Package 'Bullock'

December 14, 2019

Type Package

Title Miscellaneous helper utilities for use with John Bullock's code

Version 1.19.1.9000

Date 2019-12-04

Imports lmtest, dplyr, gdata, stringr

Suggests ivpack, multiwayvcov

Description These functions are used in John Bullock's code; they are typically needed for replication purposes. They range in complexity from a function that just removes NA values from a vector prior to summing it (sumNA) to a function that transforms regression output into LaTeX tables of the style that Bullock likes (latable).

License GPL (>= 2)

LazyLoad yes

URL https://github.com/jbullock35/Bullock

BugReports https://github.com/jbullock35/Bullock/issues

RoxygenNote 7.0.2

Encoding UTF-8

Language nob

R topics documented:

`alpha_cronbach` *Compute Cronbach's alpha for a battery of items.*

Description

This function is called by `reliability`. It generally should not be called by end users.

Usage

```
alpha_cronbach(S)
```

Arguments

`S`	Variance-covariance matrix of responses to a battery of measurements.

Author(s)

Joseph F. Lucke

`factorToDummyMatrix` *Perl-like qw() function for quoting a list of words*

Description

`factorToDummyMatrix` takes a factor of x levels and length n and returns an n-by-x matrix. The columns of the matrix have value 1, 0, or NA.

Usage

```
factorToDummyMatrix(fac)
```

Arguments

`fac`	factor

Value

Matrix. The column names of the matrix are the levels of the factor.

Note

For factors that have no missing data, conversion to a matrix of dummy variables can easily be accomplished by `model.matrix`. But by default, `model.matrix` omits NA values, returning a matrix that has rows for only those cases that were not NA in the factor. Moreover, `model.matrix` does not have an "na.action" argument.

This function temporarily changes the global na.action argument to permit `model.matrix` to return a matrix in which factor values of NA are matched by NA in every column.

Author(s)

John G. Bullock

`latable` *Print LaTeX table of regression results*

Description

Takes a list of regression models and returns a table of regression output formatted for LaTeX. There are two columns per regression: one for the coefficient estimates, another for standard errors.

Usage

```
latable(tables, substrings.to.remove = NULL, rows.to.remove=NULL, npmakebox = TRUE)
```

Arguments

`tables`	List of regression models. Supports models of class `glm`, `ivreg`, `lm`, `negbin`, `polr`, `vglm`, and `zeroinfl`.
`substrings.to.remove`	List of strings or regular expressions. If it is not a list, it will be coerced to a list with as.list(). Substrings in the row names that match any element in `substrings.to.remove` will be removed before the output is created.
`rows.to.remove`	Should be a list of strings or regular expressions. If it is not a list, it will be coerced to a list with as.list(). Rows that contain substrings matching any element in `rows.to.remove` will be removed from the output table before it is returned by the function. This is useful for creating "incomplete" regression tables that do not contain rows for some variables, e.g., control variables.
`npmakebox`	Improves formatting of the "Number of observations" row, mainly by ensuring that the Ns for each regression aren't decimal-aligned with the coefficient estimates. Requires the `numprint` package to be loaded in LaTeX.

Value

Returns a table of regression output formatted for LaTeX. The table is designed to be copied directly into LaTeX.

Note

The format of the tables produced by `latable` is inspired by "Estimates of relative survival rates, by cancer site," a table in Edward Tufte's essay on "The Cognitive Style of PowerPoint."

The current version works well for `lm` and `ivreg` models. It may be buggy when applied to models of other classes.

The current version produces buggy output if the name of the intercept row (typically "(Intercept)" or "Intercept" is modified by `substrings.to.remove` or `rows.to.remove`.

Author(s)

John G. Bullock

See Also

There are other packages that perform similar functions. See the `xtable` and `apsrtable` functions for alternatives.

`latexTable` *Create a LaTeX table from a matrix.*

Description

`latexTable` takes a single matrix, `mat`. By default, it returns a LaTeX macro that creates a well-formatted LaTeX table. It can take many arguments to adjust the table's formatting.

Usage

```
latexTable(
  mat,
  SE_table = TRUE,
  headerFooter = TRUE,
  commandName = "myTable",
  callCommand = TRUE,
  label = commandName,
  landscape = if (SE_table) ncol(mat)/2 >= 6 else ncol(mat) >= 6,
  starredFloat = FALSE,
  horizOffset = "-0in",
  rowNames = rownames(mat),
 footerRows = if (is.null(rowNames)) NULL else c("Number of observations", rep("000",
    ncol(mat)/2)),
 colNames = if (SE_table) colnames(mat)[seq(1, ncol(mat), by = 2)] else colnames(mat),
  colNameExpand = FALSE,
  extraRowHeight = if (SE_table) "2pt" else "4pt",
  spacerColumns = NULL,
  spacerColumnsWidth = ".5em",
  spacerRows = NULL,
```

```
  spacerRowsHeight = ".15in",
  tabColSep = "2.75pt",
  spaceBetweenColNameRows = TRUE,
  columnTierSeparator = "  ",
  printCaption = TRUE,
  caption = paste0("\\", label, "Caption"),
  captionMargins = NULL,
  formatNumbers = TRUE,
  decimalPlaces = 2,
  SE_fontSizeString = "\\fontsize{10.3bp}{10.3bp}\\selectfont",
  NA_text = "",
  writeToClipboard = FALSE
)
```

Arguments

`mat`	Matrix of information to be displayed in a LaTeX table.
`SE_table`	Logical variable that indicates whether the table contains pairs of columns, with the first column in each pair containing estimates, and the second column containing the corresponding standard errors. Defaults to TRUE. If `TRUE`, the even-numbered columns of `mat` will be rendered in smaller type than the odd-numbered columns: that is, the standard errors will be rendered in smaller type than their corresponding estimates. This default behavior can be overridden by the `SE_fontSizeString` argument.
`headerFooter`	Logical variable. If `TRUE`, which is the default, the output will be (or at least include) a LaTeX macro that generates a table. For example, you will be able to produce a table simply by calling `\myTable{p}` or `\myTable{h}` in your LaTeX code. B B B B B B B B If `headerFooter` is `FALSE`, the only output of the function will be rows from a LaTeX table (possibly including column headers). The function may not produce valid LaTeX output if both `SE_table` and `headerFooter` are `FALSE`.
`commandName`	A string. It is the name of the macro that produces the LaTeX table (if `headerFooter` is `TRUE`). By default, it is "myTable"; you can change it to something more descriptive, e.g., "mainEstimates".
`callCommand`	Logical variable. Should the last line of the `latexTable` object be a a call to the macro that creates the table? If `callCommand` is `TRUE`, which is the default, sourcing a file that contains `latexTable` output—that is, by using `\input` or `\include` in LaTeX—will produce a table when your LaTeX document is rendered. If `callCommand` is `FALSE`, the macro that can create your table will be included in your LaTeX document, but you will need to manually edit the LaTeX document to call the macro and thereby produce a table when the LaTeX document is rendered.
`label`	A string. Specifies the LaTeX label for table. It is not printed anywhere in the table or the caption, but references to the figure in your LaTeX document (for example, references created by `\ref` or `\autoref` must be include the label name. For simplicity, the default `label` is `commandName`.

`landscape` Logical variable. Determines whether the table is printed in landscape or in portrait mode. Affects the output only if if `headerFooter == TRUE` and `callCommand == TRUE`.

`starredFloat` Logical variable that indicates whether the LaTeX table should be specified with `table*` instead of `table`. The default is FALSE, but you may want to set it to TRUE if you want you are using a multi-column page layout in LaTeX and want the table to cross both columns.

`horizOffset` A string that specifies a LaTeX length, e.g., ".25in". When the LaTeX code produced by `latexTable` is rendered, the table will be moved to the right by this length (or to the left if the length is negative, e.g., "-.25in").

`rowNames` Character vector of labels for the rows in `mat`. The labels will be printed to the left of each row in `mat`. `rowNames` can be `NULL`.

`footerRows` List, or object that can be coerced to a list, of footer rows. Information about N and R^2 is typically included in `footerRows`. Each element in the list corresponds to a row in the footer. The first entry in each `footerRows` list-element should be the row name for the corresponding footer row (e.g., 'N', 'R^2').

`colNames` List, or object that can be coerced to a list, of column headings. Typically, each element in the list is a character vector, and the elements of the character vector specify the names of the table's columns.
B B B B B B B B If `SE_table` is `TRUE` (the default), each column name will appear over a pair of columns. In this case, each element in the `colNames` list should contain `ncol(mat)/2` entries.
B B B B B B B B To specify multi-line column labels, use a list with multiple elements. The entries in the first list element will then appear in the top row of the column label, the entries in the second list element will appear in the next row of the column label, and soB on.

`colNameExpand` Logical variable. By default, an entry of '' in a `colNames` list element—that is, an empty entry—indicates that a column should have no column heading. But if `colNameExpand` is `TRUE` and a text entry in a `colNames` list element is followed by one or more '' entries, the column name specified by the text entry will bridge the columns that have '' entries.
B B B B B B B B `colNameExpand` and `spacerColumns` do not play well together. If you run `latexTable` with `colNameExpand == TRUE` and a non-NULL `spacerColumns` argument, you will get LaTeX output, but you will probably need to edit the "\multicolumn" and "\cmidrule" commands in the output so that LaTeX can render the output.

`extraRowHeight` A string that specifies a length that LaTeX recognizes, e.g., '2pt' or '.25in'. The `extrarowheight` length in LaTeX will be set to `extraRowHeight`. In practice, this means that the vertical space between every row will be increased by `extraRowHeight`. This argument has no effect if `headerFooter` is `FALSE`.

`spacerColumns` A vector of integers. Specifies columns in `mat` after which to insert columns that contain no entries. These "spacer columns" are used to insert horizontal space into the typeset table.

B B B B B B B B To add a spacerColumn between the rownames and the first data column, make 0 one of the values in spacerColumns.
B B B B B B B B colNameExpand and spacerColumns do not play well together. If you run latexTable with colNameExpand == TRUE and a non-NULL spacerColumns argument, you will get LaTeX output, but you will probably need to edit the "\multicolumn" and "\cmidrule" commands in the output so that LaTeX can render the output.
B B B B B B B B See below for a technical note on spacerColumns and column spacing in LaTeX.

spacerColumnsWidth
: Either a single string of a recognizable LaTeX length (e.g., '.5em') or a character vector indicating the width of each spacer column. Has no effect unless headerFooter is TRUE.

spacerRows
: A vector of integers. After each row in mat whose number is in spacerRows, a vertical space of spacerRowsHeight will be printed. For example, if spacerRows == c(2, 4), a vertical space will be added after rows 2 and 4 of mat.

spacerRowsHeight
: A string that specifies a recognizable LaTeX length, e.g., ".15in".

tabColSep
: Character vector indicating a length that LaTeX recognizes, e.g., ".25in". The tabcolsep value in LaTeX will be set to this value if headerFooter is TRUE. If SE_table is TRUE, tabColSep will be the default distance between the estimate and the SE column in each column pair, and it will be half of the distance between column pairs. If SE_table is FALSE, tabColSep will simply be half of the default distance between columns. These distances between columns can be increased by the spacerColumns argument.

spaceBetweenColNameRows
: Logical variable. If TRUE, it adds a little space between the rows that specify column names. It has an effect only when the column names are split across multiple rows, i.e., when length(colNames) > 1.

columnTierSeparator
: A string. In the LaTeX code generated by latexTable, all columns are separated from each other by " & ". Column tiers – that is, pairs of columns giving the estimate and the SE for a particular coefficient – are further separated by columnTierSeparator, which defaults to two spaces (' '). This option affects only the LaTeX code produced by latexTable; it exists to make the LaTeX code more readable. It does not affect the typeset (e.g., PDF) version of the table.

printCaption
: Logical variable.

caption
: A string. It can include LaTeX commands, e.g., "\\textitResults from a minimal specification."

captionMargins
: A vector of two strings that specify the margins of the caption. The strings should be LaTeX lengths, e.g., ".25in" or ".67em". By default, captionMargins is NULL.

`formatNumbers` Logical variable. Pretty-print the entries in mat, e.g., by adjusting the number of digits after the decimal place.

`decimalPlaces` Integer. If `formatNumbers` is `TRUE`, table entries will be shown to this decimal place. For example, if `decimalPlaces==2`, both "3.0035" and "3" will become "3.00."
B B B B B B B B If `formatNumbers` is `FALSE`, entries will not be adjusted, but `decimalPlaces` will still be used to determine the widths of columns and some aspects of column spacing.

`SE_fontSizeString`
A string. Indicates how standard errors are to be formatted when `SE_table` is `TRUE`. Defaults to `\\fontsize{10.3bp}{10.3bp}\\selectfont`, which renders standard errors in slightly smaller type than the corresponding estimates.

`NA_text` A string. `NA` entries in `mat` will be replaced by the string.

`writeToClipboard`
Logical variable. Copy entire output to clipboard. Useful if you want to paste the output directly into a `.tex` file. Works only on Windows.

Details

The point of `latexTable` is to maximize flexibility in the formatting of LaTeX tables. The function's arguments permit much flexibility, and because the returned object is of the `character` class (in addition to the `latexTable` class), it can easily be tweaked "by hand" after it is generated.

One benefit of `latexTable` is that, by default, it will produce regression tables in which standard errors are positioned to the right of their corresponding estimates, and in smaller type. This design of regression tables is in contrast to conventional design, whereby standard errors appear in parentheses beneath the corresponding estimates. This is a "Tufte" design: to my knowledge, it was first used in his Edward Tufte's essay on "The Cognitive Style of PowerPoint."

A second benefit of `latexTable` is that it uses sane defaults for table formatting. That is, it produces tables with (a)B no vertical rules, (b)B few horizontal rules, and (c)B sensible spacing between rows and columns. The result is tables that are easier to read than normal tables: when looking at a table created with `latexTable`, you will not need to squint, or indeed to work at all, to figure out whether a given number corresponds to this variable or to thatB one.

A third benefit of `latexTable` is that, by default, it returns not just the LaTeX code for a table but a LaTeX macro that produces the table. The macro can be placed at any point in your LaTeX document; it does not need to be placed where you want the table to appear. To put the table where you want it in your LaTeX document, you need only use a single line of LaTeX code. For example, if `latexTable` produces a macro called "myTable", you can place the table in your LaTeX document by inserting the line `\myTable{p}` anywhere in your document. And because the macro that defines the table can be placed elsewhere, your LaTeX document can be far less cluttered than it would be if you had to define the entire table in the middle of your document.

A fourth benefit of `latexTable` is that it produces well-formatted LaTeX code. In other words, you won't just get tables that look good when they are rendered (for example, as PDF). You'll also get LaTeX code that is easy to read and to modify in the LaTeX editor of your choice.

Some tweaking of the output by hand may still be necessary to get the desired appearance. In particular, the formatting of each column is specified in the LaTeX code by rules given by the `numprint` LaTeX package, and these rules may need to be tweaked. For example, if `SE_table == TRUE` and a column-pair has a long column names (that is, a long `colNames` entry), you may need to modify the `latexTable` object that this function produces. Specifically, you may want to change `N{2}{2}` in the estimate-column specification to `N{3}{2}` or `N{4}{2}` to get the column pair centered beneath its heading. See the documentation for the `numprint` LaTeX package for more information on `numprint` column specifications like `N{2}{2}`.

`latexTable` tables can be transformed to PDF with `latexTablePDF`.

Value

An object of class `latexTable` and `character`. The returned object is a vector of strings of LaTeX code; each string is a row in a LaTeX table.

Note

Required LaTeX packages. The LaTeX code produced by the `latexTable` makes use of capabilities provided by the `array`, `booktabs`, and `numprint` LaTeX packages. If you haven't installed those LaTeX packages, you won't be able to render the tables produced by `latexTable`.

Column spacing in LaTeX. Ordinary methods for inserting space between columns involve the `\tabcolsep` and `\extracolsep` LaTeX lengths. Unfortunately, `\cmidrule` and other `booktabs` commands don't recognize that these LaTeX lengths are spaces *between* columns. As a result, rules (horizontal lines) drawn by `booktabs` commands extend into the intercolumn region if `\tabcolsep` and `\extracolsep` are used to provide intercolumn space. A similar problem occurs if `\hspace` is used to provide intercolumn space.
B B B B B B B B Thus, to fine-tune the spacing of the LaTeX tables produced by `latexTable`, blank columns can be inserted at arbitrary positions via the `spacerColumns` argument. This is a clunky way to adjust intercolumn space, but it solves the problem of positioning horizontal rules. In addition, no other approach affords the flexibility to insert horizontal space at arbitrary positions (useful for distinguishing tiers of columns from each other), and no other approach allows variation in the widths of the spaces between columns.

Changes from pre-release versions. The names of some arguments have changed slightly since the pre-release versions of this function. They have been changed to enforce consistency: camelCase is used for all arguments, and every acronym is followed by an underscore (_) character. We thus have `SE_table` instead of `SEtable`, `tabColSep` instead of `tabcolsep`, and soB on.

Examples

```
data(iris)
```

```
lm1 <- lm(Sepal.Length ~ Petal.Length,               data = iris)
lm2 <- lm(Sepal.Length ~ Petal.Length + Petal.Width, data = iris)
rT1 <- regTable(list(lm1, lm2))
latexTable(rT1)
latexTable(rT1, headerFooter = FALSE, spacerColumns = c(0, 2))
latexTable(rT1, colNames = qw("(1) (2)"))
latexTable(mat = matrix(1:16, nrow=4), colNames = 1:2)
latexTable(
  mat = matrix(1:16, nrow=2),
  colNames = c('1', '', '', 4))
latexTable(
  mat = matrix(1:16, nrow=2),
  colNames = c('1', '', '3', '4'),
  colNameExpand = TRUE)
latexTable(
  mat           = matrix(1:16, nrow=4),
  colNames      = c('One big heading', ''),
  colNameExpand = TRUE)
latexTable(
  mat           = matrix(1:16, nrow=4),
  rowNames      = 1:4,
  colNames      = c('One big heading', ''),
  colNameExpand = TRUE)
latexTable(
  mat           = matrix(1:16, nrow=4),
  colNames      = 1:2,
  rowNames      = qw("a b c d"),
  spacerColumns = c(0, 2))
```

---

lNA *Calculate length of vector after omitting NA values*

Description

Calculate length of vector after omitting NA values.

Usage

```
lNA(x)
```

Arguments

x

Author(s)

John G. Bullock

`lsos` *Improved version of* `ls`

Description

Pretty-printed version of `ls` that indicates the size of every object in an environment.

Usage

```
.ls.objects(pos = 1, pattern, order.by, decreasing = FALSE, head = FALSE, n=5)
lsos(..., n = 8)
```

Arguments

`pos`	position, on the search path, of the environment to search
`pattern`	regular expression. Only names matching `pattern` are returned.
`order.by`	object of `character` class. Valid arguments are `Type`, `Size`, `Rows`, and `Columns`. If argument is unspecified, information on objects will be returned in alphabetical order.
`decreasing`	logical value. Has no effect unless `order.by` is specified.
`head`	logical value. IF `TRUE`, information on only `n` objects will be returned.
`n`	number of objects for which to report information. Has no effect unless `head == TRUE`.
`...`	arguments that are passed to `.ls.objects`.

Details

`lsos` is a wrapper to `.ls.objects`. The main use of these functions is to see which objects are taking up the most memory.

Value

The returned object is a data frame.

Author(s)

Dirk Edelbuettel, JD Long

References

Function created by Dirk Edelbuettel and modified by JD Long. See http://stackoverflow.com/questions/1358003/ for details.

See Also

`ls`

meanNA *Calculate mean of vector after omitting NA values*

Description

Calculate mean of vector after omitting NA values.

Usage

```
meanNA(x)
```

Arguments

x

Author(s)

John G. Bullock

merge_fac *Merge factors*

Description

Fill in missing values in one factor with missing values from another.

Usage

```
merge_fac(fac.names, ...)
```

Arguments

fac.names	character vector of factor names
...	arguments passed to get()

Details

All factors should be of the same length. Missing values in the first factor named in fac.names are filled in with corresponding values from the second factor. Missing values in this merged factor are filled in with corresponding values from the third factor. And so on.

Value

Returned object is a factor.

Note

Merging factors in this way is trickier than just using a command like `fac1[is.na(fac1)] <- fac2[is.na(fac1)]` because fac1 and fac2 may have different factor levels. This commands takes care of the problem by merging the levels among different factors.

If a file that uses `merge.fac` is sourced into an environment other than the global environment (e.g., by `sys.source`), the `fac.names` variables may not be found unless the argument `envir = environment()` is also passed to `merge.fac`. In other words, it may be necessary to run a command like `merge_fac(fac.names=x, envi`

Author(s)

John G. Bullock

`modal_value` *Find modal value of a vector*

Description

Find modal value of a vector.

Usage

```
modal_value(x, na.rm = FALSE)
```

Arguments

`x`	a vector
`na.rm`	Remove NAs before modal value is determined

Note

If there are multiple values, this function will return only the first.

Author(s)

Ken Williams. Function copied from http://stackoverflow.com/a/8189441/697473.

`move.to.df` *Move a list of variables into a data frame.*

Description

Copy variables matching the pattern into a data frame, and perhaps delete the free-standing original variables.

Usage

```
move.to.df(pattern = NULL, move = TRUE)
```

Arguments

pattern	object of class character. Can specify a regular expression.
move	logical variable.

Details

IF move == TRUE, the variables in the environment will be deleted after they are moved into the data frame.

Value

Returned object is a data frame.

---

noNAmatrix *Perform listwise deletion on a matrix.*

Description

noNAmatrix performs "listwise deletion" on a matrix, removing all rows that contain any missing (NA) values.

Usage

```
noNAmatrix(x)
```

Arguments

x	a matrix

Details

This function is deprecated. Use na.omit instead.

Examples

```
noNAmatrix(matrix(c(1:8, NA), nrow=3))
```

push *Perl-like stack utilities for R*

Description

Perl-like stack utilities for R: `new_stack`, `push()`, `pop()`, `shift()`, and `unshift()`.

Usage

```
new_stack(value = NULL)
push(stack, value)
pop(stack)
shift(stack, value)
unshift(stack)
```

Arguments

`stack`	Object of class `stack`, created with `new_stack`.
`value`	For `new_stack`, the initial value of a `stack` object. For `push` and `shift`, something to be added to a `stack` object.

Value

`new_stack` returns an object of class `stack`. `unshift` and `pop` return the first and last values of `stack`, respectively.

Author(s)

Jeffrey A. Ryan, John G. Bullock

References

Adapted from Jeffrey A. Ryan's code at http://www.lemnica.com/esotericR/Introducing-Closures/.

See Also

See http://stackoverflow.com/questions/14488206 for related discussion, including a simpler implementation of `push` and `pop` by Matthew Plourde.

Examples

```
nb <- new_stack()
push(nb, 1:3)
nb$.Data     # [1] 1 2 3

pop(nb)      # from the back
unshift(nb)  # from the front
shift(nb, 3)
push(nb, 1)
nb$.Data     # [1] 3 2 1
```

qw *Perl-like qw() function for quoting a list of words*

Description

qw takes a string of words separated by spaces. It returns a vector in which each element is a word. The point of the function is to speed the creation of vectors of words.

Usage

```
qw(x)
```

Arguments

x	character string

Value

Character vector.

Author(s)

Florent Delmotte

References

Code taken from post by Florent Delmotte ("flodel") at http://stackoverflow.com/questions/520810/.

Examples

```
qw("You can type    text here
  with    linebreaks if you
  wish")
#  [1] "You"        "can"        "type"       "text"
#  [5] "here"       "with"       "linebreaks" "if"
#  [9] "you"        "wish"
```

regTable *Create a matrix of regression output from a list of regression models.*

Description

regTable takes a list of regression models, objList. It returns a matrix in which the columns are estimates and standard errors – two columns for each model.

Usage

```
regTable(
  objList,
  colNames = NULL,
  rowsToRemove = NULL,
  rowsToKeep = NULL,
  clusterSEs = FALSE,
  clusterVar = NULL
)
```

Arguments

objList	list of regression objects. They may be of class lm, plm, or ivreg. This is the only required argument.
colNames	A vector of strings as long as length(objList).
rowsToRemove	A vector of strings, which may specify regular expressions. Variables in the regressions whose names match the strings will be omitted from the regTable output. This argument overrides rowsToKeep.
rowsToKeep	A vector of strings, which may specify regular expressions. Variables in the regressions whose names match the strings will be kept in the regTable output. All other variables will be omitted. Before regTable was incorporated into this package, it used the rowsToKeep argument differently: variables were kept only if the beginnings of their names matched the strings in rowsToKeep.
clusterSEs	A logical scalar. If TRUE, the reported standard errors will be clustered at the level specified by clusterVar.
clusterVar	A list of length length(objList). Each element in the list indicates the clusters for the corresponding regression object in objList. If the regressions in objList are of class lm, clusterVar is passed to multiwayvcov::cluster.vcov. If the regressions in objList are instead of class ivreg, clustervar is passed to ivpack::cluster.robust.se.

Examples

```
data(iris)
lm1 <- lm(Sepal.Length ~ Petal.Length,               data = iris)
lm2 <- lm(Sepal.Length ~ Petal.Length + Petal.Width, data = iris)
regTable(list(lm1, lm2))
regTable(list(lm1, lm2), colNames = c("Sepal length", "Sepal width"))
regTable(list(lm1, lm2), rowsToKeep = 'Length')
regTable(list(lm1, lm2), rowsToKeep = c('Intercept', 'Length'))
```

---

reliability *Compute Cronbach's alpha for a battery of items.*

Description

Compute Cronbach's alpha for a battery of items, and show the reliability for all different batteries that might be created by removing one item from the original battery.

Usage

```
reliability (x, ...)
```

Arguments

x	Matrix of measurements, e.g., survey responses. Cannot have missing data.
...	Arguments to be passed to `alpha.cronbach()`. Currently serves no function.

Author(s)

Peter Ellis

---

rescale *Rescale a variable*

Description

Linear rescaling of numeric vectors. For example, a variable that ranges from 1 to 7 can be rescaled to range from 0 to 1.

Usage

```
rescale(x, newrange)
```

Arguments

x	numeric object
newrange	two-element numeric vector

Author(s)

Simon D. Jackman

Examples

```
vec <- 1:10
vecRescaled <- rescale(vec, c(2:5))
range(vecRescaled)  # 2 5
```

sdNA — *Calculate standard deviation of vector after omitting NA values*

Description

Calculate standard deviation of vector after omitting NA values

Usage

```
sdNA(x, na.rm = TRUE)
```

Arguments

x	a numeric vector or an R object which is coercible to one by as.vector.
na.rm	logical. Should missing values be removed?

See Also

sd

split_fac — *Create dummy variables for each level of a factor.*

Description

Create dummy variables for each level of a factor.

Usage

```
split_fac(
  fac,
  prefix = paste(deparse(substitute(NES.year.fac)), '.', sep = ''),
  env    = .GlobalEnv,
  ...)
```

Arguments

fac	factor variable
prefix	substring that begins the name of each created dummy variable
env	environment in which the dummy variables are created
...	arguments passed to assign()

Value

split_fac returns nothing. Instead, it creates, as a side effect, a set of logical variables – one for each level of fac.

Author(s)

John G. Bullock

Examples

```
 fac <- factor(rep(1:3, each = 3))
split_fac(fac, prefix = 'fac') # creates logical variables fac1, fac2, and fac3 in .GlobalEnv
```

---

sumNA *Calculate sum of vector after omitting NA values*

Description

Calculate sum of vector after omitting NA values.

Definition is `function(x) { return(sum(x, na.rm=TRUE)) }`.

Usage

```
sumNA(x)
```

Arguments

x	logical, integer, numeric, or complex vector

Value

The sum. If all elements of `x` are of type integer or logical, then the sum is an integer. Otherwise it is a length-one numeric or complex vector.

See Also

`sum`

---

table.sep *helper function for latable()*

Description

Interleaves columns between the columns of a table. Typically used to pretty-print tables.

Usage

```
table.sep(table, separator = "&", sig.digits = 2)
```

Arguments

table	object of class `table`
separator	object of class `character`
sig.digits	integer

varNA *Calculate variance of vector after omitting NA values*

Description

Calculate variance of vector after omitting NA values

Usage

```
varNA(x)
```

Arguments

x	numeric vector, matrix, or data frame

Details

The definition of varNA is `function(x) {var(x, na.rm = TRUE)}`.

See Also

var

%IN% *Value matching*

Description

%IN% returns a logical vector indicating whether there is a match for its left operand. It is like %in%, but it has one crucial difference: if there are NA values in the left operand, the corresponding values in the returned vector will also be NA (rather than FALSE, as with %in%.)

Usage

```
x %IN% table
```

Arguments

x	vector or NULL: the values to be matched.
table	vector or NULL: the values to be matched against.

Value

A logical vector of the same length as x. It indicates whether a match was found for each non-NA element of x. NA elements of x are matched by NA elements in the returned vector.

Note

The ordinary binary match operator, `%in%`, can be misleading because it seems more closely related to == than it is. The problem is that == will return NA in some (expected) cases, but `%in%` will never return NA. Instead, when using `%in%`, the returned vector will be FALSE for every NA value in the left operand.

Like ==, `%IN%` will return NA when there are NA values in the left operand. See below for an example.

`%IN%` will always return TRUE values when `%in%` would do so, and vice versa. The two operators differ only in the sense that `%IN%` returns FALSE in some cases where `%in%` returns NA.

Author(s)

John G. Bullock

See Also

`%in%`

Examples

```
tmp <- c(1, 2, 3, NA)
tmp == 1      # TRUE FALSE FALSE NA
tmp %in% 1:2  # TRUE TRUE  FALSE FALSE
tmp %IN% 1:2  # TRUE TRUE  FALSE NA
```

Index