Package 'Bullock'

January 10, 2020

Type Package

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

Version 2.0.0.9000

Date 2019-12-04

Imports lmtest, magrittr, rlang, stringr

Suggests ivpack,
multiwayvcov,
testthat,
knitr,
rmarkdown

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 removes NA values from a vector prior to summing it
(sumNA) to a trio of functions that produce well-formatted regression-table
output (regTable, latexTable, and latexTablePDF).

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

VignetteBuilder knitr

R topics documented:

`Bullock-package` *Bullock: convenience functions and tools for table generation*

Description

This package provides tools that I use in my code. Most of the tools are simple convenience functions. Three of the functions—`regTable()`, `latexTable()`, and `latexTablePDF()`—are more elaborate tools for making tables, and they are designed to be used together. See the Building better tables in less time vignette for an overview.

Details

Of the convenience functions, I most often use `qw()`, `reliability()`, `meanNA()`, and `sumNA()`.

Author(s)

- Maintainer: John Bullock (`<john@johnbullock.org>`)
- Florent Delmotte [contributor]
- Dirk Edelbuettel [contributor]
- Peter Ellis [contributor]
- Simon Jackman [contributor]
- JD Long [contributor]
- Joseph Lucke [contributor]
- Matt Pettis [contributor]
- Ken Williams [contributor]

See Also

Useful links:

- `https://github.com/jbullock35/Bullock`
- Report bugs at `https://github.com/jbullock35/Bullock/issues`

`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

`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.

Usage

```
latexTable(
  mat,
  SE_table = TRUE,
  headerFooter = TRUE,
  commandName = "myTable",
  callCommand = TRUE,
  label = commandName,
  floatPlacement = "p",
  landscape = if (SE_table) ncol(mat)/2 >= 6 else ncol(mat) >= 6,
  starredFloat = FALSE,
  horizOffset = "-0in",
  rowNames = rownames(mat),
  footerRows = lt_footer(),
  colNames = lt_colNames_default(),
  colNameExpand = FALSE,
  extraRowHeight = if (SE_table) "2pt" else "4pt",
  spacerColumns = lt_spacerColumns_default(),
  spacerColumnsWidth = ".67em",
  spacerRows = NULL,
  spacerRowsHeight = ".15in",
  tabColSep = "2.75pt",
  spaceBetweenColNameRows = TRUE,
  columnTierSeparator = "  ",
  printCaption = TRUE,
  caption = paste0(label, " caption goes here."),
  captionMargins = NULL,
  formatNumbers = TRUE,
  decimalPlaces = 2,
  SE_fontSizeString = "\\fontsize{10.3bp}{10.3bp}\\selectfont",
  NA_text = "",
  clipboard = FALSE
)
```

Arguments

`mat`	Matrix of information to be displayed in a LaTeX table.
`SE_table`	Logical variable that indicates whether `mat` contains pairs of columns, with the first column in each pair containing estimates, and the second column containing the corresponding standard errors. (Matrices returned by `regTable()` have this form.) 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 type sizing 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 master LaTeX document. If `headerFooter` is `FALSE`, the only output of the function will be LaTeX code for "data rows"—one row for each row of `mat`.
`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 your master LaTeX document—will produce a table when that master LaTeX document is rendered. If `callCommand` is `FALSE`, sourcing the file will make the macro available in your LaTeX document, but it will not call the macro. (You will need to call the macro yourself by adding a line like `\myTable{p}` to your LaTeX document.)
`label`	A string. Specifies the LaTeX label for a table. It is not printed anywhere in the table, but references to the figure in your LaTeX document (for example, references created by `\ref` or `\autoref`) must include the label name. For simplicity, the default `label` is the same as the `commandName` argument.
`floatPlacement`	Character vector of lengthB 1. Acceptable values are `p` (the default, which places each table on its own page), `h`, `H`, `t`, `b`, and `!`. Affects the output only if `landscape` is `FALSE`. See the LaTeX wikibook for more on float placement in LaTeX.

`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., '$F$', '$R^2$'). By default, the only footer row indicates the number of observations for each model in `mat`.
`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. 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. 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. By default, column names will be taken from `colnames(mat)`. If `colnames(mat)` is `NULL`, columns will be numbered "(1)", "(2)", etc. See `lt_colNames_default()` for more information.
`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. `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 prac-

tice, 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. By default, spacerColumns are specified by a helper function, `spacerColumns_default()`:

  - If `SE_table` is `FALSE`, there is a spacer column between every column in `mat`.
  - If `SE_table` is `TRUE`, there is a spacer column after every even-numbered column in `mat`, except for the last column.
  - If `rowNames` is not `NULL`, a spacer column is inserted between the table's row names and the first column of data.

    To add a spacerColumn between the rownames and the first data column, make 0 one of the values in spacerColumns.

    `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.

    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." It can also include references to other labeled parts of your LaTeX document, e.g., "\\autorefSomeFigure". See the examples.
`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." 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.
`clipboard`	Logical variable. Copy entire output to clipboard. Useful if you want to paste the output directly into a `.tex` file. Works only on Windows.

Value

An object of classes `latexTable` and `character`. The returned object is a vector of strings of LaTeX code; each string is a line in a LaTeX macro that can create a table. (There is one small exception. If `callCommand` is `TRUE`, the last line is not part of the macro; instead, it calls the macro, thereby telling LaTeX to display the table).

Note

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

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.
- Some default arguments have changed. In particular, the default `spacerColumns` argument is no longer `NULL`. Instead, the default is to insert spacer columns in appropriate places. See documentation of the `spacerColumns` argument for details.

See Also

Other functions for making tables: `regTable()`, `latexTablePDF()`. See also the Building better tables in less time vignette.

Examples

```
data(iris)
lm1 <- lm(Sepal.Length ~ Petal.Length,               data = iris)
lm2 <- lm(Sepal.Length ~ Petal.Length + Petal.Width, data = iris)
lm3 <- lm(Sepal.Length ~ Petal.Length * Petal.Width, data = iris)
rT1 <- regTable(list(lm1, lm2, lm3))
latexTable(rT1)
latexTable(rT1, SE_table = FALSE, colNames = lt_colNumbers())
lt2 <- latexTable(
  mat      = rT1,
  colNames = list(qw("model model model"), qw("1 2 3")))
lt3 <- latexTable(
  mat         = rT1,
  colNames    = lt_colNumbers(),
  rowNames    = c(
    "Intercept",
    "Petal length",
    "Petal width",
    "Petal length $\\times$ petal width"),
  footerRows  = list(lt_nobsRow(), lt_rSquaredRow()),
  commandName = 'mainEstimates',
 caption    = "Each entry is an estimate or a standard error from a separate OLS regression.")
lt4 <- update(
  lt3,
  commandName = 'myEstimates',  # change name of LaTeX macro
  spacerRows  = 1)              # add vertical space after intercept row
```

`latexTablePDF` *Renders a LaTeX table as a PDF file.*

Description

`latexTablePDF()` takes an object produced by `latexTable()` and writes a PDF file. It can also write the corresponding .tex file.

Usage

```
latexTablePDF(
  latexTable,
  container = TRUE,
  containerFilename = "tableContainer.tex",
  outputFilenameStem = "latexTable",
  writePDF = TRUE,
  writeTex = FALSE,
  overwriteExisting = FALSE,
  verbose = FALSE,
  continuedFloat = FALSE,
  continuedFloatStar = FALSE,
  firstPageEmpty = TRUE,
  firstTableNumber = NULL,
  openPDFOnExit = TRUE
)
```

Arguments

`latexTable`	Object of class `latexTable` (typically produced by `latexTable()`) or a list of such objects.
`container`	Logical variable. Should the LaTeX code in `latexTable` be inserted into a LaTeX file that contains a complete LaTeX preamble and the `\begin{document}` and `\end{document}` tags? If you want `latexTablePDF` to produce a PDF file, `container` must be `TRUE`. But if you just want to write `latexTable` to disk as a .tex file that you will insert into your own larger LaTeX document, `container` should probably be `FALSE`.
`containerFilename`	A string. Specifies the path of the "container" LaTeX file into which the `latexTable` table or tables will be inserted to make a complete LaTeX file that can be rendered as PDF. By default, it is "tableContainer.tex", which is included in this package.
`outputFilenameStem`	A string. It is the path and name of the file is to be saved to disk, up to the extension. For example, if you want to save "myTable.pdf" to disk, set `outputFilenameStem = "mytable"`.
`writePDF`	Logical variable. Should a PDF file be saved to disk?
`writeTex`	Logical variable. Should a .tex file be saved to disk?
`overwriteExisting`	Logical variable. Should files to be saved overwrite existing files that have the same names?
`verbose`	Logical variable. `latexTablePDF` calls `pdflatex` to render PDF files, and if `verbose` is `TRUE`, all of the output from `pdflatex` will be printed to screen. Useful for debugging.
`continuedFloat`	Logical variable. Should be `TRUE` if the table or tables to be rendered are part of a series and should all share the same number. For example, `continuedFloat`

should be true if `latexTable` is a list of tables and you all want them to be numbered as TableB 3.

`continuedFloatStar`
: Logical variable. Should be `TRUE` if the table or tables to be rendered are part of a series, and if all tables should share the same number but be distinguished by some secondary character. For example, `continuedFloatStar` should be true if `latexTable` is a list of tables and you all want them to be numbered as TableB 3a, TableB 3b, etc.

  `continuedFloat` and `continuedFloatStar` cannot both be `TRUE`.

`firstPageEmpty`
: Logical variable. If `TRUE`, the page that contains the first table in `latexTable` will have an empty header and footer. (Technically, `latexTablePDF` will insert `\thispagestyle{empty}` into the code block that contains the first table in `latexTable`.)

`firstTableNumber`
: Integer. What number should the first table in `latexTable` have?

  By default, the table numbering will be "natural." That is, the number of the first `latexTable` table will be determined by the number of preceding tables in the document. For example, if you use `latexTablePDF()` to create a .tex file that you then insert into a larger LaTeX document, and if the .tex file is preceded in the LaTeX document by two tables, the first table created by `latexTable()` will be TableB 3.

`openPDFOnExit`
: Logical variable. Open the PDF file after it is created by `latexTablePDF`? This argument has an effect only on Windows, and only if `writePDF` is `TRUE`.

Details

Although `latexTablePDF` produces PDF files by default, it is also useful for creating .tex files. For example, you may have a list of tables produced by `latexTable()` and a complex LaTeX document that contains many different sections and tables. When `latexTablePDF` is used with `writeTex = TRUE`, it will produce a single file that contains LaTeX code for all of the tables in your list. You can then insert those tables into your LaTeX document by adding a single `\input` or `\include` command to your LaTeX document. For details, see `vignette("tables", package = "Bullock")`.

Note

Required LaTeX tools. If `writePDF` is `TRUE`, `pdflatex` must be installed on your system. (It is part of almost every LaTeX installation.) The array, booktabs, caption, and `numprint` packages must also be installed. In addition:

- If `writePDF` is `TRUE` and `containerFilename` is "tableContainer.tex" (the default), the fancyhdr, `footmisc`, geometry, and `ragged2e` packages must be installed.
- If `writePDF` is `TRUE` and you are producing a landscaped table, the `afterpage` and `pdflscape` packages must be installed.

Most of these packages, and perhaps all of them, are already part of your LaTeX installation.

See Also

Other functions for making tables: `latexTable()`, `latexTablePDF()`. See also the Building better tables in less time vignette.

Examples

```
## Not run:
  data(iris)
  lm1 <- lm(Sepal.Length ~ Petal.Length,               data = iris)
  lm2 <- lm(Sepal.Length ~ Petal.Length + Petal.Width, data = iris)
  lm3 <- lm(Sepal.Length ~ Petal.Length * Petal.Width, data = iris)
  lmList <- list(lm1, lm2, lm3)
  rT1 <- regTable(lmList)
  lT1 <- latexTable(
    mat        = rT1,
    colNames   = lt_colNumbers(),
    rowNames   = c(
      "Intercept",
      "Petal length",
      "Petal width",
      "Petal length $\\times$ petal width"),
    footerRows = list(lt_nobsRow(), lt_rSquaredRow()),
    spacerRows = 1,  # insert white space between Intercept row and other rows
    caption    = paste0(
      "\\textit{Sepal length as a function of petal length and petal width.} ",
      "Entries are estimates and standard errors from OLS regressions..."
    )
  )
  latexTablePDF(lT1, outputFilenameStem = "irisData")


  # Create a PDF or .tex file that contains two tables:
  lm1v <- update(lm1, subset = (Species == 'versicolor'))
  lm2v <- update(lm2, subset = (Species == 'versicolor'))
  lm3v <- update(lm3, subset = (Species == 'versicolor'))
  rT2  <- regTable(list(lm1v, lm2v, lm3v))
  lT2  <- update(lT1, mat = rT2, commandName = "tableVersicolor")

  latexTablePDF(                                # PDF with two pages
    lT2,
    outputFilenameStem = "irisData_twoTables")
  latexTablePDF(                                # add .tex file with code for two tables
    lT2,
    outputFilenameStem = "irisData_twoTables",
    writeTex           = TRUE)

## End(Not run)
```

`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`

`lt_colNames_default` *Compute default column names for latexTable objects.*

Description

If `colnames(mat)` is not `NULL`, this function will use `colnames(mat)` as the `colNames` argument in `latexTable()`. If `colnames(mat)` is `NULL`, column names will be determined by `lt_colNumbers()`.

Usage

```
lt_colNames_default(
  mat = parent.frame()$mat,
  SE_table = parent.frame()$SE_table
)
```

Arguments

`mat`	A matrix, typically a `regTable` object.
`SE_table`	Logical variable. See `latexTable()`.

Details

The function is not exported and is intended to be called only by `latexTable()`.

Value

A vector of strings. Each string is a columnB name.

`lt_colNumbers` *Automatically determine column names of the form (1), (2), etc.*

Description

Given `mat` and `SE_table`, this function determines appropriate column-number names of the form "(1)", "(2)", etc.

Usage

```
lt_colNumbers(mat = parent.frame()$mat, SE_table = parent.frame()$SE_table)
```

Arguments

`mat`	A matrix, typically a `regTable` object.
`SE_table`	Logical variable. See `latexTable()`.

Value

A vector of strings. If `SE_table` is `TRUE`, the vector elements are "(1)", "(2)", etc., where the last column number is `ncol(mat)/2`. If `SE_table` is `FALSE`, the vector elements are "(1)", "(2)", etc., where the last column number is simply `ncol(mat)`.

lt_footer *Compute default footers for latexTable() objects.*

Description

The default footer row or rows are determined in this way: if SE_table is FALSE or rowNames is NULL, no footer rows are produced. Otherwise, a footer row will be added for each of the following attributes of mat: "r.squared", "SER", and "N". If mat lacks one of those attributes, there will be no corresponding footerB row. The function is not exported and is intended to be called only by latexTable().

Usage

```
lt_footer(
  mat = parent.frame()$mat,
  rowNames = parent.frame()$rowNames,
  SE_table = parent.frame()$SE_table,
  decimalPlaces = parent.frame()$decimalPlaces
)
```

Arguments

mat	A matrix, typically a regTable object.
rowNames	Character vector. See latexTable().
SE_table	Logical variable. See latexTable().
decimalPlaces	Integer. See See latexTable().

Value

A list of string vectors. Each list element contains the strings needed to produce a footerB row.

lt_nobsRow *Specify a footer row that indicates the number of observations for each regression.*

Description

Given a mat produced by regTable(), this function returns a footer row that indicates the number of observations for each model in mat.

Usage

```
lt_nobsRow(mat = parent.frame()$mat)
```

Arguments

mat	A regTable object.

Value

A vector of strings. The first element in the vector is "Number of observations". The remaining elements are the numbers of observations for each regression in `mat`.

`lt_rSquaredRow` *Specify a footer row that indicates $R\hat{}2$ for each regression.*

Description

Given a `mat` produced by `regTable()` in which all regressions are of class `lm`, this function returns a footer row that indicates R^2 for each model in `mat`.

Usage

```
lt_rSquaredRow(
  mat = parent.frame()$mat,
  decimalPlaces = parent.frame()$decimalPlaces
)
```

Arguments

`mat`	A matrix, typically a `regTable` object.
`decimalPlaces`	Integer. See `latexTable`.

Value

A vector of strings. The first element in the vector is "R$^2$". The remaining elements are strings that indicate R^2 for each model in `mat`. The strings are rounded to the number of digits specified by the `decimalPlaces` argument.

`lt_SER_row` *Specify a footer row that indicates the standard error of regression for each model*

Description

Given a `mat` produced by `regTable()` in which all regressions are of class `lm`, this function returns a footer row that indicates the standard error of regression (i.e., σ, the "residual standard error") for each model in `mat`.

Usage

```
lt_SER_row(
  mat = parent.frame()$mat,
  decimalPlaces = parent.frame()$decimalPlaces
)
```

Arguments

`mat`	A matrix, typically a `regTable` object.
`decimalPlaces`	Integer. See `latexTable`.

Value

A vector of strings. The first element in the vector is "Std. error of regression". The remaining elements are strings that indicate the SER for each model in `mat`. The strings are rounded to the number of digits specified by the `decimalPlaces` argument.

`lt_spacerColumns_default`

Compute default positions of spacer columns in calls to latexTable().

Description

`spacerColumns_default` specifies the default `spacerColumns` argument in calls to `latexTable`. It takes the values of `mat`, `SE_table`, and `rowNames` that are passed to `latexTable`. From these values, it computes the default positions of spacer columns:

- If `SE_table` is `FALSE`, there is a spacer column between every column in `mat`.
- If `SE_table` is `TRUE`, there is a spacer column after every even-numbered column in `mat`, except for the last column.
- If `rowNames` is not `NULL`, a spacer column is inserted between the table's row names and the first column of data.

Usage

```
lt_spacerColumns_default(
  mat = parent.frame()$mat,
  SE_table = parent.frame()$SE_table,
  rowNames = parent.frame()$rowNames
)
```

Arguments

`mat`	Matrix.
`SE_table`	Logical variable.
`rowNames`	Vector.

Details

The function is not exported and is intended to be called only by `latexTable()`.

Value

A vector of integers.

mergeFac — *Fill in missing values of one factor with corresponding values from another.*

Description

Given a factor of length n that contains some missing values, fill in the missing values with the corresponding values of others factors.

Usage

```
mergeFac(x, ...)
```

Arguments

x	Factor variable.
...	Other factor variables.

Details

x and the factors named in otherFactors must all have the same length. If they do, missing values in x will be filled in with the corresponding values of the first factor in otherFactors. If the corresponding values of that factor are also missing, mergeFac() will look to the corresponding values of the next factor, and soB on.

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.

Examples

```
fac1 <- factor(c("a", NA,  "b", NA,  NA))
fac2 <- factor(c("y", "y", "y", NA,  NA))
fac3 <- factor(c(NA,  "z", "z", "z", NA))
mergeFac(fac1, fac2)       # [1] a    y    b    <NA> <NA>
mergeFac(fac1, fac2, fac3)  # [1] a    y    b    z    <NA>
```

missingPackageString — *Checks for existence of required LaTeX packages.*

Description

The code produced by latexTable() can be rendered only if certain LaTeX packages are installed. (See the note at the end of the latexTablePDF() help file for details.) If packages are missing, this function generates an informative string that can be used in warning or error messages.

Usage

```
missingPackageString(
  installedPackageList,
  requiredPackageList,
  writePDF,
  writeTex
)
```

Arguments

`installedPackageList, requiredPackageList`	Character vectors.
`writePDF, writeTex`	Logical variables. See the `latexTable()` documentation for further information about these arguments.

Details

This function is not exported. It is called by `latexTablePDF()` only if packages are missing. It generates either an error (if `writePDF` is `TRUE`) or a warning (if `writePDF` is `FALSE` but `writeTex` is `TRUE`).

`missingValueFunctions` *Missing-value helper functions.*

Description

Functions to make code a little clearer. These are mainly ordinary functions, like `mean()`, with `na.rm` set to `TRUE`. For example, `meanNA()` is defined as `function(x) mean(x, na.rm = TRUE)`.

Usage

```
lNA(x, verbose = FALSE)

lNAv(x)

meanNA(x)

sdNA(x)

sumNA(x)

varNA(x)
```

Arguments

`x`	An R object.
`verbose`	Logical variable. If `TRUE`, `lNA` will print the lengths of `x` before and after `NA` values have been removed.

Details

`lNA(x)` returns its value silently. `lNAv` is shorthand for `lNA(x, verbose = TRUE)`; it returns the same value as `lNA(x)` but also prints the lengths of the vector before and after NAs are removed.

Examples

```
x <- c(1:3, NA, 5)
lNA(x)
lNAv(x)

sum(x)    # returns NA
sumNA(x)  # returns 11
meanNA(x)

sdNA(x)
varNA(x)
```

`modalValue` *Find modal value of a vector.*

Description

Find modal value of a vector. If there are multiple modal values, all will be returned.

Usage

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

Arguments

`x`	Vector.
`na.rm`	Logical variable.

Details

Before this package was released, it returned only the first mode if there were multiple modes. It now returns all modes. See the examples.

Author(s)

Ken Williams. See http://stackoverflow.com/a/8189441/697473.

Examples

```
modalValue(qw("a b b"))       # [1] "b"
modalValue(qw("a a b b c"))  # [1] "a" "b"
modalValue(1:3)              # [1] 1 2 3
```

`moveToDF` *Move or copy "freestanding" variables into a data frame.*

Description

Variables are specified by `pattern`, which is a regular expression. The modal length of all variables in the calling environment that match `pattern` is determined. Matching variables are then moved to a data frame (or copied to a data frame if `move` is `FALSE`). `pattern` may be `NULL`, in which case all variables in the calling environment will be examined.

Usage

```
moveToDF(pattern = NULL, move = TRUE)
```

Arguments

`pattern`	String that specifies a regular expression. It is `NULL` by default, in which case all variables in the environment are examined for inclusion in the data frame.
`move`	Logical variable.

Details

If there are multiple modal lengths of the objects in the calling environment, all modes will be used. For example, if the calling environment has 20 objects of lengthB 1, 20 objects of lengthB 2, and one object of lengthB 3, the returned data frame may have as many as 40 columns.

Variables that have the "dim" attribute – for example, arrays and matrices – will not be moved into the new data frame. Functions will never be moved into the new data frame, either.

Value

Data frame containing all one-dimensional variables that have names matching `pattern` and that have the modal length of those variables.

`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 objects, such as those created by `lm()`. 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,
  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. If `rowsToRemove` is specified, this argument has no effect.

`clusterVar`	A list of lengthB 1 or `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`. Can be `NULL` (the default), in which case standard errors won't be clustered.

Value

A matrix in which the columns are estimates and standard errors – two columns for each model. The matrix has an "N" attribute that indicates the number of observations for each regression. If all regressions were of class `lm`, it also has the "r.squared" and "SER" attributes. (The "SER" attribute indicates the standard error of regression – AKA σ or the "residual standard error" — for each model.)

Note

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`.

See Also

Other functions for making tables: `latexTable()`, `latexTablePDF()`. See also the Building better tables in less time vignette.

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'))
regTable(list(lm1, lm2), clusterVar = list(iris$Species))
```

`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 vector to have a specified minimum and maximum.*

Description

Rescale a vector to have a specified minimum and maximum.

Usage

```
rescale(x, newRange = c(0, 1))
```

Arguments

`x`	Numeric object.
`newRange`	Numeric vector of lengthB 2.

Author(s)

Simon D. Jackman

See Also

`scale()` and `scales::rescale()`

Examples

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

`stackUtilities` *Perl-like stack utilities for R.*

Description

You can "pop" data off the end of a vector, one element at a time, with `pop()`. You can also "push" new data onto the end of the vector with `push()`. The analogous functions for working at the start of a vector are `shift()` and `unshift()`: `shift()` removes the first element of your vector, and `unshift()` prepends new elements to your vector.

Usage

```
push(x, values)

pop(x)

unshift(x, values)

shift(x)
```

Arguments

`x`	Object, typically a vector or a list.
`values`	Object to be added to `x`.

Details

An important and unusual feature of `push()`, `pop()`, `shift()`, and `unshift()` is that they modify objects "in place"—that is, even when no explicit assignment is done. For example, `pop(x)` will return the last value of `x`, but it will also remove the last value from the `x` object. The examples illustrate this point.

These functions are adapted from Matt Pettis's code at `https://gist.github.com/mpettis/b7bfeff282e3b052684f`.

Previous versions of these functions were adapted from Jeffrey A. Ryan's code at `http://www.lemnica.com/esotericR/Introducing-Closures/`. That code works but is based on the creation of "stack" objects that contain their own environments. One consequence is that changing a copy of a stack object changes the original stack object, and vice versa. Note too that, in Ryan's code, the traditional meanings of `shift()` and `unshift()` are reversed: he uses `shift()` to concatenate objects, `unshift()` to remove a value from an object.

Value

`pop()` and `shift()` will return a scalar, that is, an object of lengthB 1. `push()` and `unshift()` don't return anything.

Author(s)

Matt Pettis

John G. Bullock

See Also

Thomas Leeper's Gist describes his own implementation of `pop()` and `push()` and includes links to six other implementations. Some of these implementations of `pop()` and `push()` do not have the modify-in-place characteristic of the corresponding Perl functions. This quality is also absent from the constructor function at `https://stackoverflow.com/a/14489296/697473`.

Examples

```
myStack <- 1:3
push(myStack, 4)
myStack  # [1] 1 2 3 4

pop(myStack)    # [1] 4
shift(myStack)  # [1] 1
myStack         # [1] 2 3

unshift(myStack, "hello")
myStack         # [1] "hello" "2" "3"
```

update.latexTable — *Update a latexTable object with new arguments.*

Description

Each latexTable object stores, as an attribute, the call that producedB it. update.latexTable() updates the call by replacing arguments or adding new ones. It then calls latexTable() to produce a new latexTable object.

Usage

```
## S3 method for class 'latexTable'
update(object, ...)
```

Arguments

object	A latexTable object
...	Arguments to latexTable(), e.g., colNames, caption.

Details

update.latexTable() is adapted from stats::update.default(). It is a method for the generic update().

Value

A latexTable object.

Examples

```
lT1 <- latexTable(matrix(1:16, nrow = 4))
lT2 <- update(lT1, mat = matrix(2:17, nrow = 8), commandName = "intTable")
```

%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