R Commands Cheat Sheet

by BarplotNorm via cheatography.com/203294/cs/43288/

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		

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-else-block

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

function

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

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'.

Tables

`table( data)`	get absolute frequencies of values
`as.numeric(tab); as.vector (tab)`	Extract values and their absolute frequencies from table
`tab/length(data)`	Compute relative frequencies

Matrices

`m = matrix(x, nrow = 3, ncol = 3)`	Create a matrix from vector `x`
`t(m)`	transpose
`m %*% n`	Matrix multiplication
`solve(m, n)`	Find x in `m * x = n`
`det(m)`	Determinant
`eigen(m)`	Find eigen vectors and values

Data sets

Data=data.frame(price=c(11,20,14,15), number=c(40,50,60,20))	Create a data set
Interacting with data sets	
`col_1 = data$col_1_name`	Access column data
`Data[1,2]; Data[,2]; Data[[1]]`	Access data with index notation
I/O	

R Commands Cheat Sheet

by BarplotNorm via cheatography.com/203294/cs/43288/

Data sets (cont)

`data = read.csv("file.csv", header =FALSE, sep="")`	Read csv (function arguments similar to that used in pandas)
`write.csv (data, "data.csv", row.names =FALSE, sep="")`	Write data set as csv

Filter

`df[df$kids == "Jack",]`	Filter data frame
`subset(df,  kids== "John" & Grade == 1.3)`	Filter multiple columns
`subset(df, kids %in% c("Jack", "John"))`	Filter a column with multiple values
`unique (housing[, c("State", "region")])`	Extract unique rows

Sort

`housing[order(housing$Home.Value), ]`	Order data frame in ascending order
`housing[order(housing$Home.Value, decreasing = TRUE), ]`	order data frame in descending order

Meta

`dim(df)`	Check the dimensions of a data frame
`colnames(d)`	Return the column names

Manipulate data

`Data_Frame_New <- Data_Frame[- c(1), -c(1)]`	Remove columns and/or rows from data frame
`rbdind (df_1, df_2)`	Combine data frames vertically

I/O

`write( data, "mydata.dat")`	Write data as binary.
`scan("mydata.dat")`	Read binary data.
`getcwd()`	Current working directory

Random Numbers

`sample (1:3,prob=c(1/6,1/3,1/2),replace=TRUE,20)`	Draw 20 balls, labeled from 1 to 3, from box with replacement.
`r<distr. ID>`(n, params)	Draw `n` numbers from distribution `<distr. ID>` with parameters `params`
*(see **Distributions in R** for more details)*	

Characteristics of data sequences

`mean(x)`	Arithmetic mean of the data sequence
`var(x)`	Variance
`median(x)`	Median
`quantile(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
`quantile(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
`cov(x,y)`	Covariance
`cor(x,y)`	Correlation

Distributions in R

General usage

`d<distr. ID>(params)`	density function
`q<distr. ID>(params)`	quantile function. Always computes the smallest quantile
`p<distr. ID>(params)`	cumulative distribution function
`r<distr. ID>(params)`	random variate generation

Distributions

`dbinom(x, size=p, prob=p)`	Binomial
`dchisq(x, df, ncp=0)`	Chi-squared
`dexp(x, rate=1)`	Exponential
`dgamma(x, shape=r, rate=1)`	Gamma
`dgeom(x, prob=p)`	Geometric
`dnbinom(x, size, prob)`	Negative binomial
`dnorm(x, mean=0, sd=1)`	Normal
`dpois(x, lambda)`	Poisson
`dt(x, df, ncp)`	t-distribution
`dunif(x, min=0, max=1)`	Uniform

Plotting

Basic plots

`plot(data)`	Plot quick overview.
`plot(x, y, xlab="m u", ylab="P owe r", type="l ", col="re d", ylim=c (0,1), lwd=1.5)`	Plot data with custom style options

Lines and curves

`abline (a, b,c ol= " red ")`	Add a red line with intercept `a` and slope `b` to the plot.
`abline (v= a,c ol= " red ")`	add vertical line at `x=a`
`abline (h= b,c ol= " red ")`	add horizontal line at `y=b`
`lines(x, y, col="gr een ", lwd=1.5)`	Add a generic line
`curve( sin ,-p i,p i,a dd= TRUE)`	Draw a curve of a function over the specified interval

Data visualization

`plot.e cdf (data)`	Plot ECDF.
`barplot(x, main="T itl e", xlab="x label")`	Annotated barplot of absolute frequencies
`hist(data, prob=TRUE, breaks=30)`	Histogram of relative frequencies (30 bins).
`rug(data)`	1D-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. Additional arguments such as `df`, `ncp` can also be specified.
`legend (x,y, legend =c( " n=1 0"), col =c( " red "), lty=1, cex=0.8)`	Add legend to plot as position `(x, y)`

Statistical hypothesis testing

One-Sample tests

`t.test (x, mu= mu0 ,al t="l ess ", conf.l eve l=1 -alpha)`	Performs one and two sample t-tests on vectors of data.

By **BarplotNorm**

cheatography.com/barplotnorm/

Statistical hypothesis testing (cont)

`power.t.t est(n = 100, delta=0.1, sd=2, sig.le vel =0.1, type="o ne.s am ple ", alt="on e .s ide d")`	Compute the power of the one- or two-sample t test, or determine parameters to obtain a target power.
`binom.t es t(s um( x), n,p 0,a lt= " gre ate r", conf.l eve l=1 -alpha)`	Performs an exact test of a simple null hypothesis about the probability of success in a Bernoulli experiment. It might happen that the decision based on the p-value differs from that of the confidence interval. Choose the decision based on the p-value in such cases.

Two-Sample tests

`t.test (sh oes $A, sho es$ B,p air ed= FALSE, var.eq ual =TRUE)`	Example for an unpaired sample t-test
`var.te st( x,y ,co nf.l ev el= 1-a lpha)`	Performs an F test to compare the variances of two samples from normal populations.

GOF tests

`shapir o.t est(x)`	Performs the Shapiro-Wilk test of normality.
`chisq.t es t(t abl e(x), p=p_0)`	Test for distribution with probabilities `p_0`. If `p` is not specified, R tests for a uniform distribution
`chisq.t es t(t abl e(x), p=p_0, simula te.p.v alu e=TRUE)`	Do not use Chi^2-approximation to calculate the p-value

`pwr.chisq.test(w=ncp, df=s-1, sig.level=alpha, power=0.9)`	Determine the number of samples needed to reach the desired power at the given significance level
`ks.test(x, "pnorm", 0, 1)`	One-sample Kolmogorov-Smirnov test against hypothetical distribution
`lillie.test(x)`	Lilliefors (Kolmogorov-Smirnov) test for the composite hypothesis of normality

Tests of independence

`chisq.test(M)`	Chi^2-test of independence. M hast to be a matrix! (contingency table)

Statistical hypothesis testing (cont)

`fisher.test(M)`	Exact test of Fisher. If the table entries are too large, use `simulate.p.value=TRUE`
`runs.test(x)`	Runs test of independence. `x` hast to be a factor (use `as.factor()` if necessary)

Runs Test of Randomness

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

Optimization

`nlm(E2,0.5)`	Carries out a minimization of the function f using a Newton-type algorithm. May not give all solutions. The function must be vectorized
`E2vec= Vectorize(E2, vectorize.args= c("n"))`	vectorize a function. `vectorize.args`: explicitly state arguments to be vectorized.
Distribution Fit	
`fitdistr(x, "Poisson")`	Maximum-likelihood fitting of univariate distributions, allowing parameters to be held fixed if desired. (`library(MASS)`)
Regression	
`reg=lm (x~t)`	Fit a linear function `x=a+bt`
`summary(reg)`	Obtain further information about regression result **Important fields:** - Residual standard error: sd of residuals (with normalization **`n-2`**) -t value: Test null hypothesis "estimate is 0" with assumption of a normally distributed random mechanism -multiple R-squared: squared corr. coef. Null hypothesis r^2=0 is tested with F-statistic

Optimization (cont)

`reg=lm(x ~ t+ I(t^2))`	Fit a polynomial function. `I()` inhibits R from interpreting `t^2` as a formula
`form=x ~ a/(1+e xp( -b* (t-c)))` `reg=nl s(form, data=U SPop, start= c(a =40 0,b =0.0 2, - c=2 000))`	perform non-linear least-squares regression
`plot(t ,pr edi ct( reg))`	Plot regression result

Root finding

`res = uniroo t(func, c(0,10))`	Searches interval for a root of the function `func`. `res$root` and `res$f.root` give the location of the root and the value of the function

Help

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

Miscellaneous

Printing

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

Libraries

`librar y(MASS)`	Load package MASS
`uniroot(f, interval)`	find 1D root

Step functions

`stepfu nc(x,y)`	Given the vectors $(x_1, ..., x_n)$ and $(y_0,...,y_n)$ (one value more!), returns an interpolating ‘step’ function
`knots(x)`	returns jump positions of stepfunction