Short R Reference Card

Please, send your corrections to petr.nazarov@crp-sante.lu

Data import/export

Command	Description
`getwd()` `no parameters`	Shows the current working folder. *Example:* getwd()
`setwd(`*`name`*`)` `variable or constant – name`	Shows the current working folder. *Example:* setwd("d://Data/Lecture2")
`scan(…)` `file, what, sep, quote, dec, etc.`	Read data into a vector or list from the console or file. **file** – location and name of the file to be loaded (on disk or URL); **what** – he type of **what** gives the type of data to be read. **sep** – separator of the values: " "-space, "\t"-tab, ","-comma, ""-all mentioned; **quote** – which symbol is used as a quote, "\"" is a good choice; **dec** – decimal separator: "." or ","; *Example:* vec = scan("currency.txt", what ="zzz")
`read.table(…)` `file, header, sep, quote, dec, row.names, col.names, as.is, skip, comment.char etc.`	Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file. **file** – location and name of the file to be loaded (on disk or URL); **header** – presence if the header: T if header is present then, otherwise F; **sep** – separator of the values: " "-space, "\t"-tab, ","-comma, ""-all mentioned; **quote** – which symbol is used as a quote, "\"" is a good choice; **dec** – decimal separator: "." or ","; **row.names** – if row names are present then T, otherwise F; **col.names** – if column names are present then T, otherwise F; **as.is** – set T if you want to keep strings as strings, and F – to transform to factors; **skip** – specify a value of the row you would like to skip (if nessesary); **comment.char** – which letter you want to use to "comment" string. Set "" to remove this. *Example:* data = read.table("mice.txt",header=T,sep="\t", comment.chat="")
`write.table(…)` `x, file, sep, quote, dec, eol, row.names, col.names, etc.`	Prints ***x*** (after converting it to a data frame if it is not one nor a matrix) to a file or connection. **x** – data frame or matrix; **file** – location and name of the file to be loaded (on disk or URL); **sep** – separator of the values: " "-space, "\t"-tab, ","-comma, ""-all mentioned; **eol** – end of line characters: "\n" – standard, "\r\n" – Windows-like, "/r" – MacOS-like; **row.names** – assign FALSE to skip row names, TRUE to use row names of **x**, or a character vector of row names to be written; **col.names** – assign FALSE to skip column names, TRUE to use column names of **x**, or a character vector of column names to be written.
`load(`*`file`*`)`	Reload datasets written with the function `save` format.
`save(…)` *`list, file, etc.`*	Compress and saves datasets into a specified file. **list** – variable or a list of variable names; **file** – location and name of the file to be saved;

Validation of the data

Command	Description
`fix(data)`	Show *data* in a special window and allows its edition
`str(data)` `variable – data`	Show the structure of the variable *data*. *Example:* str(data)
`names(data)` `data – data frame`	Return the column names of data frame *data*. It also allows assigning new names. *Example:* names(data)
`row.names(data)` `data – data frame`	Return the row names of data frame *data*. It also allows assigning new names. *Example:* row.names(data)

`rownames(matr)`
`colnames(matr)`
`matr – matrix`
	Return the column and row names of matrix *matr*. I also allows assigning new row/column names.
	Example: colnames(A)

`summary(data)` `variable – data`	Show the mean value and 5-number summary for *data*, if *data* is a numerical variable. Shows the frequency distribution if *data* is a factor. *Example:* summary(data)
`ncol(data)` `nrow(data)`	If data is a matrix or data-frame, it returns the number of columns and rows. *Example:* ncol(data)
`length(vec)` `variable – vec`	If *vec* is a vector, shows its length. *Example:* length(data$Sex)
`ls()` `no parameters`	Show names of all variables. *Example:* ls()
`rm(…)` `variable or list of variable names`	If variable is given – remove the variable. If list is given – removes the list of variables. *Example:* rm(list = ls()) # removes all variables

Types of Variables

Command	Description
`class(x)`	Shows the type of *x*

`as.integer(x)`
`as.double(x)`
`as.character(x)`
`as.factor(x)`
	Change scalar types. When applied to vectors and matrixes – generates vector of the desired type. Function as.factor() should be applied only to vectors.
	Example: as.double("-1.345")

`as.matrix(x)`
`as.data.frame(x)`
`as.list(x)`
	Change types of matrixes, data frames and lists.
	Example: as.list(Data)

`is.numeric(x)`
`is.character(x)`
`is.matrix(x)`
`etc.`
	Check whether x is of a specified class.

Mathematical Functions

Command	Description
`pi`	π constant (~3.141593)
`exp(x)`	Calculate e constant (~2.71828) in the power of x.

`log10(x)`
`log2(x)`
`log(x,…)`
`x, base`
	Calculate log.
	base – if specified, use as a base for logarithm. Default is set to ***e*** constant
	Example: log(100,10)

`sqrt(x)`	Calculate square root of x. If x<0 generates NaN value.

`cos(x)`
`sin(x)`
`tan(x)`
`acos(x)`
`asin(x)`
`atan(x)`
	These functions give the obvious trigonometric functions. They respectively compute the cosine, sine, tangent, arc-cosine, arc-sine, arc-tangent.
	Example: sin(pi/6)

Work with Strings

Command	Description
`paste(…)` `strings, variables, sep`	Concatenate several strings and values. Non-string values are transferred into strings. **sep** – separator used between strings and values. Set to **sep=""** to avoid separation *Example:* paste("The dataset has",5,"columns")
`sprintf(fmt,…)` `fmt, variables`	Generate a string using *fmt* template (like in C). After template the variables should be given. If a variable is a vector, sprint generates a vector of strings. **fmt** – string template. It contains the following special fields: **%d** – integer, **%f, %e, %g** – different double formats **%s** – string *Example:* sprintf("We have %d mice, with average weight %g g",6,33.3)
`nchar (…)` `x, start, stop`	Return number of characters **start, stop** – substring from letter number start to latter number stop *Example:* substr("Hello, world!",1,5)
`substr(…)` `x, start, stop`	Return substring of *x*. **start, stop** – substring from character number **start** to character number **stop** *Example:* substr("Hello, world!",1,5)
`strsplit(…)` `x, split, etc.`	Split the elements of a character variable `x` into substrings according to the matches to substring `split`. *Example:* strsplit("Hello, world!",split=NULL)

`sub(…)`
`gsub(…)`
`pattern, replace, x, etc.`
	Replace the first (sub) and all (gsub) substrings of *x*.
	pattern – regular expression for the replaced substring;
	replace – substring used for replacement;
	x – string or vector of strings in which replacement should be done.
	Example: gsub("l", "L", "Hello, world!")

Data Visualization

Command	Description

`x11()`
`windows(…)`
`width, height, xpos, ypos`
	Opens a new drawing window (called device). x11() works in Linux and Windows; windows(…) – works only in Windows but allows settings:
	width, height – width and height of the window;
	xpos, ypos – position of the window on the screen. If negative values are given – distance from the right and bottom sides of the screen;
	Example: x11()

`pdf(…)`
`png(…)`
`file, etc`
	Draw plots into PDF or PNG file. To stop drawing call `dev.off()`
	Example: png("test%d.png")

`par(…)` `mfcol, mfrow, new, etc. see help for more`	Settings function. See help to read about all parameters. The important parameters are *mfcol* and *mfrow*. Use them to specify how many frames you want to have in one window. Specify *new* = T to draw on top of the current plot. *Example:* par(mfcol = c(2,2))
`plot(…)` `x, y, xlim, ylim, main, xlab, ylab, type, lwd, lty, pch, col`	Plots data. **x, y** – vectors with coordinates; **xlim, ylim** – 2-element vectors for min and max values on the axes; **main, xlab, ylab** – title of the plot, x-axis label, y-axis label; **type** – plot type: "p"-points, "l"-line, "b"-both, "h"-histogram, "s"-stairs, etc; **lwd, lty, pch, col** – line width, type, point type and color *Example:* plot(x, y, type="b", pch=19, col=2, ylim=c(50,200), main="Plot")
`density(vec)` `vector vec, width, na.rm, etc.`	Generates a "probability density object". Use it inside `plot` to plot the resulted probability density. Set na.rm=T if data contain missing values. *Example:* plot(density(x, na.rm=T))
`lines(…)` `x, y, xlim, ylim, main, xlab, ylab, lwd, lty, col`	Add line-plot to the existing plot. Parameters are the same as in plot() *Example:* lines(x, y, lwd=3, col=2, ylim=c(50,200))
`points(…)` `x, y, xlim, ylim, main, xlab, ylab, lwd, lty, col`	Add points-plot to the existing plot. Parameters are the same as in plot() *Example:* points(x, y, pch=19, col=2, ylim=c(50,200))
`abline(…)` `a,b,v,h lwd, lty, col`	Plot a line on the existing plot. If *a* and *b* are specified, plots $y = a+bx$. If *v* is specified – plots a vertical line. If *h* is specified – a horizontal one. *Example:* abline(h = 1, col = 2, lty = 2)

Statistics

Command	Description

`mean(x)`
`median(x)`
`std(x)`
`var(x)`
`min(x)`
`max(x)`
`sum(x)`
`na.rm = T`
	Main commands for the basic statistical calculations. Majority of meanings is obvious; std stands for standard deviation; var – for variance.
	na.rm – set T to remove missed values. Otherwise the result will be always NA
	Example: mean(x, na.rm=T)

`cov(x,y)`
`cor(x,y)`
`use = "pairwise.complete.obs"`
	Measures of dependence: covariation and correlation.
	use – describes how to handle NA values. Use `"pairwise.complete.obs"`
	Example: cor(x, y, use = "pairwise.complete.obs")

`t.test(…)`
`x, y,`
`conf.level, paired`
	Performs a Student test for the means of two populations.
	x, y – samples (data) for two populations;
	conf.level – confidence level;
	paired – if set to T – the paired test is performed. If missed or F – unpaired.
	Example: t.test(x,y)

`var.test(…)`
`x, y,`
`ratio,`
`conf.level`
	Performs an F test to compare the variances of two samples from normal populations.
	x, y – vectors with coordinates;
	ratio – the hypothesized ratio of the population variances of x and y (usually skipped);
	conf.level – confidence level;
	Example: var.test(data[,1],data[,2])

`chisq.test(…)`
`x, p` or `y,`
`rescale.p`
	Performs chi-squared contingency table tests and goodness-of-fit tests.
	x – vector with number of observations;
	p – expected observations or expected probabilities;
	rescale.p– if **p** contains observations, then **rescale.p** should be set to T;
	or
	x – experimental data for the factor 1;
	y – experimental data for the factor 2;
	Example: chisq.test(x = expr, p = ctrl, rescale.p = T)
	chisq.test(x = data$Gender, y = data$Beer)

`pearson.test(…)`
`shapiro.test(…)`
`data`
	Tests for normality (Pearson and Shapiro-Wilk). Note that library "nortest" is needed for Pearson test.
	Example: pearson.test(data)
	shapiro.test(data)

`ks.test(…)`
`x, y,`
`param`
	Kolmogorov-Smirnov test for the distribution.
	x – vector with experimental data;
	y – either a numeric vector of data values, or a character string naming a cumulative distribution function or an actual cumulative distribution function such as pnorm;
	param – parameters of the distribution (mean, st.dev).
	Example: ks.test(data,"pnorm",mean(data),sd(data))

`cor.test(…)`
`x, y,`
`method`
	Test hypotheses about correlation.
	x, y – vectors with experimental data;
	method – which correlation measure to use: pearson, kendall or spearman
	Example: cor.test(x, y, method = "pearson")

`aov(…)`
`formula,`
`data`
	Build a linear model for ANOVA.
	formula – ANOVA equation using the factors;
	data – data table with the specified factors in columns;
	Example: aov(Ending.weight ~ Sex + Strain + Sex*Strain, data)

`lm(…)` `formula`	Build a linear model. **formula** – equation, showing the dependent and independent variables; *Example:* lm(y~x)
`predict(…)` `object, int`	Build a prediction or confidence interval for the linear model. **object** – linear model; **int** – type of interval: "confidence" – for confidence interval or "pred" – for prediction intervals ; *Example:* predict(lm(y~x), int = "confidence")
