Introducción a R

Juan Carlos Herrera

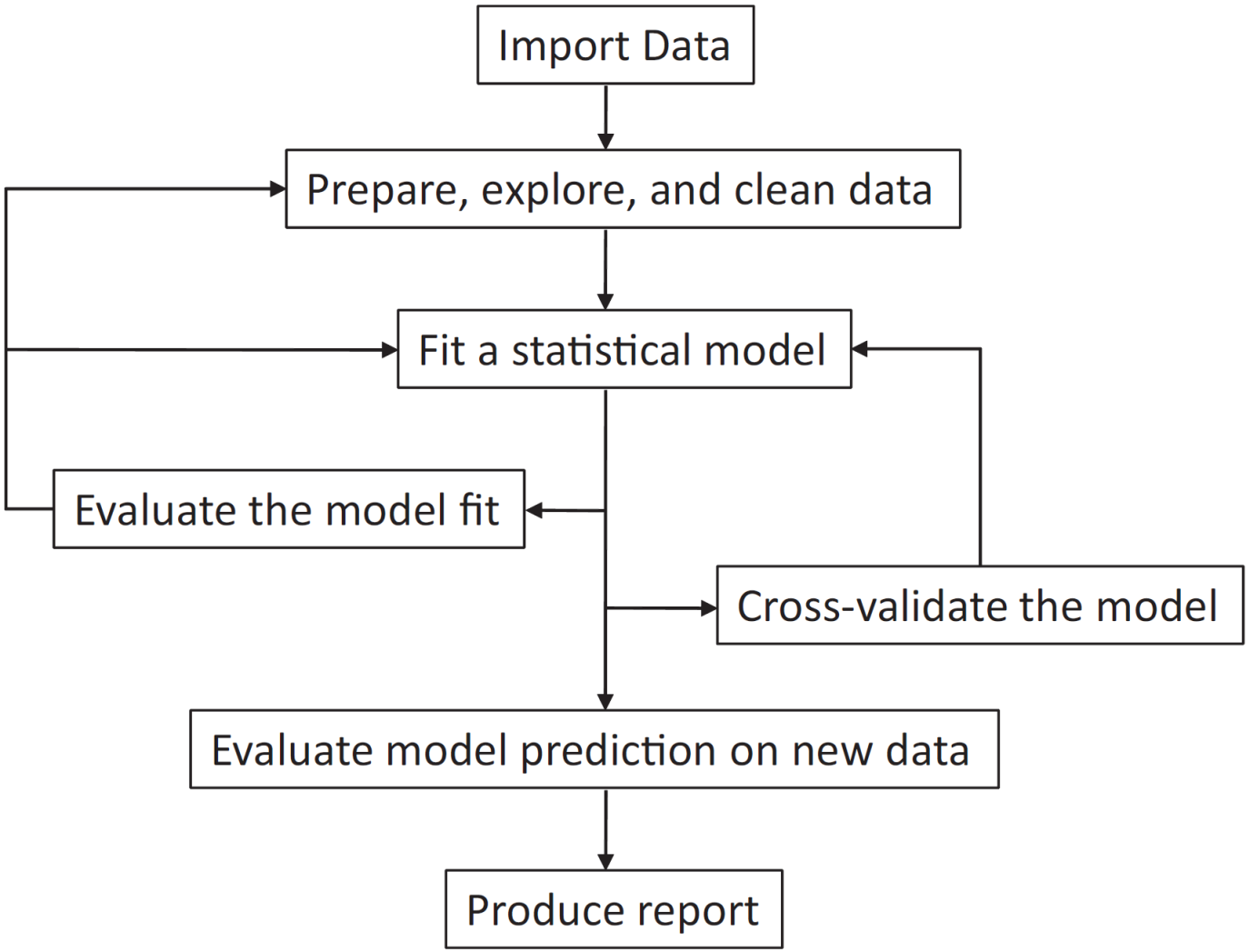
24 Agosto 2017

Contenidos

- Introducción y Preliminares
- Manipulaciones sencillas: Números y vectores
- Vectores y Matrices
- Estructura de datos
- Lectura de datos
- Distribuciones de Probabilidad
- Bucles y ejecuciones condicionales
- Escribir funciones
- Análisis de Datos

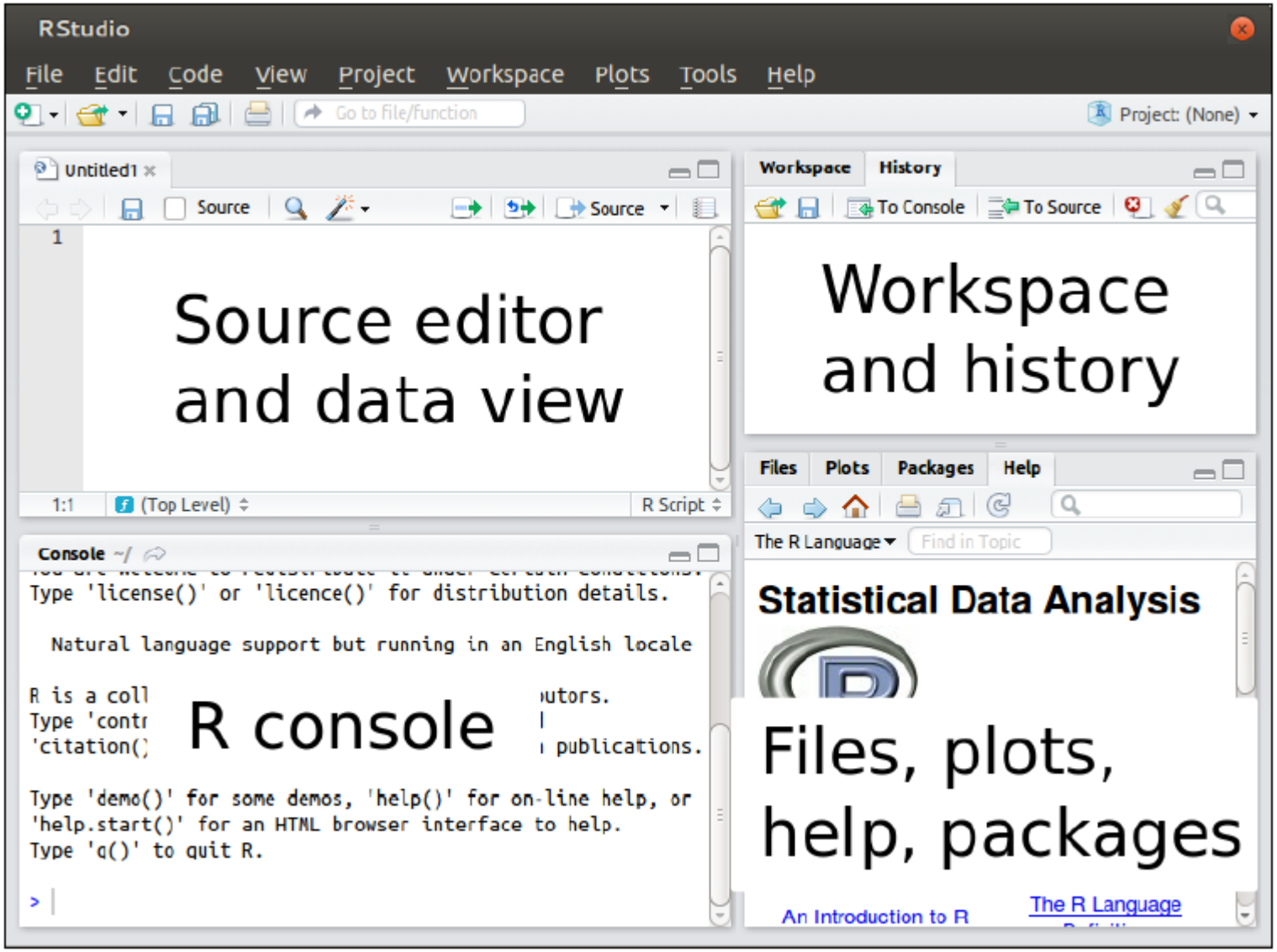
Introducción y Preliminares

Motivación



Introducción y Preliminares

Interfaz Rstudio



Introducción y Preliminares

Tabla con funciones de Ayuda

Function	Action
`help.start()`	General help.
`help("foo")` or `?foo`	Help on function *foo* (the quotation marks are optional).
`help.search("foo")` or `??foo`	Search the help system for instances of the string *foo*.
`example("foo")`	Examples of function *foo* (the quotation marks are optional).
`RSiteSearch("foo")`	Search for the string *foo* in online help manuals and archived mailing lists.
`apropos("foo", mode="function")`	List all available functions with *foo* in their name.
`data()`	List all available example datasets contained in currently loaded packages.
`vignette()`	List all available vignettes for currently installed packages.
`vignette("foo")`	Display specific vignettes for topic *foo*.

Introducción y Preliminares

Tabla con funciones para manipular el espacio de trabajo

Function	Action
`getwd()`	List the current working directory.
`setwd("mydirectory")`	Change the current working directory to *mydirectory*.
`ls()`	List the objects in the current workspace.
`rm(objectlist)`	Remove (delete) one or more objects.
`help(options)`	Learn about available options.
`options()`	View or set current options.
`history(#)`	Display your last # commands (default = 25).
`savehistory("myfile")`	Save the commands history to *myfile* (default = `.Rhistory`).
`loadhistory("myfile")`	Reload a command's history (default = `.Rhistory`).
`save.image("myfile")`	Save the workspace to myfile (default = `.RData`).
`save(objectlist, file="myfile")`	Save specific objects to a file.
`load("myfile")`	Load a workspace into the current session (default = `.RData`).
`q()`	Quit R. You'll be prompted to save the workspace.

Manipulaciones sencillas: Números y vectores

Aritmética simple

```
> x <- 2
> y <- 3
> x+y
 [1] 5
```

Vectores y asignación

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

Vectores Lógicos

```
z <- c(TRUE, TRUE, FALSE)
```

Valores Faltantes

```
q4 <- c(5, 5, 5, NA, 2)
```

Operaciones con Vectores

Suma, resta, producto cruzado, hacer algunos ejemplos..!!!

Vectores y Matrices

Operaciones con Matrices

- `A %*% B` : producto de matrices
- `t(A)` : transpuesta de la matriz `A`
- `solve(A,b)` : solución del sistema de ecuaciones `Ax=b`.
- `solve(A)` : inversa de la matriz `A`
- `svd(A)` : descomposición en valores singulares
- `qr(A)` : descomposición QR
- `eigen(A)` : valores y vectores propios
- `diag(b)` : matriz diagonal (`b` es un vector)
- `diag(A)` : matriz diagonal (`A` es una matriz)
- `A %o% B == outer(A,B)` : producto exterior de dos vectores o matrices

hacer algunos ejemplos..!!!

Vectores y Matrices

Creating matrices

```
  > y <- matrix(1:20, nrow=5, ncol=4)
> y
     [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20
> cells    <- c(1,26,24,68)
> rnames   <- c("R1", "R2")
> cnames   <- c("C1", "C2")
> mymatrix <- matrix(cells, nrow=2, ncol=2, byrow=TRUE,
                     dimnames=list(rnames, cnames))
> mymatrix
   C1 C2
R1  1 26
R2 24 68
> mymatrix <- matrix(cells, nrow=2, ncol=2, byrow=FALSE,
                     dimnames=list(rnames, cnames))
> mymatrix
  C1 C2
R1  1 24
R2 26 68
```

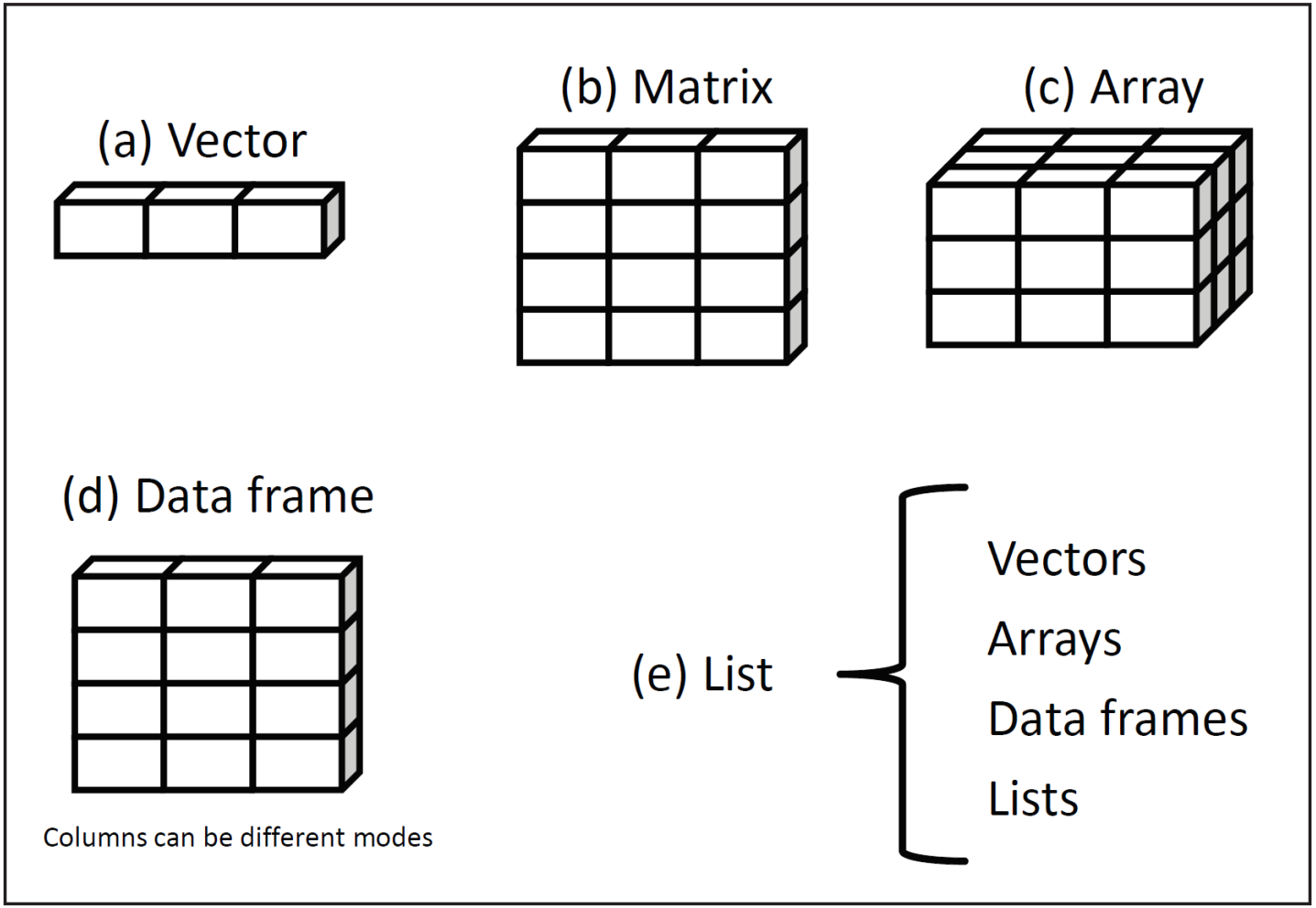
1 Create a 5x4 matrix

2 2x2 matrix filled by rows

3 2x2 matrix filled by columns

Estructuras de Datos

Estructura de datos con las que trabaja R

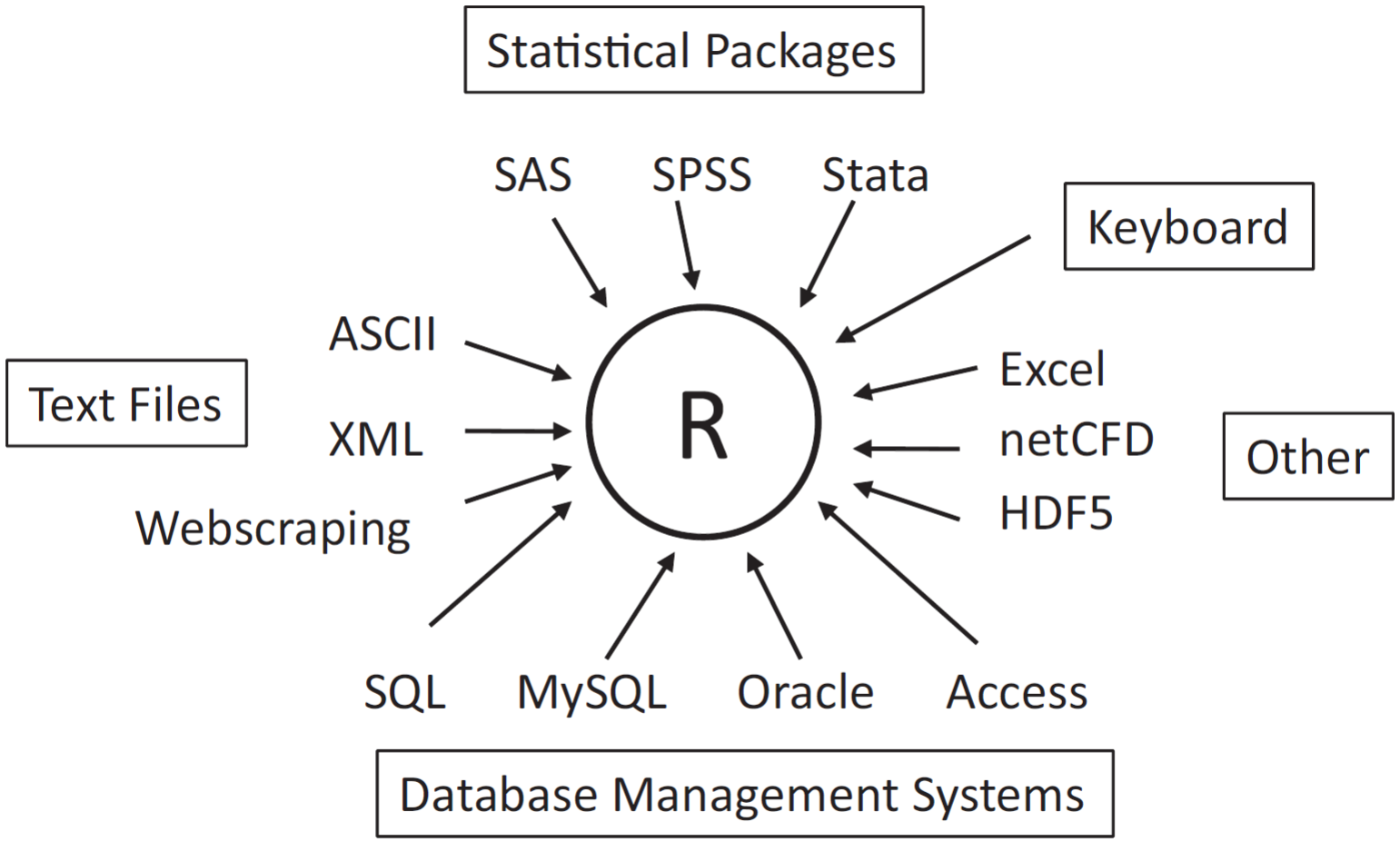


Creating a data frame

```
> patientID <- c(1, 2, 3, 4)
> age <- c(25, 34, 28, 52)
> diabetes <- c("Type1", "Type2", "Type1", "Type1")
> status <- c("Poor", "Improved", "Excellent", "Poor")
> patientdata <- data.frame(patientID, age, diabetes, status)
> patientdata
  patientID age diabetes    status
1         1  25    Type1      Poor
2         2  34    Type2  Improved
3         3  28    Type1 Excellent
4         4  52    Type1      Poor
```

Lectura de datos

Fuentes de datos que pueden ser importadas a R



Lectura de datos

Funciones útiles para trabajar con objetos de datos

Function	Purpose
`length(object)`	Number of elements/components.
`dim(object)`	Dimensions of an object.
`str(object)`	Structure of an object.
`class(object)`	Class or type of an object.
`mode(object)`	How an object is stored.
`names(object)`	Names of components in an object.
`c(object, object,...)`	Combines objects into a vector.
`cbind(object, object, ...)`	Combines objects as columns.
`rbind(object, object, ...)`	Combines objects as rows.
`object`	Prints the object.
`head(object)`	Lists the first part of the object.
`tail(object)`	Lists the last part of the object.
`ls()`	Lists current objects.
`rm(object, object, ...)`	Deletes one or more objects. The statement `rm(list = ls())` will remove most objects from the working environment.
`newobject <- edit(object)`	Edits object and saves as newobject.
`fix(object)`	Edits in place.

Funciones matemáticas útiles

Function	Description
`abs(x)`	Absolute value `abs(-4)` returns 4.
`sqrt(x)`	Square root `sqrt(25)` returns 5. This is the same as `25^(0.5)`.
`ceiling(x)`	Smallest integer not less than *x* `ceiling(3.475)` returns 4.
`floor(x)`	Largest integer not greater than *x* `floor(3.475)` returns 3.
`trunc(x)`	Integer formed by truncating values in *x* toward 0 `trunc(5.99)` returns 5.
`round(x, digits=n)`	Round *x* to the specified number of decimal places `round(3.475, digits=2)` returns 3.48.
`signif(x, digits=n)`	Round *x* to the specified number of significant digits `signif(3.475, digits=2)` returns 3.5.
`cos(x), sin(x), tan(x)`	Cosine, sine, and tangent `cos(2)` returns –0.416.
`acos(x), asin(x), atan(x)`	Arc-cosine, arc-sine, and arc-tangent `acos(-0.416)` returns 2.
`cosh(x), sinh(x), tanh(x)`	Hyperbolic cosine, sine, and tangent `sinh(2)` returns 3.627.
`acosh(x), asinh(x), atanh(x)`	Hyperbolic arc-cosine, arc-sine, and arc-tangent `asinh(3.627)` returns 2.

`log(x,base=n)`
`log(x)`
`log10(x)`
	Logarithm of *x* to the base *n*
	For convenience
	`log(x)` is the natural logarithm.
	`log10(x)` is the common logarithm.
	`log(10)` returns 2.3026.
	`log10(10)` returns 1.

Operadores Lógicos

Operator	Description
`<`	Less than
`<=`	Less than or equal to
`>`	Greater than
`>=`	Greater than or equal to
`==`	Exactly equal to
`!=`	Not equal to
`!x`	Not *x*
`x | y`	*x* or *y*
`x & y`	*x* and *y*
`isTRUE(x)`	Test if *x* is `TRUE`

Funciones de conversión de tipo

Test	Convert
`is.numeric()`	`as.numeric()`
`is.character()`	`as.character()`
`is.vector()`	`as.vector()`
`is.matrix()`	`as.matrix()`
`is.data.frame()`	`as.data.frame()`
`is.factor()`	`as.factor()`
`is.logical()`	`as.logical()`

Distribuciones de Probabilidad

Funciones estadísticas útiles

Function	Description
`mean(x)`	Mean `mean(c(1,2,3,4))` returns 2.5.
`median(x)`	Median `median(c(1,2,3,4))` returns 2.5.
`sd(x)`	Standard deviation `sd(c(1,2,3,4))` returns 1.29.
`var(x)`	Variance `var(c(1,2,3,4))` returns 1.67.
`mad(x)`	Median absolute deviation `mad(c(1,2,3,4))` returns 1.48.

Funciones estadísticas útiles

Function	Description

`quantile(x, probs)`	Quantiles where *x* is the numeric vector where quantiles are desired and *probs* is a numeric vector with probabilities in [0,1].
	`# 30th and 84th percentiles of x`
	`y <- quantile(x, c(.3,.84))`

`range(x)`	Range `x <- c(1,2,3,4)` `range(x)` returns c(1,4). `diff(range(x))` returns 3.
`sum(x)`	Sum `sum(c(1,2,3,4))` returns 10.
`diff(x, lag=n)`	Lagged differences, with `lag` indicating which lag to use. The default lag is 1. `x<- c(1, 5, 23, 29)` `diff(x)` returns `c(4, 18, 6)`.
`min(x)`	Minimum `min(c(1,2,3,4))` returns 1.
`max(x)`	Maximum `max(c(1,2,3,4))` returns 4.
`scale(x, center=TRUE, scale=TRUE)`	Column center `(center=TRUE)` or standardize `(center=TRUE, scale=TRUE)` data object *x*. An example is given in listing 5.6.

Descriptive statistics via `summary()`

```
> summary(mtcars[vars])
      mpg             hp              wt
 Min.   :10.4   Min.   : 52.0   Min.   :1.51
 1st Qu.:15.4   1st Qu.: 96.5   1st Qu.:2.58
 Median :19.2   Median :123.0   Median :3.33
 Mean   :20.1   Mean   :146.7   Mean   :3.22
 3rd Qu.:22.8   3rd Qu.:180.0   3rd Qu.:3.61
 Max.   :33.9   Max.   :335.0   Max.   :5.42
```

Descriptive statistics via `sapply()`

```
> mystats <- function(x, na.omit=FALSE){
                if (na.omit)
                    x <- x[!is.na(x)]
                m <- mean(x)
                n <- length(x)
                s <- sd(x)
                skew <- sum((x-m)^3/s^3)/n
                kurt <- sum((x-m)^4/s^4)/n - 3
                return(c(n=n, mean=m, stdev=s, skew=skew, kurtosis=kurt))
              }
> sapply(mtcars[vars], mystats)
            mpg      hp      wt
n        32.000  32.000 32.0000
mean     20.091 146.688  3.2172
stdev     6.027  68.563  0.9785
skew      0.611   0.726  0.4231
kurtosis -0.373  -0.136 -0.0227
```

Correlation matrix and tests of significance via `corr.test`

```
> library(psych)
> corr.test(states, use="complete")

Call:corr.test(x = states, use = "complete")
Correlation matrix
           Population Income Illiteracy Life Exp Murder HS Grad
Population       1.00   0.21       0.11    -0.07   0.34   -0.10
Income           0.21   1.00      -0.44     0.34  -0.23    0.62
Illiteracy       0.11  -0.44       1.00    -0.59   0.70   -0.66
Life Exp        -0.07   0.34      -0.59     1.00  -0.78    0.58
Murder           0.34  -0.23       0.70    -0.78   1.00   -0.49
HS Grad         -0.10   0.62      -0.66     0.58  -0.49    1.00

Sample Size
[1] 50
Probability value
           Population Income Illiteracy Life Exp Murder HS Grad
Population       0.00   0.15       0.46     0.64   0.01     0.5
Income           0.15   0.00       0.00     0.02   0.11     0.0
Illiteracy       0.46   0.00       0.00     0.00   0.00     0.0
Life Exp         0.64   0.02       0.00     0.00   0.00     0.0
Murder           0.01   0.11       0.00     0.00   0.00     0.0
HS Grad          0.50   0.00       0.00     0.00   0.00     0.0
```

Prueba de una y dos muestras

Para una muestra

Files | Plots | Packages | Help | Viewer

t.test

R: Student's t-Test Find in Topic

t.test {stats} R Documentation

Student's t-Test

Description

Performs one and two sample t-tests on vectors of data.

Usage

```
t.test(x, ...)

## Default S3 method:
t.test(x, y = NULL,
       alternative = c("two.sided", "less", "greater"),
       mu = 0, paired = FALSE, var.equal = FALSE,
       conf.level = 0.95, ...)

## S3 method for class 'formula'
t.test(formula, data, subset, na.action, ...)
```

Arguments

`x`	a (non-empty) numeric vector of data values.
`y`	an optional (non-empty) numeric vector of data values.
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"` or `"less"`. You can specify just the initial letter.
`mu`	a number indicating the true value of the mean (or difference in means if you are performing a two sample test).
`paired`	a logical indicating whether you want a paired t-test.
`var.equal`	a logical variable indicating whether to treat the two variances as being equal. If `TRUE` then the pooled variance is used to estimate the variance otherwise the Welch (or Satterthwaite) approximation to the degrees of freedom is used.
`conf.level`	confidence level of the interval.
`formula`	a formula of the form `lhs ~ rhs` where `lhs` is a numeric variable giving the data values and `rhs` a factor with two levels giving the corresponding groups.
`data`	an optional matrix or data frame (or similar: see `model.frame`) containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
`subset`	an optional vector specifying a subset of observations to be used.

Para dos muestras independientes

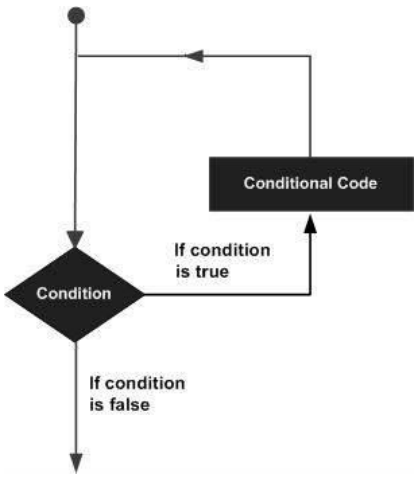
```
t.test(y ~ x, data)
```

Para dos muestras pareadas

```
t.test(y1, y2, paired=TRUE)
```

Bucles y ejecución condicional

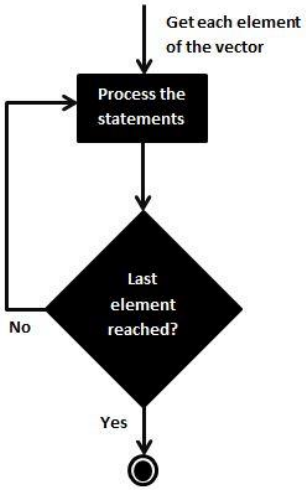
if



Ejemplo:

```
If (condition)
{
...
}
else
{
...
}
```

for

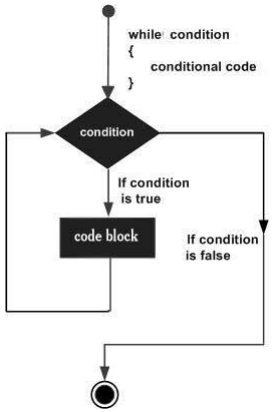


Ejemplo:

```
v <- LETTERS[1:4]
 for ( i in v)
 {
   print(i)
 }
```

Bucles y ejecución condicional

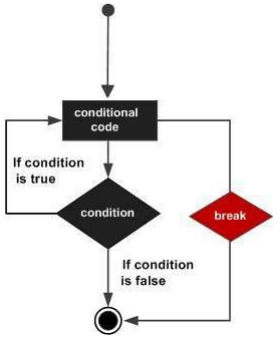
while



Ejemplo:

```
v <- c("Hello","while loop")
 cnt <- 2
while (cnt < 7)
 {
   print(v)
   cnt = cnt + 1
}
```

Repeat



Ejemplo:

```
v <- c("Hello","loop")
 cnt <- 2
 repeat
 {
  print(v)
  cnt <- cnt+1
 if(cnt > 5)
     {
      break
      }
 }
```

Escribir funciones

Resolver ecuación de segundo grado

```
solve2gr <- function(a, b, c)
{

  discr <- b^2 - 4*a*c
  if(discr > 0)
  {
    print("soluciones reales y distintas")
    x1 <- (-b + sqrt(discr)) / (2*a)
    x2 <- (-b - sqrt(discr)) / (2*a)
    return(c(x1,x2))
  }
  if(discr == 0)
  {
    print("soluciones reales iguales")
    x1 <- (-b + sqrt(discr)) / (2*a)
    x2 <- (-b - sqrt(discr)) / (2*a)
    return(c(x1,x2))
  }
  if(discr < 0)
    print("No tiene soluciones reales y distintas")

}
```

```
> sols <- solve2gr(1, 4, 2)
[1] "soluciones reales y distintas"
> sols
[1] -0.5857864 -3.4142136
> sol1 <- solve2gr(1, 4, 2)[1]
>[1] "soluciones reales y distintas"
> sol1
>[1] -0.5857864
```

Análisis de Datos

Fin ?