Basic statistics with R

Transformations

`round(x, n)`	Rounds the elements of x to n decimals
`signif(x,n)`	Rounds the elements of x to n significant digits
`ceiling(x)`	Rounds to the smallest integers not less than the corresponding elements of x
`floor(x)`	Rounds to the largest integers not greater than the corresponding elements of x
`trunc(x)`	Rounds by truncating the values in x toward 0
`abs(x)`	Computes the absolute values of x
`sqrt(x)`	Computes the (principal) square root of x
`x^0.25`	Computes the cubic root of x
`log(x, base)`	Computes logarithms with base *base* (default is natural log)
`log2(x)`	Computes binary (i.e., base 2) logarithm
`log10(x)`	Computes the base 10 logarithm
`log1p(x)`	Computes log(1+x)
`exp(x)`	Computes the exponential function.
`exp1m(x)`	Computes exp(x) - 1
`car::basicPower(()`	Computes a simple power transformation
`car::bcPower()`	Transforms the element in x using the Box-Cox family of scaled power transformations
`sin(x)`, `cos(x)`, `tan(x)`, `asin(x)`, `acos(x)`, `atan(x)`, `atan2(x)`	Trigonometric functions; they compute the cosine, sine, tangent, arc-cosine, arc-sine, arc-tangent, and the two-argument arc-tangent.
`scale(x)`	Centers and/or scales a vector or matrix x; to center only use center=FALSE, to reduce only scale=FALSE; default is center=TRUE, scale=TRUE for *z-transformation*

Descriptive statistics

Functions that return a single value (scalar)

`sum(x)`	Sum of the elements of x
`prod(x)`	Product of the elements of x
`mean(x)`	Arithmetic mean of the elements of x
`weighted.mean(x, w)`	Mean of x with weights w
`median(x)`	Median of the elements of x
`var(x)`	Variance of the elements of x (calculated on n – 1)
`sd(x)`	Standard deviation of x
`sd(x)/sqrt(length(x))`	Calculation of standard error
`min(x)`	Minimum of the elements of x
`max(x)`	Maximum of the elements of x

Functions that return a vector

`range(x)`	Returns minimum and maximum of elements of x
`pmin(x,y)`, `pmax(x,y)`	Returns (regular or parallel) minima/maxima
`quantile(x,probs)`	Sample quantiles corresponding to the given probabilities
`diff(x)`	Lagged and iterated differences of vector x
`rank(x)`	Ranks of the elements of x
`IQR(x)`	Interquartile range of the x values
`mad(x)`	Median absolute deviation
`cumsum(x)`	Cumulative sums of the elements of x
`cumprod(x)`	Cumulative product of the elements of x
`cummin(x)`	Cumulative minima of the elements of x
`cummax(x)`	Cumulative maxima of the elements of x
`colSums(x)`	Sum of each column
`rowSums(x)`	Sum of each row
`colMeans(x)`	Arithmentic mean of each column
`rowMeans(x)`	Arithmentic mean of each row

Functions that return scalars or matrices

`summary(x)`	If x is a matrix or data frame, function computes various statistics for each column
`cor(x, y)`	Linear correlation between x and y, or correlation matrix if they are matrices or data frames (Pearson's product moment correlation coefficient, Kendall's tau or Spearman's rho)
`cov(x,y)`	Covariance between x and y

Inferential statistics

Probability distributions

Gaussian (normal) distribution:

`rnorm()`	Generates random numbers (**r**xxx)
`pnorm(q)`	Returns the probabilities for a given set of quantiles (**p**xxx)
`qnorm(p)`	Returns the quantiles for a given set of probabilities (**q**xxx)
`dnorm(x)`	Gives the density function (**d**xxx)

Other standard probability distributions:
(see also `?Distributions`)

`dpois(n, lambda)`	Poisson
`dnbinom(n, size, prob)`	Negative binomial
`dbinom(n, size, prob)`	Binomial
`dmultinom(n, size, prob)`	Multinomial
`dunif(n, min=0, max=1)`	Uniform
`dexp(n, rate=1)`	Exponential
`dlnorm(n, meanlog=0, sdlog=1)`	Lognormal
`dgamma(n, shape, scale=1)`	Gamma
`dbeta(n, shape1, shape2)`	Beta
`dt(n, df)`	Student t distribution
`df(n, df1, df2)`	F-distribution
`dchisq(n, df)`	(Non-central) chi-squared distribution
`dweibull(n, shape, scale=1)`	Weibull
`dcauchy(n, location=0, scale=1)`	Cauchy
`dgeom(n, prob)`	Geometric
`dhyper(nn, m, n, k)`	Hypergeometric

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Linear models

Assumption tests

`shapiro.test(x)`	Performs the Shapiro-Wilk test of normality
`ks.test(x,y)`	Performs a one- or two-sample Kolmogorov-Smirnov test
`var.test(x)`	Performs an F test to compare the variances of two samples from normal populations
`bartlett.test()`	Performs Bartlett's test for homogeneity of variance across groups
`car::leveneTest(y, group)`	Computes Levene's test for homogeneity of variance across groups
`car::durbinWatsonTest()`	Performs Durbin-Watson test for autocorrelated errors

Simple tests

`chisq.test()`	Performs chi-squared contingency table tests and goodness-of-fit tests
1- and 2 sample tests	
`t.test(x)`	1-sample *t*-test
`t.test(x,y)`	2-sample independent *t*-test
`t.test(x,y, paired=T)`	2-sample dependent *t*-test
`wilcox.test(x)`	Non-parametric 1-sample Wilcoxon tests
`wilcox.test(x,y)`	Non-parametric 2-sample Mann Whitney *U* test (or Wilcoxon rank sum test)
`wilcox.test(x,y,paired=T)`	Non-parametric 2-sample Wilcoxon signed rank test
Multiple sample test	
`kruskal.test(x, g)`	Performs the non-parametric Kruskal-Wallis rank sum test
`friedman.test(y, groups)`	Performs a Friedman rank sum test with unreplicated blocked data
Correlation tests	
`cor.test(x, y)`	Test for correlation between paired samples

Formula

`formula()`	Provide a way of extracting formulae which have been included in other objects
`formula(y ~ x)`	y as a function of the fixed (continuous or categorical) x
`formula(y ~ Error(x))`	y as a function of the random x
`formula(y ~ x1 + x2)`	y as a function of x1 and x2
`formula(y~x1+x2+x1:x2)`	y as a function of x1 and x2, including their interaction
`formula(y ~ x1 * x2)`	Shorthand notation for the interaction inclusion
`formula(y~x1+x2%in%x1)`	y as a function of x1 and x2, which
or `x2/x1` or `x1+x1:x2`	is nested in x1
! or `x2/x1` or `x1+x1:x2`	y as a function of x1 and x2, which is nested in x1
! or `x2/x1` or `x1+x1:x2`	y as a function of x1 and x2, which is nested in x1
`update(model, .~. -x2)`	Is used to update model formulae. This typically involves adding or dropping terms

Analysis of Variance (ANOVA)

`aov(y ~ x, data)`	Fits a (type-I sequential) ANOVA by a call to lm for each stratum; response Y has to be continuous, at least on explanatory variables x has to be categorical (function can be used for ANOVA and ANCOVA)
`summary(model)`	Produces a type-I ANOVA table including SS, MS, F-ratio and p-values. Can also be defined explicitely via `summary.aov()`
`anova(lm(y ~ x, data))`	Computes (type-I) analysis of variance (or deviance) tables for one or more fitted model objects
`car::Anova(model)`	Computes type-II or type-III ANOVA tables for model objects produced by *aov, lm, glm, multinom* and other model functions
`drop1(fit,~.,test="F")`	Computes type-III marginal SS and F Tests
`model.tables(model)`	Computes summary tables for model fits, especially complex aov fits
`contrasts()`	Set and view the contrasts associated with a factor
`model.matrix()`	Creates a design (or model) matrix, e.g., by expanding factors to a set of dummy variables (depending on the contrasts) and expanding interactions similarly
`plot.design()`	Plots univariate effects of a design or model
`interaction.plot()`	Two-way interaction plot
`TukeyHSD(model)`	Computes Tukey Honest Significant Differences for *post-hoc* comparison

Linear models

`lm(y ~ x, data)`	Fits a linear model; response Y has to be continuous, explanatory variables X can be continuous or categorical (function can be used for linear regressions, ANCOVAs and ANOVAs)
`coef(model)`	Returns the estimated coefficients (sometimes with their standard-errors)
`summary(model)`	Produces a table of parameter estimates, standard errors, t-statistics and p values. Can also be defined explicitely via `summary.lm()`
`confint(model)`	Computes confidence intervals for one or more parameters in a fitted model
`broom::tidy()`	Returns tibble with estimated coefficients, std.errors, test statistic and p-value
`broom::augment()`	Returns tibble with observational data and model output (e.g. fitted values with standard error, residuals, Cook's D, etc.)
`broom::glance()`	Constructs a single row summary "glance" of a model, fit, or other object
`effects()`	Returns (orthogonal) effects from a fitted model, usually a linear model (lm or glm objects)
`residuals(model)`	Returns the model residuals (`resids(model)` can be similarly used)
`fitted(model)`	Returns the fitted values
`predict(model)`	Returns predicted values for observed or generated predictor values (via argument *newdata*); can also be defined explicitely via `predict.lm()`

Useful *modelr* functions

`modelr::add_predictions(data, model)`
Adds predictions from a single model to a data frame
`modelr::spread_predictions(data, model1, model2,..)`
Adds predictions from several models in wide format
`modelr::gather_predictions(data, model1, model2,..)`
Adds predictions from several models in long format
`modelr::data_grid(data, x1, x2)`
Generate an evenly spaced grid of points from the data
`modelr::add_residuals(data, model)`
Adds residuals from a single model to a data frame
`modelr::spread_residuals(data, model1, model2,..)`
Adds residuals from several models in wide format
`modelr::gather_residuals(data, model1, model2,..)`
Adds residuals from several models in long format

Model selection

`AIC(model1, model2)`	Computes the Akaike information criterion or AIC
`step(model)`	Select a formula-based model by AIC in a stepwise algorithm (backward and forward selection); works only for lm and glm objects
`anova(m1,m2, test="F")`	Compares two (or more) nested lm or aov models based on the F-test
`anova(m1,m2, test="Chisq")`	Compares two (or more) nested glm models based on the log-likelihood ratio test (LRT)
`drop1(model, test="F")`	Generates alternative models where each term is dropped one at a time respecting their hierarchy, fits those models and computes a table of the changes in fit including an F-test (or LRT, Rao,..)

Model diagnostics

`plot.lm(model)`	Produces diagnostic plots to check model assumptions and influential data points for an lm object
`cooks.distance(model)`	Computes the (leave-one-out deletion) diagnostics for linear and generalized linear models
`rstandard(model)`	Standardized residuals
`rstudent(model)`	Studentized residuals
`qqnorm(y)`	Produces a normal quantile-quantile plot of the values in y
`qqplot()`	Produces quantile-quantile plot
`qqline()`	Adds a line to a normal quantile-quantile plot (created with `qqplot()`), which passes through the first and third quartiles
`acf()`	Computes (and by default plots) estimates of the autocovariance or autocorrelation function
`pacf()`	Computes the partial autocorrelation function
`ccf()`	Computes the cross-correlation or cross-covariance of two univariate series
`car::vif(model)`	Calculates variance-inflation and generalized variance-inflation factors for linear, generalized linear, and other models

Experimental design

Power analysis

Compute the power or determine parameters to obtain a target power:

`power.t.test()`	One- or two- sample t test
`power.anova.test()`	Balanced one-way ANOVA
`power.prop.test()`	Two-sample test for proportions