Package 'mStats'

October 13, 2022

Type Package

Title Epidemiological Data Analysis

Version 3.4.0

Maintainer Myo Minn Oo <dr.myominnoo@gmail.com>

Description This is a tool for epidemiologist, medical data analyst, medical or public health professionals. It contains three domains of functions: 1) data management, 2) statistical analysis and 3) calculating epidemiological measures.

License GPL-2 | GPL-3

URL https://myominnoo.github.io/

BugReports https://github.com/myominnoo/mStats/issues

Depends R (>= 4.0.0)

Imports stats, utils

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Suggests knitr, rmarkdown

NeedsCompilation no

Author Myo Minn Oo [aut, cre] (<https://orcid.org/0000-0003-4089-016X>)

Repository CRAN

Date/Publication 2020-11-23 05:50:02 UTC

R topics documented:

append 2
codebook 3
duplicates 4
egen 5
expand2 6

expandtbl . . . 7
formatDate . . . 9
generate . . . 11
helpers . . . 12
histogram . . . 13
ilog . . . 14
label . . . 15
lag.data.frame . . . 16
logit . . . 17
mhor . . . 19
mhrr . . . 21
n_ . . . 24
recode . . . 25
regress . . . 26
replace . . . 30
scatterPlotMatrix . . . 31
strate . . . 32
summ . . . 33
summary . . . 35
tab . . . 36

Index **39**

`append` *Append datasets*

Description

`append()` row-combines multiple datasets of the same column names.

Usage

```
append(data, ...)
```

Arguments

`data`	data.frame
`...`	one or multiple data.frame

Details

A single or multiple datasets can be appended.

The appending datasets must have at least one variable name which is there in the master dataset.

The order of variables of the appending datasets is automatically set based on the variable arrangement of the master dataset.

Value

data.frame

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

Examples

```
x <- append(infert[, -c(3,4)], infert[, -5], infert[, -6])
## codebook(x)

## Not run:
## if no variables are matched, ERROR
append(infert, iris)

## End(Not run)
```

---

codebook *Describe the data*

Description

`codebook()` examines the variable names, labels, and data to produce a codebook for describing the dataset

Usage

```
codebook(data)
```

Arguments

`data`	data.frame

Details

It reports a description of the data with the following information.

ANNOTATIONS:

`No` = serial number

`Variable` = variable name

`Label` = variable label

`Type` = type of variable

`Obs` = number of valid observations

`NA` = number of observations with missing value `NA`

Value

a data.frame containing the codebook

Note

For haven_labelled data.frame, data types are generated using typeof().

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

Examples

```
codebook(infert)
codebook(iris)
codebook(mtcars)
```

---

duplicates *Report, tag or drop the duplicate observations*

Description

duplicates() generates a table showing the duplicate Observations as one or more copies as well as their Surplus indicating the second, third, ... copy of the first of each group of duplicates.

Usage

```
duplicates(data, ..., drop = FALSE)
```

Arguments

data	data.frame
...	variables to find the duplicate observations
drop	TRUE deletes all the duplicate observations.

Details

If no variable is specified in ..., all variables are used to find the duplicate observations.

If drop is set to TRUE, all occurrences of each group of observations except the first are deleted from the dataset.

Value

data.frame with a column dup_num, indicating the number of duplicate observations of each group of observations

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

Examples

```
x <- duplicates(iris, Species)
x <- duplicates(iris)
```

---

egen *Categorize a numerical variable*

Description

`egen()` transforms a numeric vector to a factor vector.

Usage

```
egen(data, var, cut = NULL, lbl = NULL, new_var = NULL)
```

Arguments

`data`	data.frame
`var`	existing variable
`cut`	either a number or a numeric vector
`lbl`	labels to specify
`new_var`	name of new variable to be created

Details

`egen` allows easy conversion of a numerical variable to a categorical variable.

If only a number is specified in `cut`, it categorizes into equal intervals based on that number. If no value is set for `cut`, the default interval is `10`.

Automatic naming new variable

If `new_var` is not specified, new names will be automatically created by appending `_cat` as suffix. `VARNAME`_cat``

Automatic Labelling

If `lbl` is not specified, labels are constructed in `` `##-##` ``.

Value

data.frame

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

Examples

```
x <- egen(infert, age)
tab(x, age_cat)

## Not run:
## Set cut-off points
x <- egen(infert, age, c(26, 31, 36, 41))
tab(x, age_cat)

## Add labels and give a new name
x <- egen(infert, age, c(26, 31, 36, 41),
          lbl = c("<= 25", "26 - 30", "31 - 35",
            "36 - 40", "41+"),
          new_var = age_grp)
tab(x, age_grp)

## End(Not run)
```

---

expand2 *Duplicate observations within a dataframe*

Description

expand2 generates duplicated observations within a dataframe.

Usage

```
expand2(data, n_n = NULL, copies = 2, original = TRUE)
```

Arguments

data	a data frame object
n_n	index or indexes specifying row numbers
copies	desired number of copies
original	a logical indicating whether to keep the original dataframe

Details

expand2 appends observations from the dataframe with n copies of the observations with specified indexes of observations or all data.

Value

data.frame

Author(s)

Email: <dr.myominnoo@gmail.com>
Website: https://myominnoo.github.io/

Examples

```
## create duplicates
x <- expand2(infert, 1:5, copies = 2)

## check duplicates report and rmeove dup
duplicates(x, drop = TRUE)
```

---

expandtbl *Expand* 2x2 table *into* data.frame

Description

`expandtbl()` generates a data.frame based on vectors.

Usage

```
expandtbl(
  ...,
  exp_name = "exp",
  exp_lvl = c("exposed", "unexposed"),
  case_name = "case",
  case_lvl = c("case", "control"),
  strata_name = "strata"
)

expandfreq(data, freq)
```

Arguments

`...`	vectors
`exp_name`	Name of `exp` Variable
`exp_lvl`	Names of two categories in the order of Exposed and non-exposed
`case_name`	Name of `Case` variable
`case_lvl`	names of two categories in the order of
`strata_name`	Name of stratified variable
`data`	frequency table in data.frame
`freq`	name of variable for the weighted frequency

Details

expandtbl

uses the vectors of 2x2 tables and generates a data frame of at least two columns: exp and case.

```
expandtbl(c(100, 200, 100, 200))
```

`Strata`

Multiple tables can be used to construct a dataset by specifying `strata_name` as follow. Strata can be included using multiple named vectors.

```
expandtbl(
             strata1 = c(100, 200, 100, 200),
             strata2 = c(100, 200, 100, 200),
             strata3 = c(100, 200, 100, 200),
             exp_name = "exp",
             exp_lvl = c("exposed", "unexposed"),
             case_name = "case",
             case_lvl = c("case", "control"),
             strata_name = "Strata"
)
```

`Labels for variables`

If names or levels of variables are not specified, the followings are applied.

1. exp Name: `exp`
2. exp levels: `exposed` and `unexposed`
3. case Name: `case`
4. case levels: `case` and `control`
5. Strata Name: `strata`
6. Note: Strata levels are not considered as vectors must be named.

expandfreq() uses the weighted frequencies in data.frame format and construct another data.frame based on the frequency weight. The name of the frequency weighted variable can be specified by `freq` argument.

Value

data.frame

Functions

- `expandfreq`: `expandfreq()` expands a frequency-weighted table into a data.frame.

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

Examples

```
## Asthma Example from Essential Medical Statistics
## page 160
asthma <- expandtbl(c(81, 995, 57, 867),
              exp_name = "sex",
              exp_lvl = c("woman", "man"),
              case_name = "asthma",
              case_lvl = c("yes", "no"))

## Not run:
## label variable and dataset
asthma <- label(asthma, "Hypothetical Data of Asthma Prevalence")
asthma <- label(asthma, sex = "Man or Woman",
                           asthma = "Asthma or No Asthma")

## Checking codebook
codebook(asthma)


## simple tabulation
tab(asthma)

## cross-tabulation
tab(asthma, sex, by = asthma)

## End(Not run)


## Example for expanding frequency weighted data

## Example from UCLA website
## you can download the dataset here:
## https://stats.idre.ucla.edu/stat/stata/examples/icda/afterlife.dta

x <- data.frame(gender = c(1, 1, 0, 0),
                aftlife = c(1, 0, 1, 0),
                freq = c(435, 147, 375, 134))
y <- expandfreq(x, freq)

## check the numbers by tabulation
## tab(y, gender, by = aftlife)
```

`formatDate` *Format Dates*

Description

`formatDate` converts characters or numbers to dates. `is.Date` indicates which elements are Dates.

Usage

```
formatDate(x, format = "dmY", sep = "/", century = NULL)

is.Date(x)

year(x)

month(x)

day(x)
```

Arguments

x	a character or numeric object
format	only for character vectors:
sep	separator character for date components
century	specify either 2000 or 1900 for two-digit years

Details

dmy represents dd mm YYYY format. In combination with separators from sep, this can change to several date formats. For example, dmy + - convert to dd-mm-yyyy format.

Possible conversions

1. dmy + - »> dd-mm-yyyy
2. dmy + / »> dd/mm/yyyy
3. mdy + / »> mm/dd/yyyy
4. ymd + / »> yyyy/mm/dd
5. dby + - »> dd-JAN-yy
6. dby + / »> dd/JAN/yy

Numeric conversions Origin is set at 1899-12-30.

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

Examples

```
## convert strings to dates
x <- c("2019-01-15", "2019-01-20", "2019-01-21", "2019-01-22")

# check if it is a Date format
is.Date(x)
```

```
## Not run:
y <- formatDate(x, "Ymd", "-")

# check if it is a Date format
is.Date(y)
y


## another format
x <- c("22-JAN-19", "24-MAR-20")
y <- formatDate(x, "dby", "-")
is.Date(y)
y


## convert numbers to dates
x <- 42705:42710
y <- formatDate(x)
is.Date(y)
y


## get day, month or year
day(y)
month(y)
year(y)

## End(Not run)
```

`generate` *Create a new variable*

Description

`generate()` creates a new variable either by deriving from existing variables or with a constant value.

Usage

```
generate(data, var, expr = NULL)
```

Arguments

`data`	data.frame
`var`	name for the new variable
`expr`	a constant value, name of an existing variable or an expression for simple arithmetic or logical operations:

Details

The values of the variable are specified by `expr`.

Label

The newly created variable is automatically labeled with the expression specified.

Value

data.frame with the new variable

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

Examples

```
## generate variable with a constant value
generate(mtcars, new_var, NA)
generate(mtcars, new_var, 99)

## generate variable from an existing variable
generate(mtcars, new_var, mpg)

## generate variable with arithmetic operations
generate(iris, Length, Sepal.Length + Petal.Length)
```

`helpers` *Helper functions*

Description

These are helper functions for `mStats`.

Usage

```
helpers(...)

clear()
```

Arguments

`...`	further arguments to be passed to or from methods

`histogram` *Histograms with overlay normal curve*

Description

`histogram()` draws a histogram with formatted texts and adds a normal curve over the histogram.

Usage

```
histogram(
  data,
  var,
  breaks = NULL,
  xlab = NULL,
  main = NULL,
  sub = NULL,
  labels = TRUE,
  freq = TRUE,
  curve = TRUE,
  ...
)
```

Arguments

`data`	Dataset
`var`	variable
`breaks`	hist
`xlab`	hist
`main`	hist
`sub`	hist
`labels`	hist
`freq`	hist
`curve`	logical. If `TRUE` (default), a normal curve is overlaid over the histogram.
`...`	hist

Details

If `freq` is set to `FALSE`, probability densities, component density, are plotted (so that the histogram has a total area of one). In this case, normal curve will not be generated.

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

Examples

```
# histogram(infert, age)
# histogram(infert, age, labels = FALSE)
# histogram(infert, age, freq = FALSE)
```

---

ilog *Create a copy of your output in a text format*

Description

`ilog()` creates a text copy of your output. `ilog.close()` closes the `ilog()` function. `ilog.clear()` clears for the prompt error caused when the environment is removed.

Usage

```
ilog(logfile = "log.txt", append = FALSE)

ilog.close()

ilog.clear()
```

Arguments

`logfile`	Name of desired log file in `.txt` format
`append`	logical value

Details

`ilog` is a two-step function that allows you a record of your console. A log is a file containing what you type and console output. If a name is not specified, then `ilog` will use the name `<unnamed>.txt`.

`ilog` opens a log file and `ilog.close` close the file.

Warnings:

However, clearing objects from the workspace along with hidden objects removes `ilog`'s `.logenv` environment, hence throwing an error when it's attempted to be closed. An error message `Error in (function (cmd, res, s, vis) : object '.logenv' not found` will be thrown.

In that case, console prompt is stuck at `log>` . If this error occurs, use `ilog.clear()` function to revert back to normal.

Author(s)

Email: `<dr.myominnoo@gmail.com>`

Website: https://myominnoo.github.io/

Examples

```
## Not run:
## my first log
ilog("../myFirstLog.tx")
str(infert)
str(iris)
ilog.close()

## in case of error: ".logenv" not found
# ilog.clear()

## End(Not run)
```

`label` *Attach labels to data and variables*

Description

`label()` manipulates labels

Usage

```
label(data, ...)
```

Arguments

`data`	data.frame
`...`	For variable label, `Var = "Var Label"`: For data label, `"Example data lable"`.

Details

Attach labels

It has two inputs. If only one label is specified, that label is attached to the data. Otherwise, the pattern `Var = "Var Label"` are used to attach labels to variables.

Remove labels

`NA` or `NULL` is used to remove labels.

Value

data.frame

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

Examples

```
## Variable label
x <- label(infert,
           education = "Education levels",
           age = "Age in years of case",
           parity = "count",
           stratum = "1-83",
           pooled.stratum = "1-63")

## Data label
x <- label(x, "Infertility and Abortion Dataset")
codebook(x)
```

`lag.data.frame` *Lag a variable*

Description

creates lagged version of an existing variable.

Usage

```
## S3 method for class 'data.frame'
lag(x, var, by = NULL, new_var = NULL, last_obs = FALSE, ...)
```

Arguments

`x`	data.frame
`var`	variable to be lagged
`by`	variable for grouped lagged version
`new_var`	name of new lagged variable
`last_obs`	`TRUE`retrieves the last observation per group.
`...`	further arguments to be passed to or from methods.

Details

This is often encountered in time-related analysis. In a lagged variable, values from earlier points in time are placed in later rows of dataset.

Value

data.frame

Note

Before using `lagRows`, the dataset needs to be sorted by a id variable or similar variable.

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

Examples

```
set.seed(100)
## create a dataset with dates
x <- data.frame(
    hospid = 1:100,
    docid = round(runif(100, 1, 10)),
    dis_date = formatDate(runif(100, 42700, 42800))
)

## lagged dis_date, not specifed "by"
lag(x, dis_date)

## Not run:
## lagged dis_date by docid
## first we need to sort
y <- x[order(x$docid), ]
y

## lag dates within groups
lag(y, dis_date, by = docid, new_var = lag_date)
lag(y, dis_date, by = docid, lag_date, TRUE)

## End(Not run)
```

---

logit *Logistic Regression Model*

Description

`logit()` produces summary of the model with coefficients or odds ratios (`OR`) and 95% Confident Intervals.

Usage

```
logit(model, or = TRUE, digits = 5)
```

Arguments

`model`	glm or lm model
`or`	`TRUE`reports odds ratios instead of coefficients
`digits`	specify rounding of numbers. See `round`.

Details

`logit()` is based on `glm` with `binomial` family. All statistics presented in the function's output are derivatives of `glm`, except AIC value which is obtained from `AIC`.

Outputs

Outputs can be divided into three parts.

1. `Info of the model`: Here provides number of observations (Obs.), chi value from Likelihood Ratio test (LR chi2) and its degree of freedom, p-value from LR test, Pseudo R Squared, log likelihood and AIC values.
2. `Regression Output`: Coefficients from summary of model are tabulated here along with 95\ confidence interval.

Value

a list containing

1. `info` - info and error tables
2. `reg` - regression table
3. `model` - raw model output from `lm()`
4. `fit` - formula for fitting the model
5. `lbl` - variable labels for further processing in `summary`.

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

Examples

```
mylogit <- glm(case ~ education + age + parity, family = binomial,
           data = infert)
logit(mylogit)

## Not run:
## Example from UCLA website:
## LOGIT REGRESSION | R DATA ANALYSIS EXAMPLES
## https://stats.idre.ucla.edu/r/dae/logit-regression/

mydata <- read.csv("https://stats.idre.ucla.edu/stat/data/binary.csv")
mydata <- replace(mydata, rank, factor(rank))
mydata <- label(mydata, gre = "GRE", gpa = "GPA score", rank = "Ranking")
mylogit <- glm(admit ~ gre + gpa + rank, data = mydata, family = "binomial")

## Showing Odds Ratios
logit(mylogit)

## Showing coefficients
logit(mylogit, or = FALSE)
```

```
## End(Not run)
```

---

mhor *Calculating Odds Ratios*

Description

`mhor()` calculates odds ratios, Mantel Haenszel pooled estimates and 95% CI.

Usage

```
mhor(
  data,
  exp,
  case,
  strata = NULL,
  exp_value = NULL,
  case_value = NULL,
  digits = 4
)
```

Arguments

`data`	data.frame
`exp`	exposure or independent variables
`case`	case or dependent variables (outcomes)
`strata`	if specified, MH OR is calculated.
`exp_value`	value for exposure as reference
`case_value`	value for outcome as reference
`digits`	specify rounding of numbers. See `round`.

Details

Rows and Columns can be rearranged by specifying `exp_value` and `case_value`. This is used when the exposed and case values are not at the right place in 2x2 tables.

Reference row value can be specified in `exp_value`.

Attributable fractions, `Attr. Frac. Exp` and `Attr. Frac. Pop` among exposed and population are calculated when OR is greater than or equal to 1. If OR is less than 1, preventable fractions, `Prev. Frac. Exp` and `Attr. Frac. Pop` are calculated.

It produces a table with Odds Ratio, 95% CI as well as p-value. If `strata` is specified, `Mantel-Haenzsel` Pooled estimates of `Odds Ratio` is generated along with Chi-squared test for homogeneity.

Odds Ratio, OR

$$OR = (D1xH0)/(D0xH1)$$

Error Factor, EF using Woolf's formula

$$95\%CI = OR/EForORxEF$$

$$EF = exp(1.96xSE(log(OR)))$$

$$SE(log(OR)) = \sqrt{1/D1 + 1/H1 + 1/D0 + 1/H0}$$

Calculating p-value from Wald's z test

$$z = logOR/SE(logOR)$$

Mantel-Haenszel's OR

$$ORMH = Q/R$$

$$Q = \sum (D1ixH0i)/ni$$

$$R = \sum (D0ixH1i)/ni$$

Calculating CI for MH-OR

$$95\%CI = OR/EForORxEF$$

$$SE(ORMH) = \sqrt{V/(QxR)}$$

$$V = \sum (DixHixn0ixn1i)/((ni)^2x(ni-1))$$

Chi-square test for MHOR, df = 1

$$X^2(MH), Chi - squarevalue = U^2/V$$

$$U = O - E$$

$$O = \sum D1i$$

$$E = \sum Dixn1i/ni$$

Chi-square test for Heterogeneity

$$X^2 = \sum (D1ixH0i - ORMHxD0ixH1i)^2/ORMHxVixni^2$$

Value

data.frame

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

References

1. Betty R. Kirkwood, Jonathan A.C. Sterne (2006, ISBN:978–0–86542–871–3)
2. B. Burt Gerstman (2013, ISBN:978-1-4443-3608-5)
3. Douglas G Altman (2005, ISBN:0 7279 1375 1)

Examples

```
### Example from Essential Medical Statistics
# Page 178, Chapter 18: Controlling for confounding: Stratification
lepto <- expandtbl(
    male = c(36, 14, 50, 50), female = c(24, 126, 10, 90),
    exp_name = "area", exp_lvl = c("Rural", "Urban"),
    case_name = "ab", case_lvl = c("Yes", "No"),
    strata_name = "gender"
)

## label variables and data
lepto <- label(lepto, "Prevalence survey of leptospirosis in West Indies")
lepto <- label(lepto, area="Type of area", ab = "Leptospirosis Antibodies",
                  gender="Gener: Male or Female")

## Calculate OR
mhor(lepto, area, ab)

## Calculate MHOR
mhor(lepto, area, ab, gender)
```

---

mhrr *Calculating Risk Ratios*

Description

`mhrr()` calculates different measures of risk including risk ratios (RR) as well as Mantel-Haenszel pooled estimates.

Usage

```
mhrr(
  data,
  exp,
  case,
  strata = NULL,
  exp_value = NULL,
  case_value = NULL,
  digits = 4
)
```

Arguments

data	data.frame
exp	exposure or independent variables
case	case or dependent variables (outcomes)
strata	if specified, MH OR is calculated.
exp_value	value for exposure as reference
case_value	value for outcome as reference
digits	specify rounding of numbers. See round.

Details

Rows and Columns can be rearranged by specifying exp_value and case_value. This is used when the exposed and case values are not at the right place in 2x2 tables.

Reference row value can be specified in exp_value.

Attributable fractions, Attr. Frac. Exp and Attr. Frac. Pop among exposed and population are calculated when RR is greated than or equal to 1. If RR is less than 1, preventable fractions, Prev. Frac. Exp and Attr. Frac. Pop are calculated.

It produces a table with Risk Ratio, 95% CI as well as p-value. If strata is specified, Mantel-Haenzsel Pooled estimates of Risk Ratio is generated along with Chi-squared test for homogeneity.

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

References

1. Betty R. Kirkwood, Jonathan A.C. Sterne (2006, ISBN:978–0–86542–871–3)
2. B. Burt Gerstman (2013, ISBN:978-1-4443-3608-5)
3. Douglas G Altman (2005, ISBN:0 7279 1375 1)

Examples

```
### Example from Essential Medical Statistics
# Page 178, Chapter 18: Controlling for confounding: Stratification
lepto <- expandtbl(
    male = c(36, 14, 50, 50), female = c(24, 126, 10, 90),
    exp_name = "area", exp_lvl = c("Rural", "Urban"),
    case_name = "ab", case_lvl = c("Yes", "No"),
    strata_name = "gender"
)

## label variables and data
lepto <- label(lepto, "Prevalence survey of leptospirosis in West Indies")
lepto <- label(lepto, area="Type of area", ab = "Leptospirosis Antibodies",
                  gender="Gener: Male or Female")

## Calculate RR
mhrr(lepto, area, ab)

## Calculate MHRR
mhrr(lepto, area, ab, gender)


## Not run:
### Demonstration: Calculating Risk Ratios

## Essential Medical Statistics, Betty R. Kirkwood, Second Edition
## Chapter 16, Table 16.4, Page 154
## For Risk Ratio
lung <- expandtbl(
    c(39, 29961, 6, 59994),
    exp_name = "smoking",
    exp_lvl = c("Smokers", "Non-smokers"),
    case_name = "cancer",
    case_lvl = c("Yes", "No")
)

## label variable and dataset
lung <- labelVar(lung, smoking="Yes or No", cancer="Yes or no")
lung <- labelData(lung, "Follow up lung cancer study")

## check dataset
codebook(lung)

## calculate RR
mhrr(lung, smoking, cancer, exp_value = "Smokers", case_value = "Yes")



## Simpson's paradox
## Burt Gerstman's Epidemiology, page 326, table 14.1
```

```
simpson <- expandtbl("1" = c(1000, 9000, 50, 950),
                     "2" = c(95, 5, 5000, 5000),
                     exp_name = "trt",
                     exp_lvl = c("new", "standard"),
                     case_name = "case",
                     case_lvl = c("alive", "dead"),
                     strata_name = "clinic")

## calculate RR
mhrr(simpson, trt, case, exp_value = "new", case_value = "alive")

## calculate MH RR
mhrr(simpson, trt, case, clinic)


## End(Not run)
```

---

n_ *Count from* n_ *to* N_

Description

n_() generates the current observation number per specified group. It is regarded as grouped serial numbers.

N_() generates total number of observation per group. It is regarded as grouped total number.

Usage

```
n_(data, ...)

N_(data, ...)
```

Arguments

data	data.farme
...	variables for grouping

Details

If no variable is set in ..., all variables in the datset is used for grouping.

Value

data.frame

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

Examples

```
x <- n_(iris, Species)
## Not run:
x
codebook(x)

x <- N_(iris, Species)
x
codebook(x)

## End(Not run)
```

---

recode *Recode a variable*

Description

`recode()` changes the values of a variable.

Usage

```
recode(data, var, ...)
```

Arguments

`data`	data.frame
`var`	variable name
`...`	specify in pattern: `old value` / `new value`.

Details

It changes the values of a variable according to the old values specified. Values that does not meet any of the conditions, they are left unchanged.

Using colon : to indicate a range of numeric numbers

A numeric vector can be indicated by using `:` in `old value`. The function automatically filters the values that meet the range and assigns a specified new value to these.

Value

a data.frame

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

Examples

```
x <- recode(infert, case, 0/"No", 1/"Yes")
tab(x, case)

## Not run:
## recode a factor
x <- recode(infert, education, "0-5yrs"/1, "6-11yrs"/2, "12+ yrs"/3)
tab(x, education)

## recode numeric vectors
x <- recode(infert, age, 21:28.9/1, 29:34.9/2, 35:44/3)
tab(x, age)

## recode NA
infert[4:20, "case"] <- NA
x <- recode(infert, case, NA/"Missing value")
tab(infert, case)

## End(Not run)
```

---

regress *Linear Regression Model*

Description

`regress()` produces summary of the model with coefficients and 95% Confident Intervals.

`` `predict.regress` `` a S3 method for `predict` to generate statistics related to the prediction of the linear model using the output from the `regress` function of the `mStats`.

`` `plot.regress` `` is a S3 method for `plot()` to create graphs for checking diagnostics of linear model using the output from the `regress` function of the `mStats`.

`` `ladder` `` converts a variable into a normally distributed one.

`` `hettest` `` performs the Breusch-Pagan test for heteroskedasticity. It presents evidence against the null hypothesis that t=0 in Var(e)=sigma^2 exp(zt). The formula are based on the `bptest` function in `lmtest` package.

`` `linkTest` `` determines whether a model in R is 'well specified' using the `STATA`'s `linkTest`.

Usage

```
regress(model, vce = FALSE, digits = 5)

## S3 method for class 'regress'
predict(object, ...)

## S3 method for class 'regress'
plot(x, ...)

ladder(data, var)

hettest(regress, studentize = FALSE)

linkTest(model, vce = FALSE, digits = 5)
```

Arguments

model	glm or lm model
vce	if TRUE, robust standard errors are calculated.
digits	specify rounding of numbers. See round.
object	a model object for which prediction is desired.
...	additional arguments affecting the predictions produced.
x	the coordinates of points in the plot. Alternatively, a single plotting structure, function or *any* R *object with a* plot *method* can be provided.
data	dataset
var	variable name
regress	output from regress
studentize	logical. If set to TRUE Koenker's studentized version of the test statistic will be used.

Details

regress is based on lm. All statistics presented in the function's output are derivatives of lm, except AIC value which is obtained from AIC. It uses lm() function to run the model.

Outputs

Outputs can be divided into three parts.

1. Info of the model: Here provides number of observations (Obs.), F value, p-value from F test, R Squared value, Adjusted R Squared value, square root of mean square error (Root MSE) and AIC value.
2. Errors: Outputs from anova(model) is tabulated here. SS, DF and MS indicate sum of square of errors, degree of freedom and mean of square of errors.
3. Regression Output: Coefficients from summary of model are tabulated here along with 95\ confidence interval.

using Robust Standard Errors

if heteroskedasticity is present in our data sample, the ordinary least square (OLS) estimator will remain unbiased and consistent, but not efficient. The estimated OLS standard errors will be biased and cannot be solved with a larger sample size. To remedy this, robust standard errors can be used to adjusted standard errors.

The `regress` uses sandwich estimator to estimate Huber-White's standard errors. The calculation is based on the tutorial by Kevin Goulding.

$$VarianceofRobust = (N/N-K)(X'X)^(-1) \sum XiX'iei^2(X'X)^(-1)$$

where N = number of observations, and K = the number of regressors (including the intercept). This returns a Variance-covariance (VCV) matrix where the diagonal elements are the estimated heteroskedasticity-robust coefficient variances — the ones of interest. Estimated coefficient standard errors are the square root of these diagonal elements.

`` `predict.regress` `` generates an original data with statistics for model diagnostics:

1. `fitted` (Fitted values)
2. `resid` (Residuals)
3. `std.resid` (Studentized Residuals)
4. `hat` (leverage)
5. `sigma`
6. `cooksd` (Cook's Distance)

The `Breusch-Pagan test` fits a linear regression model to the residuals of a linear regression model (by default the same explanatory variables are taken as in the main regression model) and rejects if too much of the variance is explained by the additional explanatory variables. Under H_0 the test statistic of the Breusch-Pagan test follows a chi-squared distribution with `parameter` (the number of regressors without the constant in the model) degrees of freedom.

The code for `` `linkTest` `` has been modified from Keith Chamberlain's linktext. www.ChamberlainStatistics.com https://gist.github.com/KeithChamberlain/8d9da515e73a27393effa3c9fe571c3f

Value

a list containing

1. `info` - info and error tables
2. `reg` - regression table
3. `model` - raw model output from `lm()`
4. `fit` - formula for fitting the model
5. `lbl` - variable labels for further processing in `summary`.

Note

Credits to Kevin Goulding, The Tarzan Blog.

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

References

T.S. Breusch & A.R. Pagan (1979), A Simple Test for Heteroscedasticity and Random Coefficient Variation. *Econometrica* **47**, 1287–1294

R. Koenker (1981), A Note on Studentizing a Test for Heteroscedasticity. *Journal of Econometrics* **17**, 107–112.

W. Krämer & H. Sonnberger (1986), *The Linear Regression Model under Test*. Heidelberg: Physics

Examples

```
fit <- lm(Ozone ~ Wind, data = airquality)
regress(fit)

## Not run:
## labelling variables
airquality2 <- label(airquality, Ozone = "Ozone level", Wind = "Wind Speed")
fit2 <- lm(Ozone ~ Wind, data = airquality2)
reg <- regress(fit2)
str(reg)

## End(Not run)


## Not run:
predict(reg)

## End(Not run)


## Not run:
plot(reg)

## End(Not run)


ladder(airquality, Ozone)


## Not run:
hettest(reg)

## End(Not run)


## Not run:
```

```
linkTest(fit)

## End(Not run)
```

`replace` *Change contents of an existing variable*

Description

`replace()` alters the contents of a variable when specified conditions are met.

Usage

```
replace(data, var, value, ...)
```

Arguments

`data`	data.frame
`var`	variable
`value`	value for replacement
`...`	`if` conditions or expressions

Details

If only `value` is specified, the whole variable is assigned with the `value`. Multiple conditions can be specified within the three dots.

Value

data.frame

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

Examples

```
x <- replace(infert, case, 2, case == 0)
tab(x, case)

x <- replace(infert, parity, 4, parity > 4)
tab(x, parity)

## Not run:
## More examples
```

```
## replacing mpg with standardized values of mpg
replace(mtcars, mpg, mpg / mean(mpg))

## replacing mpg with NA if < 10 or > 20
replace(mtcars, mpg, NA, mpg < 10 | mpg > 20)

## replacing education levels with one value
replace(infert, education, "6+yrs",
        education == "6-11yrs" | education == "12+ yrs")

## replacing mpg with NA if mpg is from 10 and 20.
replace(mtcars, mpg, NA, mpg >= 10, mpg <= 20)

## End(Not run)
```

`scatterPlotMatrix` *Scatter plot matrices with histogram and correlation coefficients*

Description

A matrix of scatter plots is produced with Scatter plots with smooth regression line in lower panel, histograms in diagonal panel and Pearson's correlation coefficients in upper panel.

Usage

```
scatterPlotMatrix(data, main = NULL, pch = 21, ...)
```

Arguments

`data`	data.frame.
`main`	The main title (on top) using font, size (character expansion) and color `par(c("font.main", "cex.main", "col.main"))`.
`pch`	numeric: point symbol
`...`	further arguments to be passed to or from methods

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

Examples

```
## iris data
# scatterPlotMatrix(iris)
```

`strate` *Calculate Incidence Rates from time-to-event data*

Description

`strate()` calculates incidence rates and Corresponding 95\

Usage

```
strate(data, time, var, ..., fail = NULL, per = 1, digits = 5)
```

Arguments

`data`	Dataset
`time`	person-time variable
`var`	outcome variable: preferably 1 for event, 0 for censored
`...`	variables for stratified analysis
`fail`	a value or values to specify failure event
`per`	units to be used in reported rates
`digits`	Rounding of numbers

Details

Rates of event occurrences, known as incidence rates are outcome measures in longitudinal studies. In most longitudinal studies, follow-up times vary due to logistic reasons, different periods of recruitment, delay enrollment into the study, lost-to-follow-up, immigration or emigration and death.

Follow-up time in longitudinal studies

Period of observation (called as follow-up time) starts when individuals join the study and ends when they either have an outcome of interest, are lost-to- follow-up or the follow-up period ends, whichever happens first. This period is called **person-year-at-risk**. This is denoted by *PY* in `strate` function's output and number of event by *D*.

Rate

is calculated using the following formula:

$$\lambda = D/PY$$

Confidence interval of rate

is derived using the following formula:

$$95\%CI(rate) = ratexErrorFactor$$

$$ErrorFactor(rate) = exp(1.96/\sqrt{D})$$

`plot`, if `TRUE`, produces a graph of the rates against the numerical code used for categories of `by`.

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

References

Betty R. Kirkwood, Jonathan A.C. Sterne (2006, ISBN:978–0–86542–871–3)

Examples

```
## Not run:

## Using the diet data (Clayton and Hills 1993) described in STATA manual
import diet data: require haven package to read dta format.
magrittr package for piping operation
diet <- haven::read_dta("https://www.stata-press.com/data/r16/diet.dta")

diet <- generate(diet, time, (dox - doe) / 365.25)
diet <- replace(diet, time, as.numeric(time))
diet <- generate(diet, age, as.numeric(doe - dob) / 365.25)
diet <- egen(diet, age, c(41, 51, 61, 71), new_var = ageband)
diet <- egen(diet, month, c(3, 6, 8), new_var = monthgrp)

## calculate overall rates and 95% Confidence intervals
strate(diet, time, fail, fail = c(1, 3, 13))

## per 100 unit
strate(diet, time, fail, fail = c(1, 3, 13), per = 100)

## calculate Stratified rates and 95% Confidence Intervals
strate(diet, time, fail, job, fail = c(1, 3, 13))
strate(diet, time, fail, job, ageband, monthgrp, fail = c(1, 3, 13))

## per 100 unit
strate(diet, time, fail, job, ageband, monthgrp, fail = c(1, 3, 13), per = 100)

## End(Not run)
```

---

summ *Summary statistics*

Description

summ() calculates and displays a variety of summary statistics. If no variables are specified, summary statistics are calculated for all the variables in the dataset.

Usage

```
summ(data, ..., by = NULL, na.rm = FALSE, digits = 1, detail = FALSE)
```

Arguments

data	data.frame
...	variable name or names of multiple variables
by	variable name for bivariate analysis
na.rm	logical: if TRUE, it removes observations with missing values.
digits	specify rounding of numbers.
detail	logical: if TRUE, it displays a full spectrum of summary statistics such as inter-quartile range, and p-value from normality test.

Details

It calculates seven number summary statistics, and p-values from relevant statistical tests of association.

ANNOTATIONS

Obs = Number of observations

NA = Number of observations with missing value

Mean = Mean

Std.Dev = Standard deviation

Median = Median value

25% = First quartile or percentile

75% = Third quartile or percentile

Min = Minimum value

Max = Maximum value

Normal = p-value from Shapiro-Wilk Normality Test

Grouped summary statistics

If a strata variable by is specified, grouped summary statistics are calculated. In addition, based on the levels of by, relevant statistical tests of association such as Student's t-test and Wilcoxon, ANOVA and Kruskal-Wallis tests are calculated and their associated p-values are displayed.

Tabulating the whole dataset

This is helpful when the dataset has been processed and finalized. The final dataset can be fed into the function without inputting any variables. This automatically filters and generates tables on variables with possible data types for summary statistics. These data types include numeric, double, integer, and logical.

Using colon : to summarize multiple variables

A colon separator : can be used to summarize variables more efficiently.

Labels

Labels for corresponding variables are displayed below the table.

Value

A list with summ class containing three sets of data.frame type:

1. summary result,
2. summary result without any format,
3. labels for corresponding variables.

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

Examples

```
## Univariate summary statistics
summ(iris, Sepal.Length)
summ(iris, Sepal.Length:Petal.Width)

## Bivariate summary statistics
summ(iris, Sepal.Length:Petal.Width, by = Species)

## Not run:
## Using the whole dataset
summ(iris)
summ(iris, by = Species)

## Detailed summary statistics
summ(iris, detail = TRUE)
summ(iris, by = Species, detail = TRUE)

## End(Not run)
```

---

summary *Table Format for Publication*

Description

summary() organizes the output and print a favorable format to the console, which is used with rmarkdown package to produce publication-ready tables.

Usage

```
## S3 method for class 'tab'
summary(object, ...)

## S3 method for class 'summ'
summary(object, ...)
```

Arguments

object	an object for which a summary is desired.
...	additional arguments affecting the summary produced.

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

Examples

```
## Not run:
## Summary for tabulation
x <- tab(infert, education, parity:spontaneous)
summary(x)

x <- tab(infert, education, parity:spontaneous, by = case)
summary(x)

## Summary for summary statistics
x <- summ(iris)
summary(x)

x <- summ(iris, by = Species)
summary(x)

x <- summ(iris, by = Species, detail = TRUE)
summary(x)

## End(Not run)
```

---

tab *Tabulation*

Description

tab() generates one-way or two-way tabulation of variables. If no variables are specified, tabulations for all the variables in the dataset are generated.

Usage

```
tab(data, ..., by = NULL, row.pct = TRUE, na.rm = FALSE, digits = 1)
```

Arguments

data	data.frame
...	variable name or names of multiple variables
by	variable name for bivariate analysis
row.pct	TRUE, FALSE or NULL.
na.rm	logical: if TRUE, it removes observations with missing values.
digits	specify rounding of numbers.

Details

One-way tabulation

If by is not specified, tab generates one-way tabulation of a variable or multiple variables. The table is displayed in Freq. (frequency), Percent (Relative Frequency) and Cum. (Cumulative Relative frequency).

Two-way tabulation

Specifying by leads to two-way tabulation. By default, row percentages are displayed along with count data. If row.pct is set to NULL, it shows a count table without percentages. If set to FALSE, a table with column percentages is generated. P-values from Chi-squared and Fisher's Exact tests are also shown, regardless of displaying percentages.

Tabulating the whole dataset

This is helpful when the dataset has been processed and finalized. The final dataset can be fed into the function without inputting any variables. This automatically filters and generates tables on variables with possible data types for tabulation. These data types include character, factor, order factor, and logical.

Using colon : to tabulate multiple variables

A colon separator : can be used to generate one-way or two-way tables efficiently.

Labels

Labels for corresponding variables are displayed below the table.

Value

A list with tab class containing three sets of data.frame type:

1. tabulation result,
2. tabulation result without any format,
3. labels for corresponding variables.

Author(s)

Email: <dr.myominnoo@gmail.com>

Website: https://myominnoo.github.io/

Examples

```
## One-way tabulation
tab(infert, education)
tab(infert, education, parity:spontaneous)
tab(infert)

## Two-way tabulation
tab(infert, education, by = case)
tab(infert, education, parity:spontaneous, by = case)
tab(infert, by = case)
```

Index

AIC, *18*, *27*
append, 2

clear *(*helpers*)*, 12
codebook, 3

day *(*formatDate*)*, 9
duplicates, 4

egen, 5
expand2, 6
expandfreq *(*expandtbl*)*, 7
expandtbl, 7

formatDate, 9

generate, 11
glm, *18*

helpers, 12
hettest *(*regress*)*, 26
hist, *13*
histogram, 13

ilog, 14
is.Date *(*formatDate*)*, 9

label, 15
ladder *(*regress*)*, 26
lag.data.frame, 16
linkTest *(*regress*)*, 26
lm, *27*
logit, 17

mhor, 19
mhrr, 21
month *(*formatDate*)*, 9

N_ *(*n_*)*, 24
n_, 24

plot.regress *(*regress*)*, 26
predict.regress *(*regress*)*, 26

recode, 25
regress, 26
replace, 30
round, *17*, *19*, *22*, *27*

scatterPlotMatrix, 31
strate, 32
summ, 33
summary, 35

tab, 36

year *(*formatDate*)*, 9