Short Reference Sheet

Andreas Plank (version March 29, 2010)

Legend: Note	information/help, note
$f(x)$	a function
	a package or library
	an example
v2.8+	indicates a R-version

R is developed by humans who are sometimes lazy to write endless texts or commands and instructions. Therefore you'll find a lot of functions just briefly named like `c(..)` that *c*ombines something ;-) . . .

Basics (1)

> Help (1.1)

See FAQ either in the local HTML-Help system or online at http://cran.r-project.org/faqs.html.

```
?plot # for plot function (in LOADED libraries/packages)
??plot # searches in ANY library/package (v2.8+)
??utils::data # all for data-f(x) explicit in utils
# additional help settings
  options(help_type = "html") # browser is forced
  options(browser="kfmclient newTab") # alter default
# apropos f(x) lists currently available functions + vars
  apropos('plot')  # '*plot*'; BTW regular ↩
     expressions can be used:
  apropos('^plot') # 'plot*' - i.e. plot-something
  apropos('plot$') # '*plot' - i.e. something-plot
  ls() # lists all userdefined things
# getting examples/demos
  example(plot) # watch the example of plot-f(x)
  demo() # listing of all demos currently available
  demo(graphics) # R's graphics capabilities
  demo(image)    # image-like graphics of R
  demo(persp)    # extended persp() examples
  demo(plotmath) # examples of mathematical annotation
```

> Special characters (1.2)

Specific characters ensure the 'communication' with R. Space characters are ignored. See also `?Syntax`.

```
# Comments are ignored by R (there are no multi-line)
?
  ?plot # launch help system for plot-f(x)
:
  2:-1 # is short-cut for sequence, returns 2, 1, 0, -1
  utils::data() # explicit use of data-f(x) in utils
.
  2.5 # dot is decimal point; take care when import data
  help.search(..) # can be used in function names
,
  # separates f(x)-arguments, entries, ...
  c(2, 6, 10) # c-f(x) combines everything: some numbers
;
  plot(1:5); points(5:1)  # multiple in a single line
'' ""
  c('text strings', "Hi") # c-f(x) combines some text
$ @
  mydata$columname # accessing columns in a data.frame
  object@name      # accessing within objects
+ - * / ^ # operational characters: 3 * 5 or 4^3 is 4^3
<- ->   # assigning things
  a <- 10; 10 -> a # both is the same
<= => >= == != ! # comparison; != is NOT ! negates
|| | && &         # logical comparison: OR AND
```

"Saving" or assigning things. `save.image()` saves workspace.

```
a <- 10  # assign 10 to 'a' locally
a = 10   # assign 10 to 'a' locally
10 = a   # assign 10 to 'a' locally
10 -> a  # assign 10 to 'a' locally
a <<- 10 # assign 10 to 'a' globally
10 ->> a # assign 10 to 'a' globally
a # returns now still 10
a <- append(a, 0:1) # special append-f(x)
  # would be the same: a <- c(a, 0:1)
a # returns now 10  0  1
a <- c(1, 2, 3)   # assigning numbers in a vector
a <- c(1, 2, '3') # treated as characters for ALL
a <- list(1,2,'3')# treated as 1, 2, '3' as list object
```

Brackets: {...} [...] (...)

```
(...) # round brackets:
  (mydata) # forces mydata to be printed in R-console
  plot(...) # indicates ANY function
{...} # curly brackets:
  {# start grouping
    ... # a lot of R-Code
  }# end grouping
[..] [..,..] [[..]] # access an R-object's index
  myvector[2] # second element
  # in data frames, lists
  mydata[r,c]  # access either row or column or both
```

"> Accessing/filtering/combining data" on the following page.

> Functions (1.3)

```
# EACH function has round braces
aFunctionName(arg1 = setting, arg2 = setting)
 a <- 1:3; b <- 4:2
  plot(a, b) # will plot plot(x=a, y=b)
  plot(b, a) # will plot plot(x=b, y=a)
  plot(y=b, x=a)# same as plot(a, b) but explicit assig-
  # ning as 'x=a' doesn't worry about an argument's in-
  # ternal defined position, i.e. is sometimes safer
args('plot.default')# accessing arguments of a function
getAnywhere('plot') # accessing the code of plot-f(x)
```

> Predefined words (1.4)

Reserved words, see also `?Reserved`

```
TRUE T FALSE F # boolean values
Inf  # indicates an infinite value
NaN  # means 'not a number'
NA   # indicates an entry to be not available
NULL # represents the null object
LETTERS    # vector "A" "B" "C" "D" ...
letters    # vector "a" "b" "c" "d" ...
month.abb  # "Jan" "Feb" "Mar" "Apr"...
month.name # "January" "February" ...
```

Reserved words in R's parser, see also `?Control`

```
function # used for programming a function
if # if-f(x)
  else   # used in combination with if-f(x)
while    # while-f(x) indicates a while loop
repeat   # indicates a repeat loop
for      # for-f(x) indicates a for loop
next     # used in loops
break    # used in loops
```

> Plotting (1.5)

High level plots create anew whereas low level plots *add* things.

Common high level plots.

```
a <- 1:3; b <- 4:2
plot(a, b) # will do plot(x=a, y=b); see also ?pairs()
hist(x) # histogram of frequencies
barplot(x) # histogram as bars
dotchart(x) # stacked plots line by line
pie(x) # circular pie-chart
boxplot(x) # box-and-whiskers
sunflowerplot(x, y) # identical points as 'flowers'
coplot(x~y | z)# conditioning plots + marginal intervals
matplot(x,y) # plot matrices/more than 2 columns at once
plot.ts(x) # time series if x is an ts-object
contour(x, y, z) # contour plot '3D'
filled.contour(x, y, z) # filled contour plot '3D'
image(x, y, z) # colored rectangles
persp(x, y, z) # 3D perspective plots
stars(x) # star (spider/radar) plots & segment diagrams
symbols(x,y,...)# squares,stars,thermometers,boxplots...
termplot(mod.obj) # plot regression terms
```

Common low level plots.

```
points(x, y) # adds points or lines
lines(x, y)  # identical but with lines
polygon(x, y) # draws a polygon
text(x, y, labels, ...) # text at x, y
thigmophobe.labels() # plotrix; no overlapping labels
mtext(text, side=3, line=0, ...) # adds marginal text
segments(x0, y0, x1, y1) # draws line segments
arrows(x0, y0, x1, y1, angle=30, code=2)# 1:← 2:→ 3:↔
abline(a,b) # draws a line of y = b * x + a
abline(h=3) # draws a horizontal line at 3
abline(v=1) # draws a vertical line at 1
abline(lm.obj) # draws a regression line
rect(x1, y1, x2, y2) # draws a rectangles
legend('topleft', legend="...") # adds the legend
title('My title') # adds a title (and sub-title)
axis(side=3) # 3 adds on top; 1 2 3 4 → bo le to re
box() # draw a box around the current plot
rug(x)# draws small vertical indication lines of x, y
locator(n,type="n",...)#n-coordinates of mouse-click(s)
identify(x,y) # data points with mouse-click(s)
```

>> Plotting > Saving images (1.5.1)

See also in help ?Devices. File paths: '../path' or '..\\path'

```
getwd() # get current working directory and...
setwd("C:\\path\\to\\my\\working directory") # sets it
dir() # lists content of current working directory
#EPS/PDF is safest vector format; jpeg should be avoided
dev.copy(dev.copy2eps, # a device function: png, pdf,...
  file='filename.eps', # 1 level up: '..\\filename.eps'
  width=par()$din[1], height=par()$din[2],# current w/h
  title= "Title in EPS' or PDF's"
)# end dev.copy()
dev.off() # image now saved
```

> Packages/libraries (1.6)

```
install.packages("analogue", dependencies=TRUE)# install
update.packages("analogue", dependencies=TRUE) # update
remove.packages("Rcmdr", lib="C:\\path\\to\\R\\library")
library(analogue) # loads analogue
require(analogue) # loads analogue + returns TRUE/FALSE
detach(package::analogue) # remove analogue from search path
```

Data, lists, vectors, R-objects (2)

> Accessing/filtering/combining data (2.1)

Accessing is achieved by using syntax of [..] or [..,..]

```
x[n] # the n-th element
x[-n] # WITHOUT the n-th element
x[1:n] # elements 1 to n
x[-(1:n)] # all EXCEPT elements 1 to n
x[c(1,4,2)] # combined elements 1, 4 and 2
x["name"] # element named "name"
x[x > 3] # all elements greater than 3
x[x > 3 & x < 5] # all elements inbetween 3 and 5
x[x %in% c("factor1","factor2")] # searches strings
x[order(x)] # sorting
```

```
attach(mydata) # include row-/columnnames in search path
mydata[r,c] # element in row r and column c
mydata[2,] # elements in row 2
mydata[-2,] # elements EXCEPT row 2
mydata[,3:5] # elements in columns 3, 4, 5
mydata[,-(3:5)] # all EXCEPT columns 3, 4, 5
mydata[,c(1,3)] # columns 1 and 3 combined
mydata["myrowname",] # access row name "myrowname"
mydata[["myname"]] # column named "myname"
mydata$name # column named "name"
mydata[,order('mycolname')] # needs attach(mydata)
detach(mydata) # remove mydata from search path
cbind(..,..) rbind(..,..) # combine columns/rows
```

> Dealing with data (2.2)

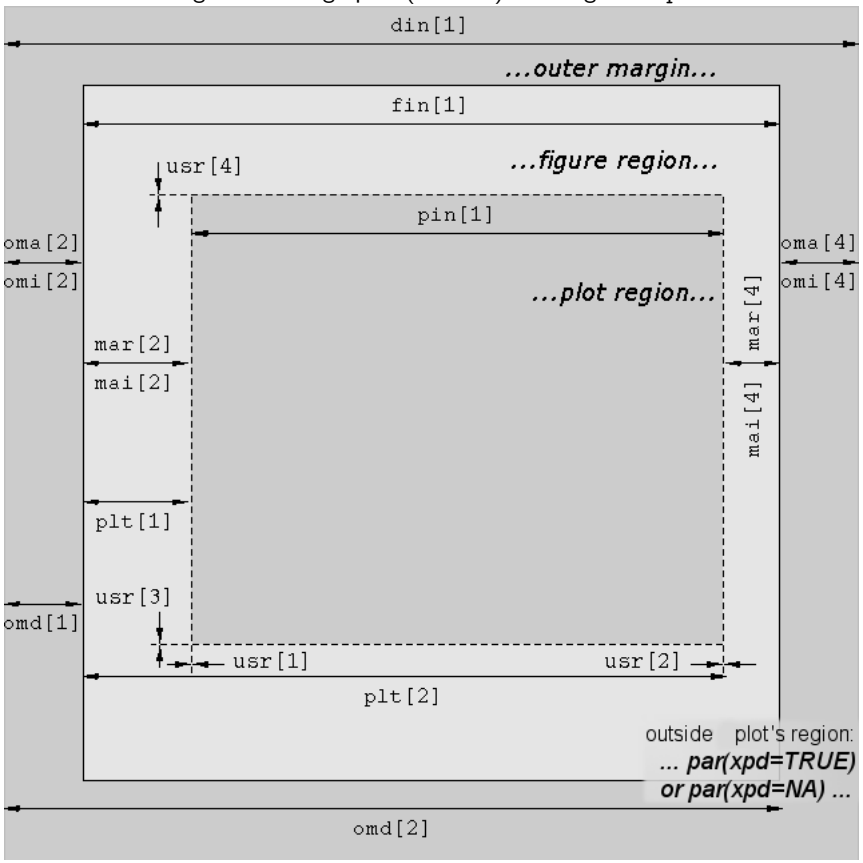
```
ls() # list all current data, variables, ...
str(mydata) # show internal data structure
attach(mydata) # adds mydata to search path
  # col-/row names are now directly available
detach(mydata) # remove mydata from search path
```

> Editing data (2.3)

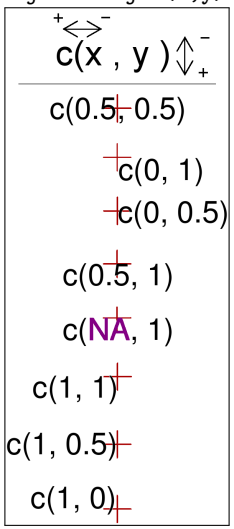
```
fix(mydata) # opens a data editor
```

Following graphics are modified from Paul Murrell "R Graphics". A book on the core graphics facilities of the R language and environment for statistical computing and graphics (Chapman & Hall/CRC, August 2005). See http://www.stat.auckland.ac.nz/~paul/RGraphics/rgraphics.html.

Margins for *one* graphic (detailed). Settings see ?par:



Text adjustment by e.g. adj=1 or adj=c(x,y):



Margins for multiple plots: