R Reference Card 2.0

Public domain, v2.0 2012-12-24.
V 2 by Matt Baggott, matt@baggott.net
V 1 by Tom Short, t.short@ieee.org
Material from *R for Beginners* by permission of Emmanuel Paradis.

Getting help and info

help(topic) documentation on topic

?topic same as above; special chars need quotes: for example **?'&&'**

help.search("topic") search the help system; same as **??topic**

apropos("topic") the names of all objects in the search list matching the regular expression "topic"

help.start() start the HTML version of help

summary(x) generic function to give a "summary" of x, often a statistical one

str(x) display the internal structure of an R object

ls() show objects in the search path; specify pat="pat" to search on a pattern

ls.str() str for each variable in the search path

dir() show files in the current directory

methods(x) shows S3 methods of x

methods(class=class(x)) lists all the methods to handle objects of class x

findFn() searches a database of help packages for functions and returns a data.frame (*sos*)

Other R References

CRAN task views are summaries of R resources for task domains at: cran.r-project.org/web/views Can be accessed via *ctv* package

R FAQ: cran.r-project.org/doc/FAQ/R-FAQ.html

R Functions for Regression Analysis, by Vito Ricci: cran.r-project.org/doc/contrib/Ricci-refcard-regression.pdf

R Functions for Time Series Analysis, by Vito Ricci: cran.r-project.org/doc/contrib/Ricci-refcard-ts.pdf

R Reference Card for Data Mining, by Yanchang Zhao: www.rdatamining.com/docs/R-refcard-data-mining.pdf

R Reference Card, by Jonathan Baron: cran.r-project.org/doc/contrib/refcard.pdf

Operators

<-	Left assignment, binary
->	Right assignment, binary
=	Left assignment, but not recommended
<<-	Left assignment in outer lexical scope; not for beginners
$	List subset, binary
-	Minus, can be unary or binary
+	Plus, can be unary or binary
~	Tilde, used for model formulae
:	Sequence, binary (in model formulae: interaction)
::	Refer to function in a package, i.e, pkg::function; usually not needed
*****	Multiplication, binary
/	Division, binary
^	Exponentiation, binary
%x%	Special binary operators, x can be replaced by any valid name
%%	Modulus, binary
%/%	Integer divide, binary
%*%	Matrix product, binary
%o%	Outer product, binary
%x%	Kronecker product, binary
%in%	Matching operator, binary (in model formulae: nesting)
! x	logical negation, NOT x
x & y	elementwise logical AND
x && y	vector logical AND
x | y	elementwise logical OR
x || y	vector logical OR
xor(x, y)	elementwise exclusive OR
<	Less than, binary
>	Greater than, binary
==	Equal to, binary
>=	Greater than or equal to, binary
<=	Less than or equal to, binary

Packages

install.packages("pkgs", lib) download and install pkgs from repository (lib) or other external source

update.packages checks for new versions and offers to install

library(pkg) loads pkg, if pkg is omitted it lists packages

detach("package:pkg") removes pkg from memory

Indexing vectors

x[n]	nth element
x[-n]	all but the nth element
x[1:n]	first n elements
x[-(1:n)]	elements from n+1 to end
x[c(1,4,2)]	specific elements
x["name"]	element named "name"
x[x > 3]	all elements greater than 3
x[x > 3 & x < 5]	all elements between 3 and 5
x[x %in% c("a","if")]	elements in the given set

Indexing lists

x[n]	list with elements n
x[[n]]	nth element of the list
x[["name"]]	element named "name"
x$name	as above (w. partial matching)

Indexing matrices

x[i,j]	element at row i, column j
x[i,]	row i
x[,j]	column j
x[,c(1,3)]	columns 1 and 3
x["name",]	row named "name"

Indexing matrices data frames (same as matrices plus the following)

X[["name"]]	column named "name"
x$name	as above (w. partial matching)

Input and output (I/O)

R data object I/O

data(x) loads specified data set; if no arg is given it lists all available data sets

save(file,...) saves the specified objects (...) in XDR platform-independent binary format

save.image(file) saves all objects

load(file) load datasets written with save

Database I/O

Useful packages: *DBI* interface between R and relational DBMS; *RJDBC* access to databases through the JDBC interface; *RMySQL* interface to MySQL database; *RODBC* ODBC database access; *ROracle* Oracle database interface driver; *RpgSQL* interface to PostgreSQL database; *RSQLite* SQLite interface for R

Other file I/O

read.table(file), read.csv(file), read.delim("file"), read.fwf("file") read a file using defaults sensible for a table/csv/delimited/fixed-width file and create a data frame from it.

write.table(x,file), write.csv(x,file) saves x after converting to a data frame

txtStart and **txtStop**: saves a transcript of commands and/or output to a text file (*TeachingDemos*)

download.file(url) from internet

url.show(url) remote input

cat(..., file="", sep=" ") prints the arguments after coercing to character; sep is the character separator between arguments

print(x, ...) prints its arguments; generic, meaning it can have different methods for different objects

format(x,...) format an R object for pretty printing

sink(file) output to file, until sink()

Clipboard I/O

File connections of functions can also be used to read and write to the clipboard instead of a file.

Mac OS: **x <- read.delim(pipe("pbpaste"))**

Windows: **x <- read.delim("clipboard")**

See also **read.clipboard** (*psych*)

Data creation

c(...) generic function to combine arguments with the default forming a vector; with recursive=TRUE descends through lists combining all elements into one vector

from:to generates a sequence; ":" has operator priority; 1:4 + 1 is "2,3,4,5"

seq(from,to) generates a sequence by= specifies increment; length= specifies desired length

seq(along=x) generates 1, 2, ..., length(along); useful in for loops

rep(x,times) replicate x times; use each to repeat "each" element of x each times; rep(c(1,2,3),2) is 1 2 3 1 2 3; **rep(c(1,2,3),each=2)** is 1 1 2 2 3 3

data.frame(...) create a data frame of the named or unnamed arguments data.frame (v=1:4, ch= c("a","B","c","d"), n=10); shorter vectors are recycled to the length of the longest

list(...) create a list of the named or unnamed arguments; list(a=c(1,2),b="hi", c=3);

array(x,dim=) array with data x; specify dimensions like dim=c(3,4,2); elements of x recycle if x is not long enough

matrix(x,nrow,ncol) matrix; elements of x recycle factor(x,levels) encodes a vector x as a factor

gl(n, k, length=n*k, labels=1:n) generate levels (factors) by specifying the pattern of their levels; k is the number of levels, and n is the number of replications

expand.grid() a data frame from all combinations of the supplied vectors or factors

Data conversion

as.array(x), as.character(x), as.data.frame(x), as.factor(x), as.logical(x), as.numeric(x), convert type; for a complete list, use **methods(as)**

Data information

is.na(x), is.null(x), is.nan(x); is.array(x), is.data.frame(x), is.numeric(x), is.complex(x), is.character(x); for a complete list, use **methods(is)**

x prints x

head(x), tail(x) returns first or last parts of an object

summary(x) generic function to give a summary

str(x) display internal structure of the data

length(x) number of elements in x

dim(x) Retrieve or set the dimension of an object; **dim(x) <- c(3,2)**

dimnames(x) Retrieve or set the dimension names of an object

nrow(x), ncol(x) number of rows/cols; **NROW(x), NCOL(x)** is the same but treats a vector as a one-row/col matrix

class(x) get or set the class of x; **class(x) <- "myclass";**

unclass(x) removes the class attribute of x

attr(x,which) get or set the attribute which of x

attributes(obj) get or set the list of attributes of obj

Data selection and manipulation

which.max(x), which.min(x) returns the index of the greatest/smallest element of x

rev(x) reverses the elements of x

sort(x) sorts the elements of x in increasing order; to sort in decreasing order: **rev(sort(x))**

cut(x,breaks) divides x into intervals (factors); breaks is the number of cut intervals or a vector of cut points

match(x, y) returns a vector of the same length as x with the elements of x that are in y (NA otherwise)

which(x == a) returns a vector of the indices of x if the comparison operation is true (TRUE), in this example the values of i for which x[i] == a (the argument of this function must be a variable of mode logical)

choose(n, k) computes the combinations of k events among n repetitions = n!/[(n − k)!k!]

na.omit(x) suppresses the observations with missing data (NA)

na.fail(x) returns an error message if x contains at least one NA

complete.cases(x) returns only observations (rows) with no NA

unique(x) if x is a vector or a data frame, returns a similar object but with the duplicates suppressed

table(x) returns a table with the numbers of the different values of x (typically for integers or factors)

split(x, f) divides vector x into the groups based on f

subset(x, ...) returns a selection of x with respect to criteria (..., typically comparisons: x$V1 < 10); if x is a data frame, the option select gives variables to be kept (or dropped, using a minus)

sample(x, size) resample randomly and without replacement size elements in the vector x, for sample with replacement use: replace = TRUE

sweep(x, margin, stats) transforms an array by sweeping out a summary statistic

prop.table(x,margin) table entries as fraction of marginal table

xtabs(a b,data=x) a contingency table from cross-classifying factors

replace(x, list, values) replace elements of x listed in index with values

Data reshaping

merge(a,b) merge two data frames by common col or row names

stack(x, ...) transform data available as separate cols in a data frame or list into a single col

unstack(x, ...) inverse of stack()

rbind(...) , cbind(...) combines supplied matrices, data frames, etc. by rows or cols

melt(data, id.vars, measure.vars) changes an object into a suitable form for easy casting, (*reshape2* package)

cast(data, formula, fun) applies fun to melted data using formula (*reshape2* package)

recast(data, formula) melts and casts in a single step (*reshape2* package)

reshape(x, direction...) reshapes data frame between 'wide' (repeated measurements in separate cols) and 'long' (repeated measurements in separate rows) format based on direction

Applying functions repeatedly

(m=matrix, a=array, l=list; v=vector, d=dataframe)

apply(x,index,fun) input: m; output: a or l**;** applies function fun to rows/cols/cells (index) of x

lapply(x,fun) input l**;** output l**;** apply fun to each element of list x

sapply(x,fun) input l**;** output v**;** user friendly wrapper for **lapply()**; see also **replicate()**

tapply(x,index,fun) input l output l; applies fun to subsets of x, as grouped based on index

by(data,index,fun) input df; output is class "by", wrapper for tapply

aggregate(x,by,fun) input df; output df; applies fun to subsets of x, as grouped based on index. Can use formula notation.

ave(data, by, fun = mean) gets mean (or other fun) of subsets of x based on list(s) by

***plyr* package functions** have a consistent names: The first character is input data type, second is output. These may be d(ataframe), l(ist), a(rray), or _(discard). Functions have two or three main arguments, depending on input:

a*ply(.data, .margins, .fun, ...)
d*ply(.data, .variables, .fun, ...)
l*ply(.data, .fun, ...)

Three commonly used functions with ply functions are **summarise()**, **mutate(), and transform()**

Math

Many math functions have a logical parameter na.rm=FALSE to specify missing data removal.

sin,cos,tan,asin,acos,atan,atan2,log,log10,exp

min(x), max(x) min/max of elements of x

range(x) min and max elements of x

sum(x) sum of elements of x

diff(x) lagged and iterated differences of vector x

prod(x) product of the elements of x

round(x, n) rounds the elements of x to n decimals

log(x, base) computes the logarithm of x

scale(x) centers and reduces the data; can center only (scale=FALSE) or reduce only (center=FALSE)

pmin(x,y,...), pmax(x,y,...) parallel minimum/maximum, returns a vector in which ith element is the min/max of x[i], y[i], . . .

cumsum(x), cummin(x), cummax(x), cumprod(x) a vector which ith element is the sum/min/max from x[1] to x[i]

union(x,y), intersect(x,y), setdiff(x,y), setequal(x,y), is.element(el,set) "set" functions

Re(x) real part of a complex number

Im(x) imaginary part

Mod(x) modulus; **abs(x)** is the same

Arg(x) angle in radians of the complex number

Conj(x) complex conjugate

convolve(x,y) compute convolutions of sequences

fft(x) Fast Fourier Transform of an array

mvfft(x) FFT of each column of a matrix

filter(x,filter) applies linear filtering to a univariate time series or to each series separately of a multivariate time series

Correlation and variance

cor(x) correlation matrix of x if it is a matrix or a data frame (1 if x is a vector)

cor(x, y) linear correlation (or correlation matrix) between x and y

var(x) or **cov(x)** variance of the elements of x (calculated on n − 1); if x is a matrix or a data frame, the variance-covariance matrix is calculated

var(x, y) or **cov(x, y)** covariance between x and y, or between the columns of x and those of y if they are matrices or data frames

Matrices

t(x) transpose

diag(x) diagonal

%*% matrix multiplication

solve(a,b) solves a %*% x = b for x solve(a) matrix inverse of a

rowsum(x), colsum(x) sum of rows/cols for a matrix-like object (consider **rowMeans(x)**, **colMeans(x)**)

Distributions

Family of distribution functions, depending on first letter either provide: r(andom sample) ; p(robability density), c(umulative probability density),or q(uantile):

rnorm(n, mean=0, sd=1) Gaussian (normal)

rexp(n, rate=1) exponential

rgamma(n, shape, scale=1) gamma

rpois(n, lambda) Poisson

rweibull(n, shape, scale=1) Weibull

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

rbeta(n, shape1, shape2) beta

rt(n, df) 'Student' (t)

rf(n, df1, df2) Fisher-Snedecor (*F*) **(!!!^{2})**

rchisq(n, df) Pearson

rbinom(n, size, prob) binomial

rgeom(n, prob) geometric

rhyper(nn, m, n, k) hypergeometric

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

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

rnbinom(n, size, prob) negative binomial

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

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

Descriptive statistics

mean(x) mean of the elements of x

median(x) median of the elements of x

quantile(x,probs=) sample quantiles corresponding to the given probabilities (defaults to 0,.25,.5,.75,1)

weighted.mean(x, w) mean of x with weights w

rank(x) ranks of the elements of x

describe(x) statistical description of data (in *Hmisc* package)

describe(x) statistical description of data useful for psychometrics (in *psych* package)

sd(x) standard deviation of x

density(x) kernel density estimates of x

Some statistical tests

cor.test(a,b) test correlation; **t.test()** t test; **prop.test(), binom.test()** sign test; **chisq.test()** chi-square test; **fisher.test()** Fisher exact test; **friedman.test()** Friedman test; **ks.test()** Kolmogorov-Smirnov test... use **help.search("test")**

Models

Model formulas

Formulas use the form: response ~ termA + termB ...
Other formula operators are:

1 intercept, meaning depdendent variable has its mean value when independent variables are zeros or have no influence

: interaction term

***** factor crossing, a*b is same as a+b+a:b

^ crossing to the specified degree, so (a+b+c)^2 is same as (a+b+c)*(a+b+c)

- removes specified term, can be used to remove intercept as in resp ~ a - 1

%in% left term nested within the right: a + b %in% a is same as a + a:b

I() operators inside parens are used literally: I(a*b) means a multiplied by b

| conditional on, should be parenthetical

Formula-based modeling functions commonly take the arguments: data, subset, and na.action.

Model functions

aov(formula, data) analysis of variance model
lm(formula, data) fit linear models;
glm(formula, family, data) fit generalized linear models; family is description of error distribution and link function to be used; see ?family
nls(formula, data) nonlinear least-squares estimates of the nonlinear model parameters
lmer(formula, data) fit mixed effects model (*lme4*); see also **lme()** (*nlme*)
anova(fit, data...) provides sequential sums of squares and corresponding F-test for objects
contrasts(fit, contrasts = TRUE) view contrasts associated with a factor; to set use: **contrasts(fit, how.many) <- value**
glht(fit, linfct) makes multiple comparisons using a linear function linfct (*mutcomp*)
summary(fit) summary of model, often w/ t-values
confint(parameter) confidence intervals for one or more parameters in a fitted model.
predict(fit,...) predictions from fit
df.residual(fit) returns residual degrees of freedom
coef(fit) returns the estimated coefficients (sometimes with standard-errors)
residuals(fit) returns the residuals
deviance(fit) returns the deviance
fitted(fit) returns the fitted values
logLik(fit) computes the logarithm of the likelihood and the number of parameters
AIC(fit), BIC(fit) compute Akaike or Bayesian information criterion
influence.measures(fit) diagnostics for lm & glm
approx(x,y) linearly interpolate given data points; x can be an xy plotting structure
spline(x,y) cubic spline interpolation
loess(formula) fit polynomial surface using local fitting
optim(par, fn, method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN") general-purpose optimization; par is initial values, fn is function to optimize (normally minimize)
nlm(f,p) minimize function f using a Newton-type algorithm with starting values p

Flow control

if(cond) expr
if(cond) cons.expr else alt.expr
for(var in seq) expr
while(cond) expr repeat expr
break
next
switch
Use braces {} around statements
ifelse(test, yes, no) a value with the same shape as test filled with elements from either yes or no
do.call(funname, args) executes a function call from the name of the function and a list of arguments to be passed to it

Writing functions

function(arglist) expr function definition,
missing test whether a value was specified as an argument to a function
require load a package within a function
<<- attempts assignment within parent environment before search up thru environments
on.exit(expr) executes an expression at function end
return(value) or **invisible**

Strings

paste(vectors, sep, collapse) concatenate vectors after converting to character; sep is a string to separate terms; collapse is optional string to separate "collapsed" results; see also str_c below
substr(x,start,stop) get or assign substrings in a character vector. See also str_sub below
strsplit(x,split) split x according to the substring split
grep(pattern,x) searches for matches to pattern within x; see **?regex**
gsub(pattern,replacement,x) replace pattern in x using regular expression matching; sub() is similar but only replaces the first occurrence.
tolower(x), toupper(x) convert to lower/uppercase
match(x,table) a vector of the positions of first matches for the elements of x among table
x %in% table as above but returns a logical vector
pmatch(x,table) partial matches for the elements of x among table
nchar(x) # of characters. See also str_length below

***stringr* package** provides a nice interface for string functions:
str_detect detects the presence of a pattern; returns a logical vector
str_locate locates the first position of a pattern; returns a numeric matrix with col start and end. (str_locate_all locates all matches)
str_extract extracts text corresponding to the first match; returns a character vector (str_extract_all extracts all matches)
str_match extracts "capture groups" formed by () from the first match; returns a character matrix with one column for the complete match and one column for each group
str_match_all extracts "capture groups" from all matches ; returns a list of character matrices
str_replace replaces the first matched pattern; returns a character vector
str_replace_all replaces all matches.
str_split_fixed splits string into a fixed number of pieces based on a pattern; returns character matrix
str_split splits a string into a variable number of pieces; returns a list of character vectors
str_c joins multiple strings, similar to paste
str_length gets length of a string, similar to nchar
str_sub extracts substrings from character vector, similar to substr

Dates and Times

Class **Date** is dates without times. Class **POSIXct** is dates and times, including time zones. Class **timeDate** in *timeDate*includes financial centers.

***lubridate* package** is great for manipulating time/dates and has 3 new object classes:

interval class: time between two specific instants. Create with **new_interval()** or subtract two times. Access with **int_start()** and **int_end()**

duration class: time spans with <u>exact</u> lengths **new_duration()** creates generic time span that can be added to a date; other functions that create duration objects start with d: dyears(), dweeks()…

period class: time spans that may not have a consistent lengths in seconds; functions include: years(), months(), weeks(), days(), hours(), minutes(), and seconds()

ymd(date, tz), mdy(date, tz), dmy(date, tz) transform character or numeric dates to POSIXct object using timezone tz (*lubridate*)

Other time packages: *zoo, xts, its* do irregular time series; *TimeWarp* has a holiday database from 1980+; *timeDate* also does holidays; *tseries* for analysis and computational finance; *forecast* for modeling univariate time series forecasts; *fts* for faster operations; *tis* for time indexes and time indexed series, compatible with FAME frequencies.

Date and time formats are specified with:

%a, %A	Abbreviated and full weekday name.
%b, %B	Abbreviated and full month name.
%d	Day of the month (01-31)
%H	Hours (00-23)
%I	Hours (01-12)
%j	Day of year (001-366)
%m	Month (01-12)
%M	Minute (00-59)
%p	AM/PM indicator
%S	Second as decimal number (00-61)
%U	Week (00-53); first Sun is day 1 of wk 1
%w	Weekday (0-6, Sunday is 0)
%W	Week (00-53); 1st Mon is day 1 of wk 1
%y	Year without century (00-99) Don't use
%Y	Year with century
%z	(output only) signed offset from Greenwich; -0800 is 8 hours west of
%Z	(output only) Time zone as a character string

Graphs

There are three main classes of plots in R: base plots, grid & lattice plots, and *ggplot2* package. They have limited interoperability. Base, grid, and lattice are covered here. *ggplot2* needs its own reference sheet.

Base graphics

Common arguments for base plots:

add=FALSE if TRUE superposes the plot on the previous one (if it exists)

axes=TRUE if FALSE does not draw the axes and the box

type="p" specifies the type of plot, "p": points, "l": lines, "b": points connected by lines, "o": same as previous but lines are over the points, "h": vertical lines, "s": steps, data are represented by the top of the vertical lines, "S": same as previous but data are represented by the bottom of the vertical lines

xlim=, ylim= specifies the lower and upper limits of the axes, for example with xlim=c(1, 10) or xlim=range(x)

xlab=, ylab= annotates the axes, must be variables of mode character main= main title, must be a variable of mode character

sub= sub-title (written in a smaller font)

Base plot functions

plot(x) plot of the values of x (on the y-axis) ordered on the x-axis

plot(x, y) bivariate plot of x (on the x-axis) and y (on the y-axis)

hist(x) histogram of the frequencies of x

barplot(x) histogram of the values of x; use horiz=TRUE for horizontal bars

dotchart(x) if x is a data frame, plots a Cleveland dot plot (stacked plots line-by-line and column-by-column)

boxplot(x) "box-and-whiskers" plot

stripplot(x) plot of the values of x on a line (an alternative to boxplot() for small sample sizes)

coplot(x˜y | z) bivariate plot of x and y for each value or interval of values of z

interaction.plot (f1, f2, y) if f1 and f2 are factors, plots the means of y (on the y-axis) with respect to the values of f1 (on the x-axis) and of f2 (different curves); the option fun allows to choose the summary statistic of y (by default fun=mean)

matplot(x,y) bivariate plot of the first column of x vs. the first one of y, the second one of x vs. the second one of y, etc.

fourfoldplot(x) visualizes, with quarters of circles, the association between two dichotomous variables for different populations (x must be an array with dim=c(2, 2, k), or a matrix with dim=c(2, 2) if k=1)

assocplot(x) Cohen-Friendly graph showing the deviations from independence of rows and columns in a two dimensional contingency table

mosaicplot(x) 'mosaic' graph of the residuals from a log-linear regression of a contingency table

pairs(x) if x is a matrix or a data frame, draws all possible bivariate plots between the columns of x

plot.ts(x) if x is an object of class "ts", plot of x with respect to time, x may be multivariate but the series must have the same frequency and dates

ts.plot(x) same as above but if x is multivariate the series may have different dates and must have the same frequency

qqnorm(x) quantiles of x with respect to the values expected under a normal distribution

qqplot(x, y) diagnostic plotr of quantiles of y vs. quantiles of x; see also qqPlot in *cars* package and distplot in *vcd* package

contour(x, y, z) contour plot (data are interpolated to draw the curves), x and y must be vectors and z must be a matrix so that dim(z)= c(length(x), length(y)) (x and y may be omitted). See also filled.contour, image, and persp

symbols(x, y, ...) draws, at the coordinates given by x and y, symbols (circles, squares, rectangles, stars, thermometers or "boxplots") with sizes, colours . . . are specified by supplementary arguments

termplot(mod.obj) plot of the (partial) effects of a regression model (mod.obj)

colorRampPalette creates a color palette (use: colfunc <- colorRampPalette(c("black", "white")); colfunc(10)

Low-level base plot arguments

points(x, y) adds points (the option type= can be used)

lines(x, y) same as above but with lines

text(x, y, labels, ...) adds text given by labels at coordinates (x,y); a typical use is: plot(x, y, type="n"); text(x, y, names)

mtext(text, side=3, line=0, ...) adds text given by text in the margin specified by side (see axis() below); line specifies the line from the plotting area segments(x0, y0, x1, y1) draws lines from points (x0,y0) to points (x1,y1)

arrows(x0, y0, x1, y1, angle= 30, code=2) same as above with arrows at points (x0,y0) if code=2, at points (x1,y1) if code=1, or both if code=3; angle controls the angle from the shaft of the arrow to the edge of the arrow head

abline(a,b) draws a line of slope b and intercept a abline(h=y) draws a horizontal line at ordinate y abline(v=x) draws a vertical line at abcissa x

abline(lm.obj) draws the regression line given by lm.obj

rect(x1, y1, x2, y2) draws a rectangle with left, right, bottom, and top limits of x1, x2, y1, and y2, respectively

polygon(x, y) draws a polygon linking the points with coordinates given by x and y

legend(x, y, legend) adds the legend at the point (x,y) with the symbols given by legend

title() adds a title and optionally a sub-title

axis(side, vect) adds an axis at the bottom (side=1), on the left (2), at the top (3), or on the right (4); vect (optional) gives the abcissa (or ordinates) where tick-marks are drawn

rug(x) draws the data x on the x-axis as small vertical lines

locator(n, type="n", ...) returns the coordinates (x, y) after the user has clicked n times on the plot with the mouse; also draws symbols (type="p") or lines (type="l") with respect to optional graphic parameters (...); by default nothing is drawn (type="n")

Plot parameters

These can be set globally with par(...); many can be passed as parameters to plotting commands.

adj controls text justification (0 left-justified, 0.5 centred, 1 right-justified)

bg specifies the colour of the background (ex. : bg="red", bg="blue", . . the list of the 657 available colours is displayed with colors())

bty controls the type of box drawn around the plot, allowed values are: "o", "l", "7", "c", "u" ou "]" (the box looks like the corresponding character); if bty="n" the box is not drawn

cex a value controlling the size of texts and symbols with respect to the default; the following parameters have the same control for numbers on the axes, cex.axis, the axis labels, cex.lab, the title, cex.main, and the sub-title, cex.sub

col controls the color of symbols and lines; use color names: "red", "blue" see colors() or as "#RRGGBB"; see rgb(), hsv(), gray(), and rainbow(); as for cex there are: col.axis, col.lab, col.main, col.sub

font an integer that controls the style of text (1: normal, 2: italics, 3: bold, 4: bold italics); as for cex there are: font.axis, font.lab, font.main, font.sub

las an integer that controls the orientation of the axis labels (0: parallel to the axes, 1: horizontal, 2: perpendicular to the axes, 3: vertical)

lty controls the type of lines, can be an integer or string (1: "solid", 2: "dashed", 3: "dotted", 4: "dotdash", 5: "longdash", 6: "twodash", or a string of up to eight characters (between "0" and "9") that specifies alternatively the length, in points or pixels, of the drawn elements and the blanks, for example lty="44" will have the same effect than lty=2

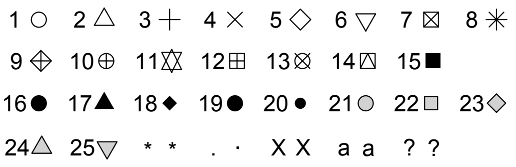
lwd numeric that controls the width of lines, default 1

mar a vector of 4 numeric values that control the space between the axes and the border of the graph of the form c(bottom, left, top, right), the default values are c(5.1, 4.1, 4.1, 2.1)

mfcol a vector of the form c(nr,nc) that partitions the graphic window as a matrix of nr lines and nc columns, the plots are then drawn in columns

mfrow same as above but the plots are drawn by row

pch controls the type of symbol, either an integer between 1 and 25, or any single char within ""



ps an integer that controls the size in points of texts and symbols

pty a character that specifies the type of the plotting region, "s": square, "m": maximal

tck a value that specifies the length of tick-marks on the axes as a fraction of the smallest of the width or height of the plot; if tck=1 a grid is drawn

tcl a value that specifies the length of tick-marks on the axes as a fraction of the height of a line of text (by default tcl=-0.5)

xaxt if xaxt="n" the x-axis is set but not drawn (useful in conjonction with axis(side=1, ...))

yaxt if yaxt="n" the y-axis is set but not drawn (useful in conjonction with axis(side=2, ...))

Lattice graphics

Lattice functions return objects of class trellis and must be printed. Use print(xyplot(...)) inside functions where automatic printing doesn't work. Use lattice.theme and lset to change Lattice defaults. In the normal Lattice formula, y x|g1*g2 has combinations of optional conditioning variables g1 and g2 plotted on separate panels. Lattice functions take many of the same args as base graphics plus also data= the data frame for the formula variables and subset= for subsetting. Use panel= to define a custom panel function (see apropos("panel") and ?llines).

xyplot(y˜x) bivariate plots (with many functionalities)

barchart(y˜x) histogram of the values of y with respect to those of x

dotplot(y˜x) Cleveland dot plot (stacked plots line-by-line and column-by-column)

densityplot(˜x) density functions plot histogram(˜x) histogram of the frequencies of x bwplot(y˜x) “box-and-whiskers” plot

qqmath(˜x) quantiles of x with respect to the values expected under a theoretical distribution

stripplot(y˜x) single dimension plot, x must be numeric, y may be a factor

qq(y˜x) quantiles to compare two distributions, x must be numeric, y may be numeric, character, or factor but must have two ‘levels’

splom(˜x) matrix of bivariate plots

parallel(˜x) parallel coordinates plot

levelplot(z˜x*y|g1*g2) coloured plot of the values of z at the coordinates given by x and y (x, y and z are all of the same length)

wireframe(z˜x*y|g1*g2) 3d surface plot

cloud(z˜x*y|g1*g2) 3d scatter plot