R reference card

Essentials

`q()` quit. You will be asked if "Save workspace?" type "y" to save to .RData in current directory

`<-` or `=` assignment, e.g.: `x <- 13.76`

`help(command1)` gives syntax, details & examples

Extensions

`help.start()` start browser help

`apropos("topic1")` lists commands relevant to topic1

`help.search("topic1")` like apropos, but gives short description

`RSiteSearch("topic1")` like help.search plus a google search on topic1 at the R-project site. Returns output to your browser.

`example(command1)` examples of `command1`

`demo(package1)` demos related to package1

Numbers and Matrices

`v1 <- c(1,2,3.4)` creates a string of numbers with no dimension

`1:3` a string of integers `1,2,3` (with no dimensions)

`rep(x1,n1)` repeats the vector `x1 n1` times

`matrix(v1,r1,c1)` make `v1` into a matrix with `r1` rows and `c1` columns.
Note: matrices are stored as stacked columns.

`cbind(a1,b1,c1)` binds columns into a matrix

`rbind(a1,b1,c1)` binds rows into a matrix

`dim(matrix1)` dimensions of `matrix1`

`length(v1)` length of `v1`

`m1[4,3]` element of matrix `m1` in 4th row, 3rd column

`m1[,2]` column 2 of matrix `m1`

`m1[,2:5]` or `m1[,c(2,3,4,5)]` columns 2 thru 5

`m1[6:4,]` or `m1[,c(6,5,4)]` rows 6 thru 4

`t(m1)` transpose matrix, switch rows and columns

`dimnames(m1)` returns or assigns names to rows/columns of `m1`

`%*%` matrix multiplication

Arithmetic

−, +, `*`, and `/` are applied element–wise to matrices. The shorter of two vectors is recycled to the length of the longer. A warning is printed if lengths are not even multiples. Use `options(warn=2)` to make this an error.

`^` exponents, `sqrt()` square root

`%/%` integer divide: `27 %/%4 = 6`

`%%` modulus or remainder: `27 %% 4 = 3`.

Statistics

`max(), min(), mean(), median(), sum(), var()` as named

`cor(m1), cor(x1,y1)` show correlations within matrix `m1` or between `x1` and `y1`

`summary(x1)` prints quartiles, mean, min, and max

`summary(data.frame)` prints summary of each column

`sort()` sort, also see help for `order()`

`quantiles(x1, .9)` find the 90th percentile

`rnorm(n1, mean,sd)` generate `n1` random normals

`rchisq(), rf(), runif(), rbinom()` generate random variates

`pnorm(), pchisq(), pf()` (CDF) Statistical tables for p-values. Use 1 - these to get upper tail probs.

`qnorm(), qchisq(), qf()` quantiles, inverse CDF.

`by()` apply function to data frame by factor e.g. `by(x1, g1, mean)`

`apply(x1,n1,function1)` apply `function1` (e.g. `mean`) to `x1` by rows (`n1=1`) or columns (`n1=2`)

`tapply(x1,list1,function1)` apply `function` to `x1` split by `list1`

`table(f1, f2)` make a table of occurrence counts

Data Frames

`read.table("file1")` read data from `file1` into a dataframe, which is a special type of list.

`data.frame(x=x1, y=y1)` creates a dataframe with 2 columns, x and y

`m1$a1` variable `a1` in data frame `m1`

`NA` missing data (use in a data file)

`is.na(x1)` returns true if `x1 == NA`, i.e.`x1` is missing

Input and Output

`source("file1")` run the commands in `file1`.

`data.entry(x1,y1)` pops up a primitive spreadsheet allowing modification to `x1` or `y1`.

`scan("file1")` read a file (or keyboard input if "file" is omitted) into a single vector

`sink("file1")` output to `file1`, until `sink()`

`write(object, "file1")` writes an object to `file1`

`write.table(dataframe1,"file1")` writes a table or matrix see its options for quotes, format, and labels

Managing Variables and Objects

`ls()` lists all objects in workspace.

`rm(object1)` removes `object1` from workspace

`search()` view your search path

`attach(x1)` put variables in dataframe `x1` into search path so that `a1` can be used for `x1$a1`.

`detach(x1)` remove from search path

`library(nlme)` load (e.g.) the `nlme` package

`as.matrix(), as.numeric()` conversions

`factor(x1), ordered(x1)` convert numeric `x1` to a factor or ordered factor

`is.factor(), is.matrix(), is.numeric()` look for attributes

`which(x1==a1)` returns indices of `x1` where `x1==a1`

Basic Statistical Analysis

`t.test(x1,y1)` t test (1 or 2 samples)

`wilcox.test(x1)` Wilcoxson's median test

`lm()` linear models: regression, anova, ancova

`aov(formula)` specialized anova function

`anova()` compares two or more linear models (LRT).
`kruskal.test(x1,g1)` Kruskal-Wallis test for equal medians in `x1` over groups `g1`.

Programming

`function(x1,v1)` build a function with 2 args
e.g. `sd <- function(x1){ sqrt(var(x1)) }`
`for (i1 in 1:n1) { stuff }` repeat "stuff" `n1` times
Logical Comparisons: `==, <=, >=` Note 2 ='s. Usage:
`if (condition1) {somestuff} else {otherstuff}`
`while (condition1) {stuff}` repeat "stuff" until condition1 is false
`break` jumps out of a loop
`switch` avoids several `if` statements
`next` jumps to end of a loop
`ifelse` applies condition to every element of a vector

Graphics

`plot(x1,y1)` scatterplot, alternatively:
`plot(y1 ~ x1, data = df1)`
Options within `plot()`: (separate with commas)
`type="p"` for points, "l" for lines, or "b" for both
`xaxt="n"` omit x axis, `yaxt="n"` omit y axis
`lty = 2` dashed lines use integers > 1
`pch = 15` set plotting character to letter or integer
`main = "String")` add a main title
`xlab = "Lab1", ylab="Lab2"` set axis labels

`abline(int1, slope1)` add a line to plot
`abline(h=0), abline(v=22)` horiz. or vert. line
`points(x1,y1)` add more points to a plot
`lines(x1,y1)` add lines to an existing plot
add smoother: `lines(loess(x1, y1))`
`text(x1,y1, text1)` add text to plot
`axis()` or `mtext()` to create an axis
`legend(x1, y1, labels1, lty=lty1, pch = pch1)` add a legend at coordinates `x1, y1`.
`stem(x1), hist(x1)` stem-and-leaf and histogram
`boxplot(x1)` box-whisker plot (single)
`boxplot(x1 ~ g1)` box-whisker plot by group
`pairs(m1)` matrix of scatterplots
`qqnorm(x1), qqline(x1)` compare `x1` to normal dist'n
`interaction.plot(Xfactor1, TraceFactor2, y1)` plot means for 2-way anova

Plotting Devices

`x11()` open a plot window on Unix system
`windows()` same for MSWindows. Note different menus when plotting window is active.
`postscript("file1.ps", horiz=F, height=6, width=6, paper="special")` open a device to save plots to `file1.ps`
`dev.off()` to finish the file
Jpeg, png, and other formats available, see `?Devices`.

Lattice Graphics

`library(lattice)` load the library
`xyplot(y1 ~ x1|g1)` scatterplot of `y1` over `x1` separated by group `g1`
`bwplot(y1 ~ g1)` box-whisker plot
`barchart() dotplot() stripplot()` and others
`trellis.par.set(theme = col.whitebg())` set white background

Linear Models

`lm(y1 ~ x1, data = df1)`
If `x1` is quantitative, a regression of `y1` on `x1`.
If x1 is a factor, the analysis of variance.
Formula: the first argument of `lm()` can have the form
`y ~ x1 + x2 + x3` main effects for 3 predictors
`y ~ x1 + x2 + x1:x2` main effects and interactions
shorthand versions:
`y ~ x1 * x2` or
`y ~ (x1 + x2)^2`
To enforce arithmetic within a formula use `I()` as in
`y ~ x1 + I(x1^2)` (quadratic in `x1`)

`lm1 <- lm(formula1)` a linear models object
`summary(lm1)` prints coefficient estimates and F test for $H_0: \boldsymbol{\beta} = \mathbf{0}$
`update(lm1, formula2)` shortcut to modify `lm1`
`anova(lm1, lm2)` gives LRT for nested models
`predict(lm1, newdata = df2)` prediction and confidence intervals for new x values
`par(mfrow=c(2,2)); plot(lm1)` plots 4 plots:
Residuals vs Fitted to look for curvature
Normal Q-Q plot to examine normality assumptions
Scale-Location plot to look for non-constant variance
Cook's distance plot to look for influential points

Mixed Models

`lme(fixed=formula1, data=df1, random=formula2, corr = structure, weights = variance.structure)` linear mixed effects
Example formulae:
`random = ~ 1| g1` random intercept for each group
`random = ~ x1 | g1` random intercept & slope (over x1) for each group
`corr= corCompSymm(form = ~ 1| g1)` same correlation within group
`corr = corAR1(form = ~ 1| Subj)` AR1 correlations w/in Subject
`weights= varIdent(form = ~ 1|Year)` variance changes with year
`weights= varPower(form = ~ fitted(.) | g1)` variance increases as power of $E(Y)$, powers vary with group.
`gls(formula1, data=df1, corr = structure, weights = variance.structure)` generalized least squares. Use `corr` and `weights` as with lme.
`nlme()` nonlinear mixed models

Setting Options

`par(mfrow = c(2,3))` 6 plots/page (2 rows, 3 cols)
`options(contrasts = c("contr.treatment", "contr.poly" ))` set treatment contrast option

Jim Robison-Cox, August 2005 jimrc@math.montana.edu