R: Guía de Referencia Rápida

Angelo Santana*

Índice

*Esta guía de referencia ha sido compilada a partir de las guías (disponibles en inglés) de Tom Short, Vito Ricci y Jonathan Barron, con la adición de algunas funciones nuevas.

1. Sistemas de Ayuda

help() Nos muestra como funciona la ayuda.

help(palabra) o ?palabra Nos muestra ayuda sobre la palabra especificada.

apropos("xyz") Muestra todas las funciones que contienen `"xyz"` en su nombre.

help.search("xyz") Similar al anterior, pero muestra los resultados en un documento html.

help.start() Arranca una página de ayuda general en formato html.

help(package=nombre-libreria) Arranca una página html con ayuda sobre la librería especificada.

demo(nombre-paquete) Hace una demostración de las posibilidades de uso del paquete especificado. `demo()` sin argumentos muestra una lista de los paquetes que incluyen demostraciones.

vignette(tema,package,...) Muestra un documento pdf sobre el tema especificado, asociado al paquete que se indica. No todos los paquetes contienen documentos pdf de esta clase. Puede obtenerse una lista de dichos documentos mediante `browseVignettes()`.

citation() Muestra como citar R en publicaciones. Para citar una librería particular se usa `citation("nombre-librería")`.

2. Entrada/Salida

data(nombre-data-frame) Carga en memoria el conjunto de datos (`data.frame`) especificado, si está disponible en alguna de las librerías cargadas en nuestra sesión de R . El comando data(), sin argumentos, muestra una lista de los data.frames disponibles.

save(objeto,file=nombre-de-archivo.Rdata) Guarda `objeto` en el formato binario propio de R . Es recomendable que el archivo tenga la extensión `.Rdata`

load(nombre-de-archivo.Rdata) Lee los objetos guardados en el formato binario propio de R mediante `save()`

read.table(file,sep,dec,header,as.is,na.strings,...) Lee un archivo ASCII en formato tabular (cada columna una variable, cada fila un caso). El argumento `file="nombre-del-archivo"` especifica el archivo a leer. El resto de argumentos por defecto son:

- `sep=""` indica que el símbolo para separar variables en el archivo es el espacio;
- `dec="."` indica que el símbolo decimal es el punto;
- `header=FALSE` indica que el nombre de las variables se encuentra en la cabecera de cada columna;
- `as.is=FALSE` indica que las variables alfanuméricas se convierten en factores al ser leídas;
- `na.strings="NA"` especifica que los valores perdidos se han codificado como "NA".

read.csv2(nombre-de-archivo,...) Lee archivos de texto csv (comma separated values). Idem que `read.table()`, pero con los valores `sep=";"`, `dec=","` y `header=TRUE` ya incorporados por defecto, por lo que no es preciso especificarlos. Ideal para leer archivos de datos csv con las codificaciones habituales en español. Se pueden especificar adicionalmente los mismos argumentos que en `read.table()`.

read.csv(nombre-de-archivo,...) Idem que `read.csv2()`, pero con los valores por defecto `sep=","`, `dec="."` y `header=TRUE`.

write.table(data,file=nombre-de-archivo) Guarda en data.frame `data` en un archivo ASCII en formato tabular. Admite las mismas opciones que `read.table()` salvo que para que los nombres de las variables se incluyan en la cabecera de la tabla utiliza el argumento `col.names=TRUE` (valor por defecto). También por defecto incluye el nombre de las filas `row.names=TRUE`; si las filas no tienen nombre se guarda su número de orden a menos que se especifique `row.names=FALSE`.

write.csv2() Idem que `write.table()` con las mismas opciones por defecto que `read.csv2()`

write.csv() Idem que `write.table()` con las mismas opciones por defecto que `read.csv()`

read.fwf(file,widths,header=FALSE,sep="",as.is=FALSE) Lee un archivo de texto con columnas de ancho fijo; `widths` es un vector de enteros con las anchuras de cada una de las columnas.

scan() Lectura de datos linea a linea desde consola o archivos.

readWorksheetFromFile(file,sheet,...) Lee datos en formato tabular de archivos Excel (requiere el paquete XLConnect). Además del nombre de archivo hay que indicar el nombre o índice de la hoja a leer.

writeWorksheetToFile(file,sheet,...) escribe datos en formato tabular en archivos Excel (requiere el paquete XLConnect). Además del nombre de archivo hay que indicar el nombre o índice de la hoja a guardar.

read.spss(file,to.data.frame=FALSE) Importa datos de archivos SPSS (paquete `foreign`). Se debe especificar la opción `to.data.frame=TRUE` (por defecto es `FALSE`) si se desea importar los datos directamente a un objeto con estructura de data.frame.

sink(nombre-de-archivo) Tras ejecutar esta función, todas las salidas de R se derivan al archivo indicado, hasta que se ejecuta nuevamente `sink()`, sin argumentos.

readLines(archivo) Lee archivos de texto linea a linea (devuelve un vector cuyos elementos son las líneas del archivo).

writeLines(x,."rchivo") Guarda los elementos del vector `x` en un archivo de texto; cada elemento de `x` ocupa una línea del archivo.

3. Entorno de trabajo

getwd() Permite ver el directorio de trabajo.

setwd() Establece el directorio de trabajo.

dir() Muestra los archivos del directorio de trabajo.

ls() Muestra una lista de todos los objetos que se encuentran en memoria.

rm(x) Elimina (borra) el objeto `x`.

rm(list=ls()) Borra todos los objetos en memoria.

source(nombre-script.R) Ejecuta todos los comandos del archivo script especificado.

library(nombre-librería) Carga en memoria las funciones contenidas en la librería especificada.

require(nombre-libreria) Comprueba si la librería indicada está cargada en memoria, y la carga si no lo está.

detach(package:nombre-librería) Elimina la librería especificada de la memoria de R .

search() Muestra una lista de los paquetes y data.frames cargados en memoria.

attach(nombre-dataframe) Añade el contenido de las variables en el data.frame especificado a la ruta de búsqueda de R . Ello significa que R puede acceder directamente por su nombre a las variables dentro del data.frame.

detach(nombre-dataframe) Elimina las variables del data.frame especificado de la ruta de búsqueda de R , de tal forma que tales variables dejan de estar directamente accesibles por su nombre.

quit() o q() Salir de la sesión de R .

install.packages("nombre-paquete") Instala el paquete especificado descargándolo desde un repositorio de CRAN. También se puede instalar desde un archivo local descargado previamente.

4. Creación y manipulación de datos

Inf Representa el valor "infinito" (∞).

NA Representa un valor perdido.

pi El número π.

c() Crea un vector con los elementos que se especifiquen entre paréntesis, o añade elementos a un vector existente.

cbind(x,y) "Pega" los objetos `x` e `y` por columnas.

rbind(x,y) "Pega" los objetos `x` e `y` por filas.

n_1:n_2 Crea un vector con los valores sucesivos de n_1 a n_2.

seq(n_1,n_2,by,length) Crea una secuencia de valores de n_1 a n_2. Si se especifica `by=k` la secuencia se genera con los valores de k en k. Alternativamente, si se especifica `length=r` se genera una secuencia de `r` valores equiespaciados entre n_1 y n_2.

rep(x,n) Repite `n` veces el vector `x`.

array(x,dim=c(n_1,n_2,...)) Crea un vector de una, dos ó más dimensiones con los valores en `x`.

matrix(x,ncol,byrow=FALSE) Crea una matriz de `n` columnas con los valores del vector `x`; la matriz se va rellenando por columnas con dichos valores; si se especifica `byrow=TRUE` la matriz se rellena por filas.

factor(x,levels,labels) Convierte una variable `x` en *factor* (variable categórica, utilizada habitualmente para realizar clasificaciones de los datos, estableciendo su pertenencia a los grupos o categorías determinados por los niveles del factor).

gl(n,k,length=n*k,labels=1:n) Genera un factor con *n* niveles, cada uno replicado *k* veces. Opcionalmente se pueden especificar etiquetas para los niveles y el número total `length` de observaciones del factor.

cut(x,breaks) Divide `x` en intervalos y crea un factor cuyos niveles son dichos intervalos; `breaks` es el número de intervalos, o un vector que contiene los puntos de corte.

data.frame() Crea una tabla o conjunto de datos. Es similar a una matriz, pero sus columnas pueden ser de distinta clase (*numéric*, *character*, *factor*).

list() Crea una lista o colección de objetos de distintas clases: valores, vectores, matrices, data.frames, funciones, ...

tapply(x,indice,fn) Crea un vector con el resultado de aplicar la función `fn` a los valores del vector `x` separados según los grupos definidos por el factor `indice`.

outer(x,y,FUN) Dados los dos vectores `x` e `y`, crea una matriz `z` de dimensión `dim(z)=c(length(x),length(y))` tal que `z[i,j]=FUN(x[i],y[j])`.

5. Información sobre objetos.

str() Muestra la estructura de un objeto

class() Muestra la clase de un objeto

dim() Muestra o asigna la dimensión de un array.

dimnames() Muestra o asigna nombres a las dimensiones de un array.

length() Muestra la longitud de un vector

ncol() Número de columnas de una matriz

nrow() Número de filas de una matriz

colnames() Muestra o asigna nombres de columna en un array o matriz.

names() Muestra o asigna los nombres de columna en un data.frame o lista.

attributes(x) Muestra o asigna los atributos del objeto x.

attr(x,atrib) Muestra o asigna el atributo `atrib` del objeto `x`

unique() Muestra una lista de los valores distintos que ocurren en un vector.

which(condición) Posiciones de los elementos de un vector que satisfacen la condición especificada. `which(x==2)` devuelve las posiciones de los valores de (x) que son iguales a 2; `which(x<=2)` los que son menores o iguales que 2; etc.

6. Indexación de datos

6.1. Vectores.

x[n] n-ésimo elemento del vector `x`.

x[-n] Todos los elementos de `x` excepto el n-ésimo.

x[1:n] Primeros n elementos de `x`.

x[-(1:n)] Todos los elementos de `x` excepto los que van de 1 a n.

x[c(3,5,7)] Elementos de `x` que ocupan las posiciones 3, 5 y 7.

x[x > 3] Todos los elementos mayores que 3.

x[x > 3 & x < 5] Todos los elementos entre 3 y 5.

x[x %in% c(1,3,5,8)] Elementos de `x` que están en el conjunto especificado.

append(x, values, after = length(x)) Añade o inserta valores en un vector `x` a partir de una posición especificada.

6.2. Listas

x[[n]] *n*-ésimo componente de la lista `x`

x[["nombre"]] **o** **x$nombre** Componente de la lista con el nombre especificado

6.3. Matrices

x[i,j] Elemento de la fila i, columna j.

x[i,] Fila i.

x[,j] Columna j

x[,c(1,3)] Columnas 1 y 3

x["nombre",] Fila con el nombre especificado

6.4. Data frames

Valen los mismos métodos de indexación que para matrices, y además:

x[["nombre"]] **o** **x$nombre** columna con el nombre indicado

7. Miscelánea.

system("comando") Ejecuta un comando del sistema operativo.

is.na(x) Función que devuelve `TRUE` para aquellos valores de `x` que sean valores perdidos, y `FALSE` en caso contrario.

is.finite(x), is.infinite(x) Determina para cada valor en `x` si es finito o infinito.

sort(x) Ordena los valores del vector `x`.

order(x) Muestra los números de orden de los valores del vector `x`.

which.max(x) Devuelve la posición del máximo del vector `x`.

which.min(x) Devuelve la posición del mínimo del vector `x`.

rev(x) Invierte el orden de los elementos de `x`.

match(x,y) devuelve un vector de la misma longitud que `x` pero sólo con los elementos de `x` que están en `y`. El resto se sustituye con NA.

sample(x, size, replace=FALSE) Extrae sin reemplazamiento una muestra aleatoria de tamaño `n` de valores del vector `x`. La opción `replace=TRUE` realiza el muestreo con reemplazamiento.

replicate(n,expr) Replica `n` veces la expresión `expr`. Ésta puede ser cualquier objeto de R. Resulta particulamente interesante en simulación cuando `expr` es una función.

with(object,expr) Evalúa la expresión `expr` utilizando los valores que se encuentran en el entorno definido por el objeto `object`.

integrate(fn,lower,upper) Calcula numéricamente la integral de la función `fn` entre los valores `lower` y `upper`.

uniroot(fn,lower,upper) Encuentra una raíz de la función `fn` en el intervalo entre `lower` y `upper`.

polyroot() Calcula numéricamente las raíces de un polinomio real o complejo.

system.time(expr) Muestra el tiempo de cálculo que consume la evaluación de `expr`.

combinations(n, r, v=1:n, set=TRUE, repeats=FALSE) Genera todas las combinaciones de `n` elementos (especificados en el vector `v`, que puede ser de texto) tomados de `r` en `r`, con o sin repetición según se especifique en `repeats` (paquete `gtools`).

permutations(n, r, v=1:n, set=TRUE, repeats=FALSE) Genera todas las variaciones de `n` elementos (especificados en el vector `v`, que puede ser de texto) tomados de `r` en `r`, con o sin repetición según se especifique en `repeats` (paquete `gtools`). Cuando `n=r` se obtienen las permutaciones de `n` elementos.

8. Presentación de objetos.

nombre-de-objeto Muestra en la consola el contenido del objeto.

head(objeto) Muestra la primera parte de un objeto; en el caso de vectores, muestra los 6 primeros valores; en el caso de matrices, tablas o data.frames, muestra las 6 primeras filas; en el caso de una función muestra las 6 primeras líneas.

tail(objeto) Similar al anterior, pero muestra la parte final del objeto (últimos 6 valores, filas o líneas).

print(nombre-del-objeto, ...) Muestra el contenido del objeto especificado; el tipo de presentación depende de la clase del objeto (cada clase de objeto lleva un método de presentación asociado).

cat(...,file,sep,...) convierte en caracteres los argumentos especificados y los muestra por consola si no se especifica ningún valor para `file`, o los guarda en el fichero que se especifique en su caso. `sep` es el carácter que separa los argumentos.

format(x,...) formatea un objeto para presentarlo de forma elegante.

9. Operaciones con data.frames

merge() Combina data.frames.

subset() extrae un subconjunto de datos de un data.frame

by(data,indices,fn) Aplica la función `fn` a la(s) variable(s) especificada(s) en `data` según los grupos definidos por la variable `indices`.

aggregate(data,by,fn,...) similar a `by` pero más flexible y con otra presentación de resultados. Aplica la función `fn` a `data` según los grupos definidos por `by`.

stack() Toma varias columnas (variables) de un data.frame y las coloca una a continuación de otra en un único vector; crea además una variable índice que especifica de qué columna original procedía cada valor.

unstack() lo contrario que la función `stack()`.

reshape() similar a `stack()` y `unstack()`, pero permite más flexibilidad.

apply(A,1,f,...) Aplica la función `f` a cada fila del data.frame A.

apply(A,2,f,...) Aplica la función `f` a cada columna del data.frame A.

10. Funciones para variables tipo cadena.

nchar(x) Muestra el número de caracteres en la cadena `x`.

paste(...) concatena valores después de convertirlos en caracteres (si no lo eran ya previamente). La opción `sep=` especifica el carácter utilizado entre los valores que se concatenan. Adicionalmente, para concatenar todos los téminos de un vector, en un único valor se utiliza la opción `collapse=` indicando entre comillas el carácter que separa los valores concatenados.

substr(x,start,stop) extrae de `x` los caracteres entre las posiciones `start` y `stop`.

strsplit(x,split) Parte la cadena `x` en los lugares en que aparece la subcadena `split`.

grep(pattern,x) Muestra qué componentes del vector `x` contienen a la cadena `pattern`. Ver `?regex` para ver las distintas formas de especificar patrones de búsqueda.

gsub(pattern,replacement,x) Sustituye en `x` todas las apariciones de `pattern` por `replacement`.

sub() Igual que `gsub` pero solo reemplaza la primera ocurrencia.

tolower(x) Convierte `x` a minúsculas.

toupper(x) Convierte `x` a mayúsculas.

11. Funciones matemáticas.

Algunas de las siguientes funciones precisan que se especifique la opción `na.rm=TRUE` (acrónimo de *NA remove*) para eliminar los valores perdidos del objeto `x` antes de aplicar la función.

+,-,*,/,^ Operaciones aritméticas básicas.

%/% División entera.

%% Resto de la división.

sin(),cos(),tan(),asin(),acos(),atan() Funciones trigonométricas.

log(),log10() Logaritmo natural, logaritmo en base 10.

exp(x) e^x.

sqrt() Raiz cuadrada.

trunc(x),ceiling(x),floor(x) Trunca a la parte entera, redondea por exceso, redondea por defecto.

round(x, n) redondea todos los elementos de `x` a `n` decimales.

factorial(n) factorial de `n`

choose(n, k) Calcula el número de combinaciones de n elementos tomados de k en k.

gamma(x) Función Gamma: $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$

beta(x) Función Beta: $B(x,y) = \int_0^1 t^{x-1}(1-t)^{y-1}\,dt = \Gamma(x)\,\Gamma(y)\,/\Gamma(x+y)$

max(x) valor máximo del vector `x`.

min(x) valor mínimo de `x`.

range(x) rango de `x` (valores mínimo y máximo).

sum(x) Suma de los valores de `x`.

diff(x) Vector formado por las diferencias entre los sucesivos valores de `x`.

prod(x) Producto de todos los elementos de `x`.

log(x, base) logaritmo de `x` en la base que se especifica.

cumsum(x) Sumas acumuladas de los valores de un vector `x`. El término i-ésimo es la suma de los valores `x[1]` hasta `x[i]`.

cumprod(x) Idem que `cumsum`, pero para el producto.

union(x,y), intersect(x,y), setdiff(x,y), setequal(x,y), is.element(x,set) Funciones para realizar operaciones de conjuntos.

Re(x) Parte real de un número complejo.

Im(x) Parte imaginaria.

Mod(x) o abs(x) Módulo de `x`.

Arg(x) Ángulo en radianes del número complejo `x`.

Conj(x) Complejo conjugado de `x`.

convolve(x,y) Varios tipos de convoluciones de los vectores `x` e `y`.

fft(x) Transformada rápida de Fourier de un vector `x`.

mvfft(x) FFT de cada columna de una matriz `x`.

filter(x,filter) aplica un filtro lineal a una serie temporal, o a cada una de las series temporales dispuestas en columnas en una matriz.

sapply(x,f,...) Aplica la función `f` a cada elemento del vector `x`.

12. Operaciones con matrices.

t(A) Traspuesta de la matriz A.

diag(A) Diagonal de A.

diag(1,n) Matriz Identidad de dimensión `nxn`.

A %* %B Producto de matrices

A %^ %n Potencia de una matriz, A^n (requiere el paquete `expm`).

expm(A) exponencial de una matriz e^A (requiere el paquete `expm`).

eigen(A) Calcula autovalores y autovectores de la matriz A.

det(A) Determinante de la matriz A.

solve(A,b) Muestra el vector `x` solución del sistema $Ax = b$.

solve(A) Matriz inversa de A.

rowSums(A) Vector que contiene las sumas de cada fila de la matriz A.

colSums(A) Idem para columnas

rowMeans(A) Vector que contiene las medias de cada fila de la matriz A

colMeans(A) Idem para columnas

apply(A,1,f,...) Aplica la función `f` a cada fila de A.

apply(A,2,f,...) Aplica la función `f` a cada columna de A.

13. Estadística Descriptiva.

Muchas de las siguientes funciones pueden aplicarse no sólo a variables individuales, sino a data.frames completos. En muchos casos es preciso especificar la opción `na.rm=TRUE` para eliminar los valores perdidos del objeto `x` antes de aplicar la función.

table(x) Tabla de frecuencias de `x`.

prop.table(table(x)) Tabla de frecuencias relativas.

table(x,y) Tabla de frecuencias cruzadas de `x` por `y`.

prop.table(table(x),margin=i) Tabla de frecuencias relativas. Las frecuencias relativas se calculan por fila si `margin=1`, por columna si `margin=2` o globales si no se especifica `margin`.

mean(x) Media de los elementos de `x`

median(x) Mediana de los elementos de `x`.

quantile(x,probs=) cuantiles muestrales correspondientes a las probabilidades especificadas, p. ej. `probs=c(0.25, 0.5,0.75,0.95)`.

weighted.mean(x, w) Media de `x` ponderada por `w`.

rank(x) rangos de los elementos de `x`.

var(x) Varianza muestral de `x` (se usa $n-1$ como divisor);

sd(x) Desviación típica de `x`.

summary(x) Muestra un resumen de estadísticos descriptivos: mínimo, máximo, media, mediana y primer y tercer cuartil para variables continuas; tabla de frecuencias para variables discretas (factores).

cov(x,y) Covarianza entre las variables `x` e `y`.

cov(A) matriz de varianzas-covarianzas del data.frame A.

cor(x, y, method, use) Correlación lineal entre las variables `x` e `y`. Como método se puede elegir `"pearson"`, `"spearman"` o `"kendall"`. Por defecto se calcula la correlación de Pearson. El parámetro `use` permite especificar la acción a realizar en presencia de valores perdidos.

cor(A) Matriz de correlaciones de las variables del data.frame A.

scale(x) Tipificación de los valores de `x` (se les resta su media y se dividen por su desviación típica); si se añade la opción `center=FALSE` sólo se cambian de escala, dividiendo por la desviación típica; si se añade `scale=FALSE` sólo se les resta la media.

14. Distribuciones de probabilidad.

A continuación se muestran funciones para simular variables aleatorias con distintas distribuciones de probabilidad. Si la letra inicial `r` se sustituye por `d` se obtienen los valores de la función de densidad o de probabilidad; si se sustituye por `p` se obtienen los valores de la función de distribución acumulada; y si se sustituye por `q` se obtienen los percentiles que se especifiquen. Consultar la ayuda sobre cada distribución para completar esta información.

14.1. Distribuciones Discretas

rbinom(n, size, prob) Binomial

rhyper(nn, m, n, k) Hipergeométrica

rgeom(n, prob) Geométrica

rnbinom(n, size, prob) Binomial negativa.

rpois(n, lambda) Poisson

14.2. Distribuciones Continuas

runif(n, min=0, max=1) Uniforme

rnorm(n, mean=0, sd=1) Distribución normal (gaussiana).

rexp(n, rate=1) Exponencial

rgamma(n, shape, scale=1) Gamma

rweibull(n, shape, scale=1) Weibull

rt(n, df) t de Student.

rf(n, df1, df2) F de Fisher–Snedecor.

rchisq(n, df) χ^2 (Chi-cuadrado) de Pearson

rlogis(n, location=0, scale=1) Logística

rlnorm(n, meanlog=0, sdlog=1) Lognormal

rcauchy(n, location=0, scale=1) Cauchy

rbeta(n, shape1, shape2) Beta

rwilcox(nn, m, n) y **rsignrank(nn, n)** Wilcoxon

14.3. Distribuciones multivariantes

rnorm2d(n,rho) Normal bivariante (paquete `fMultivar`).

mvrnorm(n,mu,sigma) Simulación de la distribución normal multivariante (paquete `MASS`).

14.4. Ajuste de distribuciones

fitdistr(x, densfun, start, ...) Estimación máximo-verosímil de los parámetros de una distribución de probabilidad especificada por la función de densidad `densfun`, a partir de los datos recogidos en el vector `x` (paquete `MASS`).

15. Métodos de Inferencia Estadística.

binom.test() Contraste sobre una proporción en experimentos de Bernoulli.

prop.test() Contraste de proporciones en poblaciones independientes con muestras grandes.

fisher.test() Test exacto de Fisher. Contrasta proporciones en muestras pequeñas.

mcnemar.test() Test de McNemar. Contrasta proporciones en muestras emparejadas.

t.test(expr) Contrastes de la t de Student sobre la media de poblaciones normales, incluyendo intervalos de confianza. El término `expr` puede ser:

- Una única variable `t.test(x, mu=mu0, alternative, ...)`. Se compara la media de la población de la que se ha extraído la muestra `x` con el valor `mu0`. En `alternative` se especifica si la hipótesis alternativa es $\mu = \mu_0$, $\mu > \mu_0$ o $\mu < \mu_0$.
- Dos variables `t.test(x,y,alternative,paired)`. Se contrasta si las medias de las poblaciones que han generado las muestras `x` e `y` son iguales, o difieren en la forma en que se señale como `alternative`. La opción `paired=TRUE` o `paired=FALSE` permite especificar si las muestras son emparejadas o independientes.
- Una variable continua `x` y un factor `g` que define los dos grupos cuyas medias se comparan. La sintaxis a emplear en este caso es `t.test(x ~g)`

wilcox.test() Test de comparación de tendencias centrales en dos poblaciones no normales. Equivalente al test de Mann-Whitney.

chisq.test() Test de la chi-cuadrado sobre tablas de contingencia.

cor.test() Test sobre el coeficiente de correlación lineal entre dos variables.

aov(formula) Análisis de la varianza. Los términos expresados en `formula`, salvo la variable respuesta, deben ser definidos como factores.

leveneTest() Test de homoscedasticidad de Levene (requiere cargar el paquete `car`)

bartlett.test() Test de homoscedasticidad de Bartlett.

duncan.test() Test de comparaciones múltiples de Duncan (paquete `agricolae`).

scheffe.test() Test de comparaciones múltiples de Scheffe (paquete `agricolae`).

SNK.test() Test de comparaciones múltiples de Student-Newman-Keuls (paquete `agricolae`).

HSD.test() Test HSD (Honestly Significant Difference) de Tukey para comparaciones múltiples (paquete `agricolae`).

LSD.test() Test LSD (Least Significant Difference) de comparaciones múltiples (paquete `agricolae`).

kruskal.test() Test de Kruskal-Wallis para la comparación de tendencias centrales en más de dos poblaciones no normales.

friedman.test() Test de Friedman para la comparación de tendencias centrales en múltiples grupos con un diseño en bloques sin replicación.

pairwise.t.test() Contrastes múltiples sobre las medias de poblaciones normales, utilizando la t de Student y corrigiendo por el número de comparaciones.

power.t.test() Permite calcular la potencia de un t-test para un tamaño de muestra, nivel de significación, desviación típica y diferencia a detectar prefijados. Asimismo permite calcular también el tamaño de muestra necesario para alcanzar una potencia determinada con un nivel de significación, desviación típica y diferencia a detectar prefijados.

power.prop.test() Cálculos de potencia y tamaños de muestra en la comparación de proporciones.

power.anova.test() Cálculos de potencia y tamaños de muestra en en análisis de la varianza.

shapiro.test() Test de normalidad de Shapiro-Wilk

ks.test() Tests de Kolmogorov-Smirnov

glht() Contrastes de hipótesis lineales generales y comparaciones múltiples para modelos paramétricos, incluyendo modelos lineales generalizados, modelos de efectos mixtos y modelos de supervivencia (requiere el paquete `multcomp`).

density(x) Devuelve la estimación de núcleo de la función de densidad que ha generado los valores contenidos en el vector `x`.

16. Optimización y ajuste de modelos

optimize(fn, interval=) Busca un máximo o un mínimo de la función `fn` en el intervalo especificado.

optim(par, fn, method = , ...) Obtiene los valores de los parámetros para los que se alcanza un mínimo de la función `fn`. El vector `par` contiene los valores iniciales de los parámetros.

nlm(fn,par, ...) Minimiza la función `fn` utilizando un algoritmo tipo Newton. `par` es el vector que contiene los valores iniciales de los parámetros sobre los que se desea alcanzar el mínimo.

lm(formula, ...) Ajuste de modelos lineales (regresión, análisis de la varianza y de la covarianza). `formula` es de la forma `response ~x+y+...`. Puede usarse p.ej. `I(x*y)+I(x^2)` para añadir términos construidos con funciones no lineales.

glm(formula,family=, ...) Ajuste de modelos lineales generalizados. `family=` especifica la función de enlace y la distribución del tipo de error. Ver `?family` para las funciones disponibles.

step() Selección paso a paso de variables en modelos lineales (`lm`, `glm`) mediante el criterio de Akaike.

vif() Calcula el Factor de Inflación de la Varianza en modelos lineales y modelos lineales generalizados (paquete `car`).

boxcox() Transformación Box-Cox para conseguir normalidad (paquete `MASS`).

gam(formula,family=, ...) Ajuste de modelos aditivos generalizados. Debe cargarse el paquete `mgcv`.

lme(formula, ...): Ajuste de modelos lineales de efectos mixtos (fijos y aleatorios), permitiendo además efectos anidados. Los errores intragrupos pueden ser correlados o/y heteroscedásticos. Requiere el paquete `nlme`.

nls(formula, ...) Estimación por mínimos cuadrados de funciones no lineales.

nlme(formula, ...): Ajuste de modelos no lineales de efectos mixtos (fijos y aleatorios), permitiendo además efectos anidados. Los errores intragrupos pueden ser correlados o/y heteroscedásticos. Requiere el paquete `nlme`.

gls(modelo,data, ...) Ajuste de modelos lineales mediante mínimos cuadrados generalizados. Los residuos pueden estar correlados y ser heteroscedásticos (paquete `nlme`).

approx(x,y, ...) Interpolación lineal dados un conjunto de puntos `x,y`.

spline(x,y, ...) Interpolación mediante splines cúbicos.

loess(formula, ...) Interpolación mediante ajuste polinómico local

Muchas de las funciones para el ajuste de modelos tienen argumentos comunes:

- `data`: el data-frame que contiene los datos especificados en la formulación del modelo.
- `subset`: el subconjunto de datos utilizado, en su caso, para el ajuste del modelo.

- `na.action`: la acción a seguir con los valores perdidos; puede ser `"na.fail"`, `"na.omit"`, `"na.exclude"` o una función.

Suponiendo que **fit** es el objeto resultante de la aplicación de alguno de los modelos anteriores, en muchas ocasiones es posible aplicar las siguientes funciones a dicho objeto:

summary(fit) resumen del ajuste del modelo, incluyendo estimaciones de los parámetros, contrastes y coeficientes de ajuste.

predict(fit,new-data...) predicciones para un nuevo conjunto de datos `new-data` basadas en el ajuste `fit` realizado a partir de los datos originales.

df.residual(fit) Número de grados de libertad residuales.

coef(fit) Coeficientes estimados (a veces acompañados de sus errores estándar).

confint(fit) Intervalos de confianza para los parámetros del modelo.

residuals(fit) Devuelve los residuos del modelo.

fitted(fit) Devuelve los valores ajustados del modelo.

logLik(fit) Devuelve el valor máximo alcanzado por la función de log-verosimilitud utilizada en la estimación del modelo, así como el número de parámetros.

AIC(fit) Calcula el criterio de información de Akaike.

anova(fit,...) Devuelve la tabla resultante de aplicar el análisis de la varianza (o de la *deviance*) al resultado de ajustar un modelo a unos datos, mostrando las varianzas (*deviances*) explicadas por los distintos componentes del modelo, así como la varianza residual.

anova(fit1,fit2) Idem que el anterior, para modelos anidados; permite contrastar la significación de las variables excluidas en el modelo más simple.

plot(fit) Presenta cuatro gráficos para validar el ajuste del modelo; 1: residuos vs valores ajustados Fitted (para detectar falta de linealidad, o curvatura); 2: Gráfico cuantil-cuantil (QQplot) con los cuantiles empíricos frente a los cuantiles de la distribución Normal (para examinar la hipótesis de normalidad en los residuos); 3: Gráfico localización-escala (para comprobar si la varianza residual es o no constante); 4: Gráfico de distancias de Cook (para detectar posibles outliers influyentes).

17. Métodos de análisis multivariante

hclust(d) Análisis Cluster jerárquico basado en una matriz de disimilaridades `d` obtenida normalmente con la función `dist()`

dist(A, method) Calcula la matriz de disimilaridades entre las filas de una matriz o data.frame `A`. Pueden utilizarse diversos métodos para el cálculo de disimilaridades; por defecto se usa la distancia euclídea.

kmeans(A) Aplica el método de las k-medias para formar clusters con los objetos representados por las filas de la matriz `A`.

prcomp(A, ...), princomp(A, ...) Análisis de componentes principales.

factanal(A, factors, ...) Análisis factorial por máxima verosimilitud.

varimax(), promax() Rotaciones para los factores en análisis factorial. Los paquetes `psych` y `GPArotation` incluyen otros métodos de rotación y distintos procedimientos gráficos.

cca() Análisis de correspondencias (paquete `vegan`).

corresp(), mca() Análisis de correspondencias simple y múltiple (paquete `MASS`).

cancor(x,y) Correlaciones canónicas entre las matrices `x` e `y`.

lda() Análisis Discriminante lineal (paquete `MASS`).

qda() Análisis Discriminante cuadrático (paquete `MASS`).

18. Series Temporales.

ts() Crea un objeto de clase “ts” (time series).

diff() Diferencia una serie temporal el número de veces que se especifique.

lag() Desfasa una serie temporal en el número de periodos que se indiquen.

acf() Calcula la función de autocorrelación.

pacf() Calcula la función de autocorrelación parcial.

ccf() Calcula la función de covarianzas o correlaciones cruzadas.

spec.pgram() Estimación del espectro de una serie mediante suavizado del periodograma.

spectrum() Estimación de la densidad espectral de una serie temporal.

sfilter() Elimina la fluctuación estacional de una serie utilizando medias móviles.

decompose() Descompone una serie temporal en las componentes estacional, de tendencia e irregular, utilizando medias móviles.

stl() Descompone una serie temporal en las componentes estacional, de tendencia e irregular, utilizando suavización ’loess’.

adf.test() Aplica el método de Dickey-Fuller Aumentado para contrastar si la serie tiene una raíz unidad (paquete `tseries`).

Box.test() Test de Box-Pierce o de Ljung-Box para contrastar la hipótesis de independencia en una serie temporal.

bds.test() Test para contrastar si una serie temporal es una sucesión de v.a.i.i.d. (paquete `tseries`).

bptest() Test de Breusch-Pagan sobre la heteroscedasticidad de los residuos del modelo ajustado a una serie temporal (paquete `lmtest`)

kpss.test() Test KPSS (Kwiatkowski-Phillips-Schmidt-Shin) para contrastar la estacionariedad de una serie temporal (paquete `tseries`).

ar() Ajusta un modelo autoregresivo.

arima() Ajusta un modelo ARIMA.

garch() Ajusta un modelo Autoregresivo Generalizado Heteroscedástico Condicional GARCH(p,q) a la serie temporal que se especifique (paquete `tseries`).

tsdiag() Validación del ajuste del modelo de series temporales.

durbinWatsonTest() Test de autocorrelaciones de Durbin-Watson (paquete `car`).

arima.sim() Simula un proceso ARIMA.

19. Programación.

Definición de funciones.

function(args) `f=function(args) {expr; return(output)}`

Expresiones condicionales

if(cond) `if(cond) expr else expr_alternativa`

ifelse(condicion,exprT,exprF) Evalúa una condición para cada uno de los valores de un vector `x`; devuelve un vector de la misma longitud que `x`, aplicando `exprT` en aquellos valores de `x` en que se cumple la condición, y `exprF` en los que no se cumple.

switch(expr,args) Muestra el valor de alguno de sus argumentos (`args`, pueden ser también funciones) dependiendo de cuál sea el resultado de evaluar `expr`.

Bucles

for(var in seq) `expr`

while(cond) `expr`

repeat `expr`

break Fuerza la salida de un bucle.

next Fuerza a la iteración siguiente en un bucle.

Llamadas a programas en otros lenguajes.

do.call(funname, args)

20. Gráficos

20.1. Gráficos elementales.

plot(x) Representa la sucesión de valores de `x`.

plot(x, y) Nube de puntos de los pares `(x,y)`.

hist(x) Histograma de frecuencias de `x`.

barplot(x,horiz=FALSE) Diagrama de barras de los valores de `x`; las barras se representan en horizontal mediante la opción `horiz=TRUE`.

pie(x) Diagrama de sectores.

boxplot(x) Gráfico "caja y bigotes".

stripplot(x) Gráfico de los valores de `x` en una línea.

interaction.plot(f1, f2, y) Representación gráfica de la media de `y` para cada combinación de niveles de los factores `f1` y `f2`; los valores de `f1` se representan en el eje `x`, y los valores de `f2` se representan mediante lineas distintas. Se puede añadir la opción `fun` para especificar un estadístico distinto de la media.

matplot(d1,d2) Representa, en el mismo gráfico, una nube de puntos con la primera columna del data.frame `d1` frente a la primera columna del `d2`, la segunda de `d1` frente a la segunda de `d2`, etc. Cada pareja de variables se representa mediante un símbolo distinto. Ambos data.frames deben tener el mismo número de columnas.

pairs(d) Si `d` es un data.frame o una matriz, dibuja todos los posibles gráficos bivariados entre las columnas de `d`.

plot.ts(x) Si `x` es un objeto de clase "ts" (serie temporal), represente los valores de `x` frente al tiempo. Si `x` es multivariante, todas las series que la componen deben tener las mismas fechas y frecuencias.

ts.plot(x) Similar la anterior pero en el caso multivariante las series pueden tener distintas fechas (aunque la misma frecuencia).

qqnorm(x) Gráfico de los cuantiles de `x` frente a los valores esperados de dichos cuantiles bajo una distribución normal.

qqline(x) Añade al gráfico obtenido con `qqnorm` una línea que pasa por las coordenadas correspondientes al primer y tercer cuartil.

qqplot(x, y) Gráfico de los cuantiles de `y` frente a los cuantiles de `x`.

contour(x, y, z) Representación mediante lineas de nivel de los valores de `z` sobre el plano `XY` (los datos se interpolan para dibujar las isolíneas). `x` e `y` deben ser vectores y `z` una matriz de dimensión `dim(z)=c(length(x), length(y))`, que contiene el valor de `z` para cada par de valores `(x,y)`.

filled.contour(x, y, z) Similar al anterior, pero coloreando el espacio entre líneas de nivel.

image(x, y, z) Crea una malla de rectángulos sobre el plano XY, coloreando cada rectángulo de acuerdo al valor especificado en la matriz `z` para cada punto `(x,y)`.

persp(x, y, z) Similar al anterior pero representando los rectángulos en perspectiva 3D.

symbols(x, y, ...) Dibuja símbolos (círculos, cuadrados, rectángulos, estrellas o boxplots) en las coordenadas espeficadas por `x` e `y`. Se pueden especificar argumentos adicionales como color o tamaño.

20.2. Argumentos adicionales de las funciones gráficas.

Estos argumentos pueden añadirse a muchas de las funciones anteriores. Se muestran los valores por defecto (cuando los tienen) de dichos argumentos.

add=FALSE Permite, si se declara como `TRUE`, superponer el gráfico que se está generando ahora al último gráfico que se haya generado previamente (si existe).

axes=TRUE Si se declara como `FALSE` no se dibujan los ejes ni el marco alrededor del gráfico.

type="p" Especifica el tipo de gráfico; *"p"*: puntos; *"l"*: líneas; *"b"*: (*both*) puntos conectados por líneas; "o": igual, pero las líneas se superponen a los puntos; *"h"*: líneas verticales; *"s"*: (*steps*), gráfico escalonado, con el valor representado en la parte alta del escalón; *"S"*: idem, pero con el valor representado en la parte baja del escalón.

xlim=, ylim= Permite especificar los límites superior e inferior en cada eje, como en el ejemplo `xlim=c(0,10)`.

xlab=, ylab= Etiquetas para los ejes (se especifican entre comillas).

main= Título para el gráfico.

sub= Subtítulo para el gráfico.

20.3. Funciones para la gestión de colores.

colors() Muestra la lista de colores (con nombre propio) disponibles en R .

palette() Permite ver o manipular la paleta de colores disponible para el trazado de gráficos.

colorRamp() Permite definir una nueva paleta de colores mediante la interpolación de colores.

terrain.colors(), heat.colors(), topo.colors(), cm.colors(), rainbow() Paletas de colores predefinidas.

rgb() Crea un nuevo color a partir de las intensidades relativas de rojo (R), verde (G) y azul (B).

20.4. Adición de puntos, líneas, texto, polígonos, ... al gráfico activo.

grid() Superpone una malla al gráfico activo.

points(x, y) Añade puntos a un gráfico existente.

lines(x, y) Añade a un gráfico existente las líneas resultantes de unir los puntos especificados en `x` e `y`.

text(x, y, labels, ...) Añade el texto especificado en `labels` en las coordenadas `(x,y)` del gráfico activo.

mtext(text, side, line, ...) Añade texto en el margen especificado (1: abajo, 2: izquierda, 3: arriba, 4: derecha); `line` especifica la línea (contada desde el marco del gráfico) en que se añadirá el texto.

segments(x0, y0, x1, y1) Dibuja segmentos entre los puntos `(x0,y0)` y los puntos `(x1,y1)`

arrows(x0, y0, x1, y1, angle, code) Idem que el anterior, pero añadiendo una punta de flecha; la flecha apunta a `(x0,y0)` si `code=2`, a `(x1,y1)` si `code=1`, o a ambos extremos si `code=3`; `angle` especifica el la amplitud del ángulo de la punta de la flecha.

abline(a,b) Dibuja una recta de pendiente `b` y ordenada `a`.

abline(h=y) Dibuja una línea horizontal en la ordenada `y`.

abline(v=x) Dibuja una línea vertical en la ordenada `x`.

abline(recta) Dibuja la recta de regresión, usualmente obtenida como `recta=lm(y~x)`.

rect(x1, y1, x2, y2) Dibuja un rectángulo de vértice inferior izquierdo `(x1,y1)` y vértice superior derecho `(x2,y2)`.

polygon(x, y) Dibuja un polígono uniendo con lineas los puntos cuyas coordenadas se especifican en `x` e `y`.

legend(x, y, legend, ...) Añade la leyenda `legend` en el punto `(x,y)`. En lugar de `(x,y)` puede especificarse `"topright"`, `"topleft"`, `"bottomright"` o `"bottomleft"`.

title() Permite añadir un título o subtítulo.

axis(side, vect) (para gráficos que no tengan ejes dibujados) Añade un eje en la parte inferior de gráfico (`side=1`), a la izquierda (`side=2`), arriba (`side=3`) o a la derecha (`side=4`); `vect` (opcional) especifica las posiciones en abcisa (u ordenadas) donde se dibujan las *tickmarks*.

rug(x) Dibuja los valores de `x` sobre el eje de ordenadas como pequeñas lineas verticales.

locator(m, type="n", ...) Devuelve *m* coordenadas de la forma `(x,y)` correspondientes a *m* posiciones sobre el gráfico en las que haya clicado el usuario con el ratón; también permite dibujar puntos `(type="p")` o líneas `(type="l")`; por defecto no se dibuja nada `(type="n")`.

20.5. Parámetros gráficos.

Los siguientes parámetros se fijan globalmente mediante la función **par()**, de la forma par(adj=,bg=,mfrow=, ...) *antes* de la construcción del primer gráfico al que afecten. Mientras no se modifiquen estos parámetros, se seguirán utilizando en los sucesivos gráficos. En algunos casos pueden pasarse también como argumentos de muchas de las funciones gráficas.

adj Controla la justificación del texto (0: justificación izquierda, 0.5: centrado , 1:justificación derecha)

bg Especifica el color de fondo.

bty Controla el tipo de marco que se dibuja alrededor del gráfico. Pueden utilizarse los valores: "o" (valor por defecto), "l", "7", "c", "u", "]" (el marco se muestra en cada caso como la correspondiente letra mayúscula); si bty="n" no se dibuja ningún marco.

cex Valor que controla el tamaño del texto y los símbolos con respecto al tamaño por defecto. Para controlar el tamaño de los números en los ejes se utiliza `cex.axis`; para el tamaño de las etiquetas de los ejes `cex.lab`, para el título `cex.title` y para el subítulo `cex.sub`.

col Controla el color de símbolos y líneas. Para los colores de ejes, etiquetas, título y subtítulo se usan, respectivamente, `col.axis`, `col.lab`, `col.main` y `col.sub`. Para ver los nombres de los colores disponibles utilizar `colors()`.

font Valor entero que controla el estilo del texto (1: normal, 2: itálica, 3: negrilla, 4: negrilla itálica); los estilos de ejes, etiquetas, título y subtítulo se especifican mediante `font.axis`, `font.lab`, `font.main` y `font.sub`.

las Valor entero que controla la orientación de las etiquetas de los ejes; 0: paralela a los ejes; 1: horizontal; 2: perpendicular a los ejes; 3: vertical.

lty Valor que controla el tipo de linea; puede ser de tipo entero o carácter (1: "solid", 2: "dashed" 3: "dotted", 4: "dotdash", 5: "longdash", 6: "twodash"), o una cadena de hasta ocho caracteres (entre "0" y "9") que especifica, alternativamente, la longitud (en puntos o pixels) de los trazos dibujados y de los espacios en blanco entre ellos.

lwd Valor entero que controla la anchura de las líneas.

mar Vector de 4 valores numéricos que controlan el espacio entre los ejes y los bordes del gráfico, de la forma `c(abajo, izquierda, arriba, derecha)`. Los valores por defecto son `c(5.1, 4.1, 4.1, 2.1)`.

mfcol Un vector de la forma `c(nf, nc)` que divide la ventana gráfica en una matriz de `nf` filas y `nc` columnas, de tal forma que pueden representarse nf·nc gráficos. Los gráficos se van dibujando en la ventana por columnas.

mfrow Idem que el anterior, pero los gráficos se van dibujando por filas.

pch controla el tipo de símbolo que se utiliza para representar puntos; puede ser un entero entre 1 y 25, o un carácter ASCII (letra, cifra o símbolo). Mediante `help(points)` podemos ver la lista de símbolos.

ps Valor entero que controla el tamaño (en pixels) del texto.

srt Valor que especifica ángulo de rotación para cadenas de caracteres.

tcl Valor que especifica la longitud de los *tickmarks* en los ejes como una fracción de la altura de una línea de texto.

20.6. Dispositivos/Formatos de salida de gráficos

Por defecto la salida de gráficos de R se deriva hacia una ventana en la pantalla del ordenador. Si se desea que el gráfico se genere en un formato específico –postscript, tiff, bmp, jpg, etc– es preciso especificar el tipo y el archivo de salida *antes* de generar el gráfico, mediante alguno de los comandos que se reseñan a continuación. Una vez generado el gráfico, debe ejecutarse el comando `dev.off()` para cerrar el archivo donde se ha guardado la figura.

postscript(file, onefile=TRUE, width, height, horizontal=TRUE, paper, ...) Deriva la salida gráfica de R hacia el archivo postscript especificado en `file`; `width` y `height` especifican, respectivamente, la altura y anchura del gráfico (en pulgadas): `onefile=TRUE` (valor por defecto) permite la inclusión de múltiples gráficos en un único archivo; `horizontal=TRUE` especifica la orientación por defecto de la figura en el papel; `paper` especifica el tamaño del papel en la impresora ("a4","letter", "legal", "executive","special"). Si el gráfico se va a utilizar por otra aplicación (e.g. LaTeX), debe elegirse paper="special" de tal forma que el tamaño de la figura sea exactamente el especificado en `width` and `height`.

setEPS(horizontal=FALSE,onefile=FALSE,paper="special") Permite establecer valores por defecto para los gráficos postscript; en particular los valores indicados son los adecuados para gráficos a incluir en documentos LaTeX.

pdf(file,width,height,onefile=TRUE,horizontal=TRUE ...) Deriva la salida gráfica hacia un archivo `pdf`. La configuración de sus parámetros es análoga al comando `postscript`.

jpeg(file,width,height, ...) Deriva la salida gráfica hacia un archivo `jpg`.

bmp(file,width,height, ...) Deriva la salida gráfica hacia un archivo `bmp`.

tiff(file,width,height, ...) Deriva la salida gráfica hacia un archivo `tiff`.

png(file,width,height, ...) Deriva la salida gráfica hacia un archivo `png`.

x11() Abre una nueva ventana gráfica en la pantalla del ordenador (Linux y Mac).

windows() Abre una nueva ventana gráfica en la pantalla del ordenador (Windows)

dev.off() Cierra el dispositivo –o archivo– gráfico activo.

20.7. Gráficos en el paquete car

scatter3d(x,y,z,surface=TRUE,fit,revolutions...) Representa una nube de puntos en el espacio; `surface=TRUE` (valor por defecto) superpone una superficie a la nube de puntos; `fit` especifica el tipo de superficie que se ajusta ("linear", "quadratic", "smooth", "additive"). La nube de puntos puede girarse moviendo el ratón; `revolutions=3` ejecuta una animación haciendo girar la figura sobre sí misma el número de veces especificado.

scatter3d(formula,...) Igual que el anterior, salvo que las variables a representar se especifican mediante una fórmula, por ejemplo, `z ~x+y` o bien `z ~x+y|g` si se desea representar los datos en grupos definidos por un factor `g`.

qqPlot(x, dist) Representa los cuantiles empíricos de la variable `x` (o los residuos estudentizados de un modelo lineal) frente a los cuantiles teóricos de la distribución que se especifique.

scatterplot(x,y,...) Representa la nube de puntos `(x,y)` (puede utilizarse también una fórmula como en `scatter3d`), añadiendo boxplots en los márgenes del gráfico, recta de regresión, y línea de regresión suavizada mediante ajuste polinómico-local.

scatterplot(y~x|g, reg.line = lm, smooth = TRUE) representa en un único gráfico los puntos `(x,y)` dibujando de color distinto los grupos definidos por el factor `g`; reg.line=lm superpone la recta de regresión a cada grupo; smooth=TRUE superpone además la linea de regresión suavizada por ajuste polinómico local; `smooth=FALSE` suprime dicha linea.

residualPlots(modelo) Muestra gráficos de los residuos del modelo que se especifique.

20.8. Gráficos en el paquete plotrix.

El paquete `plotrix` contiene muchas funciones para la construcción de gráficos, algunas de las cuales mostramos a continuación:

barp() Diagrama de barras con efectos (forma cilíndrica, sombras, etc.)

pie3D() Diagrama de sectores en 3D.

ablineclip() Permite trazar un segmento de una recta predefinida.

addtable2plot Permite añadir una tabla a un gráfico.

boxed.labels(),trigmophobe.labels() Permite añadir etiquetas a un gráfico; `boxed.labels()` las presenta rodeadas de un marco y `trigmophobe.labels()` emplea un algoritmo que coloca las etiquetas de forma que no se solapen ni se superpongan a los símbolos que acompañan.

draw.arc(),draw.circle(),draw.ellipse() Dibuja arcos, círculos o elipses.

arctext() permite escribir texto siguiendo un arco de circunferencia.

gap.barplot(), gap.boxplot(), gap.plot() Gráficos con un hueco (salto) en alguno(s) de los ejes.

gradient.rect() Dibuja un rectángulo con gradiente de colores.

legendg() Muestra una leyenda con varios símbolos y/o colores.

multhist() Muestra múltiples histogramas, lado a lado.

plotCI() Representa intervalos de confianza o barras de error.

pyramid.plot() Representación de pirámides de población.

tab.title() Muestra el título de un gráfico en una banda coloreada.

20.9. Gráficos en el paquete lattice (gráficos Trellis)

Los gráficos Trellis permiten visualizar datos multivariados fundamentalmente mediante gráficos múltiples condicionados, de tal forma que cada gráfico bivariado se descompone en un panel de gráficos según el valor de una tercera variable (o combinación de variables) categórica. Los datos a representar se describen mediante una fórmula que en general es una combinación de expresiones como las siguientes:

Fórmula	Representación
`~x`	Se representa la variable `x`
`y~x`	Se representa `y` en función de `x`
`y~I(x^2)`	Se representa `y` en función de x^2
`y~x+z`	Se representa `y` en función de `x+z`
`y~x|g`	Se representa `y` en función de `x` con un gráfico distinto para cada valor de `g`
`y~x|g1+g2`	Se representa `y` en función de `x` con un gráfico distinto para cada combinación de valores de `g1` y `g2`

Algunos argumentos adicionales de uso general para las funciones del paquete lattice.

groups El argumento adicional `groups=gr`, que hace que dentro de cada panel los datos se muestren de distinto color según los grupos definidos por el factor `gr`.

data Especifica el conjunto de datos a utilizar.

subset Permite escoger un subconjunto de los datos.

panel Permite definir funciones personalizadas para la representación de gráficos en los paneles Trellis; `apropos("panel")` muestra un listado de las funciones predefinidas. Si se van a definir funciones que requieran dibujar puntos, lineas, texto, etc., en los paneles deben utilizarse funciones específicas de bajo nivel; consultar `help(llines)`.

Otros argumentos Los gráficos Trellis incluyen muchos argumentos opcionales, por lo que es recomendable consultar la ayuda y los ejemplos que ésta ofrece.

Gráficos Trellis Univariados.

barchart(formula, ...) Diagramas de barras.

bwplot(formula, ...) Diagramas de caja y bigotes.

densityplot(~x, ...) Estimación de la función de densidad de `x`.

histogram(~x, ...) Histogramas.

stripplot(formula, ...) Gráfica de dispersión unidimensional

qqmath(~x, ...) Gráficas de los cuantiles empíricos de `x` frente a distintas distribuciones de probabilidad teóricas.

Gráficos Trellis Bivariados

xyplot(formula, ...) gráfico (o panel de gráficos) de dispersión de `y` frente a `x`.

splom(formula, ...) (**s***catter* **plo***t* **m***atrix*) matriz con todos los gráficos bivariados posibles entre el conjunto de variables especificado.

contourplot(formula) Curvas de nivel.

levelplot(formula) Similar al anterior, pero coloreando el espacio entre isolíneas.

Gráficos Trellis 3d

cloud(formula) Gráficos de dispersión 3-D

wireframe(formula) Superficies 3-D

20.10. Gráficos en el paquete ggplot2

El paquete `ggplot2` es un paquete de gráficos de gran potencia y versatilidad porque se apoya en una auténtica *gramática de gráficos*. Se muestra a continuación simplemente parte de la sintaxis de la función `qplot` y se recomienda visitar `http://had.co.nz/ggplot2/` para una visión más completa del paquete.

qplot(x, y = NULL, z=NULL, data, facets = . ~., geom = "auto", colour, size, shape, xlim, ylim, main, xlab , ylab, ...)

Realiza un gráfico de la(s) variable(s) especificada(s) (`x, y, z`). Los argumentos de esta función son:

data Nombre del conjunto de datos del que se leen las variables.

geom Tipo de gráfico: "bar","histogram","boxplot","density", ...

facets=g1~g2 Se dibuja un panel de gráficos, con una figura para cada combinación de niveles de los factores `g1` (filas) y `g2` (columnas); si no se especifica, se dibuja un único gráfico.

colour=f Si los datos están separados en grupos definidos por el factor `f`, los puntos, boxplots, barras, etc. correspondientes se dibujan con colores distintos.

shape=f Idem que el anterior utilizando formas distintas.

size=x En gráficos de puntos, cada punto se dibuja de tamaño proporcional al valor de la variable `x`.

xlim,ylim,xlab,ylab,main Igual que en `plot`.

20.11. Gráficos en el paquete rgl.

`rgl` es una librería especializada en gráficos 3D.

open3d() Prepara el dispositivo gráfico (una nueva ventana) para generar una figura 3D.

bg3d() Fondo para el gráfico 3D.

material3d() Propiedades "materiales" (color, textura, brillo, ...) de la figura a representar.

persp3d() Representación de una superficie en 3D.

plot3d() Nube de puntos en 3D.

planes3d() Añade un plano al gráfico.

points3d(), lines3d(), segments3d(), triangles3d(), quads3d() funciones que añaden, respectivamente, puntos, lineas, segmentos, triángulos o cuadriláteros al gráfico 3D activo.

spheres3d(), ellipse3d() Traza esferas o elipsoides en 3D.

text3d() Adición de texto al gráfico

title3d() Título del gráfico.

axis3d() Manipulación de los ejes.

play3d() Animación de figuras 3D.

spin3d() Función que se usa en conjunción con `play3d` para hacer girar una figura sobre un eje.

movie3d() Similar a `play3D()`, permite grabar la animación.

light3d() Añade una fuente de luz al gráfico.

view3d() Establece el punto de vista desde el que se observa la figura.