - wavelets — A package of funtions for computing wavelet filters, wavelet transforms and multiresolution analyses
  - waveslim — Basic wavelet routines for one-, two- and three-dimensional signal processing
  - FKF — Fast Kalman Filter
  - KFAS — Multivariate Kalman filter and smoother, simulation smoother and forecasting of state space models. State smoothing and approximate likelihood of exponential family state space models
  - robfilter — Robust Time Series Filters
  - sapa — Insightful Spectral Analysis for Physical Applications
  - biOps — Image processing and analysis
  - biOpsGUI — GUI for Basic image operations
  - PET — Simulation and Reconstruction of PET Images
  - ReadImages — Image Reading Module for R
  - rimage — Image Processing Module for R
  - ripa — R Image Processing and Analysis
  - class — Functions for Classification

Trabajos de exposición (15 minutos)

- A Handbook of Statistical Analyses Using R. Brian S. Everitt and Torsten Hothorn, CRC, 2006.
- > install.packages("HSAUR")
- > library("HSAUR")

http://cran.r-project.org/web/packages/HSAUR/index.html

- Chapter 2. Simple Inference: Guessing Lengths, Wave Energy, Water Hardness, Piston Rings, and Rearrests of Juveniles
- Chapter 3. Conditional Inference: Guessing Lengths, Suicides, Gastrointestinal Damage, and Newborn Infants
- Chapter 4. Analysis of Variance: Weight Gain, Foster Feeding in Rats, Water Hardness and Male Egyptian Skulls
- Chapter 5. Multiple Linear Regression: Cloud Seeding
- Chapter 6. Logistic Regression and Generalised Linear Models: Blood Screening, Women's Role in Society, and Colonic Polyps
- Chapter 7. Density Estimation: Erupting Geysers and Star Clusters
- Chapter 8. Recursive Partitioning: Large Companies and Glaucoma Diagnosis
- Chapter 9. Survival Analysis: Glioma Treatment and Breast Cancer Survival
- Chapter 10. Analysing Longitudinal Data I: Computerised Delivery of Cognitive Behavioural Therapy–Beat the Blues
- Chapter 11. Analysing Longitudinal Data II – Generalised Estimation Equations: Treating Respiratory Illness and Epileptic Seizures
- Chapter 12. Meta-Analysis: Nicotine Gum and Smoking Cessation and the Efficacy of BCG Vaccine in the Treatment of Tuberculosis
- Chapter 13. Principal Component Analysis: The Olympic Heptathlon
- Chapter 14. Multidimensional Scaling: British Water Voles and Voting in US Congress
- Chapter 15. Cluster Analysis: Classifying the Exoplanets