Basic Math

`exp(x)`	Exponential	`sum(x)`	Sum
`log(x)`	Natural log	`cumsum(x)`	Cumulative Sum
`max(x)`	Largest element	`ceil(x)`	Round up
`min(x)`	Smallest element	`floor(x)`	Round down
`x %% y`	Modulo		

I/O

`write( data, " myd ata.da t")`	Write data as binary.
`scan("m yda ta.d at ")`	Read binary data.
`getcwd()`	Current working directory

Plotting

`plot(data)`	Plot quick overview.
`barplot(x, main="T itl e", xlab="x label ")`	Annotated barplot of absolute frequencies
`plot.e cdf (data)`	Plot ECDF.
`hist(data, prob=TRUE, breaks=30)`	Histogram of relative frequencies (30 bins).
`rug(data)`	1D-plot
`abline (a, b,c ol= " red ")`	Add a red line with intercept `a` and slope `b` to the plot.
`boxplo t(d ata1, data2, ... ,range =1.5)`	Plot boxplots of one or more data sequences in one window. `range` determines the extend of the whiskers. Default `range=1.5`, i.e. 1.5 x IQR
`qqnorm(x)`	QQ-Plot against standard normal distribution
`qqPlot(x, dist="u nif ")`	QQ-Plot against any R-standard distribution

Vectors

Creating Vectors

`c(2, 4, 6)`	Join elements into a vector
`2:6`	An integer sequence (end inclusive!)
`seq(2,3, by=0.5)`	Complex sequence (s. np.linspace)
`rep(1:2, 3)`	Repeat vector
`rep(1:2, 3:4)`	Repeat each element

Functions

`sort(x)`	Return x sorted.
`rev(x)`	Return x reversed.
`unique(x)`	See unique values.
`length(x)`	Length of x.

Selecting Vector Elements

Vectors (cont)

By Position	
`x[4]`	The fourth element
`x[-4]`	All but the fourth.
`x[2:4]`	Elements two to four
`x[-(2:4)]`	All elements except 2 to four
`x[c(1, 5)]`	Elements one and five.
By Value	
`x[x == 10]`	All elements equal to 10
`x[x < 10]`	All elements less than 10.
`x[x %in% c(1, 2, 5)]`	Elements in the given set.
Named Vectors	
`x['apple']`	Element with name 'apple'.

Control Flow

`for (variable in sequence) {...}`	for-loop. If the loop body contains only a single line, the curly brackets can be omitted.
`while (condi tion) {...}`	while-loop

`if (i > 5) {`
`...`
`else {`
`...`
`}`
	if-else-block

`foo = functi on( arg1, arg2, ...) {`
`...`
`return (var)`
`}`
	function

Runs Test of Randomness

`rle(x)`	Compute the lengths and values of runs of equal values in a vector .
`rle(x) $le ngths`	Vector containing the length of each run.
`rle(x) $values`	Vector of the same length as lengths with the corresponding values.

Help

`?sqrt`	Display documentation of the command `sqrt``
`?'%%'`	use quotation marks for special characters

Not published yet.
Last updated 17th December, 2024.

Miscellaneous

Printing

`print( " Tex t")`	Default print
`sprint f("F orm atted %s: %.3f", object, mean)`	Formatted print
`(x=3)`	enclose an R command with brackets to directly print the result
`edit(x)`	Invoke text editor on R object
`librar y(MASS)`	Load package MASS

Random Numbers

`sample (1: 3,p rob =c( 1/6 ,1/ 3,1 /2) ,re pla c e= TRU E,20)`	Draw 20 balls, labeled from 1 to 3, from box with replacement.
`r<d istr. ID>(n, params)`	Draw `n` numbers from distribution `<distr. ID>` with parameters `params`

(see Distributions in R for more details)

Tables

`table( data)`	get absolute frequencies of values
`as.num eri c(tab); as.vec tor (tab)`	Extract values and their absolute frequencies from table
`tab/le ngt h(data)`	Compute relative frequencies

Characteristics of data sequences

`mean(x)`	Arithmetic mean of the data sequence
`var(x)`	Variance
`median(x)`	Median
`quanti le(x, type=7 )`	Quantile. `type=7` is the default computation algorithm, i.e. the function returns the value at position $k=1+p(n-1)$, if this is an integer. Otherwise, R computes a weighted mean of the two neighboring integers
`qunati le(x, type=1 )`	General inverse function of the ECDF (smallest p-quantile). Largetst p-quantile can be obtained indirectly by slightly increasing p
`summary(x)`	Overview of important measures

By **BarplotNorm**

cheatography.com/barplotnorm/

Not published yet.
Last updated 17th December, 2024.
Page 3 of 4.

Data sets

Interacting with data sets

`col_1 = data$c ol_ 1_name`	Access column data

I/O

`data = read.c sv( " fil e.c sv", header =FALSE, sep="")`	Read csv (function arguments similar to that used in pandas)
`write.c sv (data, " dat a.c sv", row.na mes =FALSE, sep= " ")`	Write data set as csv

Distributions in R

d<d>