

Package ‘rigr’

April 3, 2026

Title Regression, Inference, and General Data Analysis Tools in R

Version 1.0.9

Description A set of tools to streamline data analysis. Learning both R and introductory statistics at the same time can be challenging, and so we created 'rigr' to facilitate common data analysis tasks and enable learners to focus on statistical concepts. We provide easy-to-use interfaces for descriptive statistics, one- and two-sample inference, and regression analyses. 'rigr' output includes key information while omitting unnecessary details that can be confusing to beginners. Heteroscedasticity-robust ("sandwich") standard errors are returned by default, and multiple partial F-tests and tests for contrasts are easy to specify. A single regression function can fit both linear and generalized linear models, allowing students to more easily make connections between different classes of models.

License MIT + file LICENSE

BugReports <https://github.com/statdivlab/rigr/issues/>

Depends R (>= 3.5.0)

Imports sandwich, stats, survival

Suggests testthat (>= 3.0.0), knitr, rmarkdown, car

VignetteBuilder knitr

RoxygenNote 7.3.3

Encoding UTF-8

Config/testthat/edition 3

LazyData true

URL <https://statdivlab.github.io/rigr/>

NeedsCompilation no

Author Amy D Willis [aut, cre] (ORCID:
<<https://orcid.org/0000-0002-2802-4317>>),
Taylor Okonek [aut],
Charles J Wolock [aut],
Brian D Williamson [aut],
Scott S Emerson [aut],
Andrew J Spieker [aut],
Yiqun T Chen [aut],

Travis Y Hee Wai [ctb],
 James P Hughes [ctb],
 R Core Team [ctb],
 Akhil S Bhel [ctb],
 Thomas Lumley [ctb]

Maintainer Amy D Willis <adwillis@uw.edu>

Repository CRAN

Date/Publication 2026-04-03 21:30:02 UTC

Contents

rigr-package	3
anova.uRegress	3
cooks.distance.uRegress	5
descrip	5
dfbeta.uRegress	8
dfbetas.uRegress	9
dummy	9
fev	10
hatvalues.uRegress	11
lincom	11
mri	13
polynomial	15
predict.uRegress	16
proptest	17
proptesti	18
psa	20
regress	21
residuals.uRegress	23
rstandard.uRegress	25
rstudent.uRegress	25
salary	26
ttest	27
ttesti	29
U	30
wilcoxon	31

Index	34
--------------	-----------

Description

Developed by Scott S. Emerson, Andrew J. Spieker, Brian D. Williamson, and Travis Y. Hee Wai at the University of Washington Department of Biostatistics. Currently maintained by Prof. Amy Willis at the University of Washington Department of Biostatistics. Previously maintained by Charles Wolock and Taylor Okonek, also at the University of Washington Department of Biostatistics. Aims to facilitate regression, descriptive statistics, and one- and two-sample inference by implementing more intuitive layout and functionality for existing R functions.

Details

Package: rigr
Type: Package
Version: 1.0.0
Date: 2021-09-10
License: MIT

A set of tools designed to facilitate easy adoption of R for students in introductory classes with little programming experience. Compiles output from existing routines together in an intuitive format, and adds functionality to existing functions. For instance, the regression function can perform linear models and generalized linear models. The user can also specify multiple-partial F-tests to print out with the model coefficients, and robust standard errors are provided automatically. We also provide functions for descriptive statistics and one- and two-sample inference with improved, legible output.

Author(s)

Scott S. Emerson, Andrew J. Spieker, Brian D. Williamson, Amy D. Willis, Charles Wolock, and Taylor Okonek

Maintainer: Amy Willis <adwillis@uw.edu>

Description

Compute analysis of variance (or deviance) tables for two fitted, nested uRegress objects. The model with more parameters is referred to as the full model (or the larger model), and the model with fewer parameters is referred to as the null model (or the smaller model).

Usage

```
## S3 method for class 'uRegress'
anova(object, full_object, test = "LRT", robustSE = TRUE, useFdstn = TRUE, ...)
```

Arguments

<code>object</code>	an object of class <code>uRegress</code> , the model with fewer parameters (i.e. the null model).
<code>full_object</code>	an object of class <code>uRegress</code> , the model with more parameters (i.e. the full model).
<code>test</code>	a character string specifying the test statistic to be used. Can be one of 'Wald' or 'LRT', which corresponds to Wald or likelihood (partial likelihood for hazard regressions) ratio tests. Note that currently the Wald test is only supported for symbolically nested models; that is, when the larger model contains all the covariates (with the same names) in the smaller model.
<code>robustSE</code>	a logical value indicating whether or not to use robust standard errors in calculation. Defaults to TRUE. If TRUE, then <code>robustSE</code> must have been TRUE when <code>reg</code> was created.
<code>useFdstn</code>	a logical indicator that the F distribution should be used for test statistics instead of the chi squared distribution. Defaults to FALSE. This option is not supported when input <code>reg</code> is a hazard regression (i.e., <code>fnctl="hazard"</code>).
<code>...</code>	argument to be passed in

Value

A list of class `anova.uRegress` with the following components:

printMat A formatted table with inferential results (i.e., test statistics and p-values) for comparing two nested models.

null_model The null model in the comparison.

full_model The full model in the comparison.

Examples

```
# Loading required libraries
library(sandwich)

# Reading in a dataset
data(mri)

# Linear regression of LDL on age and stroke (with robust SE by default)
testReg_null <- regress("mean", ldl~age+stroke, data = mri)

# Linear regression of LDL on age, stroke, and race (with robust SE by default)
testReg_full <- regress("mean", ldl~age+stroke+race, data = mri)
# Comparing the two models using the Wald test with robust SE
anova(testReg_null, testReg_full, test = "Wald")
```

 cooks.distance.uRegress

Calculate Cook's distances from uRegress objects

Description

Extracts Cook's distances from uRegress objects by relying on functionality from the stats package.

Usage

```
## S3 method for class 'uRegress'
cooks.distance(model, ...)
```

Arguments

model an object of class uRegress, as returned by [regress](#).
 ... other arguments to pass to stats::cooks.distance

Value

a vector of Cook's distances

descrip

Descriptive Statistics

Description

Produces table of relevant descriptive statistics for an arbitrary number of variables of class integer, numeric, Surv, Date, or factor. Descriptive statistics can be obtained within strata, and the user can specify that only a subset of the data be used. Descriptive statistics include the count of observations, the count of cases with missing values, the mean, standard deviation, geometric mean, minimum, and maximum. The user can specify arbitrary quantiles to be estimated, as well as specifying the estimation of proportions of observations within specified ranges.

Usage

```
descrip(
  ...,
  strata = NULL,
  subset = NULL,
  probs = c(0.25, 0.5, 0.75),
  geomInclude = FALSE,
  replaceZeroes = FALSE,
  restriction = Inf,
```

```

above = NULL,
below = NULL,
labove = NULL,
rbelow = NULL,
lbetween = NULL,
rbetween = NULL,
interval = NULL,
linterval = NULL,
rinterval = NULL,
lrinterval = NULL
)

```

Arguments

...	an arbitrary number of variables for which descriptive statistics are desired. The arguments can be vectors, matrices, or lists. Individual columns of a matrix or elements of a list may be of class <code>numeric</code> , <code>factor</code> , <code>Surv</code> , or <code>Date</code> . Factor variables are converted to integers. Character vectors will be coerced to numeric. Variables may be of different lengths, unless <code>strata</code> or <code>subset</code> are non-NULL. A single <code>data.frame</code> or <code>tibble</code> may also be entered, in which case each variable in the object will be described.
<code>strata</code>	a vector, matrix, or list of stratification variables. Descriptive statistics will be computed within strata defined by each unique combination of the stratification variables, as well as in the combined sample. If <code>strata</code> is supplied, all variables must be of that same length.
<code>subset</code>	a vector indicating a subset to be used for all descriptive statistics. If <code>subset</code> is supplied, all variables must be of that same length.
<code>probs</code>	a vector of probabilities between 0 and 1 indicating quantile estimates to be included in the descriptive statistics. Default is to compute 25th, 50th (median) and 75th percentiles.
<code>geomInclude</code>	if not <code>FALSE</code> (the default), includes the geometric mean in the descriptive statistics.
<code>replaceZeroes</code>	if not <code>FALSE</code> (the default), this indicates a value to be used in place of zeroes when computing a geometric mean. If <code>TRUE</code> , a value equal to one-half the lowest nonzero value is used. If a numeric value is supplied, that value is used for all variables.
<code>restriction</code>	a value used for computing restricted means, standard deviations, and geometric means with censored time-to-event data. The default value of <code>Inf</code> will cause restrictions at the highest observation. Note that the same value is used for all variables of class <code>Surv</code> .
<code>above</code>	a vector of values used to dichotomize variables. The descriptive statistics will include an estimate for each variable of the proportion of measurements with values greater than each element of <code>above</code> .
<code>below</code>	a vector of values used to dichotomize variables. The descriptive statistics will include an estimate for each variable of the proportion of measurements with values less than each element of <code>below</code> .

<code>labove</code>	a vector of values used to dichotomize variables. The descriptive statistics will include an estimate for each variable of the proportion of measurements with values greater than or equal to each element of <code>labove</code> .
<code>rbelow</code>	a vector of values used to dichotomize variables. The descriptive statistics will include an estimate for each variable of the proportion of measurements with values less than or equal to each element of <code>rbelow</code> .
<code>lbetween</code>	a vector of values with <code>-Inf</code> and <code>Inf</code> appended is used as cutpoints to categorize variables. The descriptive statistics will include an estimate for each variable of the proportion of measurements with values between successive elements of <code>lbetween</code> , with the left-hand endpoint included in each interval.
<code>rbetween</code>	a vector of values with <code>-Inf</code> and <code>Inf</code> appended is used as cutpoints to categorize variables. The descriptive statistics will include an estimate for each variable of the proportion of measurements with values between successive elements of <code>rbetween</code> , with the right-hand endpoint included in each interval.
<code>interval</code>	a two-column matrix of values in which each row is used to define intervals of interest to categorize variables. The descriptive statistics will include an estimate for each variable of the proportion of measurements with values between two elements in a row, with neither endpoint included in each interval.
<code>linterval</code>	a two-column matrix of values in which each row is used to define intervals of interest to categorize variables. The descriptive statistics will include an estimate for each variable of the proportion of measurements with values between two elements in a row, with the left-hand endpoint included in each interval.
<code>rinterval</code>	a two-column matrix of values in which each row is used to define intervals of interest to categorize variables. The descriptive statistics will include an estimate for each variable of the proportion of measurements with values between two elements in a row, with the right-hand endpoint included in each interval.
<code>lrinterval</code>	a two-column matrix of values in which each row is used to define intervals of interest to categorize variables. The descriptive statistics will include an estimate for each variable of the proportion of measurements with values between two elements in a row, with both endpoints included in each interval.

Details

This function depends on the `survival` R package. You should execute `library(survival)` if that library has not been previously installed. Quantiles are computed for uncensored data using the default method in `quantile()`. For variables of class `factor`, descriptive statistics will be computed using the integer coding for factors. For variables of class `Surv`, estimated proportions and quantiles will be computed from Kaplan-Meier estimates, as will be restricted means, restricted standard deviations, and restricted geometric means. For variables of class `Date`, estimated proportions will be labeled using the Julian date since January 1, 1970.

Value

An object of class `uDescriptives` is returned. Descriptive statistics for each variable in the entire subsetted sample, as well as within each stratum if any is defined, are contained in a matrix with rows corresponding to variables and strata and columns corresponding to the descriptive statistics. Descriptive statistics include

- N: the number of observations.
- Msng: the number of observations with missing values.
- Mean: the mean of the nonmissing observations (this is potentially a restricted mean for right-censored time-to-event data).
- Std Dev: the standard deviation of the nonmissing observations (this is potentially a restricted standard deviation for right-censored time to event data).
- Geom Mn: the geometric mean of the nonmissing observations (this is potentially a restricted geometric mean for right-censored time to event data). Nonpositive values in the variable will generate NA, unless `replaceZeroes` was specified.
- Min: the minimum value of the nonmissing observations (this is potentially restricted for right-censored time-to-event data).
- Quantiles: columns corresponding to the quantiles specified by `probs` (these are potentially restricted for right-censored time-to-event data).
- Max: the maximum value of the nonmissing observations (this is potentially restricted for right-censored time-to-event data).
- Proportions: columns corresponding to the proportions as specified by `above`, `below`, `labove`, `rbelow`, `lbetween`, `rbetween`, `interval`, `linterval`, `rinterval`, and `lrinterval`.
- `restriction`: the threshold for restricted means, standard deviations, and geometric means.
- `firstEvent`: the time of the first event for censored time-to-event variables.
- `lastEvent`: the time of the last event for censored time-to-event variables.
- `isDate`: an indicator that the variable is a Date object.

Examples

```
# Read in the data
data(mri)

# Create the table
descrip(mri)
```

<code>dfbeta.uRegress</code>	<i>Calculate dfbeta from uRegress objects</i>
------------------------------	---

Description

Extracts `dfbeta` from `uRegress` objects by relying on functionality from the `stats` package. Note that `dfbeta` and `dfbetas` are not the same (`dfbetas` are less than the `dfbeta` values by a scaling factor that reflects both the leverage of the observation in question and the residual model error).

Usage

```
## S3 method for class 'uRegress'
dfbeta(model, ...)
```

Arguments

model an object of class uRegress, as returned by [regress](#).
 ... other arguments to pass to stats::dfbeta

Value

a matrix of dfbeta values, with a row for each observation and a column for each model coefficient

dfbetas.uRegress *Calculate dfbetas from uRegress objects*

Description

Extracts dfbetas from uRegress objects by relying on functionality from the stats package. Note that dfbeta and dfbetas are not the same (dfbetas are less than the dfbeta values by a scaling factor that reflects both the leverage of the observation in question and the residual model error).

Usage

```
## S3 method for class 'uRegress'
dfbetas(model, ...)
```

Arguments

model an object of class uRegress, as returned by [regress](#).
 ... other arguments to pass to stats::dfbetas

Value

a matrix of dfbetas values, with a row for each observation and a column for each model coefficient

dummy *Create Dummy Variables*

Description

Creates dummy variables using a reference value.

Usage

```
dummy(
  x,
  subset = rep(TRUE, length(x)),
  reference = sort(unique(x[!is.na(x)])),
  includeAll = FALSE
)
```

Arguments

<code>x</code>	<code>y</code> variable used to create the dummy variables.
<code>subset</code>	cluster a subset of the data, if desired.
<code>reference</code>	the reference value for the dummy variables to compare to.
<code>includeAll</code>	logical value indicating whether all of the dummy variables should be returned (including the reference).

Value

A matrix containing the dummy variables.

Examples

```
data(mri)

# Create a dummy variable for chd
dummy(mri$chd)
```

fev	<i>FEV dataset</i>
-----	--------------------

Description

Data from a study of 654 children on the relationship between smoking status and lung function (measured by FEV). Each row corresponds to a single clinic visit and contains information on age, height, sex, FEV, and smoking status. More information, including a coding key, is available at rct-design.com (Teaching Materials, Datasets section).

Usage

```
fev
```

Format

A data frame with 654 rows and 7 variables:

seqnbr case number (the numbers 1 to 654)
subjid subject identification number (unique for each different child)
age subject age at time of measurement (years)
fev measured forced exhalation volume (liters per second)
height subject height at time of measurement (inches)
sex subject sex
smoke smoking habits ("yes" or "no")

Source

Dataset originally distributed at rct-design.com (Teaching Materials, Datasets section).

hatvalues.uRegress	<i>Calculate the hat-values (leverages) from uRegress objects</i>
--------------------	---

Description

Extracts hat-values (leverages) from uRegress objects by relying on functionality from the stats package.

Usage

```
## S3 method for class 'uRegress'
hatvalues(model, ...)
```

Arguments

model	an object of class uRegress, as returned by regress .
...	other arguments to pass to stats::hatvalues

Value

a vector of hat-values (leverages)

lincom	<i>Tests of Linear Combinations of Regression Coefficients</i>
--------	--

Description

Produces point estimates, interval estimates, and p-values for linear combinations of regression coefficients using a uRegress object.

Usage

```
lincom(
  reg,
  comb,
  null.hypoth = 0,
  conf.level = 0.95,
  robustSE = TRUE,
  joint.test = FALSE,
  useFdstdn = FALSE,
  eform = reg$fnctl != "mean"
)
```

Arguments

<code>reg</code>	an object of class <code>uRegress</code> .
<code>comb</code>	a vector or matrix containing the values of the constants which create the linear combination of the form $c_0 + c_1\beta_1 + \dots$ Zeroes must be given if coefficients aren't going to be included. For testing multiple combinations, this must be a matrix with number of columns equal to the number of coefficients in the model.
<code>null.hypoth</code>	the null hypothesis to compare the linear combination of coefficients against. This is a scalar if one combination is given, and a vector or matrix otherwise. The default value is \emptyset .
<code>conf.level</code>	a number between 0 and 1, indicating the desired confidence level for intervals.
<code>robustSE</code>	a logical value indicating whether or not to use robust standard errors in calculation. Defaults to TRUE. If TRUE, then <code>robustSE</code> must have been TRUE when <code>reg</code> was created.
<code>joint.test</code>	a logical value indicating whether or not to use a joint Chi-square test for all the null hypotheses. If <code>joint.test</code> is TRUE, then no confidence interval is calculated. Defaults to FALSE.
<code>useFdstn</code>	a logical indicator that the F distribution should be used for test statistics instead of the chi squared distribution. Defaults to TRUE. This option is not supported when input <code>reg</code> is a hazard regression (i.e., <code>fnct1="hazard"</code>).
<code>eform</code>	a logical value indicating whether or not to exponentiate the estimated coefficient. By default this is performed based on the type of regression used.

Value

A list of class `lincom` (`joint.test` is `False`) or `lincom.joint` (`joint.test` is `True`). For the `lincom` class, `comb` entries in the list are labeled `comb1`, `comb2`, etc. for as many linear combinations were used. Each is a list with the following components:

printMat A formatted table with inferential results for the linear combination of coefficients. These include the point estimate, standard error, confidence interval, and t-test for the linear combination.

nms The name of the linear combination, for printing.

null.hypoth The null hypothesis for the linear combination.

Examples

```
# Loading required libraries
library(sandwich)

# Reading in a dataset
data(mri)

# Linear regression of LDL on age (with robust SE by default)
testReg <- regress ("mean", ldl~age+stroke, data = mri)
```

```

# Testing coefficient created by .5*age - stroke (the first 0 comes from excluding the intercept)
testC <- c(0, 0.5, -1)
lincom(testReg, testC)

# Test multiple combinations:
# whether separately whether .5*age - stroke = 0 or Intercept + 60*age = 125
testC <- matrix(c(0, 0.5, -1, 1, 60, 0), byrow = TRUE, nrow = 2)
lincom(testReg, testC, null.hypoth = c(0, 125))

# Test joint null hypothesis:
# H0: .5*age - stroke = 0 AND Intercept + 60*age = 125
lincom(testReg, testC, null.hypoth = c(0, 125), joint.test = TRUE)

```

mri

MRI dataset

Description

Data from an observational study of the incidence of cardiovascular disease (especially heart attacks and congestive heart failure) and cerebrovascular disease (especially strokes) in the U.S. elderly. More information, including a coding key, is available at rct-design.com (Teaching Materials, Datasets section).

Usage

```
mri
```

Format

A data frame with 735 rows and 30 variables:

ptid Participant identification number.

mridate The date on which the participant underwent MRI scan in MMDDYY format.

age Participant age at time of MRI, in years.

sex The sex of the participant. Only ‘Male’ and ‘Female’ are represented.

race Participant’s race. One of the following: ‘White’, ‘Black’, ‘Asian’, or ‘Subject did not identify as White, Black or Asian’. It is unclear if study participants self-identified their race, or if it was guessed by the study organisers.

weight Participant’s weight at time of MRI (pounds).

height Participant’s height at time of MRI (centimeters).

packyrs Participant smoking history in pack years (1 pack year = smoking 1 pack of cigarettes per day for 1 year). A participant who has never smoked has 0 pack years.

yrsquit Number of years since quitting smoking. A current smoker will have a nonzero packyrs and a 0 for yrsquit. A never smoker will have a zero for both variables.

- alcoh** Average alcohol intake for the participant for the two weeks prior to MRI (drinks per week, where one drink is 1 oz. whiskey, 4 oz. wine, or 12 oz. beer).
- physact** Physical activity of the participant for the week prior to MRI (1,000 kcal).
- chf** Indicator of whether the participant had been diagnosed with congestive heart failure prior to MRI (0=no, 1=yes).
- chd** Indicator of whether the participant had been diagnosed with coronary heart disease prior to MRI (0=no, 1=diagnosis of angina, 2=diagnosis of myocardial infarction).
- stroke** Indicator of whether the participant had been diagnosed with a cerebrovascular event prior to MRI (0=no, 1=diagnosis of a transient ischemic attack, 2=diagnosis of stroke).
- diabetes** Indicator of whether the participant had been diagnosed with diabetes prior to MRI (0=no, 1=yes).
- genhlth** an indicator of the participant's view of their own health (1=excellent, 2=very good, 3=good, 4=fair, 5=poor)
- ldl** a laboratory measure of low density lipoprotein (a kind of cholesterol) in the participant's blood at the time of MRI (mg/dL).
- alb** a laboratory measure of albumin, a kind of protein, in the participant's blood at the time of MRI (g/L).
- crt** a laboratory measure of creatinine, a waste product, in the participant's blood at the time of MRI (mg/dL).
- plt** a laboratory measure of the number of platelets circulating in the participant's blood at the time of MRI (1000 per cubic mm).
- sbp** a measurement of the participant's systolic blood pressure in their arm at the time of MRI (mm Hg).
- aa1** the ratio of systolic blood pressure measured in the participant's ankle at time of MRI to the systolic blood pressure in the participant's arm.
- fev** a measure of the forced expiratory volume in the participant at the time of MRI (L/sec).
- dsst** a measure of cognitive function (Digit Symbol Substitution Test) for the participant at the time of MRI. Maximum score possible is 100.
- atrophy** a measure of loss of neurons estimated by the degree of ventricular enlargement relative to the predicted ventricular size; with 0 indicating no atrophy and 100 indicating the most severe degree of atrophy.
- whgrd** a measure of white matter changes detected on MRI. 0 means no changes, 9 means marked changes.
- numinf** a count of the number of distinct regions identified on MRI scan which were suggestive of infarcts.
- volinf** a measure of the total volume of infarct-like lesions found on MRI scan (cubic cm).
- obstime** the total time (in days) that the participant was observed on study between the date of MRI and death or September 16, 1997, whichever came first.
- death** an indicator that the participant was observed to die while on study. If 1, the number of days recorded in `obstime` is the number of days from that participant's MRI to their death. If 0, the number of days in `obstime` is the number of days between that participant's MRI and September 16, 1997.

Source

Dataset originally distributed at rct-design.com (Teaching Materials, Datasets section).

polynomial	<i>Create Polynomials</i>
------------	---------------------------

Description

Creates polynomial variables, to be used in regression. Will create polynomials of degree less than or equal to the degree specified, and will mean center variables by default.

Usage

```
polynomial(x, degree = 2, center = mean(x, na.rm = TRUE))
```

Arguments

x	variable used to create the polynomials.
degree	the maximum degree polynomial to be returned. Polynomials of degree \leq degree will be returned.
center	the value to center the polynomials at.

Value

A matrix containing the linear splines.

Examples

```
# Reading in a dataset
data(mri)

# Create a polynomial on ldl
polynomial(mri$ldl, degree=3)

# Use a polynomial in regress
regress("mean", atrophy ~ polynomial(age, degree = 2), data = mri)
```

predict.uRegress *Prediction Intervals for uRegress objects*

Description

Produces prediction intervals for objects of class uRegress.

Usage

```
## S3 method for class 'uRegress'  
predict(object, interval = "prediction", level = 0.95, ...)
```

Arguments

object	an object of class uRegress.
interval	Type of interval calculation
level	Tolerance/confidence level
...	other arguments to pass to the appropriate predict function for the class of object\$fit. See predict.lm , or predict.glm for more details.

Value

Returns a matrix with the fitted value and prediction interval for the entered X.

See Also

[regress](#)

Examples

```
# Loading required libraries  
library(survival)  
library(sandwich)  
  
# Reading in a dataset  
data(mri)  
  
# Linear regression of LDL on age (with robust SE by default)  
testReg <- regress ("mean", ldl~age, data = mri)  
  
# 95% Prediction Interval for age 50  
predict(testReg)
```

 proptest

Test of proportions with improved layout

Description

Performs a one- or two-sample test of proportions using data. This test can be approximate or exact.

Usage

```
proptest(
  var1,
  var2 = NULL,
  by = NULL,
  exact = FALSE,
  null.hypoth = ifelse(is.null(var2) && is.null(by), 0.5, 0),
  alternative = "two.sided",
  conf.level = 0.95,
  correct = FALSE,
  more.digits = 0
)
```

Arguments

var1	a (non-empty) vector of binary numeric (0-1), binary factor, or logical data values
var2	an optional (non-empty) vector of binary numeric (0-1), binary factor, or logical data values
by	a variable of equal length to that of var1 with two outcomes (numeric or factor). This will be used to define strata for a prop test on var1.
exact	If true, performs a test of equality of proportions using exact binomial probabilities.
null.hypoth	a number specifying the null hypothesis for the mean (or difference in means if performing a two-sample test). Defaults to 0.5 for a one-sample test and 0 for a two-sample test.
alternative	a string: one of "less", "two.sided", or "greater" specifying the form of the test. Defaults to a two-sided test.
conf.level	confidence level of the test. Defaults to 0.95.
correct	a logical indicating whether to perform a continuity correction
more.digits	a numeric value specifying whether or not to display more or fewer digits in the output. Non-integers are automatically rounded down.

Details

Missing values must be given by "NA"s to be recognized as missing values. Numeric data must be given in 0-1 form. This function also accepts binary factor variables, treating the higher level as 1 and the lower level as 0, or logical variables.

Value

A list of class `proptest`. The print method lays out the information in an easy-to-read format.

tab A formatted table of descriptive and inferential results (total number of observations, number of missing observations, sample proportion, standard error of the proportion estimate), along with a confidence interval for the underlying proportion.

zstat the value of the test statistic, if using an approximate test.

pval the p-value for the test

var1 The user-supplied first data vector.

var2 The user-supplied second data vector.

by The user-supplied stratification variable.

par A vector of information about the type of test (null hypothesis, alternative hypothesis, etc.)

See Also

[prop.test](#)

Examples

```
# Read in data set
data(psa)
attach(psa)

# Define new binary variable as indicator
# of whether or not bss was worst possible
bssworst <- bss
bssworst[bss == 1] <- 0
bssworst[bss == 2] <- 0
bssworst[bss == 3] <- 1

# Perform test comparing proportion in remission
# between bss strata
proptest(factor(inrem), by = bssworst)
```

proptesti

Test of proportions from summary statistics

Description

Performs a one- or two-sample test of proportions using counts of successes and trials, rather than binary data. This test can be approximate or exact.

Usage

```

proptesti(
  x1,
  n1,
  x2 = NULL,
  n2 = NULL,
  exact = FALSE,
  null.hypoth = ifelse(is.null(x2) && is.null(n2), 0.5, 0),
  conf.level = 0.95,
  alternative = "two.sided",
  correct = FALSE,
  more.digits = 0
)

```

Arguments

<code>x1</code>	Number of successes in first sample
<code>n1</code>	Number of trials in first sample
<code>x2</code>	Number of successes in second sample
<code>n2</code>	Number of trials in second sample
<code>exact</code>	If true, performs a test of equality of proportions with Exact Binomial based confidence intervals.
<code>null.hypoth</code>	a number specifying the null hypothesis for the mean (or difference in means if performing a two-sample test). Defaults to 0.5 for one-sample and 0 for two-sample.
<code>conf.level</code>	confidence level of the test. Defaults to 0.95
<code>alternative</code>	a string: one of "less", "two.sided", or "greater" specifying the form of the test. Defaults to a two-sided test. When either "less" or "greater" is used, the corresponding one-sided confidence interval is returned.
<code>correct</code>	a logical indicating whether to perform a continuity correction
<code>more.digits</code>	a numeric value specifying whether or not to display more or fewer digits in the output. Non-integers are automatically rounded down.

Details

If `x2` or `n2` are specified, then both must be specified, and a two-sample test is run.

Value

A list of class `proptesti`. The print method lays out the information in an easy-to-read format.

tab A formatted table of descriptive and inferential results (total number of observations, sample proportion, standard error of the proportion estimate), along with a confidence interval for the underlying proportion.

zstat the value of the test statistic, if using an approximate test.

pval the p-value for the test

par A vector of information about the type of test (null hypothesis, alternative hypothesis, etc.)

Examples

```
# Two-sample test
proptesti(10, 100, 15, 200, alternative = "less")
```

psa

PSA dataset

Description

Data from a study of 50 men having hormonally treated prostate cancer. Includes information on PSA levels, tumor characteristics, remission status, age, and disease state. More information, including a coding key, is available at rct-design.com (Teaching Materials, Datasets section).

Usage

```
psa
```

Format

A data frame with 50 rows and 9 variables:

ptid patient identifier

nadirpsa lowest PSA value attained post therapy (ng/ml)

pretxpsa PSA value prior to therapy (ng/ml)

ps performance status (0= worst, 100= best)

bss bone scan score (1= least disease, 3= most)

grade tumor grade (1= least aggressive, 3= most)

age patient's age (years)

obstime time observed in remission (months)

inrem Indicator whether patient still in remission at last follow-up (yes or no)

Source

Dataset originally distributed at rct-design.com (Teaching Materials, Datasets section).

regress

*General Regression for an Arbitrary Functional***Description**

Produces point estimates, interval estimates, and p values for an arbitrary functional (mean, geometric mean, proportion, odds, hazard) of a variable of class `integer`, or `numeric` when regressed on an arbitrary number of covariates. Multiple Partial F-tests can be specified using the `U` function.

Usage

```
regress(
  fnctl,
  formula,
  data,
  intercept = TRUE,
  weights = rep(1, nrow(data.frame(data))),
  subset = rep(TRUE, nrow(data.frame(data))),
  robustSE = TRUE,
  conf.level = 0.95,
  exponentiate = fnctl != "mean",
  replaceZeroes,
  useFdstn = TRUE,
  suppress = FALSE,
  na.action,
  method = "qr",
  qr = TRUE,
  singular.ok = TRUE,
  contrasts = NULL,
  init = NULL,
  ties = "efron",
  offset,
  control = list(...),
  ...
)
```

Arguments

<code>fnctl</code>	a character string indicating the functional (summary measure of the distribution) for which inference is desired. Choices include "mean", "geometric mean", "odds", "rate", "hazard".
<code>formula</code>	an object of class <code>formula</code> as might be passed to <code>lm</code> , <code>glm</code> , or <code>coxph</code> . Functions of variables, specified using <code>dummy</code> or <code>polynomial</code> may also be included in <code>formula</code> .
<code>data</code>	a data frame, matrix, or other data structure with matching names to those entered in <code>formula</code> .

<code>intercept</code>	a logical value indicating whether a intercept exists or not. Default value is TRUE for all functionals. Intercept may also be removed if a "-1" is present in formula. If "-1" is present in formula but <code>intercept = TRUE</code> is specified, the model will fit without an intercept. Note that when <code>fnctl = "hazard"</code> , the intercept is always set to FALSE because Cox proportional hazards regression models do not explicitly estimate an intercept.
<code>weights</code>	vector indicating optional weights for weighted regression.
<code>subset</code>	vector indicating a subset to be used for all inference.
<code>robustSE</code>	a logical indicator that standard errors (and confidence intervals) are to be computed using the Huber-White sandwich estimator. The default is TRUE.
<code>conf.level</code>	a numeric scalar indicating the level of confidence to be used in computing confidence intervals. The default is 0.95.
<code>exponentiate</code>	a logical indicator that the regression parameters should be exponentiated. This is by default true for all functionals except the mean.
<code>replaceZeroes</code>	if not FALSE, this indicates a value to be used in place of zeroes when computing a geometric mean. If TRUE, a value equal to one-half the lowest nonzero value is used. If a numeric value is supplied, that value is used. Defaults to TRUE when <code>fnctl = "geometric mean"</code> . This parameter is always FALSE for all other values of <code>fnctl</code> .
<code>useFdstn</code>	a logical indicator that the F distribution should be used for test statistics instead of the chi squared distribution even in logistic regression models. When using the F distribution, the degrees of freedom are taken to be the sample size minus the number of parameters, as it would be in a linear regression model.
<code>suppress</code>	if TRUE, and a model which requires exponentiation (for instance, regression on the geometric mean) is computed, then a table with only the exponentiated coefficients and confidence interval is returned. Otherwise, two tables are returned - one with the original unexponentiated coefficients, and one with the exponentiated coefficients.
<code>na.action, qr, singular.ok, offset, contrasts, control</code>	optional arguments that are passed to the functionality of <code>lm</code> or <code>glm</code> .
<code>method</code>	the method to be used in fitting the model. The default value for <code>fnctl = "mean"</code> and <code>fnctl = "geometric mean"</code> is "qr", and the default value for <code>fnctl = "odds"</code> and <code>fnctl = "rate"</code> is "glm.fit". This argument is passed into the <code>lm()</code> or <code>glm()</code> function, respectively. You may optionally specify <code>method = "model.frame"</code> , which returns the model frame and does no fitting.
<code>init</code>	a numeric vector of initial values for the regression parameters for the hazard regression. Default initial value is zero for all variables.
<code>ties</code>	a character string describing method for breaking ties in hazard regression. Only <code>efron</code> , <code>breslow</code> , or <code>exact</code> is accepted. See more details in the documentation for this argument in the <code>survival::coxph</code> function. Default to <code>efron</code> .
<code>...</code>	additional arguments to be passed to the <code>lm</code> function call

Details

Regression models include linear regression (for the "mean" functional), logistic regression with logit link (for the "odds" functional), Poisson regression with log link (for the "rate" functional),

linear regression of a log-transformed outcome (for the “geometric mean” functional), and Cox proportional hazards regression (for the hazard functional).

Currently, for the hazard functional, only ‘coxph’ syntax is supported; in other words, using ‘dummy’, ‘polynomial’, and **U** functions will result in an error when ‘fnctl = hazard’.

Note that the only possible link function in ‘regress’ with ‘fnctl = odds’ is the logit link. Similarly, the only possible link function in ‘regress’ with ‘fnctl = "rate"' is the log link.

Objects created using the **U** function can also be passed in. If the **U** call involves a partial formula of the form $\sim \text{var1} + \text{var2}$, then `regress` will return a multiple-partial F-test involving `var1` and `var2`. If an F-statistic will already be calculated regardless of the **U** specification, then any naming convention specified via `name ~ var1` will be ignored. The multiple partial tests must be the last terms specified in the model (i.e. no other predictors can follow them).

Value

An object of class `uRegress` is returned. Parameter estimates, confidence intervals, and p values are contained in a matrix `$augCoefficients`.

See Also

Functions for fitting linear models ([lm](#)), and generalized linear models ([glm](#)). Also see the function to specify multiple-partial F-tests, [U](#).

Examples

```
# Loading dataset
data(mri)

# Linear regression of atrophy on age
regress("mean", atrophy ~ age, data = mri)

# Linear regression of atrophy on sex and height and their interaction,
# with a multiple-partial F-test on the height-sex interaction
regress("mean", atrophy ~ height + sex + U(hs=~height:sex), data = mri)

# Logistic regression of sex on atrophy
mri$sex_bin <- ifelse(mri$sex == "Female", 1, 0)
regress("odds", sex_bin ~ atrophy, data = mri)

# Cox regression of age on survival
library(survival)
regress("hazard", Surv(obstime, death)~age, data=mri)
```

residuals.uRegress *Extract Residuals from uRegress objects*

Description

Extracts residuals (unstandardized, standardized, studentized, or jackknife) from `uRegress` objects.

Usage

```
## S3 method for class 'uRegress'  
residuals(object, type = "", ...)
```

Arguments

object	an object of class <code>uRegress</code> , as returned by regress .
type	denotes the type of residuals to return. Default value is "", which returns unstandardized residuals. "standardized", "studentized", and "jackknife" return the expected type of residuals.
...	other arguments

Details

Relies on functionality from the `stats` package to return residuals from the `uRegress` object. "studentized" residuals are computed as internally studentized residuals, while "jackknife" computes the externally studentized residuals.

Value

Returns the type of residuals requested.

See Also

[regress](#), [rstudent](#), [rstandard](#)

Examples

```
# Reading in a dataset  
data(mri)  
  
# Create a uRegress object, regressing ldl on age  
ldlReg <- regress("mean", age~ldl, data=mri)  
  
# Get the studentized residuals  
residuals(ldlReg, "studentized")  
  
# Get the jackknifed residuals  
residuals(ldlReg, "jackknife")
```

rstandard.uRegress *Extract standardized residuals from uRegress objects*

Description

Extracts standardized residuals from uRegress objects by relying on functionality from the stats package.

Usage

```
## S3 method for class 'uRegress'  
rstandard(model, ...)
```

Arguments

model an object of class uRegress, as returned by [regress](#).
... other arguments to pass to residuals.uRegress

Value

a vector of standardized residuals

rstudent.uRegress *Extract Studentized residuals from uRegress objects*

Description

Extracts Studentized residuals from uRegress objects by relying on functionality from the stats package.

Usage

```
## S3 method for class 'uRegress'  
rstudent(model, ...)
```

Arguments

model an object of class uRegress, as returned by [regress](#).
... other arguments to pass to residuals.uRegress

Value

a vector of Studentized residuals

salary

Salary dataset

Description

Data from a study of 1,597 faculty members at a single US university. Includes information on monthly salary each year from 1976 through 1995, as well as sex, highest degree attained, year of highest degree, field, year hired, rank, and administrative duties. More information, including a coding key, is available at rct-design.com (Teaching Materials, Datasets section).

Usage

salary

Format

A data frame with 19792 rows and 11 variables:

case case number

id identification number for the faculty member

sex M (male) or F (female)

deg highest degree attained: PhD, Prof (professional degree, eg, medicine or law), or Other (Master's or Bachelor's degree)

yrdeg year highest degree attained

field Arts (Arts and Humanities), Prof (professional school, e.g., Business, Law, Engineering or Public Affairs), or Other

startyr year in which the faculty member was hired (2 digits)

year year (2 digits)

rank rank of the faculty member in this year: Assist (Assistant), Assoc (Associate), or Full (Full)

admin Indicator of whether the faculty member had administrative duties (eg, department chair) in this year: 1 (yes), or 0 (no)

salary monthly salary of the faculty member in this year in dollars

Source

Dataset originally distributed at rct-design.com (Teaching Materials, Datasets section).

ttest

*T-test with Improved Layout***Description**

Performs a one- or two-sample t-test using data. In the two-sample case, the user can specify whether or not observations are matched, and whether or not equal variances should be presumed.

Usage

```
ttest(
  var1,
  var2 = NA,
  by = NA,
  geom = FALSE,
  null.hypoth = 0,
  alternative = "two.sided",
  var.eq = FALSE,
  conf.level = 0.95,
  matched = FALSE,
  more.digits = 0
)
```

Arguments

var1	a (non-empty) numeric vector of data values.
var2	an optional (non-empty) numeric vector of data.
by	a variable of equal length to that of var1 with two outcomes. This will be used to define strata for a t-test on var1.
geom	a logical indicating whether the geometric mean should be calculated and displayed.
null.hypoth	a number specifying the null hypothesis for the mean (or difference in means if performing a two-sample test). Defaults to zero.
alternative	a string: one of "less", "two.sided", or "greater" specifying the form of the test. Defaults to a two-sided test.
var.eq	a logical value, either TRUE or FALSE (default), specifying whether or not equal variances should be presumed in a two-sample t-test. Also controls robust standard errors.
conf.level	confidence level of the test. Defaults to 0.95.
matched	a logical value, either TRUE or FALSE, indicating whether or not the variables of a two-sample t-test are matched. Variables must be of equal length.
more.digits	a numeric value specifying whether or not to display more or fewer digits in the output. Non-integers are automatically rounded down.

Details

Missing values must be given by NA to be recognized as missing values.

Value

a list of class `ttest`. The print method lays out the information in an easy-to-read format.

tab A formatted table of descriptive and inferential statistics (total number of observations, number of missing observations, mean, standard error of the mean estimate, standard deviation), along with a confidence interval for the mean.

df Degrees of freedom for the t-test.

p P-value for the t-test.

tstat Test statistic for the t-test.

var1 The user-supplied first data vector.

var2 The user-supplied second data vector.

by The user-supplied stratification variable.

par A vector of information about the type of test (null hypothesis, alternative hypothesis, etc.)

geo A formatted table of descriptive and inferential statistics for the geometric mean.

call The call made to the `ttest` function.

See Also

[t.test](#)

Examples

```
# Read in data set
data(psa)
attach(psa)

# Perform t-test
ttest(pretxpsa, null.hypoth = 100, alternative = "greater", more.digits = 1)

# Define new binary variable as indicator
# of whether or not bss was worst possible
bssworst <- bss
bssworst[bss == 1] <- 0
bssworst[bss == 2] <- 0
bssworst[bss == 3] <- 1

# Perform t-test allowing for unequal
# variances between strata -#
ttest(pretxpsa, by = bssworst)

# Perform matched t-test
ttest(pretxpsa, nadirpsa, matched = TRUE, conf.level = 99/100, more.digits = 1)
```

ttesti

T-test Given Summary Statistics with Improved Layout

Description

Performs a one- or two-sample t-test given summary statistics. In the two-sample case, the user can specify whether or not equal variances should be presumed.

Usage

```
ttesti(
  obs,
  mean,
  sd,
  obs2 = NA,
  mean2 = NA,
  sd2 = NA,
  null.hypoth = 0,
  conf.level = 0.95,
  alternative = "two.sided",
  var.eq = FALSE,
  more.digits = 0
)
```

Arguments

obs	number of observations for the first sample.
mean	the sample mean of the first sample.
sd	the sample standard deviation of the first sample.
obs2	number of observations for the second sample (this is optional).
mean2	if obs2 is supplied, then sample mean of the second sample must be supplied.
sd2	if obs2 is supplied, then sample standard deviation of the second sample must be supplied.
null.hypoth	a number specifying the null hypothesis for the mean (or difference in means if performing a two-sample test). Defaults to zero.
conf.level	confidence level of the test. Defaults to 0.95.
alternative	a string: one of "less", "two.sided", or "greater" specifying the form of the test. Defaults to a two-sided test.
var.eq	a logical value, either TRUE or FALSE (default), specifying whether or not equal variances should be presumed in a two-sample t-test.
more.digits	a numeric value specifying whether or not to display more or fewer digits in the output. Non-integers are automatically rounded down.

Details

If `obs2`, `mean2`, or `sd2` is specified, then all three must be specified and a two-sample t-test is run.

Value

a list of class `ttesti`. The print method lays out the information in an easy-to-read format.

tab A formatted table of descriptive and inferential statistics (number of observations, mean, standard error of the mean estimate, standard deviation), along with a confidence interval for the mean.

df Degrees of freedom for the t-test.

p P-value for the t-test.

tstat Test statistic for the t-test.

par A vector of information about the type of test (null hypothesis, alternative hypothesis, etc.)

twosamp A logical value indicating whether a two-sample test was performed.

call The call made to the `ttesti` function.

Examples

```
# t-test given sample descriptives
ttesti(24, 175, 35, null.hypoth=230)

# two-sample test
ttesti(10, -1.6, 1.5, 30, -.7, 2.1)
```

 U

Create a Partial Formula

Description

Creates a partial formula of the form `~var1 + var2`. The partial formula can be named by adding an equals sign before the tilde.

Usage

```
U(...)
```

Arguments

... partial formula of the form `~var1 + var2`.

Value

A partial formula (potentially named) for use in [regress](#).

See Also[regress](#)**Examples**

```
# Reading in a dataset
data(mri)

# Create a named partial formula
U(ma=~male+age)

# Create an unnamed partial formula

U(~male+age)
```

wilcoxon

*Wilcoxon Signed Rank and Mann-Whitney-Wilcoxon Rank Sum Test***Description**

Performs Wilcoxon signed rank test or Mann-Whitney-Wilcoxon rank sum test depending on data and logicals entered. Relies heavily on the function [wilcox.test](#). Adds formatting and variances.

Usage

```
wilcoxon(
  var1,
  var2 = NULL,
  alternative = "two.sided",
  null.hypoth = 0,
  paired = FALSE,
  exact = FALSE,
  correct = FALSE,
  conf.int = FALSE,
  conf.level = 0.95
)
```

Arguments

var1	numeric vector of data values. Non-finite (missing or infinite) values will be omitted.
var2	optional numeric vector of data values. Non-finite (missing or infinite) values will be omitted.
alternative	specifies the alternative hypothesis for the test; acceptable values are "two.sided", "greater", or "less".
null.hypoth	the value of the null hypothesis.

<code>paired</code>	logical indicating whether the data are paired or not. Default is FALSE. If TRUE, data must be the same length.
<code>exact</code>	logical value indicating whether or not an exact test should be computed.
<code>correct</code>	logical indicating whether or not a continuity correction should be used and displayed.
<code>conf.int</code>	logical indicating whether or not to calculate and display a confidence interval
<code>conf.level</code>	confidence level for the interval. Defaults to 0.95.

Details

In the one-sample case, the returned confidence interval (when `conf.int = TRUE`) is a confidence interval for the pseudo-median of the underlying distribution. In the two-sample case, the function returns a confidence interval for the median of the difference between samples from the two distributions. See [wilcox.test](#) for more information.

Value

A list of class `wilcoxon` is returned. The print method lays out the information in an easy-to-read format.

statistic the value of the test statistic with a name describing it.

parameter the parameter(s) for the exact distribution of the test statistic.

p.value the p-value for the test (calculated for the test statistic).

null.value the parameter `null.hypoth`.

alternative character string describing the alternative hypothesis.

method the type of test applied.

data.name a character string giving the names of the data.

conf.int a confidence interval for the location parameter (only present if the argument `conf.int=TRUE`).

estimate an estimate of the location parameter (only present if the argument `conf.int=TRUE`).

table a formatted table of rank sum and number of observation values, for printing.

vars a formatted table of variances, for printing.

hyps a formatted table of the hypotheses, for printing.

inf a formatted table of inference values, for printing.

See Also

[wilcox.test](#)

Examples

```
#- Create the data -#
cf <- c(1153, 1132, 1165, 1460, 1162, 1493, 1358, 1453, 1185, 1824, 1793, 1930, 2075)
healthy <- c(996, 1080, 1182, 1452, 1634, 1619, 1140, 1123, 1113, 1463, 1632, 1614, 1836)

#- Perform the test -#
```

```
wilcoxon(cf, healthy, paired=TRUE)  
  
#- Perform the test -#  
wilcoxon(cf, healthy, conf.int=TRUE)
```

Index

- * **datasets**
 - fev, 10
 - mri, 13
 - psa, 20
 - salary, 26
- * **package**
 - rigr-package, 3
- anova.uRegress, 3
- cooks.distance.uRegress, 5
- descrip, 5
- dfbeta (dfbeta.uRegress), 8
- dfbeta.uRegress, 8
- dfbetas.uRegress, 9
- dummy, 9, 21
- fev, 10
- fitted.uRegress (regress), 21
- glm, 23
- hatvalues.uRegress, 11
- lincom, 11
- lm, 23
- mri, 13
- plot.proptest (proptest), 17
- polynomial, 15, 21
- predict (predict.uRegress), 16
- predict.glm, 16
- predict.lm, 16
- predict.uRegress, 16
- print.augCoefficients (regress), 21
- print.lincom (lincom), 11
- print.proptest (proptest), 17
- print.proptesti (proptesti), 18
- print.ttest (ttest), 27
- print.ttesti (ttesti), 29
- print.uDescriptives (descrip), 5
- print.uRegress (regress), 21
- print.wilcoxon (wilcoxon), 31
- prop.test, 18
- proptest, 17
- proptesti, 18
- psa, 20
- regress, 5, 9, 11, 16, 21, 24, 25, 30, 31
- residuals (residuals.uRegress), 23
- residuals.uRegress, 23
- rigr (rigr-package), 3
- rigr-package, 3
- rstandard, 24
- rstandard.uRegress, 25
- rstudent, 24
- rstudent.uRegress, 25
- salary, 26
- t.test, 28
- ttest, 27
- ttesti, 29
- U, 21, 23, 30
- uLRtest (regress), 21
- uWaldtest (regress), 21
- wilcox.test, 31, 32
- wilcoxon, 31