R cheat sheet

Copied in part from https://cran.r-project.org/doc/contrib/Short-refcard.pdf

Getting help

`?topic` gives documentation on given topic.

Input & output

`install.packages("name")` installs the given package to this computer
`library(name)` load add-on packages
`read.table(file)` reads a file in table format and creates a data frame from it
`write.table(x,file="<file>")` saves data frame to file
`setwd(), getwd()` sets and returns the working directory

Data creation

`c(...)` generic function to combine arguments into a vector
`from:to` generates a sequence
`seq(from,to)` generates a sequence by= specifies increment; length= specifies desired length
`seq(along=x)` generates `1, 2, ..., length(along)`
`rep(x,times)` replicate x times; use each= to repeat "each" element of x each times
`data.frame(...)` create a data frame of the named or unnamed arguments; shorter vectors are recycled to the length of the longest
`list(...)` create a list of the named or unnamed arguments
`factor(x,levels=)` encodes a vector x as a factor
`gl(n,k,labels=<labels>)` generate levels (factors) by specifying the pattern of their levels; n is the number of levels, and k is the number of replications
`rbind(...)` combine arguments by rows for matrices, data frames, and others
`cbind(...)` idem by columns

Slicing & extracting data

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 the 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","and","the")]`	elements in the given set

Indexing lists

`x[n]`	list with elements n
`x[[n]]`	nth element of the list
`x[["name"]]`	element of the list named "name"
`x$name`	idem

Indexing data frames

`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"
`x[["name"]]`	column named "name"
`x$name`	idem

Variable conversion

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

Variable information

`is.na(x), is.null(x), is.array(x), is.data.frame(x), is.numeric(x), is.complex(x), is.character(x), ...` test for type; for a complete list, use `methods(is)`
`length(x)` number of elements in x
`dim(x)` retrieve or set the dimension of an object
`nrow(x)` number of rows
`ncol(x)` number of columns
`str(object)` display the internal *str*ucture of an R object
`summary(object)` gives a "summary" of `object`, usually a statistical summary, but it is generic, meaning it has different operations for different classes
`complete.cases(x)` gives a logical vector indicating which cases (rows of a dataframe) have no missing values
`class(x)` gives the data type of x

Data selection and manipulation

`with(x, ...)` execute statements within a local environment
`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))`
`order(x)` returns a permutation which rearranges its first argument into ascending or descending order
`cut(x,breaks)` divides x into intervals (factors)
`match(x, y)` returns a vector of the same length than x with the elements of x which are in y (NA otherwise)
`which(expr on x)` returns a vector of the indices of x if the comparison operation is true
`na.omit(x)` suppresses the observations with missing data (NA) (suppresses the corresponding line if x is a matrix or a data frame)
`unique(x)` if x is a vector or a data frame, returns a similar object but with the duplicate elements suppressed
`table(x)` returns a table with the numbers of the different values of x (typically for integers or factors)
`subset(x, ...)` returns a selection of x with respect to criteria (...); if x is a data frame, the option select gives the variables to be kept (or dropped using a minus sign)
`sample(x, size)` sample randomly from x

Math & Descriptive statistics

`max(x)` maximum of the elements of x
`min(x)` minimum of the elements of x
`range(x)` gives `c(min(x), max(x))`
`sum(x)` sum of the elements of x
`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)
`IQR(x)` interquantile range
`var(x)` or cov(x) variance of the elements of x (calculated on n−1);
`sd(x)` standard deviation of x
`round(x, n)` rounds the elements of x to n decimals
`log(x, base)` computes the logarithm of x with base base
`scale(x)` if x is a matrix, centers and reduces the data

Advanced data processing

`apply(X,INDEX,FUN=)` a vector or array or list of values obtained by applying function FUN to margins (INDEX) of X
`lapply(X,FUN)` apply FUN to each element of the list X; **`sapply`** is a wrapper for lapply that tries to simplify the result.
`tapply(X,INDEX,FUN=)` apply FUN to each cell of a ragged array given by X with indexes INDEX
`merge(a,b)` merge two data frames by common columns or row names
`aggregate(x,by,FUN)` splits the data frame x into subsets, computes summary statistics for each, and returns the result in a convenient form; by is a list of grouping elements, each as long as the variables in x

Strings

`paste(...)` concatenate vectors after converting to character
`substr(x,start,stop)` substrings in a character vector; can also assign, as `substr(x, start, stop) <- value`
`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)` replacement of matches determined by regular expression matching; **`sub()`** is the same but only replaces the first occurrence.
`tolower(x)` convert to lowercase
`toupper(x)` convert to uppercase
`x %in% table` a logical vector of the matches for the elements of x among table
`nchar(x)` number of characters

Plotting

`plot(x)` plots 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
`pie(x)` circular pie-chart
`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
`pairs(x)` if x is a matrix or a data frame, draws all possible bivariate plots between the columns of x
`qqnorm(x)` quantiles of x with respect to the values expected under a normal law
`qqplot(x, y)` quantiles of y with respect to the quantiles of x
`matplot(x, y)` plot the columns of one matrix against the columns of another, or the columns of a matrix (or dataframe).

The following parameters are common to many plotting functions:
`add=FALSE` if TRUE superposes the plot on the previous one
`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": id. but the lines are over the points, "h": vertical lines, "s": steps, the data are represented by the top of the vertical lines, "S": id. but the data are represented by the bottom of the vertical lines
`xlim=, ylim=` specifies the lower and upper limits of the axes
`xlab=, ylab=` annotates the axes with labels
`main=` main title ; **`sub=`** sub-title

Low-level plotting commands

`points(x, y)` adds points (the option type= can be used)
`lines(x, y)` id. but with lines
`text(x, y, labels, ...)` adds text given by labels at coordinates (x,y)
`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
`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

Graphical parameters

These can be set globally with `par(...)`; many can be passed as parameters to plotting commands. Type **`?par`** for more info
`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 which controls the style of text; as for cex there are: font.axis, font.lab, font.main, font.sub
`las` an integer which controls the orientation of the axis labels
`lty` controls the type of lines
`lwd` a numeric which controls the width of lines, default 1
`mar` a vector of 4 numeric values which control the space between the axes and the border of the graph: c(bottom, left, top, right)
`mfcol` a vector of the form c(nr,nc) which partitions the graphic window as a matrix of nr lines and nc columns, the plots are then drawn in columns
`mfrow` id. but the plots are drawn by row
`pch` controls the type of symbol, either an integer between 1 and 25, or any single character within ""

Optimization and model fitting

`lm(formula)` fit linear models
`loess(formula)` fit a polynomial surface using local fitting. Many of the formula-based modeling functions have several common arguments: `data=` the data frame for the formula variables, `subset=` a subset of variables used in the fit, `na.action=` action for missing values: "na.fail", "na.omit", or a function. The following generics often apply to model fitting functions:
`predict(fit,...)` predictions from fit based on input data

Statistics

`aov(formula)` analysis of variance model
`anova(fit,...)` analysis of variance (or deviance) tables for one or more fitted model objects
`density(x)` kernel density estimates of x
`binom.test(), pairwise.t.test(), power.t.test(), prop.test(), t.test(), ...` use `help.search("test")`

Distributions

`rnorm(n, mean=0, sd=1)` Gaussian (normal)
`rbinom(n, size, prob)` binomial
All these functions can be used by replacing the letter r with d, p or q to get, respectively, the probability density (dfunc(x, ...)), the cumulative probability density (pfunc(x, ...)), and the value of quantile (qfunc(p, ...), with 0 < p < 1).
NB: There are many more distributions!

Programming

`function( arglist ) expr` function definition
`return(value)`
`if(cond) expr`
`if(cond) cons.expr else alt.expr`
`for(var in seq) expr` loop over seq and put the elements in var
`while(cond) expr` loop while condition is true
`break` breaks out of loop iteration
`next` goes to next loop iteration
Use braces {} around statements
`ifelse(test, yes, no)` a value with the same shape as test filled with elements from either yes or no