R Basic Reference Card

`?topic` documentation on `topic`
`help.search("topic")` search the help system
`apropos("topic")` all objects matching the reg. exp. "topic"
`ls()` list all objects; `pat="pat"` to search on a pattern

Input and output

`source("my.R")` includes and executes my.R in this place
`data(f)` loads specified data sets
`install.package()`
`library(s)` load add-on packages
`read.table(f)` reads a file into data frame; `sep=""` for value separator; `header=TRUE` names on the first line; `as.is=TRUE` prevent string to factor conversion
`read.csv(f,header=TRUE)` same, for comma-delimited files
`read.delim(f",header=TRUE)` same, for tab-delimited files
`read.fwf(f,widths,header=FALSE,sep="",as.is=FALSE)` for *f*ixed *w*idth *f*ormatted data
`save(f,...)` saves the specified objects (...) in the XDR format
`save.image(f)` saves all objects
`load(f)` load the datasets written with `save`
`print(a, ...)` prints its arguments; *generic* function
`format(x,...)` format an R object for pretty printing
`write.table(x, file=f,row.names=TRUE, col.names=TRUE, sep=" ")` convert `x` to data frame and output to `f`; see params: `quote`, `sep`, `eol`, `na`, `col.names=NA`
`sink(f)` output to `f`, until `sink()`
Exchange tables with other apps. (Excel) via clipboard:
`df <- read.table("clipboard")`
`write.table(df,"clipboard",sep="\t",col.names=NA)`

Data creation

`c(...)` combine arguments to vector, *generic*, see param `recursive`
`from:to` generates a sequence;`2:5` returns `[1] 2 3 4 5`
`seq(from,to)` generates a sequence, `by=` set step; `length=` set length
`seq(along=x)` generates `1, 2, ..., length(x)`; useful for loops
`rep(x,n)` replicate `x` `n`-times (abcabc); `each=` to get (aabbcc)
`data.frame(...)` data frame from the list of vector parameters; `data.frame(v=1:6,ch=c("a","B"))`; recycle shorter vectors
`list(...)` create a list of the named or unnamed arguments; `list(a=c(1,2),b="hi",c=3i)`;
`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
`outer(x,y, FUN(x,y) )` create matrix by tensor product
`factor(x,levels=)` encodes a vector `x` as a factor
`gl(n,k,length=n*k,labels=1:n)` factor with `n` labels *k*-times each
`expand.grid()` data frame of all combinations of given lists
`cbind(df1, df2)`, `rbind(df1,df2)` combine arguments by columns/rows for matrix-like objects

Object operations

`rm(myvar)` removes object `myvar` from memory
`rm(list = ls(all = TRUE))` removes all objects from memory
`class(X)` class of object `X` (get type of data)
`as.array(x)`, `as.data.frame(x)`, `as.numeric(x)`, ... convert type; `methods(as)` shows a complete list
`is.na(x)`, `is.null(x)`, `is.array(x)`, `is.numeric(x)`, ... test for type; `methods(is)`
`summary(a)` gives a "summary" of `a`, *generic* function
`length(x)` number of elements in `x`
`dim(x)` get or set the dimension of an object; `dim(x) <- c(3,2)`
`dimnames(x)` get or set the dimension names of an object
`nrow(x)`, `ncol(x)` number of rows/cols of matrix (cf. `NROW`, `NCOL`)

addressing vectors

`v[n]`	n^{th} element
`v[-n]`	all *but* the n^{th} element
`v[1:n]`	first `n` elements
`v[-(1:n)]`	elements from `n+1` to the end
`v[c(1,4,2)]`	specific elements
`v["name"]`	element named `"name"`
`v[v > 3 & v < 5]`	all elements between 3 and 5
`v[v %in% set]`	elements in vector `set`

addressing lists

`x[n]` or `x[[n]]`	n^{th} element of the list
`x[["name"]]` or `x$name`	element of the list named `"name"`

addressing matrices and dataframes

`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"`
only for dataframes:	
`df[["name"]]` or `df$name`	column named `"name"`

Data selection and manipulation

`which.max(v)`, `which.min(v)` index of the max/min element of `v`
`rev(v)` reverses the elements of `v`
`sort(v)` sorts `v` (increasing); use `rev(sort(x))` for decreasing
`cut(x,breaks)` divides `x` into intervals (laels of resulting factor) `breaks` is number of intervals or vector of cut points
`match(x, y)` for every `x[i]` index of same value in `y`, `NA` otherwise
`which(x)` indices of TRUE in logical vector `x`; e.g. `which(x>7)`
`na.omit(x)` suppress `NA` values (or lines of matrix or data frame)
`na.fail(x)` returns an error message if `x` contains at least one `NA`
`unique(x)` suppress duplicate values (or lines of matrix or DF)
`table(x)` freqency/**contingency table** for vector/data frame
`subset(df, x)` lines where `x` is TRUE; e.g. `subset(df, V1 < 5)`
`sample(x, N)` random sample of size `N` from vector `x`;`replace=FALSE`
`prop.table(x,margin=)` table entries as fraction of marginal table

Math

`sin,cos,tan,asin,acos,atan,atan2,log,log10,exp`
`range(x)` shortcut for `c(min(x), max(x))`
`sum(x)` sum of the elements of `x`
`diff(x)` iterated differences of vector `x`
`prod(x)` product 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; `type=` nine types of approx.
`weighted.mean(x, w)` mean of `x` with weights `w`
`rank(x)` ranks of the elements of `x`
`var(x)` variance of the elements of `x`;
`sd(x)` standard deviation of `x`
`cov(x)` covariance matrix of the matrix or data frame`x`
`cor(x)` correlation matrix of `x` see param. `method=`
`var(x, y)` or `cov(x, y)` covariance between `x` and `y`
`cor(x, y)` linear correlation between `x` and `y` (also for matrices)
`choose(n, k)` k-combinations from n elements $= n!/[(n-k)!k!]$
`round(x, n)` rounds the elements of `x` to `n` decimals
`scale(x)` normalization of vector (matrix, df) x
`pmin(x,y,...)`, `pmax(x,y,...)` parallel minimum, maximum
`cumsum(v)` vector with i^{th} element `sum(v[:i])`
`cumprod(v)`, `cummin(v)`, `cummax(v)` ... similar
Many math functions have a logical parameter `na.rm=FALSE` to specify missing data (NA) removal.

Matrices

`t(x)` transpose
`diag(x)` diagonal
`%*%` matrix multiplication and scalar product
`solve(a,b)` solves `a %*% x = b` for `x`
`solve(a)` matrix inverse of `a`
`rowsum(x)`, `colsum(x)`, `rowmean`, `colmean` sum/mean of rows/cols for a matrix-like object

Advanced data processing

`function( arglist ) expr` function definition; `return(value)`
`lapply(X,FUN)` apply `FUN` to each element of the list `X`
`by(data,FACTOR,FUN,...)` split data frame `data` by `FACTOR` of same length and apply `FUN` to resulting data frames
`merge(a,b)` merge two data frames (default by common columns)
`ftable(xtabs(cols rows,data=x))` a contingency table from DF
`stack, unstack(x,...)` convert list of factor vectors to/from DF

Distributions

`rnorm(n, mean=0, sd=1)` Gaussian (normal)
`rlnorm(n, meanlog=0, sdlog=1)` lognormal
`rweibull(n, shape, scale=1)` Weibull
`rgamma(n, shape, scale=1)` gamma
`rbeta(n, shape1, shape2)` beta
`rt(n, df)` 'Student' (t)
`rf(n, df1, df2)` Fisher–Snedecor (F)
`rchisq(n, df)` Pearson (χ^2)
`rexp(n, rate=1)` exponential
`rpois(n, lambda)` Poisson
`rcauchy(n, location=0, scale=1)` Cauchy
`rbinom(n, size, prob)` binomial
`rhyper(nn, m, n, k)` nn-white drown, m-white, n-black, k-drown
`rgeom(n, prob)` geometric
`rnbinom(n, size, prob)` negative binomial
`rlogis(n, location=0, scale=1)` logistic
`runif(n, min=0, max=1)` uniform
`rwilcox(nn, m, n)`, `rsignrank(nn, n)` Wilcoxon's statistics
`d<distr>(x, ...)` density function
`p<distr>(x, ...)` distribution (CDF)
`q<distr>(p, ...)` quantile function ($0 \le p \le 1$)

Tests and confidence interavls

`t.test()` Student's test
`pairwise.t.test()` ...corrections for ANOVA posthoc analysis
`power.t.test()` power of t-test
`binom.test()` exact test for p
`var.test()` F-test for variance of normal distribution
`fisher.test()` exact test for independence in contingency table
`ks.test()` one or two sample Kolmogorov-Smirnov test
`kruskal.test(x)` Kruskal-Wallis ("robust" ANOVA for list `x`)
`shapiro.test()` test of normality
`apropos("test")` for full list

Statistics

`approx(x,y=)` piecewise linear or constant interpolation
`spline(x,y=)` cubic spline interpolation
`lm(formula, data=, subset=)` fit linear model; `formula` is typically of the form `response termA + termB + ...`; use `I(x*y) + I(x^2)` for terms made of nonlinear components
`aov(formula)` analysis of variance model

formulas

`x+c:d` with interaction term $y = \beta_0 + \beta_x x + \beta_{cd}$;
`a-1` without intercept
`c*d` same as `c+d+c:d` same as `(c+d)^2`
`I(x^2)` $y = \beta_0 + \beta_2 x^2$

work with fit

`predict(fit,...)` predictions from `fit` based on input data
`df.residual(fit)` number of residual degrees of freedom
`coef(fit)` returns the estimated coefficients
`residuals(fit)` returns the residuals
`deviance(fit)` returns the deviance
`fitted(fit)` returns the fitted values
`anova(fit,...)` ANOVA table
`plot(fit)`

Basic Plotting

`plot(y)` plot of the values of `y` (on the y-axis) ordered on the x-axis
`plot(vx, vy)` scatter plot; points (`vx[i]`, `vy[i]`)
`matplot(x,y)` i^{th} col of matrix x vs. i^{th} col of y taking i^{th} value from vector plot params (col, bg, pch,..)
`stripchart(x)` plot of the values of `x` on a line
`hist(x)` histogram of the frequencies of `x`
`barplot(x)` histogram of the values of `x`; horizontal: `horiz=FALSE`
`pie(x)` circular pie-chart
`boxplot(x)` "box-and-whiskers" plot
`pairs(x)` plot for every pair of columns in data frame `x`
`pairs(~V1+V2+V3,data=df)` only for columns 1,2,3
`plot.ts(x)` plot time series object `x`
`ts.plot(x)` multivariate series may have different dates
`qqnorm(x)` quantiles of `x` with respect to `qnorm()`
`qqplot(x, y)` quantiles of `y` with respect to the quantiles of `x`
`contour(x, y, z)` `x`, `y` - vectors; `z`= $z(x, y)$
`filled.contour(x, y, z)` same + fill between contours
`image(x, y, z)` plot z as colour
`persp(x, y, z)` 3D graph with perspective
`dotchart(x)` Cleveland dot plot ...TODO
`coplot(x~y | z)` conditioning scatter plots for each value or interval of values of `z`
`stars(x)` draw star for every row of df `x`; `draw.segments=`
`symbols(x, y, ...)` draw parametic symbols (termometer, circle, ...) at `x`,`y` points

Statistic Plotting

`fourfoldplot(x)` plot 2 by 2 by k contingency table $x$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
`interaction.plot (f1, f2, y)` plot means of `y` (y-axis) for factors `f1` (x-axis) and `f2` (line type)
`termplot(mod.obj)` plot (partial) effects of a regression model

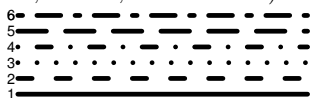
Low-level plotting commands

`points(x, y)` adds points (the option `type=` can be used)
`lines(x, y)` id. but with lines
`text(x, y, labels, ...)` add text `labels` at points (`x`, `y`)
`mtext(text, side=3, ...)` add `text` at margin specified by `side`
`segments(x0, y0, x1, y1)` lines from (`x0`,`y0`) to (`x1`,`y1`)
`arrows(x0, y0, x1, y1, angle= 30, code=2)` same with arrows
`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)` rectangle with corners (`x1`, `y2`) (`x2`, `y2`)
`polygon(x, y)` polygon linking the points (`x`, `y`)
`legend(x, y, legend)` add legend (`x`,`y`)
`title()` adds a title and optionally a sub-title
`axis(side, ...)` add axis on side (1=below, 2=left, 3=above, 4=right)
`rug(v)` draws the data on the x-axis as small vertical lines
`locator(N, type=''n'', ...)` returns coordinates (x, y) after the user has clicked `n` times on the plot; also draws symbols `type`

common plot params

`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"`: 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, 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)

Graphical parameters

`par(...)` set grapics parameters globally
`adj` adjustement of text (0=left, 0.5=center, 1=right)
`bg` background color (fill); `colors()` list color names
`bty` box type around plot, (`"o"`, `"l"`, `"7"`, `"c"`, `"u"`, `"]"`) or `bty="n"` for none
`cex` scale size of texts and symbols (also: `cex.axis`, ...)
`col` color of symbols and lines; use color names or `"#RRGGBB"`; see `rgb()`, `hsv()`, `gray()`, and `rainbow()`;
`font` style of text (1: normal, 2: italics, 3: bold, 4: bold italics)
`las` orientation of axis labels
0: parallel, 1: horizontal,
2: perpendicular, 3: vertical


`lty` line type:
`lwd` line width
`mfrow=c(nr,nc)` set matrix `nr` × `nc` of subplots
`pch` point type (integer code) or single character and 25, or any single character within `""`

1 ○ 2 △ 3 + 4 × 5 ◇ 6 ▽ 7 ⊠ 8 ✳ 9 ⊕ 10 ⊕ 11 ⧖ 12 ⊞ 13 ⊠ 14 ⧄ 15 ■
16 ● 17 ▲ 18 ◆ 19 ● 20 ● 21 ○ 22 □ 23 ◇ 24 △ 25 ▽ * ⋆ . · X X a a ? ?

`ps` size of texts
`pty` plotting region type, `"s"`: square, `"m"`: maximal
`tck` tick's length as fraction of min of plot height and width
`tcl` tick's length as fraction of text height (default `tcl=-0.5`)

Lattice (Trellis) graphics

`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