Package 'table1xls'

July 27, 2017

Type Package

Title Exports Reproducible Summary Tables to Multi-Tab Spreadsheet Files (.xls or .xlsx)

Version 0.4.0

Date 2017-07-26

Author Assaf P. Oron [cre, aut]

Maintainer Assaf P. Oron <assaf.oron@seattlechildrens.org>

Description A collection of time-saving wrappers for reproducible export of summary tables commonly used in scientific articles, to .xls/.xlsx multi-tab spreadsheets, while controlling spreadsheet layout. Powered by 'XLConnect'/'rJava' utilities.

License GPL-3

Imports XLConnect

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-07-26 22:04:36 UTC

R topics documented:

table1xls-package *Publication-Grade Tables Exported to .xls/.xlsx spreadsheets*

Description

Generate tabular summaries in formats commonly found in scientific articles, and export them to Office-compatible spreadsheet document (.xls/.xlsx format).

Details

Package:	table1xls
Type:	Package
Version:	0.4.0
Date:	2017-02-28
License:	GPL-3

Any statistician collaborating with scientists, especially in the health sciences, has to produce publication-grade summary tables. Most commonly these are "Table 1" style basic demographics, contingency tables, or regression-output summaries. As a minimum, statisticians need to provide tables, from which the lead author can easily and accurately generate these tables.

Outside of physics and mathematical fields, the vast majority of scientific investigators edit their manuscripts and tables in office-type software, usually the Microsoft suite. R can easily output data in `.csv` format; however, the analyst or someone in the investigative team still has to perform tedious work formatting the tables using office software, into a template that the lead author can use. Moreover, the manner in which regression summaries appear in manuscripts, is quite different from the typical output of `summary.glm` and similar R functions.

More important than the tedious labor involved, this manual "portage" of analysis output from the raw R/csv format to tabular format can be prone to data errors, and is not reproducible.

The package `table1xls` is meant to close this gap, enabling analysts and their collaborators to focus on the analysis and the science with peace of mind, while saving precious time. It relies upon the functionality offered by the `XLConnect` package. It can be seen as an Office-compatible baby version of the LaTex (and now also HTML) oriented `xtable` (or the SAS-inspired `tables`). `table1xls` does offer some conveniences that these other packages cannot match, in particular the possibility of packaging all your output tables as separate tabs in the same single spreadsheet document, and the ability to lay out related tables side-by-side or below each other in the same sheet.

Regarding export to non-Microsoft spreadsheet software: with my LibreOffice (Windows version) the .xls exports work perfectly fine, while .xlsx is not as reliable. I recommend using the former, of course.

You can download the latest package source code directly from GitHub, via the command

`devtools::install_github("assaforon/table1xls")`.

If you have `roxygen2` installed, then the help pages will show following a GitHub download, thanks to a hack originating/disseminated by Yihui Xie.

I will gladly accept requests for new functionalities, as well as comments and corrections on existing functions.

Thanks, Assaf

PS: XLConnect uses the rJava package. For some systems, additional configuration is needed to enable Java on R. For Windows machines tested so far it was pretty much plug-and-play; the Linux CRAN tests all seem to work; not so for some Macs.

When manipulating large spreadsheet objects, the Java Virtual Machine might choke and issue error messages. If this happens, you will need to start a new session, and before loading XLConnect (either directly or via loading table1xls), write

```
options(java.parameters = "-XmxYYYY")
```

Where YYYY is the virtual-memory size in MB. The default is only 128 MB. The string "-Xmx1g" is interpreted as 1 gigabyte. Keep in mind that you need to have substantially more RAM than the amount allocated to the JVM.

Author(s)

Assaf P. Oron.

Maintainer: Assaf P. Oron <assaf.oron.at.seattlechildrens.org>

`niceRound` *Rounding to a Predictable Number of Digits*

Description

Rounds numbers to always have the specified number of decimal digits, rather than R's "greedy" most-compact rounding convention. Includes optional "<0.0..." override adequate for representing small p-values.

Usage

```
niceRound(numbers, digits = 0, plurb = FALSE)
```

Arguments

numbers	the numbers to be rounded. Can also be a vector or numeric array.
digits	the desired number of decimal digits
plurb	logical, should the p-value-style "less-than blurb" convention be used? Default FALSE.

Details

R's standard round utility rounds to at *most* the number of digits specified. When the number happens to round more "compactly", it rounds to fewer digits. Thus, for example, `round(4.03,digits=2)` yields 4 as an answer. This is undesirable when trying to format a table, e.g., for publication.

`niceRound` solves this problem by wrapping a `format` call around the `round` call. The result will always have `digits` decimal digits. In addition, since reporting p-values always involves rounding, if the argument `plurb` is `TRUE`, then values below the rounding thresholds will be represented using the "less than" convention. For example, with `digits=3` and `plurb=TRUE`, the number 0.0004 will be represented as `<0.001`.

Author(s)

Assaf P. Oron `<assaf.oron.at.seattlechildrens.org>`

See Also

round,format

`rangeString` *Utility functions for table summaries*

Description

Various auxiliary convenience functions, mostly for `XLunivariate`. Functions calculating simple statistics and returning the output in a formatted manner, making it easier for `XLunivariate` to embed them in spreadsheet cells.

Usage

```
rangeString(x, digits = 1, sep = "-", na.rm = FALSE, ...)

iqrString(x, digits = 1, sep = "-", quantmeth = 7, na.rm = FALSE, ...)

roundmean(x, digits = 1, na.rm = FALSE, ...)

roundmedian(x, digits = 1, na.rm = FALSE, ...)

roundSD(x, digits = 1, na.rm = FALSE, ...)

emptee(x, ...)
```

Arguments

`x`	vector (usually numeric, but can be logical) on which statistics are to be calculated
`digits`	numeric: how many digits to round the output to?

sep	character: separating character for range- type functions.
na.rm	logical: should missing values be removed? (default FALSE) Passed onto the underlying functions
...	this is ignored by the functions, but enables the "mixing and matching" of extra parameters between functions called by XLunivariate, without triggering an error.
quantmeth	numeric: for functions calling quantile, the calculation method for the quantiles. Default is 7 to match the R default. Note that it is shrunk towards the median and hence biased, but typically with lower MSE. A very viable alternative is 6, the SAS/SPSS (and Stata?) default, which is unbiased. See the help on quantile for more details.

Details

This is a small collection of useful utilities called by XLunivariate. They return 1-2 summary statistics, in a format that will not require additional formatting and formula-manipulation in Excel.

For example, roundmedian returns the median rounded to the specified number of digits, while iqrString returns the 1st and 3rd quartiles, separated by at least one dash (default 3 dashes). XLunivariate can combine these functions' output to produce the formatted summary "median (Q1---Q3)" often used in research articles.

In particular, emptee returns an empty string, enabling the use of XLunivariate to produce only a single summary statistic per cell rather than a pair.

Value

The summary statistic(s), in the format specified via the arguments.

Author(s)

Assaf P. Oron <assaf.oron.at.seattlechildrens.org>

See Also

XLunivariate which is the main function calling these utilities.

Examples

```
book2<-XLwriteOpen("chick2.xls")
## Plain-vanilla
XLunivariate(book2,"weightByDiet",ChickWeight$weight,ChickWeight$Diet,
             title="Mean Weights by Diet",rowTitle="Diet")

## Replace mean/SD with median/range, put results beside previous
XLunivariate(book2,"weightByDiet",ChickWeight$weight,ChickWeight$Diet,
             title="Median Weights by Diet",rowTitle="Diet",col1=8,
          fun1=list(fun=roundmedian,name="Median"),fun2=list(fun=rangeString,name="range"))

### You can also do only one statistic... by "killing" one of the functions
XLunivariate(book2,"weightByAge",ChickWeight$weight,ChickWeight$Time,
```

```
            title="Mean Weights by Age",rowTitle="Age (Days)",seps=rep("",3),
            fun2=list(fun=emptee,name=""))
cat("Look for",paste(getwd(),"chick2.xls",sep='/'),"to see the results!\n")
```

XLaddText *Write text to a single cell*

Description

Write text to a single cell in a specified file and sheet, and save the file.

Usage

```
XLaddText(wb, sheet, text, row1 = 1, col1 = 1)
```

Arguments

wb	a workbook-class object
sheet	numeric or character: a worksheet name (character) or position (numeric) within wb.
text	character: the text to be written to file.
row1, col1	integer: the row and column for the output.

Details

Since XLConnect only exports data to spreadsheets as `data.frame`, this function sends the text as an on-the-fly `data.frame` with one column and one row, and without writing the header or the row name.

Value

The function returns invisibly, after writing the data into `sheet` and saving the file.

Note

If the specified `sheet` does not exist, the function will create it, assuming that was the user's intent (e.g., add a text-only sheet with explanations to a file.) This is hard-coded, because the inadvertent creation of single-text sheets due to typos can be easily discovered upon opening the file :)

Examples

```
t1<-XLwriteOpen("generic1.xls")
### Just a meaningless matrix; function converts to data.frame and exports.
XLgeneric(t1,"s1",matrix(1:4,nrow=2))
### Now adding row names, title, etc. Note adding the title shifts the table one row down.
XLgeneric(t1,"s1",matrix(1:4,nrow=2),col1=5,addRownames=TRUE,
          title="Another Meaningless Table",rowTitle="What?",
          rowNames=c("Hey","You!"))
```

```
###... and now adding some text
XLaddText(t1,"s1","You can also add text here...",row1=10)
XLaddText(t1,"s1","...or here.",row1=11,col1=8)
XLaddText(t1,"s2",
          "Adding text to a new sheet name will create that sheet!"
          ,row1=2,col1=2)

### A more complicated example, showing how a "flattened" 3-way table might be exported:

carnames=paste(rep(c(4,6,8),each=2),"cylinders",rep(c("automatic","manual"),3))
XLgeneric(t1,'cars',ftable(mtcars$cyl,mtcars$vs,mtcars$am),
          addRownames=TRUE,rowNames=carnames,rowTitle="Engine Type",colNames=c("S","V"))

cat("Look for",paste(getwd(),"generic1.xls",sep='/'),"to see the results!\n")
```

`XLgeneric` *Write generic rectangular data to a spreadsheet*

Description

Export a generic data frame, matrix or table to a spreadsheet and save the file.

Usage

```
XLgeneric(wb, sheet, dataset, title = NULL, addRownames = FALSE,
  rowNames = rownames(dataset), rowTitle = "Name", colNames = NULL,
  row1 = 1, col1 = 1, purge = FALSE)
```

Arguments

`wb`	a `workbook-class` object
`sheet`	numeric or character: a worksheet name (character) or position (numeric) within `wb`.
`dataset`	the rectangular structure to be written. Can be a data frame, table, matrix or similar.
`title`	character: an optional overall title to the table. Default (`NULL`) is no title.
`addRownames`	logical: should a column of row names be added to the left of the structure? (default `FALSE`)
`rowNames`	character: vector of row names. Default `rownames(dataset)`, but relevant only if `addRownames=TRUE`.
`rowTitle`	character: the title to be placed above the row name column (default "Name")
`colNames`	character: vector of column names to replace the original ones. Default `NULL`, meaning that the original names are left intact. Note that the title for the row-names column (if `addRownames=TRUE`) is *not* considered part of `colNames`, and is set separately.

row1, col1	numeric: the first row and column occupied by the output.
purge	logical: should sheet be created anew, by first removing the previous copy if it exists? (default FALSE)

Details

This function is a convenience wrapper for getting practically any rectangular data structure into a spreadsheet, without worrying about conversion or spreadsheet-writing technicalities.

If the structure is not a data frame (or inherited from one), but a table or matrix, the function will convert it into one using as.data.frame.matrix, because data frames are what the underlying function writeWorksheet can export.

See the XLtwoWay help page, for behavior regarding new-sheet creation, overwriting, etc.

Value

The function returns invisibly, after writing the data into sheet and saving the file.

Author(s)

Assaf P. Oron <assaf.oron.at.seattlechildrens.org>

See Also

For two-way contingency tables, see XLtwoWay.

Examples

```
t1<-XLwriteOpen("generic1.xls")
### Just a meaningless matrix; function converts to data.frame and exports.
XLgeneric(t1,"s1",matrix(1:4,nrow=2))
### Now adding row names, title, etc. Note adding the title shifts the table one row down.
XLgeneric(t1,"s1",matrix(1:4,nrow=2),col1=5,addRownames=TRUE,
          title="Another Meaningless Table",rowTitle="What?",
          rowNames=c("Hey","You!"))

###... and now adding some text
XLaddText(t1,"s1","You can also add text here...",row1=10)
XLaddText(t1,"s1","...or here.",row1=11,col1=8)
XLaddText(t1,"s2",
          "Adding text to a new sheet name will create that sheet!"
          ,row1=2,col1=2)

### A more complicated example, showing how a "flattened" 3-way table might be exported:

carnames=paste(rep(c(4,6,8),each=2),"cylinders",rep(c("automatic","manual"),3))
XLgeneric(t1,'cars',ftable(mtcars$cyl,mtcars$vs,mtcars$am),
          addRownames=TRUE,rowNames=carnames,rowTitle="Engine Type",colNames=c("S","V"))

cat("Look for",paste(getwd(),"generic1.xls",sep='/'),"to see the results!\n")
```

XLoneWay *One-way Contingency Tables exported to a spreadsheet*

Description

Calculates a one-way contingency table in counts and percents, exports a formatted output to a spreadsheet, and saves the file.

Usage

```
XLoneWay(wb, sheet, rowvar, table1mode = FALSE, title = NULL,
  rowTitle = "Value", rowNames = NULL, colNames = NULL, ord = NULL,
  row1 = 1, col1 = 1, digits = ifelse(length(rowvar) >= 500, 1, 0),
  combine = TRUE, useNA = "ifany", testname = NULL, testBelow = FALSE,
  margins = TRUE, ..., purge = FALSE, pround = 3)
```

Arguments

wb	a workbook-class object
sheet	numeric or character: a worksheet name (character) or position (numeric) within wb.
rowvar	vector: the categorical variable (logical, numeric, character, factor, etc.) to be tabulated.
table1mode	logical: is the function called from XLtable1? If TRUE, some modifications will be made to the output. Default FALSE.
title	character: an optional overall title to the table. Default (NULL) is no title.
rowTitle	character: the title to be placed above the row name column (default empty string)
rowNames	character: vector of row names. Default behavior (NULL): automatically determined from data
colNames	dummy argument for compatibility with calls from XLtable1(). Otherwise ignored by function.
ord	numeric vector specifying row-index order in the produced table. Default (NULL) is no re-ordering.
row1, col1	numeric: the first row and column occupied by the table (title included if relevant).
digits	numeric: how many digits (after the decimal point) to show in the percents? Defaults to 1 if n>=500, 0 otherwise.
combine	logical: should counts and percents be combined to the popular "Count(percent)" format, or presented side-by-side? (default TRUE)
useNA	How to handle missing values. Passed on to table (see help on that function for options).

testname	string, the *name* of a function to run a significance test on the table. Default NULL (no test).
testBelow	logical, should test p-value be placed right below the table? Default FALSE, which places it next to the table's right edge, one row below the column headings
margins	logical: should margins with totals be returned? Default TRUE.
...	additional arguments as needed, to pass on to get(textfun); for example, the reference frequencies for a Chi-Squared GoF test.
purge	logical should sheet be created anew, by first removing the previous copy if it exists? (default FALSE)
pround	number of significant digits in test p-value representation. Default 3.

Details

This function performs a one-way contingency table, also calculating the distribution in percents.

The table is then exported to worksheet sheet in workbook wb, either using the format "Count(percent)" (if combine=TRUE), or as two separate columns in the same table.

See the XLtwoWay help page, for behavior regarding new-sheet creation, overwriting, etc.

Value

The function returns invisibly, after writing the data into sheet and saving the file.

Author(s)

Assaf P. Oron <assaf.oron.at.seattlechildrens.org>

See Also

If interested in other descriptive summaries, see XLunivariate. For two-way contingency tables, see XLtwoWay.

Examples

```
book1<-XLwriteOpen("chick1.xls")
XLoneWay(book1,"Diets",ChickWeight$Diet)
### Now in separate columns, and with a title - note it shifts the table down.
### Also adding a Chi-Square goodness of fit (GoF) test vs. a 2:1:1:1 allocation
XLoneWay(book1,"Diets",ChickWeight$Diet,combine=FALSE,row1=10,
         rowTitle="Diet",title="Counts by Diet:",testname='chisq.test',p=c(2,1,1,1)/5)
cat("Look for",paste(getwd(),"chick1.xls",sep='/'),"to see the results!\n")
```

XLregresSummary *Regression Summary Tables exported to a spreadsheet*

Description

Takes regression effect estimates and the corresponding standard errors, transforms to "human scale" if requested, calculates confidence-intervals and p-values, and exports a standard formatted summary table to a spreadsheet.

Usage

```
XLregresSummary(wb, sheet, betas, SE = NULL, varnames = NULL, colid = 1:2,
  transfun = identity, title = NULL, effname = "Difference",
  alpha = 0.05, df = NA, roundig = 2, pround = 3, row1 = 1,
  col1 = 1, purge = FALSE)
```

Arguments

wb	a workbook-class object
sheet	numeric or character: a worksheet name (character) or position (numeric) within wb.
betas	numeric: a vector of point estimates, or a matrix containing estimates and standard errors in columns
SE	numeric: a vector of standard-error estimates for the effects. If NULL (default), user needs to specify them via the betas matrix.
varnames	character: a vector of effect names (column 1 of output table). If NULL (default), user needs to specify them via the betas matrix.
colid	integer: vector of indices for the columns containing the point estimates and SEs, respectively. Used only if betas is a matrix.
transfun	transformation function for betas,SE, to produce columns 2-3 of the output. Defaults to identity. use exp for odds ratio or relative risk.
title	character: an optional overall title to the table. Default (NULL) is no title.
effname	character: a string explaining what the effect stands for, e.g. "difference" (the default), "Odds Ratio", etc.
alpha	numeric, Type I error for CIs. Default 0.05 for 95% CIs.
df	numeric, residual degrees of freedom. If a finite value is provided, t-distribution p-value and CIs will be calculated; otherwise Normality is assumed. Default NA. ##' @param title character: an optional overall title to the table. Default (NULL) is no title.
roundig	numeric: how many digits (after the decimal point) to round the effect estimate to?

pround	numeric: how many digits (after the decimal point) to round the p-value to? P-values rounded down to zero will show up as "<" the smallest nonzero value, e.g. with the default pround=3 p-values smaller than 0.0005 will show up as "<0.001".
row1, col1	numeric: the first row and column occupied by the table (title included if relevant).
purge	logical: should sheet be created anew, by first removing the previous copy if it exists? (default FALSE)

Details

This function produces a standard scientific-article regression summary table, given the raw regression output. The resulting table has 4 columns: effect name, its (optionally transformed) magnitude, a probabilistically symmetric confidence interval (likewise transformed), and p-value. The formatted table is exported to sheet, and the file is immediately saved.

The input can be provided as separate vectors of point estimates (betas) and standard errors (SE), or as a single matrix for betas. In the latter case, as a default the effect names will be rownames(betas), unless a vector with more descriptive names is provided via varnames.

See the XLtwoWay help page, for behavior regarding new-sheet creation, overwriting, etc.

Value

The function returns invisibly, after writing the data into sheet.

Note

The default CI's are 95% and Normal. P-values are also derived from the Normal. If you run any regression whose intervals are calculated differently (e.g., linear regression with not-huge sample size), make sure to change both confac and pfun accordingly, as is shown in the example.

Author(s)

Assaf P. Oron <assaf.oron.at.seattlechildrens.org>

Examples

```
book4<-XLwriteOpen("attenu.xls")

quakenames=c("Magnitude (Richter), per unit","Distance (log km), per x10")

# Ground acceleration as a function of magnitude and distance, all log scale.
quakemod1=summary(lm(log10(accel)~mag+log10(dist),data=attenu))


## Model-scale summaries; we don't care for the intercept.
# First (wrongly) using Normal distribution for inference/CIs

XLregresSummary(book4,"ModelScale",varnames=quakenames,
                betas=quakemod1$coef[-1,1],SE=quakemod1$coef[-1,2],
```

```
                ,title="Log-Ground Acceleration Effects, Normal CIs")

# Now using t-distribution as befits linear regression

XLregresSummary(book4,"ModelScale",varnames=quakenames,
                betas=quakemod1$coef[-1,1],SE=quakemod1$coef[-1,2],
                ,title="Log-Ground Acceleration Effects",df=quakemod1$df[2],col1=6)

## Same thing, but using matrix input; no need to provide SE and names.
## It is arguably still nicer to provide your own names - but could be a reproducibility risk.
## Also, increasing the p-value resolution by changing 'pround'.

XLregresSummary(book4,"ModelScale",betas=quakemod1$coef[-1,],
                pround=6,title="Log-Ground Acceleration Effects",
                ,df=quakemod1$df[2],row1=8)

## Effects are arguably more meaningful as percent change.
## So... still same model, but different summaries.
## Also, note the combination of matrix input with names over-written via 'varnames':

XLregresSummary(book4,"PercentChange",varnames=quakenames,
                betas=quakemod1$coef[-1,],
                roundig=1,pround=6,title="Relative Ground Acceleration Effects",
                transfun=function(x) 100*(10^x-1),
                effname="Percent Change",df=quakemod1$df[2])

cat("Look for",paste(getwd(),"attenu.xls",sep='/'),"to see the results!\n")

### lm() does not take account of station or event level grouping.
### So we use a mixed model, losing 16 data points w/no station data:
### Run this on your own... and ask the authors of "lme4" about p-values at your own risk :)

# library(lme4)
# quakemod2=lmer(log10(accel)~mag+log10(dist)+(1|event)+(1|station),data=attenu)
#
# XLregresSummary(book4,"MixedModel",varnames=quakenames,betas=fixef(quakemod2)[-1],
# SE=sqrt(diag(vcov(quakemod2)))[-1],
# roundig=1,pround=6,
# title="Relative Ground Acceleration Effects",
# transfun=function(x) 100*(10^x-1),effname="Percent Change",df=160)
```

XLtable1 *"Table 1" Style List of Tables exported to a spreadsheet*

Description

Formats and exports a series of shared-structure tables, and saves the file.

Usage

```
XLtable1(wb, sheet, DF, colvar = NULL, fun = XLoneWay, title = "Table 1",
  colTitle = NULL, colNames = NULL, row1 = 1, col1 = 1, digits = NULL,
  useNA = "ifany", ..., purge = FALSE)
```

Arguments

wb	a workbook-class object
sheet	numeric or character: a worksheet name (character) or position (numeric) within wb.
DF	a rectangular array with all variables to be tabulated.
colvar	vector; specifies the variable to cross-tabulate for fun=XLtwoWay (see 'Details' for convenience options), or to stratify for XLunivariate. Has to be the entire variable, rather than just a name.
fun	The table1xls function to apply for each variable. Default XLoneWay. Other supported functions are XLtwoWay, XLunivariate.
title	character: an optional overall title to the table. Default "Table 1".
colTitle	character: the title to be placed above the first column of the column variable. Default NULL.
colNames	character: when relevant, more descriptive names for columns in case colvar is used. Default NULL, which will use the unique values of colvar as names.
row1, col1	numeric: the first row and column occupied by the table (title included if relevant).
digits	numeric: how many digits (after the decimal point) to show in the percents? Defaults to 1 if n>=500 or if using XLunivariate, and 0 otherwise.
useNA	How to handle missing values. Passed on to table (see help on that function for options).
...	additional arguments as needed, to pass on to fun; for example, non-default summary function choices for XLunivariate.
purge	logical should sheet be created anew, by first removing the previous copy if it exists? (default FALSE)

Details

Auto-generation of a series of tables of the same type for a single dataset. One-way and two-way contingency tables and numerical summaries are all supported, but all summaries call the same atomic fun.

The function employs convenience conventions for two-way tabulation: first, if colvar is specified and fun is left blank, then fun will be set to XLtwoWay. Second, if fun=XLtwoWay and colvar is left blank, then colvar will be set to the last column of DF.

For numerical summaries, use fun=XLunivariate. If you specify colvar, two-way summaries stratified by colvar will be returned.

Note that this function does not mix and match. Just make several calls to XLtable1 with different sub-datasets and different values of fun, and combine the results in your report document.

In a similar vein, two-way summaries do not return the marginal one-way summaries as a byproduct. For example, if you use `fun=XLtwoWay`, then in order to get column totals for the generated two-way output, you will need to call `XLtable1` again on the same data, using the default `fun=XLoneWay`.

See the `XLtwoWay` help page, for behavior regarding new-sheet creation, overwriting, etc.

Value

The function returns invisibly, after writing the data into `sheet` and saving the file.

Author(s)

Assaf P. Oron <assaf.oron.at.seattlechildrens.org>

See Also

`XLoneWay`,`XLtwoWay`,`XLunivariate`.

Examples

```
table1<-XLwriteOpen("table1.xls")

## A default, option-free call generates one-way tables
XLtable1(table1,'cars1',mtcars[,c(2,8:11)])
## You can prettify a bit, first by changing variable names

names(mtcars)[c(2,8:11)]=c("Cylinders","V/S","Auto/Manual","Gears","Carbureutors")
XLtable1(table1,'cars1',mtcars[,c(2,8:11)],
         title="'mtcars': Summary of Categorical Variables",col1=4)

## Now two-way, generated implicitly by specifying 'colvar' (unless fun=XLunivariate)
XLtable1(table1,'cars2',mtcars[,8:11],colvar=mtcars$Cylinders,
         title="Cylinders vs. categorical variables",colTitle="Cylinders")

## Finally, two-way numerical summaries for continuous variables
names(mtcars)[c(1,3:7)]=c('MPG','Engine Vol.','HP',"Axle Ratio","Weight","Quarter Mile")
XLtable1(table1,'carsContinuous',mtcars[,c(1,3:7)],fun=XLunivariate,colvar=mtcars$Cylinders,
         title="Cylinders vs. continuous variables",colTitle="Cylinders")

cat("Look for",paste(getwd(),"table1.xls",sep='/'),"to see the results!\n")
```

XLtwoWay *Two-way Contingency Tables exported to a spreadsheet*

Description

Produces 2-way contingency tables, optionally with percentages, exports them to a spreadsheet, and saves the file.

Usage

```
XLtwoWay(wb, sheet, rowvar, colvar, table1mode = FALSE, sumby = 1,
  rowTitle = "", rowNames = NULL, colNames = NULL, ord = NULL,
  row1 = 1, col1 = 1, title = NULL, header = FALSE, purge = FALSE,
  digits = ifelse(length(rowvar) >= 500, 1, 0), useNA = "ifany",
  percents = TRUE, combine = percents, testname = "chisq.test",
  pround = 3, testBelow = FALSE, margins = TRUE, ...)
```

Arguments

wb	an workbook-class object
sheet	numeric or character: a worksheet name (character) or position (numeric) within wb.
rowvar	vector: categorical variable (logical, numeric, character, factor, etc.) for the table's rows
colvar	vector: categorical variable (logical, numeric, character factor, etc.) for the table's columns
table1mode	logical: is the function called from XLtable1? If TRUE, some modifications will be made to the output. Default FALSE.
sumby	whether percentages should be calculated across rows (1, default) or columns (2).
rowTitle	character: the title to be placed above the row name column (default empty string)
rowNames, colNames	character vector of row and column names. Default behavior (NULL): automatically determined from data
ord	numeric vector specifying row-index order in the produced table. Default (NULL) is no re-ordering.
row1, col1	numeric: the first row and column occupied by the table (title included if relevant).
title	character: an optional overall title to the table. Default (NULL) is no title.
header	logical: should a header row with the captions "Counts:" and "Percentages:" be added right above the tables? Relevant only when combine=FALSE,percents=TRUE)
purge	logical: should sheet be created anew, by first removing the previous copy if it exists? (default FALSE)
digits	numeric: how many digits (after the decimal point) to show in the percents? Defaults to 1 if n>=500, 0 otherwise.
useNA	How to handle missing values. Passed on to table (see help on that function for options).
percents	logical: would you like only a count table (FALSE), or also a percents table side-by-side with the the count table (TRUE, default)?
combine	logical: should counts and percents be combined to the popular "Count(percent)" format, or presented side-by-side in separate tables? (default: same value as percents)

testname	string, the *name* of a function to run a significance test on the table. Default chisq.test. If you want no test, set testname=NULL
pround	number of significant digits in test p-value representation. Default 3.
testBelow	logical, should test p-value be placed right below the table? Default FALSE, which places it next to the table's right edge, one row below the column headings.
margins	logical: should margins with totals be returned? Default TRUE.
...	additional arguments as needed, to pass on to get(textfun)

Details

This function produces two-way cross-tabulated counts of unique values of rowvar, colvar, optionally with percentages, calculated either by row (sumby=1, default) or column (sumby=2). Row and column margins are also produced. ##' Tables are automatically saved to the file associated with the wb spreadsheet object.

There is an underlying asymmetry between rows and columns, because the tables are converted to data frame in order for writeWorksheet to export them. The percents can be in parentheses in the same cells as the counts (combine=TRUE, default), in an identically-sized table on the side (combine=FALSE,percents=TRUE), or absent (combine=FALSE,percents=FALSE). If you want no margins, just use the simpler function XLgeneric.

Value

The function returns invisibly, after writing the data into sheet.

Note

The worksheet sheet does not have to pre-exist; the function will create it if it doesn't already exist.

By default, if sheet exists, it will be written into - rather than completely cleared and rewritten de novo. Only existing data in individual cells that are part of the exported tables' target range will be overwritten. If you do want to clear an existing sheet while exporting the new tables, set purge=TRUE. This behavior, and the usage of purge, are the same across all table1xls export functions.

This function uses the internal function fancytab2 which produces 2-way tables with counts, percentages and margins.

Author(s)

Assaf P. Oron <assaf.oron.at.seattlechildrens.org>

See Also

Uses writeWorksheet to access the spreadsheet. See setStyleAction to control the output style. If interested in one-way tables, see XLoneWay.

Examples

```
### Contrived example looking at, e.g., the distribution of A-K-Q card counts
### in two partners' Bridge hands


hand1=rhyper(1000,12,40,13)
hand2=rhyper(1000,12-hand1,27+hand1,13)
handNames=c("0-1",2:4,"5 or more")


### The problem is ridiculously symmetric, so I de-symmetrize the presentation slightly:

book3<-XLwriteOpen("hands.xls")
XLtwoWay(book3,"PartnersAKQcounts",cut(hand1,c(0,2:6,14)-0.5),cut(hand2,c(0,2:5,14)-0.5),
         rowTitle="AKQ's in Hand 1 (rows) vs. Hand 2",
         rowNames=c(handNames[-5],5,"6 or more","Total"),
         colNames=c(handNames,"Total"),header=TRUE)

## Same table, but percents now condition on columns rather than rows,
## counts/pct header row removed - but a title added.
## Also, Chi-square p-value now placed below the table rather than the default top-right corner
XLtwoWay(book3,"PartnersAKQcounts",cut(hand1,c(0,2:6,14)-0.5),cut(hand2,c(0,2:5,14)-0.5),
         rowTitle="AKQ's in Hand 1 (rows) vs. Hand 2",
         rowNames=c(handNames[-5],5,"6 or more","Total"),
         colNames=c(handNames,"Total"),header=FALSE,row1=12,sumby=2,
         title="Now Percents are Summed by Column:",testBelow=TRUE)

cat("Look for",paste(getwd(),"hands.xls",sep='/'),"to see the results!\n")
```

`XLunivariate` *Univariate Statistics Exported to Excel*

Description

Calculates univariate summary statistics (optionally stratified), exports the formatted output to a spreadsheet, and saves the file.

Usage

```
XLunivariate(wb, sheet, calcvar, colvar = rep("", length(calcvar)),
  table1mode = FALSE, fun1 = list(fun = roundmean, name = "Mean"),
  fun2 = list(fun = roundSD, name = "SD"), seps = c("", " (", ")"),
  sideBySide = FALSE, title = NULL, rowTitle = "", rowNames = NULL,
  colNames = rowNames, ord = NULL, row1 = 1, col1 = 1, purge = FALSE,
  ...)
```

Arguments

wb	a workbook-class object
sheet	numeric or character: a worksheet name (character) or position (numeric) within wb.
calcvar	vector: variable to calculate the statistics for (usually numeric, can be logical).
colvar	vector: categorical variable to stratify calcvar's summaries over. Will show as columns in output only if sideBySide=TRUE; otherwise as rows. Default behavior if left unspecified, is to calculate overall summaries for a single row/column output.
table1mode	logical: is the function called from XLtable1? If TRUE, some modifications will be made to the output. Default FALSE.
fun1, fun2	two lists describing the utility functions that will calculate the statistics. Each list has a fun component for the function, and a name component for its name as it would appear in the column header.
seps	character vector of length 3, specifying the formatted separators before the output of fun1$fun, between the two outputs, and after the output of fun2$fun. Default behavior encloses the second output in parentheses. See 'Examples'.
sideBySide	logical: should output be arranged horizontally rather than vertically? Default FALSE.
title	character: an optional overall title to the table. Default (NULL) is no title.
rowTitle	character: the title to be placed above the row name column (default empty string)
rowNames	character vector of row names. Default behavior (NULL): automatically determined from data
colNames	column names for stratifying variable, used when sideBySide=TRUE. Default: equal to rowNames.
ord	numeric vector specifying row-index order (i.e., a re-ordering of rowvar's levels) in the produced table. Default (NULL) is no re-ordering.
row1, col1	numeric: the first row and column occupied by the table (title included if relevant).
purge	logical: should sheet be created anew, by first removing the previous copy if it exists? (default FALSE)
...	parameters passed on to fun1$fun, fun2$fun

Details

This function evaluates up to 2 univariate functions on the input vector calcvar, either as a single sample, or grouped by strata defined via colvar (which is named this way for compatibility with XLtable1). It produces a single-column or single-row table (apart from row/column headers), with each interior cell containing the formatted results from the two functions. The table is exported to a spreadsheet and the file is saved.

The cell can be formatted to show a combined result, e.g. "Mean (SD)" which is the default. Tne function is quite mutable: both `fun1$fun`, `fun2$fun` and the strings separating their formatted output can be user-defined. The functions can return either a string (i.e., a formatted output) or a number that will be interpreted as a string in subsequent formatting. The default calls `roundmean`,`roundSD` and prints the summaries in `"mean(SD)"` format.

See the `XLtwoWay` help page, for behavior regarding new-sheet creation, overwriting, etc.

Value

The function returns invisibly, after writing the data into `sheet` and saving the file.

Author(s)

Assaf P. Oron <assaf.oron.at.seattlechildrens.org>

See Also

Uses `writeWorksheet` to access the spreadsheet, `rangeString` for some utilities that can be used as `fun1$fun`,`fun2$fun`. For one-way (univariate) contingency tables, `XLoneWay`.

Examples

```
book2<-XLwriteOpen("chick2.xls")
## Plain-vanilla
XLunivariate(book2,"weightByDiet",ChickWeight$weight,ChickWeight$Diet,
             title="Mean Weights by Diet",rowTitle="Diet")

## Replace mean/SD with median/range, put results beside previous
XLunivariate(book2,"weightByDiet",ChickWeight$weight,ChickWeight$Diet,
             title="Median Weights by Diet",rowTitle="Diet",col1=8,
          fun1=list(fun=roundmedian,name="Median"),fun2=list(fun=rangeString,name="range"))

### You can also do only one statistic... by "killing" one of the functions
XLunivariate(book2,"weightByAge",ChickWeight$weight,ChickWeight$Time,
             title="Mean Weights by Age",rowTitle="Age (Days)",seps=rep("",3),
             fun2=list(fun=emptee,name=""))
cat("Look for",paste(getwd(),"chick2.xls",sep='/'),"to see the results!\n")
```

`XLwriteOpen` *Open a spreadsheet document, while deleting the previous copy.*

Description

Open a spreadsheet file (.xls or .xlsx), while deleting the previous copy if it exists.

Usage

```
XLwriteOpen(path)
```

Arguments

path	character: the spreadsheet's full filename, including the extension. Only .xls, .xlsx extensions are allowed.

Details

The XLConnect function loadWorkbook can open existing spreadsheets or create new ones if they don't exist. However, it *cannot* delete the previous copy when opening the new one – which is the default expected behavior of software such as R. As a result, analysts might inadvertently mix old and new versions of data and analyses, in the same spreadsheet.

This short utility mitigates the risk, by calling unlink first to make sure existing copies are deleted before the new spreadsheet file is opened.

Value

an XLConnect workbook object.

Note

Even though the workbook object is created, and is linked to a specific file name, it will only be saved to disk after saveWorkbook is called. See example. From table1xls version 0.3.0 on, all of the package's spreadsheet-export functions save the file by default. The example also illustrates some of the peculiarities of working with XLConnect, many of which are taken care of when using table1xls functions.

Author(s)

Assaf P. Oron <assaf.oron.at.seattlechildrens.org>

See Also

loadWorkbook, saveWorkbook

Examples

```
### Run this example in successive copy-paste batches

## Batch 1: be careful to copy and paste only the first 3 lines
# *without* the white-space below them.
cat("R will now open a new .xls worksheet for you!\n")
cat("Please enter path and filename, without extension:\n")
filestring<-readLines(n=1)


# R is waiting for you... enter the filename ... then proceed to next batch.

## Batch 2
newPath<-paste(filestring,'xls',sep='.')
blankbook<-XLwriteOpen(newPath)
```

```
# If you check to see whether the file exists - it's not there.
# The spreadsheet is only in R's memory. The next batch will save it.

## Batch 3
XLConnect::saveWorkbook(blankbook)
cat("Now there should be a blank file called",newPath, "- Check it out!\n")

## Now: writing into the file and resaving
# Make sure you close the file in case you opened it in Excel.
# We'll just write something silly there now:

## Batch 4
# Excel showed 1 blank sheet. But for R, there are 0 sheets until you create some.
XLConnect::createSheet(blankbook,"one")
XLConnect::writeWorksheet(blankbook,"Something Sillee!!!",sheet='one')
XLConnect::saveWorkbook(blankbook)

# Now it's not blank anymore - Check it out...
# You will notice XLConnect has interpreted the string
# as a data frame. Data transfer can only occur in the form of
# data frames (except some graphics).
# After closing the file run the last batch, which finally demonstrates
# what XLwriteOpen itself does (open with overwrite).
# Don't forget to close the .xls file first!


## Batch 5

blankbook2<-XLwriteOpen(newPath)
XLConnect::saveWorkbook(blankbook2)

#### Now the file is blank again - Check it out!
#### All done!
```

Index