Package 'RALSA'

October 12, 2022

Type Package

Title R Analyzer for Large-Scale Assessments

Version 1.3.0

Date 2022-07-08 14:00

Author Plamen V. Mirazchiyski [aut, cre],
INERI [aut]

Maintainer Plamen V. Mirazchiyski <plamen.mirazchiyski@ineri.org>

Description Prepare and analyze data from large-scale assessments and surveys with complex sampling and assessment design (see 'Rutkowski', 2010 <doi:10.3102/0013189X10363170>). Such studies are, for example, international assessments like 'TIMSS', 'PIRLS' and 'PISA'. A graphical interface is available for the non-technical user.The package includes functions to covert the original data from 'SPSS' into 'R' data sets keeping the user-defined missing values, merge data from different respondents and/or countries, generate variable dictionaries, modify data, produce descriptive statistics (percentages, means, percentiles, benchmarks) and multivariate statistics (correlations, linear regression, binary logistic regression). The number of supported studies and analysis types will increase in future. For a general presentation of the package, see 'Mirazchiyski', 2021a (<doi:10.1186/s40536-021-00114-4>). For detailed technical aspects of the package, see 'Mirazchiyski', 2021b (<doi:10.3390/psych3020018>).

Depends R (>= 4.0.0)

Encoding UTF-8

Imports DT, Hmisc, data.table, foreign, ggplot2, methods, openxlsx, rclipboard, readr, shiny, shinyFiles, shinydashboard, shinyjs, stringi, stringr

License GPL-2

URL http://ralsa.ineri.org/

RoxygenNote 7.2.0

NeedsCompilation no

Repository CRAN

Date/Publication 2022-07-08 12:00:07 UTC

R topics documented:

`lsa.bench` *Compute percentages of respondents reaching or surpassing certain ability cut-off scores*

Description

`lsa.bench` computes percentages of respondents reaching or surpassing certain ability cut-off scores (benchmarks/performance levels). The cut-off scores are points in the distributions of PVs defined differently in different studies and, sometimes, in different study cycles. The percentages can also be computed as cumulative percentages. There is an option to compute an average of continuous contextual/background variable.

Usage

```
lsa.bench(
  data.file,
  data.object,
  split.vars,
  PV.root.bench,
  bench.vals,
  bench.type,
  pcts.within = FALSE,
  bckg.var,
  weight.var,
  include.missing = FALSE,
  shortcut = FALSE,
  graphs = FALSE,
  save.output = TRUE,
  output.file,
```

```
  open.output = TRUE
)
```

Arguments

data.file	The file containing lsa.data object. Either this or data.object shall be specified, but not both. See details.
data.object	The object in the memory containing lsa.data object. Either this or data.file shall be specified, but not both. See details.
split.vars	Categorical variable(s) to split the results by. If no split variables are provided, the results will be for the overall countries' populations. If one or more variables are provided, the results will be split by all but the last variable and the percentages of respondents will be computed by the unique values of the last splitting variable.
PV.root.bench	The root name(s) for the set(s) of plausible values which will be used to compute the percentages of respondents reaching or surpassing certain cut-off score. See details.
bench.vals	A vector of integers representing the cut-off scores. See details.
bench.type	A character string representing how the percentages of respondents shall be computed. See details.
pcts.within	Logical value specifying if the percentages shall be computed within the groups defined by the split.vars (TRUE) or not (FALSE, default). See details.
bckg.var	Name of continuous background or contextual variable to compute the mean for. The results will be computed by all groups specified by the splitting variables and per performance group. See details.
weight.var	The name of the variable containing the weights. If no name of a weight variable is provide, the function will automatically select the default weight variable for the provided data, depending on the respondent type.
include.missing	Logical, shall the missing values of the splitting variables be included as categories to split by and all statistics produced for them? The default (FALSE) takes all cases on the splitting variables without missing values before computing any statistics. See details.
shortcut	Logical, shall the "shortcut" method for IEA TIMSS, TIMSS Advanced, TIMSS Numeracy, eTIMSS PSI, PIRLS, ePIRLS, PIRLS Literacy and RLII be applied? The default (FALSE) applies the "full" design when computing the variance components and the standard errors of the estimates.
graphs	Logical, shall graphs be produced? Default is FALSE. See details.
save.output	Logical, shall the output be saved in MS Excel file (default) or not (printed to the console or assigned to an object).
output.file	If save.output = TRUE (default), full path to the output file including the file name. If omitted, a file with a default file name "Analysis.xlsx" will be written to the working directory (getwd()). Ignored if save.output = FALSE.
open.output	Logical, shall the output be open after it has been written? The default (TRUE) opens the output in the default spreadsheet program installed on the computer. Ignored if save.output = FALSE.

Details

Either `data.file` or `data.object` shall be provided as source of data. If both of them are provided, the function will stop with an error message.

The function computes percentages of respondents which reach or surpass certain cut-off scores (benchmarks/performance levels). These percentages are computed using a set of PVs, specified in the `PV.root.bench`. Only one set of PVs can be added to `PV.root.bench` at a time. All studies (except CivED, TEDS-M, SITES, TALIS and TALIS Starting Strong Survey) have a set of PVs per content domain (e.g. in TIMSS five for overall mathematics, five for algebra, five for geometry, etc.) and cognitive domain (i.e. knowing, applying and reasoning). In some studies (say TIMSS and PIRLS) the names of the PVs in a set always start with character string and end with sequential number of the PV. For example, the names of the set of PVs for overall mathematics in TIMSS are BSMMAT01, BSMMAT02, BSMMAT03, BSMMAT04 and BSMMAT05. The root of the PVs for this set to be added to `PV.root.avg` will be "BSMMAT". The function will automatically find all the variables in this set of PVs and include them in the analysis. In other studies like OECD PISA and IEA ICCS and ICILS the sequential number of each PV is included in the middle of the name. For example, in ICCS the names of the set of PVs are PV1CIV, PV2CIV, PV3CIV, PV4CIV and PV5CIV. The root PV name has to be specified in `PV.root.bench` as "PV#CIV".

Multiple splitting variables can be added to the `split.vars`, the function will compute the percentages of respondents reaching or surpassing the cut-off scores for all formed groups and their means on the continuous variables. If no splitting variables are added, the results will be only by country.

If a continuous contextual/background variable is provided to the `bckg.var`, the average for that variable will be computed for each group formed by the splitting variables and the performance groups. Only one contextual/background variable can be added in the analysis. This argument is ignored when `bench.type = "cumulative"`.

The cut-off scores are provided as vector of integers (e.g. `c(475, 500)`) to the `bench.vals`. If no cut-off scores are provided, the function will automatically choose all benchmark values for the corresponding study and, in some cases for the data from the specific cycle. The latter applies to ICCS and PISA where the proficiency levels differ from one cycle to another.

The `bench.type` argument has two different options: `"discrete"` (default) and `"cumulative"`. Using the former will compute the percentages of respondents within the boundaries specified by the cut-off scores in `bench.vals`. Using the latter, the function will compute the percentages of respondents at or above the cut-off points in the `bench.vals`.

If the `pcts.within` if `FALSE` (default), the function will compute the percentages of respondents reaching or surpassing each of the cut-off scores defined by `bench.vals`. In this case the percentages of all respondents across the performance levels will add to 100 in each group defined by the splitting variables. On the contrary, if `pcts.within = TRUE`, the function will compute the percentages of respondents at each of the performance levels across groups defined by the splitting variables. Then the sum of percentages within a specific performance level will sum up to 100 across the groups defined by the splitting variables. For example, we can compute what is the ratio (i.e. percentages) of female and male students performing between 475 and 550 points in PIRLS – say 55 of all students performing at this level are female and 45 are male. If no split variables are provided, the percentages will be 100 for each performance level within a country. The argument is ignored if `bench.type = "cumulative"`.

If no variables are specified for `bckg.var`, the output will contain only percentages of cases in groups specified by the splitting variables and the cut-off scores.

If include.missing = FALSE (default), all cases with missing values on the splitting variables will be removed and only cases with valid values will be retained in the statistics. Note that the data from the studies can be exported in two different ways: (1) setting all user-defined missing values to NA; and (2) importing all user-defined missing values as valid ones and adding their codes in an additional attribute to each variable. If the include.missing is set to FALSE (default) and the data used is exported using option (2), the output will remove all values from the variable matching the values in its missings attribute. Otherwise, it will include them as valid values and compute statistics for them.

The shortcut argument is valid only for TIMSS, eTIMSS PSI, TIMSS Numeracy, TIMSS Advanced, PIRLS, ePIRLS, PIRLS Literacy and RLII. Previously, in computing the standard errors, these studies were using 75 replicates because one of the schools in the 75 JK zones had its weights doubled and the other one has been taken out. Since TIMSS 2015 and PIRLS 2016 the studies use 150 replicates and in each JK zone once a school has its weights doubled and once taken out, i.e. the computations are done twice for each zone. For more details see Foy & LaRoche (2016) and Foy & LaRoche (2017). If replication of the tables and figures is needed, the shortcut argument has to be changed to TRUE.

If graphs = TRUE, the function will produce graphs. If split.vars are specified, bar plots of percentages of respondents (population estimates) reaching or surpassing each benchmark level specified in bench.vals per group specified by split.vars will be produced with error bars (95% confidence) for these percentages. If bckg.var is specified, plots with 95% confidence intervals of the average for this variable will be produced. All plots are produced per country.

Value

If save.output = FALSE, a list containing the estimates and analysis information. If graphs = TRUE, the plots will be added to the list of estimates.

If save.output = TRUE (default), an MS Excel (.xlsx) file (which can be opened in any spreadsheet program), as specified with the full path in the output.file. If the argument is missing, an Excel file with the generic file name "Analysis.xlsx" will be saved in the working directory (getwd()). The workbook contains three spreadsheets. The first one ("Estimates") contains a table with the results by country and the final part of the table contains averaged results from all countries' statistics. The following columns can be found in the table, depending on the specification of the analysis:

- <Country ID> - a column containing the names of the countries in the file for which statistics are computed. The exact column header will depend on the country identifier used in the particular study.
- <Split variable 1>, <Split variable 2>... - columns containing the categories by which the statistics were split by. The exact names will depend on the variables in split.vars.
- n_Cases - the number of cases reaching or surpassing each of the benchmarks using a set of PVs. Please note that these may not be whole numbers because they are computed using each PV and then averaged.
- Sum_<Weight variable> - the estimated population number of elements per group after applying the weights. The actual name of the weight variable will depend on the weight variable used in the analysis.
- Sum_<Weight variable>_SE - the standard error of the the estimated population number of elements per group. The actual name of the weight variable will depend on the weight variable used in the analysis.

- Performance_Group - the labels for the performance groups defined by the `bench.vals`.
- Percentages_<PVs' root name> - the percentages of respondents (population estimates) reaching or surpassing each cut-off score (in case of `bench.type = "discrete"`) or the the percentages of respondents (population estimates) at or above each cut-off value (in case of `bench.type = "cumulative"`) per groups defined by the splitting variables in `split.vars`.
- Percentages_<PVs' root name>_SE - the standard errors of the percentages from above.
- Mean_<Background variable> - the average of the continuous <Background variable> specified in `bckg.var`.
- Mean_<Background variable>_SE - the standard error of the average of the continuous <Background variable> specified in `bckg.var`.
- Variance_<Background variable> - the variance for the continuous <Background variable> specified in `bckg.var`.
- Variance_<Background variable>_SE - the error of the variance for the continuous <Background variable> specified in `bckg.var`.
- SD_<Background variable> - the standard deviation for the continuous <Background variable> specified in `bckg.var`.
- SD_<Background variable>_SE - the error of the standard deviation for the continuous <Background variable> specified in `bckg.avg.var`.
- Percent_Missings_<Background variable> - the percentage of missing values for the <Background variable> specified in `bckg.var`.

The second sheet contains some additional information related to the analysis per country in columns:

- DATA - used `data.file` or `data.object`.
- STUDY - which study the data comes from.
- CYCLE - which cycle of the study the data comes from.
- WEIGHT - which weight variable was used.
- DESIGN - which resampling technique was used (JRR or BRR).
- SHORTCUT - logical, whether the shortcut method was used.
- NREPS - how many replication weights were used.
- ANALYSIS_DATE - on which date the analysis was performed.
- START_TIME - at what time the analysis started.
- END_TIME - at what time the analysis finished.
- DURATION - how long the analysis took in hours, minutes, seconds and milliseconds.

The third sheet contains the call to the function with values for all parameters as it was executed. This is useful if the analysis needs to be replicated later.

If `graphs = TRUE` there will be an additional "Graphs" sheet containing all plots.

If any warnings resulting from the computations are issued, these will be included in an additional "Warnings" sheet in the workbook as well.

References

LaRoche, S., Joncas, M., & Foy, P. (2016). Sample Design in TIMSS 2015. In M. O. Martin, I. V. S. Mullis, & M. Hooper (Eds.), *Methods and Procedures in TIMSS 2015* (pp. 3.1-3.37). Chestnut Hill, MA: TIMSS & PIRLS International Study Center. LaRoche, S., Joncas, M., & Foy, P. (2017). Sample Design in PIRLS 2016. In M. O. Martin, I. V. S. Mullis, & M. Hooper (Eds.), *Methods and Procedures in PIRLS 2016* (pp. 3.1-3.34). Chestnut Hill, MA: Lynch School of Education, Boston College.

See Also

lsa.convert.data

Examples

```
# Compute percentages of female and male students reaching or surpassing the "Intermediate"
# and "High" benchamrks in TIMSS 2015 grade 8 mathematics using data file, omit missing from
# the splitting variable (female and male as answered by the students), without shortcut, and
# open the output after the computations are done
## Not run:
lsa.bench(data.file = "C:/Data/TIMSS_2015_G8_Student_Miss_to_NA.RData", split.vars = "BSBG01",
include.missing = FALSE, PV.root.bench = "BSMMAT", bench.vals = c(475, 550),
output.file = "C:/temp/test.xlsx", open.output = TRUE)

## End(Not run)

# Repeat the analysis from above, using an object loaded in the memory, the student senate
# weight and compute the cumulative percentage, adding student feeling safe at school as a
# second splitting variable, using the shortcut method and including the missing values of
# the splitting variables
## Not run:
lsa.bench(data.object = T15_G8_student_data, split.vars = c("BSBG01", "BSBG15B"),
PV.root.bench = "BSMMAT", bench.vals = c(475, 550), weight.var = "SENWGT",
include.missing = TRUE, shortcut = TRUE, output.file = "C:/temp/test.xlsx",
open.output = TRUE)

## End(Not run)

# Compute the percentage of students reaching or surpassing the "Level 2" and "Level 3"
# in computer and information lteracy and the average of the complex background scale
# "Use of specialist applications for activities" by student sex and expected further
# level of education using ICILS 2018 data loaded in memory, include the missing values
# in the splitting variables
## Not run:
lsa.bench(data.object = ICILS_2018_student_data, split.vars = c("S_SEX", "IS2G03"),
PV.root.bench = "PV#CIL", bckg.var = "S_SPECACT", include.missing = TRUE,
output.file = "C:/temp/test.xlsx", open.output = TRUE)

## End(Not run)

# Compute the cumulative percentage of students at or above each of the (default) benchmarks
# of student overall reading achievement scores using PIRLS 2016 student data file, split the
```

```
# output by student sex, use the full design, include the missing values of the splitting
# variable (i.e. student sex), and do not open the output after the computations are finished
## Not run:
lsa.bench(data.file = "C:/Data/PIRLS_2016_Student_Miss_to_NA.RData", split.vars = "ASBG01",
PV.root.bench = "ASRREA", bench.type = "cumulative", include.missing = TRUE,
output.file = "C:/temp/test.xlsx", open.output = FALSE)

## End(Not run)
```

`lsa.bin.log.reg` *Compute binary logistic regression coefficients specified groups*

Description

`lsa.bin.log.reg` computes binary logistic regression coefficients within groups defined by one or more variables.

Usage

```
lsa.bin.log.reg(
  data.file,
  data.object,
  split.vars,
  bin.dep.var,
  bckg.indep.cont.vars,
  bckg.indep.cat.vars,
  bckg.cat.contrasts,
  bckg.ref.cats,
  PV.root.indep,
  interactions,
  standardize = FALSE,
  weight.var,
  norm.weight = FALSE,
  include.missing = FALSE,
  shortcut = FALSE,
  save.output = TRUE,
  output.file,
  open.output = TRUE
)
```

Arguments

`data.file`	The file containing `lsa.data` object. Either this or `data.object` shall be specified, but not both. See details.
`data.object`	The object in the memory containing `lsa.data` object. Either this or `data.file` shall be specified, but not both. See details.

split.vars Categorical variable(s) to split the results by. If no split variables are provided, the results will be for the overall countries' populations. If one or more variables are provided, the results will be split by all but the last variable and the percentages of respondents will be computed by the unique values of the last splitting variable.

bin.dep.var Name of a binary (i.e. just two distinct values) background or contextual variable used as a dependent variable in the model. See details.

bckg.indep.cont.vars
Names of continuous independent background or contextual variables used as predictors in the model. See details.

bckg.indep.cat.vars
Names of categorical independent background or contextual variables used as predictors in the model to compute contrasts for (see bckg.cat.contrasts and bckg.ref.cats). See details.

bckg.cat.contrasts
String vector with the same length as the length of bckg.indep.cat.vars specifying the type of contrasts to compute in case bckg.indep.cat.vars are provided. See details.

bckg.ref.cats Vector of integers with the same length as the length of bckg.indep.cat.vars and bckg.cat.contrasts specifying the reference categories for the contrasts to compute in case bckg.indep.cat.vars are provided. See details.

PV.root.indep The root names for a set of plausible values used as a independent variables in the model. See details.

interactions Interaction terms - a list containing vectors of length of two. See details.

standardize Shall the dependent and independent variables be standardized to produce beta coefficients? The default is FALSE. See details.

weight.var The name of the variable containing the weights. If no name of a weight variable is provide, the function will automatically select the default weight variable for the provided data, depending on the respondent type.

norm.weight Shall the weights be normalized before applying them, default is FALSE. See details.

include.missing
Logical, shall the missing values of the splitting variables be included as categories to split by and all statistics produced for them? The default (FALSE) takes all cases on the splitting variables without missing values before computing any statistics. See details.

shortcut Logical, shall the "shortcut" method for IEA TIMSS, TIMSS Advanced, TIMSS Numeracy, eTIMSS PSI, PIRLS, ePIRLS, PIRLS Literacy and RLII be applied? The default (FALSE) applies the "full" design when computing the variance components and the standard errors of the estimates.

save.output Logical, shall the output be saved in MS Excel file (default) or not (printed to the console or assigned to an object).

output.file If save.output = TRUE (default), full path to the output file including the file name. If omitted, a file with a default file name "Analysis.xlsx" will be written to the working directory (getwd()). Ignored if save.output = FALSE.

`open.output` Logical, shall the output be open after it has been written? The default (`TRUE`) opens the output in the default spreadsheet program installed on the computer. Ignored if `save.output = FALSE`.

Details

Either `data.file` or `data.object` shall be provided as source of data. If both of them are provided, the function will stop with an error message. The function computes binary logistic regression coefficients by the categories of the splitting variables. The percentages of respondents in each group are computed within the groups specified by the last splitting variable. If no splitting variables are added, the results will be computed only by country.

If `standardize = TRUE`, the variables will be standardized before computing any statistics to provide beta regression coefficients.

A binary (i.e. dichotomous) background/contextual variable must be provided to `bin.dep.var` (numeric or factor). If more than two categories exist in the variable, the function will exit with an error. The function automatically recodes the two categories of the `bin.dep.var` to 0 and 1 if they are not as such (e.g. as 1 and 2 as in factors). If the variable of interest has more than two distinct values (can use the `lsa.var.dict` to see them), they can be collapsed using the `lsa.recode.vars`.

Background/contextual variables passed to `bckg.indep.cont.vars` will be treated as numeric variables in the model. Variables with discrete number of categories (i.e. factors) passed to `bckg.indep.cat.vars` will be used to compute contrasts. In this case the type of contrast have to be passed to `bckg.cat.contrasts` and the number of the reference categories for each of the `bckg.indep.cat.vars`. The number of types of contrasts and the reference categories must be the same as the number of `bckg.indep.cat.vars`. The currently supported contrast coding schemes are:

- `dummy` (also called "indicator" in logistic regression) - the odds ratios show what is the probability for a positive (i.e. 1) outcome in the binary dependent variable compared to the negative outcome (i.e. 0) per category of a variable in the `bckg.indep.cat.cats` compared to the reference category of that dummy coded variable. The intercept shows the log of the odds for the reference category when all other levels are 0.
- `deviation` (also called "effect" in logistic regression) - comparing the effect of each category (except for the reference) of the deviation coded variable to the overall effect (which is the intercept).
- `simple` - the same as for the `dummy` contrast coding, except for the intercept which in this case is the overall effect.

Note that when using `standardize = TRUE`, the contrast coding of `bckg.indep.cat.vars` is not standardized. Thus, the regression coefficients may not be comparable to other software solutions for analyzing large-scale assessment data which rely on, for example, SPSS or SAS where the contrast coding of categorical variables (e.g. dummy coding) takes place by default. However, the model statistics will be identical.

Multiple continuous or categorical background variables and/or sets of plausible values can be provided to compute regression coefficients for. Please note that in this case the results will slightly differ compared to using each pair of the same background continuous variables or PVs in separate analysis. This is because the cases with the missing values are removed in advance and the more variables are provided, the more cases are likely to be removed. That is, the function support only listwisie deletion.

Computation of regression coefficients involving plausible values requires providing a root of the plausible values names in PV.root.dep and/or PV.root.indep. All studies (except CivED, TEDS-M, SITES, TALIS and TALIS Starting Strong Survey) have a set of PVs per construct (e.g. in TIMSS five for overall mathematics, five for algebra, five for geometry, etc.). In some studies (say TIMSS and PIRLS) the names of the PVs in a set always start with character string and end with sequential number of the PV. For example, the names of the set of PVs for overall mathematics in TIMSS are BSMMAT01, BSMMAT02, BSMMAT03, BSMMAT04 and BSMMAT05. The root of the PVs for this set to be added to PV.root.dep or PV.root.indep will be "BSMMAT". The function will automatically find all the variables in this set of PVs and include them in the analysis. In other studies like OECD PISA and IEA ICCS and ICILS the sequential number of each PV is included in the middle of the name. For example, in ICCS the names of the set of PVs are PV1CIV, PV2CIV, PV3CIV, PV4CIV and PV5CIV. The root PV name has to be specified in PV.root.dep or PV.root.indep as "PV#CIV". More than one set of PVs can be added in PV.root.indep.

The function can also compute two-way interaction effects between independent variables by passing a list to the interactions argument. The list must contain vectors of length two and all variables in these vectors **must also be passed as independent variables** (see the examples). Note the following:

- When an interaction is between two independent background continuous variables (i.e. both are passed to bckg.indep.cont.vars), the interaction effect will be computed between them as they are.
- When the interaction is between two categorical variables (i.e. both are passed to bckg.indep.cat.vars), the interaction effect will be computed between each possible pair of categories of the two variables, except for the reference categories.
- When the interaction is between one continuous (i.e. passed to bckg.indep.cont.vars) and one categorical (i.e. passed to bckg.indep.cat.vars), the interaction effect will be computed between the continuous variable and each category of the categorical variable, except for the reference category.
- When the interaction is between a continuous variable (i.e. passed to bckg.indep.cont.vars) and a set of PVs (i.e. passed to PV.root.indep), the interaction effect is computed between the continuous variable and each PV in the set and the results are aggregated.
- When the interaction is between a categorical variable (i.e. passed to bckg.indep.cat.vars) and a set of PVs (i.e. passed to PV.root.indep), the interaction effect is computed between each category of the categorical variable (except the reference category) and each PV in the set. The results are aggregated for each of the categories of the categorical variables and the set of PVs.
- When the interaction is between two sets of PVs (i.e. passed to PV.root.indep), the interaction effect is computed between the first PV in the first set and the first PV in the second set, the second PV in the first set and the second PV in the second set, and so on. The results are then aggregated.

If norm.weight = TRUE, the weights will be normalized before used in the model. This may be necessary in some countries in some studies extreme weights for some of the cases may result in inflated estimates due to model perfect separation. The consequence of normalizing weights is that the number of elements in the population will sum to the number of cases in the sample. Use with caution.

If include.missing = FALSE (default), all cases with missing values on the splitting variables will be removed and only cases with valid values will be retained in the statistics. Note that the data

from the studies can be exported in two different ways: (1) setting all user-defined missing values to NA; and (2) importing all user-defined missing values as valid ones and adding their codes in an additional attribute to each variable. If the include.missing is set to FALSE (default) and the data used is exported using option (2), the output will remove all values from the variable matching the values in its missings attribute. Otherwise, it will include them as valid values and compute statistics for them.

The shortcut argument is valid only for TIMSS, eTIMSS PSI, TIMSS Advanced, TIMSS Numeracy, PIRLS, ePIRLS, PIRLS Literacy and RLII. Previously, in computing the standard errors, these studies were using 75 replicates because one of the schools in the 75 JK zones had its weights doubled and the other one has been taken out. Since TIMSS 2015 and PIRLS 2016 the studies use 150 replicates and in each JK zone once a school has its weights doubled and once taken out, i.e. the computations are done twice for each zone. For more details see Foy & LaRoche (2016) and Foy & LaRoche (2017). If replication of the tables and figures is needed, the shortcut argument has to be changed to TRUE. The function provides two-tailed *t*-test and *p*-values for the regression coefficients.

Value

If save.output = FALSE, a list containing the estimates and analysis information. If save.output = TRUE (default), an MS Excel (.xlsx) file (which can be opened in any spreadsheet program), as specified with the full path in the output.file. If the argument is missing, an Excel file with the generic file name "Analysis.xlsx" will be saved in the working directory (getwd()). The workbook contains four spreadsheets. The first one ("Estimates") contains a table with the results by country and the final part of the table contains averaged results from all countries' statistics. The following columns can be found in the table, depending on the specification of the analysis:

- <Country ID> - a column containing the names of the countries in the file for which statistics are computed. The exact column header will depend on the country identifier used in the particular study.
- <Split variable 1>, <Split variable 2>... - columns containing the categories by which the statistics were split by. The exact names will depend on the variables in split.vars.
- n_Cases - the number of cases in the sample used to compute the statistics.
- Sum_<Weight variable> - the estimated population number of elements per group after applying the weights. The actual name of the weight variable will depend on the weight variable used in the analysis.
- Sum_<Weight variable>_SE - the standard error of the the estimated population number of elements per group. The actual name of the weight variable will depend on the weight variable used in the analysis.
- Percentages_<Last split variable> - the percentages of respondents (population estimates) per groups defined by the splitting variables in split.vars. The percentages will be for the last splitting variable which defines the final groups.
- Percentages_<Last split variable>_SE - the standard errors of the percentages from above.
- Variable - the variable names (background/contextual or PV root names, or contrast coded variable names).
- Coefficients - the logistic regression coefficients (intercept and slopes).

- Coefficients_SE - the standard error of the logistic regression coefficients (intercepts and slopes) for each independent variable (background/contextual or PV root names, or contrast coded variable names) in the model.
- Coefficients_SVR - the sampling variance component for the logistic regression coefficients if root PVs are specified either as dependent or independent variables.
- Coefficients_<root PV>_MVR - the measurement variance component for the logistic regression coefficients if root PVs are specified either as dependent or independent variables.
- Wald_Statistic - Wald (z) statistic.
- p_value - the p-value for the regression coefficients.
- Odds_Ratio - the odds ratios of the logistic regression.
- Odds_Ratio_SE - the standard errors for the odds ratios of the logistic regression.
- Wald_L95CI - the lower 95% model-based confidence intervals for the logistic regression coefficients.
- Wald_U95CI - the upper 95% model-based confidence intervals for the logistic regression coefficients.
- Odds_L95CI - the lower 95% model-based confidence intervals for the odds ratios.
- Odds_U95CI - the upper 95% model-based confidence intervals for the odds ratios.

When interaction terms are included, the cells with the interactions in the `Variables` column will contain the names of the two variables in each of the interaction terms, divided by colon, e.g. `ASBGSSB:ASBGHRL`.

The second sheet contains the model statistics:

- <Country ID> - a column containing the names of the countries in the file for which statistics are computed. The exact column header will depend on the country identifier used in the particular study.
- <Split variable 1>, <Split variable 2>... - columns containing the categories by which the statistics were split by. The exact names will depend on the variables in `split.vars`.
- Statistic - a column containing the Null Deviance (-2LL, no predictors in the model, just constant, also called "baseline"), Deviance (-2LL, after adding predictors, residual deviance, also called "new"), DF Null (degrees of freedom for the null deviance), DF Residual (degrees of freedom for the residual deviance), Akaike Information Criteria (AIC), Bayesian information criterion (BIC), model Chi-Square, different R-Squared statistics (Hosmer & Lemeshow - HS, Cox & Snell - CS, and Nagelkerke - N).
- Estimate - the numerical estimates for each of the above.
- Estimate_SE - the standard errors of the estimates from above.
- Estimate_SVR - the sampling variance component if PVs were included in the model.
- Estimate_MVR - the measurement variance component if PVs were included in the model.

The third sheet contains some additional information related to the analysis per country in columns:

- DATA - used `data.file` or `data.object`.
- STUDY - which study the data comes from.
- CYCLE - which cycle of the study the data comes from.

- WEIGHT - which weight variable was used.
- DESIGN - which resampling technique was used (JRR or BRR).
- SHORTCUT - logical, whether the shortcut method was used.
- NREPS - how many replication weights were used.
- ANALYSIS_DATE - on which date the analysis was performed.
- START_TIME - at what time the analysis started.
- END_TIME - at what time the analysis finished.
- DURATION - how long the analysis took in hours, minutes, seconds and milliseconds.

The fourth sheet contains the call to the function with values for all parameters as it was executed. This is useful if the analysis needs to be replicated later.

References

LaRoche, S., Joncas, M., & Foy, P. (2016). Sample Design in TIMSS 2015. In M. O. Martin, I. V. S. Mullis, & M. Hooper (Eds.), *Methods and Procedures in TIMSS 2015* (pp. 3.1-3.37). Chestnut Hill, MA: TIMSS & PIRLS International Study Center. LaRoche, S., Joncas, M., & Foy, P. (2017). Sample Design in PIRLS 2016. In M. O. Martin, I. V. S. Mullis, & M. Hooper (Eds.), *Methods and Procedures in PIRLS 2016* (pp. 3.1-3.34). Chestnut Hill, MA: Lynch School of Education, Boston College. UCLA: Statistical Consulting Group. 2020. "R LIBRARY CONTRAST CODING SYSTEMS FOR CATEGORICAL VARIABLES." *IDRE Stats - Statistical Consulting Web Resources*. Retrieved June 16, 2020 (https://stats.idre.ucla.edu/r/library/r-library-contrast-coding-systems-for-categorical-variables/). Hilbe, J. M. (2015). *Practical Guide to Logistic Regression*. CRC Press.

See Also

lsa.convert.data, , lsa.vars.dict, lsa.recode.vars, lsa.lin.reg

Examples

```
# Compute logistic regression predicting the log of the odds the students will respond
# "Agree a lot" when asked if teachers are fair (dependent variable, categorical), as a function
# of their own sense of school belonging (independent variable, continuous) using PIRLS 2016
# student data. Because the dependent variable has four categories, it needs to be recoded first
# into a dichotomous (using the \code{lsa.recode.vars}).
## Not run:
lsa.recode.vars(data.file = "C:/temp/test.RData", src.variables = "ASBG12D",
old.new = "1=2;2=2;3=1;4=1;5=3", new.variables = "ASBG12Dr",
new.labels = c("Disagree", "Agree", "Omitted or invalid"),
missings.attr = "Omitted or invalid",
variable.labels = "GEN/AGREE/TEACHERS ARE FAIR - RECODED",
out.file = "C:/temp/test.RData")

lsa.bin.log.reg(data.file = "C:/temp/test.RData", split.vars = "ASBG01",
bin.dep.var = "ASBG12Dr", bckg.indep.cont.vars = "ASBGSSB")

## End(Not run)

# Perform the same analysis from above, this time use the overall student reading achievement
```

```
# as a predictor.
## Not run:
lsa.bin.log.reg(data.object = test, split.vars = "ASBG01",
bin.dep.var = "ASBG12Dr", PV.root.indep = "ASRREA")

## End(Not run)

# Compute linear regression with interaction terms using PIRLS 2016 student data.
## Not run:
lsa.bin.log.reg(data.file = "C:/temp/test.RData", bin.dep.var = "ASBG05B",
bckg.indep.cont.vars = "ASBGSSB", bckg.indep.cat.vars = c("ASBG01", "ASBG12B"),
PV.root.indep = c("ASRREA", "ASRLIT"),
interactions = list(c("ASBG12B", "ASBGSSB"), c("ASBG01", "ASRLIT")))

## End(Not run)
```

`lsa.convert.data` *Convert Large-Scale Assessments' Datasets to .RData Format*

Description

`lsa.convert.data` converts datasets from large-scale assessments from their original formats (SPSS or ASCII text) into `.RData` files. `print` prints the properties of an `lsa.data` objects on screen. `lsa.select.countries.PISA` lets selecting PISA data from specific countries for analysis.

Usage

```
lsa.convert.data(
  inp.folder,
  PISApre15 = FALSE,
  ISO,
  missing.to.NA = FALSE,
  out.folder
)

## S3 method for class 'lsa.data'
print(x, col.nums, ...)

lsa.select.countries.PISA(data.file, data.object, cnt.names, output.file)
```

Arguments

`inp.folder`	The folder containing the IEA-like SPSS data files or ASCII text files and `.sps` import files for OECD PISA data from cycles prior to 2015. See details. If blank, the working directory (`getwd()`) is used.

PISApre15	When converting PISA files, set to TRUE if the input files are from PISA cycles prior 2015 (ASCII text format with .sps control files) or to FALSE (default) if they are in SPSS .sav format, as in the case of IEA studies and the like and OECD PISA 2015 or later. Ignored if the input folder contains IEA-like studies.
ISO	Vector containing character ISO codes of the countries' data files to convert (e.g. ISO = c("aus", "svn")). If none of the files contain the specified ISO codes in their names, the codes are ignored and a warning is shown. Ignored when converting PISA files (both for cycles prior 2015 and 2015 and later). This argument is case-insensitive, i.e. the ISO codes can be passed as lower- or upper-case. (lower or upper) as the original SPSS .sav files.
missing.to.NA	Should the user-defined missing values be recoded to NA? If TRUE, all user-defined missing values from the SPSS files (or specified in the OECD PISA import syntax files) are all imported as NA. If FALSE (default), they are converted to valid values and the missing codes are assigned to an attribute missings for each variable. See details.
out.folder	Path to the folder where the converted files will be stored. If blank, same as the inp.folder, and if the inp.folder is missing as well, this will be getwd().
x	(print only) lsa.data object.
col.nums	(print only) Which columns to print, positions by number.
...	(print only) Further arguments passed to or from other methods.
data.file	(lsa.select.countries.PISA only) Converted PISA data file to select countries' data from. Either this one or data.object must be provided, but not both. See details.
data.object	(lsa.select.countries.PISA only) PISA object in memory to filter. Either this one or data.file must be provided, but not both. See details.
cnt.names	(lsa.select.countries.PISA only) Character vector containing the names of the countries, as they exist in the data, which should stay in the PISA exported file or object in memory.
output.file	(lsa.select.countries.PISA only) Full path to the file with the filtered countries' data to be written on disk. If not provided, the PISA object will be written to memory.

Details

The lsa.convert.data function converts the originally provided data files into .RData sets. RALSA adds its own method for printing lsa.data objects on screen. The lsa.select.countries.PISA is a utility function that allows the user to select countries of interest from a converted PISA data file (or PISA object residing in memory) and remove the rest of the countries' data. This is useful when the user does not want to analyze all countries data in a PISA file.

- lsa.convert.data

  IEA studies, as well as some OECD ones and those conducted by multiple organizations, provide their data in SPSS .sav format with same or very similar structure: one file per country and type of respondent (e.g. school principal, student, teacher, etc.) per population. Cycles of OECD PISA prior to 2015, on the other hand, do not provide SPSS .sav or other binary files, but ASCII text files, accompanied with SPSS syntax (.sps) files that are used

to import the text files into SPSS. These files are per each type of respondent containing all countries' data. The `lsa.convert.data` function converts the data from either source assuring that the structure of the output `.RData` files is the same, although the structure of the input files is the different (SPSS binary files vs. ASCII text files plus import `.sps` files). The data from PISA 2015 and later, on the other hand, is provided in SPSS format (all countries in one file per type of respondent). Thus, the `PISApre15` argument needs to be specified as `TRUE` when converting data sets from PISA prior to its 2015 cycle. The default for the `PISApre15` argument is `FALSE` which means that the function expects to find IEA-like SPSS binary files per country and type of respondent in the directory in `inp.folder` or OECD PISA 2015 (or later) SPSS `.sav` files. If `PISApre15 = TRUE` and country codes are provided to `ISO`, they will be ignored because PISA files contain data from all countries together.

The files to be converted must be in a folder on their own, from a single study, single cycle and single population. In addition, if there are more than one file types per study, cycle and population, these also must be in different folders. For example, in TIMSS 2019 the grade 8 data files are main (end with "m7", electronic version of the paper administered items), bridge (end with "b7", paper administration with trend items for countries participating in previous TIMSS cycles) and Problem Solving and Inquiry (PSI) tasks (end with "z7", electronic administration only, optional for countries). These different types must be in separate folders. In case of OECD PISA prior 2015, the folder must contain both the ASCII text files and the SPSS `.sps` import syntax files. If the folder contains data sets from more than one study or cycle, the operation will break with error messages.

If the path for the `inp.folder` argument is not specified, the function will search for files in the working directory (i.e. as returned by `getwd()`). If folder path for the the `out.folder` is not specified, it will take the one from the `inp.folder` and the files will be stored there. If both the `inp.folder` and `out.folder` arguments are missing, the directory from `getwd()` will be used to search, convert and store files.

If `missing.to.NA` is set to `TRUE`, all user-defined missing values from the SPSS will be imported as `NA` which is `R`'s only kind of missing value. This will be the most often case when analyzing these data since the reason why the response is missing will be irrelevant most of the times. However, if it is needed to know why the reasons for missing responses, as when analyzing achievement items (i.e. not administered vs. omitted or not reached), the argument shall be set to `FALSE` (default for this argument) which will convert all user-defined missing values as valid ones.

- `print`

  RALSA uses its own method for printing objects of class `lsa.data` on screen. Passing just the object name to the console will print summarized information about the study's data and the first six columns of the dataset (see the Value section). If `col.nums` specifies which columns from the dataset shall be included in the output (see examples).

- `lsa.select.countries.PISA`

  `lsa.select.countries.PISA` lets the user to take a PISA dataset, either a converted file or `lsa.data` object in the memory and reduce the number of countries in it by passing the names of the countries which need to be kept as a character vector to the `cnt.names` argument. If full path (including the file name) to the resulting file is specified in the `output.file` argument, it will be written on disk. If not, the data will be written to an `lsa.object` in memory with the same name as the input file. See the examples.

Value

- `lsa.convert.data`

  `.RData` data files, containing an object with class `lsa.data`, an extension of the `data.table` class. The `data.table` object has the same name as the `.RData` file it is saved in. The object has additional attributes: study name (`study`), study cycle (`cycle`), and respondent file type (`file.type`). Each variable has its own additional attributes: its own label attached to it, if it existed in the source SPSS file. If the `missing.to.NA` was set to `TRUE`, each variable has an attribute `missings`, containing the user-defined missing values from the SPSS files.

  The object in the `.RData` file is keyed on the country ID variable.

- `print`

  Prints the information of an `lsa.data` object (study, cycle, respondent type, number of countries, key – country ID, and if the variables have user-defined missing values) and a preview of the data. The default preview (when no `col.nums`) are specified will include the first six columns.

- `lsa.select.countries.PISA`

  Writes a file containing an `lsa.object` with the data for the countries passed to the `cnt.names` argument, if the `output.file` argument is used. If the `output.file` argument is not used, the `lsa.object` will be written to the memory with the same name as the file name in `inp.file`.

Note

When downloading the `.sps` files (ASCII text and control `.sps`) for OECD PISA files prior to the 2015 cycle (say http://www.oecd.org/pisa/pisaproducts/pisa2009database-downloadabledata.htm), save them **without changing their names and without modifying the file contents**. The function will look for the files as they were named originally.

Different studies and cycles define the "I don't know" (or similar) category of discrete variables in different ways - either as a valid or missing value. The `lsa.convert.data` function sets all such or similar codes to missing value. If this has to be changed, the `lsa.recode.vars` can be used as well (also see `lsa.vars.dict`).

See Also

`lsa.vars.dict`, `lsa.recode.vars`

Examples

```
# Convert all IEA-like SPSS files in the working directory, setting all user-defined missing
# values to \code{NA}
## Not run:
lsa.convert.data(missing.to.NA = TRUE)

## End(Not run)


# Convert IEA TIMSS 2011 grade 8 data from Australia and Slovenia, keeping all user-defined
# missing values as valid ones specifying custom input and output directories
## Not run:
lsa.convert.data(inp.folder = "C:/TIMSS_2011_G8", ISO = c("aus", "svn"), missing.to.NA = FALSE,
```

```
out.folder = "C:/Data")

## End(Not run)

# Convert OECD PISA 2009 files converting all user-defined missing values to \code{NA}
# using custom input and output directories
## Not run:
lsa.convert.data(inp.folder = "/media/PISA_2009", PISApre15 = TRUE, missing.to.NA = TRUE,
out.folder = "/tmp")

## End(Not run)

# Print 20th to 25th column in PISA 2018 student questionnaire dataset loaded into memory
## Not run:
print(x = cy07_msu_stu_qqq, col.nums = 20:25)

## End(Not run)

# Select data from Albania and Slovenia from PISA 2018 student questionnaire dataset
# and save it under the same file name in a different folder
## Not run:
lsa.select.countries.PISA(data.file = "C:/PISA/cy07_msu_stu_qqq.RData",
cnt.names = c("Albania", "Slovenia"),
output.file = "C:/PISA/Reduced/cy07_msu_stu_qqq.RData")

## End(Not run)
```

`lsa.corr` *Compute correlations between variables within specified groups*

Description

`lsa.corr` computes correlation coefficients between variables within groups defined by one or more variables.

Usage

```
lsa.corr(
  data.file,
  data.object,
  split.vars,
  bckg.corr.vars,
  PV.root.corr,
  corr.type,
  weight.var,
  include.missing = FALSE,
  shortcut = FALSE,
```

```
  save.output = TRUE,
  output.file,
  open.output = TRUE
)
```

Arguments

data.file	The file containing lsa.data object. Either this or data.object shall be specified, but not both. See details.
data.object	The object in the memory containing lsa.data object. Either this or data.file shall be specified, but not both. See details.
split.vars	Categorical variable(s) to split the results by. If no split variables are provided, the results will be for the overall countries' populations. If one or more variables are provided, the results will be split by all but the last variable and the percentages of respondents will be computed by the unique values of the last splitting variable.
bckg.corr.vars	Names of continuous background or contextual variables to compute the correlation coefficients for. The results will be computed by all groups specified by the splitting variables. See details.
PV.root.corr	The root names for the sets of plausible values to compute the correlation coefficients for. See details.
corr.type	String of length one, specifying the type of the correlations to compute, either "Pearson" (default) or "Spearman".
weight.var	The name of the variable containing the weights. If no name of a weight variable is provide, the function will automatically select the default weight variable for the provided data, depending on the respondent type.
include.missing	Logical, shall the missing values of the splitting variables be included as categories to split by and all statistics produced for them? The default (FALSE) takes all cases on the splitting variables without missing values before computing any statistics. See details.
shortcut	Logical, shall the "shortcut" method for IEA TIMSS, TIMSS Advanced, TIMSS Numeracy, eTIMSS PSI, PIRLS, ePIRLS, PIRLS Literacy and RLII be applied? The default (FALSE) applies the "full" design when computing the variance components and the standard errors of the estimates.
save.output	Logical, shall the output be saved in MS Excel file (default) or not (printed to the console or assigned to an object).
output.file	If save.output = TRUE (default), full path to the output file including the file name. If omitted, a file with a default file name "Analysis.xlsx" will be written to the working directory (getwd()). Ignored if save.output = FALSE.
open.output	Logical, shall the output be open after it has been written? The default (TRUE) opens the output in the default spreadsheet program installed on the computer. Ignored if save.output = FALSE.

Details

Either `data.file` or `data.object` shall be provided as source of data. If both of them are provided, the function will stop with an error message.

The function computes correlation coefficients by the categories of the splitting variables. The percentages of respondents in each group are computed within the groups specified by the last splitting variable. If no splitting variables are added, the results will be computed only by country.

Multiple continuous background variables and/or sets of plausible values can be provided to compute correlation coefficients for. Please note that in this case the results will slightly differ compared to using each pair of the same background continuous variables or PVs in separate analysis. This is because the cases with the missing values are removed in advance and the more variables are provided to compute correlations for, the more cases are likely to be removed. That is, the function support only listwisie deletion.

Computation of correlation coefficients involving plausible values requires providing a root of the plausible values names in `PV.root.corr`. All studies (except CivED, TEDS-M, SITES, TALIS and TALIS Starting Strong Survey) have a set of PVs per construct (e.g. in TIMSS five for overall mathematics, five for algebra, five for geometry, etc.). In some studies (say TIMSS and PIRLS) the names of the PVs in a set always start with character string and end with sequential number of the PV. For example, the names of the set of PVs for overall mathematics in TIMSS are BSMMAT01, BSMMAT02, BSMMAT03, BSMMAT04 and BSMMAT05. The root of the PVs for this set to be added to `PV.root.corr` will be "BSMMAT". The function will automatically find all the variables in this set of PVs and include them in the analysis. In other studies like OECD PISA and IEA ICCS and ICILS the sequential number of each PV is included in the middle of the name. For example, in ICCS the names of the set of PVs are PV1CIV, PV2CIV, PV3CIV, PV4CIV and PV5CIV. The root PV name has to be specified in `PV.root.corr` as "PV#CIV". More than one set of PVs can be added. Note, however, that providing multiple continuous variables for the `bckg.avg.corr` argument and multiple PV roots for the `PV.root.corr` argument will affect the results for the correlation coefficients for the PVs because the cases with missing on `bckg.corr.vars` will be removed and this will also affect the results from the PVs (i.e. listwise deletion). On the other hand, using only sets of PVs to correlate should not affect the results on any PV estimates because PVs shall not have any missing values.

A sufficient number of variable names (background/contextual) or PV roots have to be provided - either two background variables, or two PV roots, or mixture of them with total length of two (i.e. one background/contextual variable and one PV root).

If `include.missing` = `FALSE` (default), all cases with missing values on the splitting variables will be removed and only cases with valid values will be retained in the statistics. Note that the data from the studies can be exported in two different ways: (1) setting all user-defined missing values to `NA`; and (2) importing all user-defined missing values as valid ones and adding their codes in an additional attribute to each variable. If the `include.missing` is set to `FALSE` (default) and the data used is exported using option (2), the output will remove all values from the variable matching the values in its `missings` attribute. Otherwise, it will include them as valid values and compute statistics for them.

The `shortcut` argument is valid only for TIMSS, eTIMSS PSI, TIMSS Advanced, TIMSS Numeracy, PIRLS, ePIRLS, PIRLS Literacy and RLII. Previously, in computing the standard errors, these studies were using 75 replicates because one of the schools in the 75 JK zones had its weights doubled and the other one has been taken out. Since TIMSS 2015 and PIRLS 2016 the studies use 150 replicates and in each JK zone once a school has its weights doubled and once taken out, i.e.

the computations are done twice for each zone. For more details see Foy & LaRoche (2016) and Foy & LaRoche (2017). If replication of the tables and figures is needed, the `shortcut` argument has to be changed to `TRUE`. The function provides two-tailed *t*-test and *p*-values for the correlation coefficients.

Value

If `save.output = FALSE`, a list containing the estimates and analysis information. If `save.output = TRUE` (default), an MS Excel (`.xlsx`) file (which can be opened in any spreadsheet program), as specified with the full path in the `output.file`. If the argument is missing, an Excel file with the generic file name "Analysis.xlsx" will be saved in the working directory (`getwd()`). The workbook contains three spreadsheets. The first one ("Estimates") contains a table with the results by country and the final part of the table contains averaged results from all countries' statistics. The results are presented as a correlation matrices by the splitting variables. The following columns can be found in the table, depending on the specification of the analysis:

- <Country ID> - a column containing the names of the countries in the file for which statistics are computed. The exact column header will depend on the country identifier used in the particular study.
- <Split variable 1>, <Split variable 2>... - columns containing the categories by which the statistics were split by. The exact names will depend on the variables in `split.vars`.
- n_Cases - the number of cases in the sample used to compute the statistics.
- Sum_<Weight variable> - the estimated population number of elements per group after applying the weights. The actual name of the weight variable will depend on the weight variable used in the analysis.
- Sum_<Weight variable>_SE - the standard error of the the estimated population number of elements per group. The actual name of the weight variable will depend on the weight variable used in the analysis.
- Percentages_<Last split variable> - the percentages of respondents (population estimates) per groups defined by the splitting variables in `split.vars`. The percentages will be for the last splitting variable which defines the final groups.
- Percentages_<Last split variable>_SE - the standard errors of the percentages from above.
- Variable - the variable names (background/contextual or PV root names) to be matched against the rows of the following columns, forming the correlation matrices together.
- Correlation_<Background variable> - the correlation coefficient of each continuous <Background variable> specified in `bckg.corr.vars` against itself and each of the variables in the column "Variable". There will be one column with correlation coefficient estimate for each variable specified in `bckg.corr.vars` and/or set of PVs specified in `PV.root.corr`.
- Correlation_<Background variable>_SE - the standard error of the correlation of each continuous <Background variable> specified in `bckg.corr.vars`. There will be one column with the SE of the correlation coefficient estimate for each variable specified in `bckg.corr.vars` and/or set of PVs specified in `PV.root.corr`.
- Correlation_<root PV> - the correlation coefficient of each set of PVs specified as PV root name in `PV.root.corr` against itself and each of the variables in the column "Variable". There will be one column with correlation coefficient estimate for each set of PVs specified in `PV.root.corr` and each other set of PVs specified in `PV.root.corr` and/or each continuous background variable specified in `bckg.corr.vars`.

- Correlation_<root PV>_SE - the standard error of the correlation of each set of PVs specified as PV root name in `PV.root.corr`. There will be one column with the SE of the correlation coefficient estimate for each set of root PVs specified in `PV.root.corr` and another set of PVs specified in `PV.root.corr` and/or each continuous background variable specified in `bckg.corr.vars`.
- Correlation_<root PV>_SVR - the sampling variance component for the correlation of the PVs with the same <root PV> specified in `PV.root.corr`. There will be one column with the sampling variance component for the correlation coefficient estimate for each set of PVs specified in `PV.root.corr` with the other variables (other sets of PVs or background/contextual variables).
- Mean_<root PV>_MVR - the measurement variance component for the correlation of the PVs with the same <root PV> specified in `PV.root.corr`. There will be one column with the measurement variance component for the correlation coefficient estimate for each set of PVs specified in `PV.root.corr` with the other variables (other sets of PVs or background/contextual variables).
- Correlation_<Background variable>_SVR - the sampling variance component for the correlation of the particular background variable with a set of PVs specified in `PV.root.corr` it is correlated with. There will be one column with the sampling variance component for the average estimate for each background/contextual variable correlated with a set of PVs specified in `PV.root.corr`.
- Correlation_<Background variable>_MVR - the measurement variance component for the correlation of the particular background variable PVs with a set of PVs specified in `PV.root.corr`. There will be one column with the measurement variance component for the correlation coefficient estimate for each background/contextual variable correlated with a set of PVs specified in `PV.root.corr`.
- t_<root PV> - the *t*-test value for the correlation coefficients of a set of PVs when correlating them with other variables (background/contextual or other sets of PVs).
- t_<Background variable> - the *t*-test value for the correlation coefficients of background variables when correlating them with other variables (background/contextual or other sets of PVs).
- p_<root PV> - the *p*-value for the correlation coefficients of a set of PVs when correlating them with other variables (background/contextual or other sets of PVs).
- p_<Background variable> - the *p*-value value for the correlation coefficients of background variables when correlating them with other variables (background/contextual or other sets of PVs).

The second sheet contains some additional information related to the analysis per country in columns:

- DATA - used `data.file` or `data.object`.
- STUDY - which study the data comes from.
- CYCLE - which cycle of the study the data comes from.
- WEIGHT - which weight variable was used.
- DESIGN - which resampling technique was used (JRR or BRR).
- SHORTCUT - logical, whether the shortcut method was used.
- NREPS - how many replication weights were used.
- ANALYSIS_DATE - on which date the analysis was performed.

- START_TIME - at what time the analysis started.
- END_TIME - at what time the analysis finished.
- DURATION - how long the analysis took in hours, minutes, seconds and milliseconds.

The third sheet contains the call to the function with values for all parameters as it was executed. This is useful if the analysis needs to be replicated later.

References

LaRoche, S., Joncas, M., & Foy, P. (2016). Sample Design in TIMSS 2015. In M. O. Martin, I. V. S. Mullis, & M. Hooper (Eds.), *Methods and Procedures in TIMSS 2015* (pp. 3.1-3.37). Chestnut Hill, MA: TIMSS & PIRLS International Study Center. LaRoche, S., Joncas, M., & Foy, P. (2017). Sample Design in PIRLS 2016. In M. O. Martin, I. V. S. Mullis, & M. Hooper (Eds.), *Methods and Procedures in PIRLS 2016* (pp. 3.1-3.34). Chestnut Hill, MA: Lynch School of Education, Boston College.

See Also

lsa.convert.data

Examples

```
# Compute correlations between the complex student background scales
# "Home Educational Resources/SCL", "Students Sense of School Belonging/SCL" and
# "Students Value Mathematics/SCL" by sex of students in TIMSS 2015 grade 8
# using data file, omit missing from the splitting variable (female and male
# as answered by the students), without shortcut, and open the output after the
# computations are done
## Not run:
lsa.corr(data.file = "C:/Data/TIMSS_2015_G8_Student_Miss_to_NA.RData", split.vars = "BSBG01",
bckg.corr.vars = c("BSBGHER", "BSBGSSB", "BSBGSVM"), include.missing = FALSE,
output.file = "C:/temp/test.xlsx", open.output = TRUE)

## End(Not run)

# Compute correlations between the complex student background scales
# "Home Educational Resources/SCL" and "Students Sense of School Belonging/SCL"
# and the plausible values in overall mathematics and overall science by student
# sex and frequency of using computer or tablet at home using TIMSS 2015 grade 8
# data loaded in memory, using the shortcut, include the missing values in the
# splitting variables, and use the senate weights
## Not run:
lsa.corr(data.object = T15_G8_student_data, split.vars = c("BSBG01", "BSBG13A"),
bckg.corr.vars = c("BSBGHER", "BSBGSSB"), PV.root.corr = c("BSMMAT", "BSSSCI"),
weight.var = "SENWGT", include.missing = FALSE, shortcut = TRUE,
output.file = "C:/temp/test.xlsx", open.output = TRUE)

## End(Not run)

# Compute the correlations between student overall reading achievement, overall mathematics
# scores (i.e. using a set of PVs) and student family wealth, using PISA 2018 student data
```

```
# loaded as object in the memory, by country, and do not open the output after the computations
# are finished
## Not run:
lsa.corr(data.object = CY07_MSU_STU_QQQ, bckg.corr.vars = "WEALTH",
PV.root.corr = c("PV#MATH", "PV#READ"), include.missing = TRUE,
output.file = "C:/temp/test.xlsx", open.output = FALSE)

## End(Not run)
```

`lsa.crosstabs` *Compute crosstabulations and design corrected chi-square statistics*

Description

`lsa.crosstabs` computes two-way tables and estimates the Rao-Scott first- and second-order adjusted chi-square.

Usage

```
lsa.crosstabs(
  data.file,
  data.object,
  split.vars,
  bckg.row.var,
  bckg.col.var,
  expected.cnts = TRUE,
  row.pcts = FALSE,
  column.pcts = FALSE,
  total.pcts = FALSE,
  weight.var,
  include.missing = FALSE,
  shortcut = FALSE,
  graphs = FALSE,
  save.output = TRUE,
  output.file,
  open.output = TRUE
)
```

Arguments

`data.file`	A file containing `lsa.data` object. Either this or `data.object` shall be specified, but not both. See details.
`data.object`	An object in the memory containing `lsa.data`. Either this or `data.file` shall be specified, but not both. See details.

`split.vars` Categorical variable(s) to split the results by. If no split variables are provided, the results will be for the overall countries' populations. If one or more variables are provided, the results will be split by all but the last variable and the percentages of respondents will be computed by the unique values of the last splitting variable.

`bckg.row.var` Name of the categorical background row variable. The results will be computed by all groups specified by the splitting variables. See details.

`bckg.col.var` Name of the categorical background column variable. The results will be computed by all groups specified by the splitting variables. See details.

`expected.cnts` Logical, shall the expected counts be computed as well? The default (`TRUE`) will compute the expected counts. If `FALSE`, only the observed counts will be included in the output.

`row.pcts` Logical, shall row percentages be computed? The default (`FALSE`) will skip the computation of the row percentages.

`column.pcts` Logical, shall column percentages be computed? The default (`FALSE`) will skip the computation of the column percentages.

`total.pcts` Logical, shall percentages of total be computed? The default (`FALSE`) will skip the computation of the total percentages.

`weight.var` The name of the variable containing the weights. If no name of a weight variable is provided, the function will automatically select the default weight variable for the provided data, depending on the respondent type.

`include.missing` Logical, shall the missing values of the splitting variables be included as categories to split by and all statistics produced for them? The default (`FALSE`) takes all cases on the splitting variables without missing values before computing any statistics. See details.

`shortcut` Logical, shall the "shortcut" method for IEA TIMSS, TIMSS Advanced, TIMSS Numeracy, eTIMSS, PIRLS, ePIRLS, PIRLS Literacy and RLII be applied? The default (`FALSE`) applies the "full" design when computing the variance components and the standard errors of the estimates.

`graphs` Logical, shall graphs be produced? Default is `FALSE`. See details.

`save.output` Logical, shall the output be saved in MS Excel file (default) or not (printed to the console or assigned to an object).

`output.file` If `save.output = TRUE` (default), full path to the output file including the file name. If omitted, a file with a default file name "Analysis.xlsx" will be written to the working directory (`getwd()`). Ignored if `save.output = FALSE`.

`open.output` Logical, shall the output be open after it has been written? The default (`TRUE`) opens the output in the default spreadsheet program installed on the computer. Ignored if `save.output = FALSE`.

Details

The function computes two-way tables between two categorical variables and estimates the Rao-Scott first- and second-order design correction of the chi-square statistics. All statistics are computed within the groups specified by the last splitting variable. If no splitting variables are added, the results will be computed only by country.

Either data.file or data.object shall be provided as source of data. If both of them are provided, the function will stop with an error message.

Only two (row and column) categorical variables can be provided. The function always computes the observed counts. If requested, the expected counts, row percentages, column percentages and total percentages can be computed as well.

If include.missing = FALSE (default), all cases with missing values on the splitting variables will be removed and only cases with valid values will be retained in the statistics. Note that the data from the studies can be exported in two different ways: (1) setting all user-defined missing values to NA; and (2) importing all user-defined missing values as valid ones and adding their codes in an additional attribute to each variable. If the include.missing is set to FALSE (default) and the data used is exported using option (2), the output will remove all values from the variable matching the values in its missings attribute. Otherwise, it will include them as valid values and compute statistics for them.

The shortcut argument is valid only for TIMSS, eTIMSS, TIMSS Advanced, TIMSS Numeracy, PIRLS, ePIRLS, PIRLS Literacy and RLII. Previously, in computing the standard errors, these studies were using 75 replicates because one of the schools in the 75 JK zones had its weights doubled and the other one has been taken out. Since TIMSS 2015 and PIRLS 2016 the studies use 150 replicates and in each JK zone once a school has its weights doubled and once taken out, i.e. the computations are done twice for each zone. For more details see Foy & LaRoche (2016) and Foy & LaRoche (2017).

If graphs = TRUE, the function will produce graphs, heatmaps of counts per combination of bckg.row.var and bckg.col.var category (population estimates) per group defined by the split.vars will be produced. All plots are produced per country. If no split.vars at the end there will be a heatmap for all countries together.

The function also computes chi-square statistics with Rao-Scott first- and second-order design corrections because of the clustering in complex survey designs. For more details, see Rao & Scott (1984, 1987) and Skinner (2019).

Value

If save.output = FALSE, a list containing the estimates and analysis information. If graphs = TRUE, the plots will be added to the list of estimates.

If save.output = TRUE (default), an MS Excel (.xlsx) file (which can be opened in any spreadsheet program), as specified with the full path in the output.file. If the argument is missing, an Excel file with the generic file name "Analysis.xlsx" will be saved in the working directory (getwd()). The workbook contains four spreadsheets. The first one ("Estimates") contains a table with the results by country and the final part of the table contains averaged results from all countries' statistics. The following columns can be found in the table, depending on the specification of the analysis:

- <Country ID> - a column containing the names of the countries in the file for which statistics are computed. The exact column header will depend on the country identifier used in the particular study.
- <Split variable 1>, <Split variable 2>... - columns containing the categories by which the statistics were split by. The exact names will depend on the variables in split.vars.
- n_Cases - the number of cases in the sample used to compute the statistics for each split combination defined by the split.vars, if any, and the bckg.row.var.

- Sum_<Weight variable> - the estimated population number of elements per group after applying the weights. The actual name of the weight variable will depend on the weight variable used in the analysis.
- Sum_<Weight variable>_SE - the standard error of the the estimated population number of elements per group. The actual name of the weight variable will depend on the weight variable used in the analysis.
- Percentages_<Row variable> - the percentages of respondents (population estimates) per groups defined by the splitting variables in `split.vars`, if any, and the row variable in `bckg.row.var`. The percentages will be for the combination of categories in the last splitting variable and the row variable which define the final groups.
- Percentages_<Row variable>_SE - the standard errors of the percentages from above.
- Type - the type of computed values depending on the logical values passed to the `expected.cnts`, `row.pcts`, `column.pcts`, and `total.pcts` arguments: "Observed count", "Expected count", "Row percent", "Column percent", and "Percent of total".
- <Column variable name Category 1>, <Column variable name Category 1>,... - the estimated values for all combinations between the row and column variables passed to `bckg.row.var` and `bckg.col.var`. There will be one column for each category of the column variable.
- <Column variable name Category 1, 2,... n>_SE - the standard errors of the estimated values from the above.
- Total - the grand totals for each of the estimated value types ("Observed count", "Expected count", "Row percent", "Column percent", and "Percent of total") depending on the logical values (`TRUE`, `FALSE`) passed to the `expected.cnts`, `row.pcts`, `column.pcts`, and `total.pcts` arguments.
- Total_SE - the standard errors of the estimated values from the above.

The second sheet contains some additional information related to the analysis per country in the following columns:

- <Country ID> - a column containing the names of the countries in the file for which statistics are computed. The exact column header will depend on the country identifier used in the particular study.
- <Split variable 1>, <Split variable 2>... - columns containing the categories by which the statistics were split by. The exact names will depend on the variables in `split.vars`.
- Statistics - contains the names of the different statistics types: chi-squares, degrees of freedom (sample and design), and p-values.
- Value - the estimated values for the statistics from above.

The third sheet contains some additional information related to the analysis per country in the following columns:

- DATA - used `data.file` or `data.object`.
- STUDY - which study the data comes from.
- CYCLE - which cycle of the study the data comes from.
- WEIGHT - which weight variable was used.
- DESIGN - which resampling technique was used (JRR or BRR).
- SHORTCUT - logical, whether the shortcut method was used.

- NREPS - how many replication weights were used.
- ANALYSIS_DATE - on which date the analysis was performed.
- START_TIME - at what time the analysis started.
- END_TIME - at what time the analysis finished.
- DURATION - how long the analysis took in hours, minutes, seconds and milliseconds.

The fourth sheet contains the call to the function with values for all parameters as it was executed. This is useful if the analysis needs to be replicated later.

If `graphs = TRUE` there will be an additional "Graphs" sheet containing all plots.

If any warnings resulting from the computations are issued, these will be included in an additional "Warnings" sheet in the workbook as well.

References

LaRoche, S., Joncas, M., & Foy, P. (2016). Sample Design in TIMSS 2015. In M. O. Martin, I. V. S. Mullis, & M. Hooper (Eds.), *Methods and Procedures in TIMSS 2015* (pp. 3.1-3.37). Chestnut Hill, MA: TIMSS & PIRLS International Study Center. LaRoche, S., Joncas, M., & Foy, P. (2017). Sample Design in PIRLS 2016. In M. O. Martin, I. V. S. Mullis, & M. Hooper (Eds.), *Methods and Procedures in PIRLS 2016* (pp. 3.1-3.34). Chestnut Hill, MA: Lynch School of Education, Boston College. Rao, J. N. K., & Scott, A. J. (1984). On Chi-Squared Tests for Multiway Contingency Tables with Cell Proportions Estimated from Survey Data. *The Annals of Statistics*, *12*(1). https://doi.org/10.1214/aos/1176346391 Rao, J. N. K., & Scott, A. J. (1987). On Simple Adjustments to Chi-Square Tests with Sample Survey Data. *The Annals of Statistics*, *15*(1), 385-397. Skinner, C. (2019). Analysis of Categorical Data for Complex Surveys. *International Statistical Review*, *87*(S1), S64-S78. https://doi.org/10.1111/insr.12285

See Also

`lsa.convert.data`

Examples

```
# Compute two-way table between student sex and how much they proud they are proud to go to
# school using PIRLS 2016 student data.
## Not run:
lsa.crosstabs(data.file = "C:/Data/PIRLS_2016_G8_Student_Miss_to_NA.RData",
bckg.row.var = "ITSEX", bckg.col.var = "ASBG12E")

## End(Not run)

# Same as the above, this time also computing the expected counts, row percentages, column
# percentages, percentages of total.
## Not run:
lsa.crosstabs(data.file = "C:/Data/PIRLS_2016_G8_Student_Miss_to_NA.RData",
bckg.row.var = "ITSEX", bckg.col.var = "ASBG12E", expected.cnts = TRUE,
row.pcts = TRUE, column.pcts = TRUE, total.pcts = TRUE)

## End(Not run)
```

lsa.data.diag *Produce data diagnostic tables*

Description

lsa.data.diag is a utility function which produces diagnostic tables for variables in an lsa.data object available in the memory or saved in an .RData file. The function can be used with regular data.frame or data.table, i.e. it is applicable not only to large-scale assessment data.

Usage

```
lsa.data.diag(
  data.file,
  data.object,
  split.vars,
  variables,
  weight.var,
  cont.freq = FALSE,
  include.missing = FALSE,
  output.file,
  open.output = TRUE,
  ...
)
```

Arguments

data.file	The file containing lsa.data object. Either this or data.object shall be specified, but not both. See details.
data.object	The object in the memory containing lsa.data object. Either this or data.file shall be specified, but not both. See details.
split.vars	Variable(s) to split the results by. If no split variables are provided, the results will be computed on country level. (if weights are used) or samples (if no weights are used). See details.
variables	Names of the variables to compute statistics for. If the variables are factors or character, frequencies will be computed, and if they are numeric, descriptives will be computed, unless cont.freq = TRUE. See details.
weight.var	The name of the variable containing the weights, if weighted statistics are needed. If no name of a weight variable is provided, the function will automatically select the default weight variable for the provided lsa.data, depending on the respondent type. "none" is for unweighted statistics. See details.
cont.freq	Shall the values of the numeric categories be treated as categorical to compute frequencies for? See details.
include.missing	Shall the NA and user-defined missing values (if available) be included as splitting categories for the variables in split.vars? The default is FALSE. See details.

output.file	Full path to the output file including the file name. If omitted, a file with a default file name "Analysis.xlsx" will be written to the working directory (getwd()).
open.output	Logical, shall the output be open after it has been written? The default (TRUE) opens the output in the default spreadsheet program installed on the computer.
...	Further arguments.

Details

The function produces data diagnostic tables for variables in an lsa.data set by the categories of splitting variables. The function is also applicable to data sets which are not of class lsa.data, a regular data.frame or a data.table are accepted as well. If the data is of class lsa.data and no split.vars variables are provided, the results will be automatically split and computed by country. The country ID variable will be added automatically, there is no need to specify it explicitly in split.vars. If the data is not of class lsa.data and no split.vars variables are provided, the results will be computed without any split.

Either data.file or data.object shall be provided as source of data. If both of them are provided, the function will stop with an error message.

If variables are provided for the split.vars argument and include.missing = TRUE, the function will automatically add the NA and user-defined missing values from the missings attribute (if available) of the split.vars variables to the categories to split by and will compute statistics for the provided variables for these as well. See the documentation on lsa.convert.data for more details on the conversion of data with and without user-defined missing values.

If no variable names are provided to variables all variables available in the data set will be added automatically, except the weighting and splitting variables, and statistics for all of them will be computed.

If the variables provided to the variables argument are factor or character, the function will compute frequencies, percentages, valid percentages, and cumulative percentages. If the variables are numeric, the computed statistics will include the total number of cases, range, minimum, maximum, mean, variance, and standard deviation. If cont.freq = TRUE, then the numeric variables will be treated as factors.

If the data set is of class lsa.data and no weight variable is provided, the computed statistics will be automatically weighted by the default weight for the respondents' data in the object. If the name of a weight variable is provided, the statistics will be weighted by it. If weight.var = "none", the computed statistics will be unweighted. If the data is not of class lsa.data and no weight.var is provided, the computed statistics will be unweighted. If a weight variable is provided, the computed statistics will be weighted by it.

Value

A MS Excel (.xlsx) file (which can be opened in any spreadsheet program), as specified with the full path in the output.file. If the argument is missing, an Excel file with the generic file name "Analysis.xlsx" will be saved in the working directory (getwd()). The first sheet in the workbook is an Index sheet. All other sheets contain the computed statistics for the variables, one sheet per variable. The Index sheet contains columns with the names of the variables for which statistics are computed and their labels, if available. The names are clickable links, if clicked, they switch to the corresponding sheet with statistics for the corresponding variable. If the data is of class lsa.data, the Index sheet also contains information with the study name, cycle, respondent type and used

weight. If the data is not of class `lsa.data`, the `Index` sheet contains information only which weight was used. Each sheet with statistics for a variable contains a clickable link to go back to the `Index` sheet, the variable name and label (if any), and the table with statistics for that variable.

Note

This function is intended only as utility function for diagnostic purposes, to inspect the variables prior to performing an actual analysis. It is **not** intended for actual analysis of large-scale assessments' data. Reporting statistics from it can and will lead to biased and erroneous conclusions.

See Also

`lsa.convert.data`

Examples

```
# Merge PIRLS 2016 school principal data for all countries
## Not run:
lsa.merge.data(inp.folder = "C:/Data", file.types = list(acg = NULL),
out.file = "C:/Merged/Merged.RData")

## End(Not run)

# Produce diagnostic tables for some factor (categorical) and numeric (continuous) variables
# by country
## Not run:
lsa.data.diag(data.file = "C:/Merged/Merged.RData",
variables = c("ACBG05A", "ACBG04", "ACBGELS", "ACBGRRS"),
output.file = "C:/temp/test.xlsx", open.output = TRUE)

## End(Not run)

# Repeat the above, splitting the results by country and percentage of students at school
# coming from economically affluent homes ("ACBG03B")
## Not run:
lsa.data.diag(data.file = "C:/Merged/Merged.RData",
split.vars = "ACBG03B", variables = c("ACBG05A", "ACBG04", "ACBGELS", "ACBGRRS"),
output.file = "C:/temp/test.xlsx", open.output = TRUE)

## End(Not run)

# Repeat the above, this time treating the numeric variables ("ACBGELS" and "ACBGRRS")
# as categorical
## Not run:
lsa.data.diag(data.file = "C:/Merged/Merged.RData",
split.vars = "ACBG03B, include.missing = TRUE,
variables = c("ACBG05A", "ACBG04", "ACBGELS", "ACBGRRS"),
output.file = "C:/temp/test.xlsx", open.output = TRUE)

## End(Not run)

# Produce diag for all variables in the data set by country and percentage of students
```

```
# coming from economically affluent homes ("ASBG03B")
## Not run:
lsa.data.diag(data.file = "C:/Merged/Merged.RData",
split.vars = "ACBG03B, output.file = "C:/temp/test.xlsx",
open.output = TRUE)

## End(Not run)
```

`lsa.lin.reg` *Compute linear regression coefficients specified groups*

Description

`lsa.lin.reg` computes linear regression coefficients within groups defined by one or more variables.

Usage

```
lsa.lin.reg(
  data.file,
  data.object,
  split.vars,
  bckg.dep.var,
  PV.root.dep,
  bckg.indep.cont.vars,
  bckg.indep.cat.vars,
  bckg.cat.contrasts,
  bckg.ref.cats,
  PV.root.indep,
  interactions,
  standardize = FALSE,
  weight.var,
  include.missing = FALSE,
  shortcut = FALSE,
  save.output = TRUE,
  output.file,
  open.output = TRUE
)
```

Arguments

`data.file`	The file containing `lsa.data` object. Either this or `data.object` shall be specified, but not both. See details.
`data.object`	The object in the memory containing `lsa.data` object. Either this or `data.file` shall be specified, but not both. See details.

split.vars — Categorical variable(s) to split the results by. If no split variables are provided, the results will be for the overall countries' populations. If one or more variables are provided, the results will be split by all but the last variable and the percentages of respondents will be computed by the unique values of the last splitting variable.

bckg.dep.var — Name of a continuous background or contextual variable used as a dependent variable in the model. See details.

PV.root.dep — The root name for a set of plausible values used as a dependent variable in the model. See details.

bckg.indep.cont.vars — Names of continuous independent background or contextual variables used as predictors in the model. See details.

bckg.indep.cat.vars — Names of categorical independent background or contextual variables used as predictors in the model to compute contrasts for (see bckg.cat.contrasts and bckg.ref.cats). See details.

bckg.cat.contrasts — String vector with the same length as the length of bckg.indep.cat.vars specifying the type of contrasts to compute in case bckg.indep.cat.vars are provided. See details.

bckg.ref.cats — Vector of integers with the same length as the length of bckg.indep.cat.vars and bckg.cat.contrasts specifying the reference categories for the contrasts to compute in case bckg.indep.cat.vars are provided. See details.

PV.root.indep — The root names for a set of plausible values used as a independent variables in the model. See details.

interactions — Interaction terms - a list containing vectors of length of two. See details.

standardize — Shall the dependent and independent variables be standardized to produce beta coefficients? The default is FALSE. See details.

weight.var — The name of the variable containing the weights. If no name of a weight variable is provide, the function will automatically select the default weight variable for the provided data, depending on the respondent type.

include.missing — Logical, shall the missing values of the splitting variables be included as categories to split by and all statistics produced for them? The default (FALSE) takes all cases on the splitting variables without missing values before computing any statistics. See details.

shortcut — Logical, shall the "shortcut" method for IEA TIMSS, TIMSS Advanced, TIMSS Numeracy, eTIMSS PSI, PIRLS, ePIRLS, PIRLS Literacy and RLII be applied? The default (FALSE) applies the "full" design when computing the variance components and the standard errors of the estimates.

save.output — Logical, shall the output be saved in MS Excel file (default) or not (printed to the console or assigned to an object).

output.file — If save.output = TRUE (default), full path to the output file including the file name. If omitted, a file with a default file name "Analysis.xlsx" will be written to the working directory (getwd()). Ignored if save.output = FALSE.

open.output Logical, shall the output be open after it has been written? The default (TRUE) opens the output in the default spreadsheet program installed on the computer. Ignored if save.output = FALSE.

Details

Either data.file or data.object shall be provided as source of data. If both of them are provided, the function will stop with an error message.

The function computes linear regression coefficients by the categories of the splitting variables. The percentages of respondents in each group are computed within the groups specified by the last splitting variable. If no splitting variables are added, the results will be computed only by country.

If standardize = TRUE, the variables will be standardized before computing any statistics to provide beta regression coefficients.

Either a background/contextual variable (bckg.dep.var) or a root name of a set of plausible values (PV.root.dep) can be provided as dependent variable but not both.

Background/contextual variables passed to bckg.indep.cont.vars will be treated as numeric variables in the model. Variables with discrete number of categories (i.e. factors) passed to bckg.indep.cat.vars will be used to compute contrasts. In this case the type of contrast has to be passed to bckg.cat.contrasts and the number of the reference categories for each of the bckg.indep.cat.vars. The number of types of contrasts and the reference categories must be the same as the number of bckg.indep.cat.vars. The currently supported contrast coding schemes are:

- dummy - the intercept is the average on the dependent variable for the respondents choosing the reference category and the slopes are the differences between intercept and the average of respondents on the dependent variable choosing every other category.
- deviation - the intercept is the grand mean on the dependent variable regardless of the group and the slopes are the differences between intercept and the average of respondents on the dependent variable choosing every other category except the reference one.
- simple - the same as for the dummy contrast coding, except for the intercept which in this case is the grand mean.

Note that when using standardize = TRUE, the contrast coding of bckg.indep.cat.vars is not standardized. Thus, the regression coefficients may not be comparable to other software solutions for analyzing large-scale assessment data which rely on, for example, SPSS or SAS where the contrast coding of categorical variables (e.g. dummy coding) takes place by default. However, the model statistics will be identical.

Multiple continuous or categorical background variables and/or sets of plausible values can be provided to compute regression coefficients for. Please note that in this case the results will slightly differ compared to using each pair of the same background continuous variables or PVs in separate analysis. This is because the cases with the missing values are removed in advance and the more variables are provided, the more cases are likely to be removed. That is, the function support only listwisie deletion.

Computation of regression coefficients involving plausible values requires providing a root of the plausible values names in PV.root.dep and/or PV.root.indep. All studies (except CivED, TEDS-M, SITES, TALIS and TALIS Starting Strong Survey) have a set of PVs per construct (e.g. in TIMSS five for overall mathematics, five for algebra, five for geometry, etc.). In some studies (say TIMSS and PIRLS) the names of the PVs in a set always start with character string and end with

sequential number of the PV. For example, the names of the set of PVs for overall mathematics in TIMSS are BSMMAT01, BSMMAT02, BSMMAT03, BSMMAT04 and BSMMAT05. The root of the PVs for this set to be added to `PV.root.dep` or `PV.root.indep` will be "BSMMAT". The function will automatically find all the variables in this set of PVs and include them in the analysis. In other studies like OECD PISA and IEA ICCS and ICILS the sequential number of each PV is included in the middle of the name. For example, in ICCS the names of the set of PVs are PV1CIV, PV2CIV, PV3CIV, PV4CIV and PV5CIV. The root PV name has to be specified in `PV.root.dep` or `PV.root.indep` as "PV#CIV". More than one set of PVs can be added in `PV.root.indep`.

The function can also compute two-way interaction effects between independent variables by passing a list to the `interactions` argument. The list must contain vectors of length two and all variables in these vectors **must also be passed as independent variables** (see the examples). Note the following:

- When an interaction is between two independent background continuous variables (i.e. both are passed to `bckg.indep.cont.vars`), the interaction effect will be computed between them as they are.
- When the interaction is between two categorical variables (i.e. both are passed to `bckg.indep.cat.vars`), the interaction effect will be computed between each possible pair of categories of the two variables, except for the reference categories.
- When the interaction is between one continuous (i.e. passed to `bckg.indep.cont.vars`) and one categorical (i.e. passed to `bckg.indep.cat.vars`), the interaction effect will be computed between the continuous variable and each category of the categorical variable, except for the reference category.
- When the interaction is between a continuous variable (i.e. passed to `bckg.indep.cont.vars`) and a set of PVs (i.e. passed to `PV.root.indep`), the interaction effect is computed between the continuous variable and each PV in the set and the results are aggregated.
- When the interaction is between a categorical variable (i.e. passed to `bckg.indep.cat.vars`) and a set of PVs (i.e. passed to `PV.root.indep`), the interaction effect is computed between each category of the categorical variable (except the reference category) and each PV in the set. The results are aggregated for each of the categories of the categorical variables and the set of PVs.
- When the interaction is between two sets of PVs (i.e. passed to `PV.root.indep`), the interaction effect is computed between the first PV in the first set and the first PV in the second set, the second PV in the first set and the second PV in the second set, and so on. The results are then aggregated.

If `include.missing = FALSE` (default), all cases with missing values on the splitting variables will be removed and only cases with valid values will be retained in the statistics. Note that the data from the studies can be exported in two different ways: (1) setting all user-defined missing values to `NA`; and (2) importing all user-defined missing values as valid ones and adding their codes in an additional attribute to each variable. If the `include.missing` is set to `FALSE` (default) and the data used is exported using option (2), the output will remove all values from the variable matching the values in its `missings` attribute. Otherwise, it will include them as valid values and compute statistics for them.

The `shortcut` argument is valid only for TIMSS, eTIMSS, TIMSS Advanced, TIMSS Numeracy, eTIMSS PSI, PIRLS, ePIRLS, PIRLS Literacy and RLII. Previously, in computing the standard errors, these studies were using 75 replicates because one of the schools in the 75 JK zones had

its weights doubled and the other one has been taken out. Since TIMSS 2015 and PIRLS 2016 the studies use 150 replicates and in each JK zone once a school has its weights doubled and once taken out, i.e. the computations are done twice for each zone. For more details see Foy & LaRoche (2016) and Foy & LaRoche (2017). If replication of the tables and figures is needed, the `shortcut` argument has to be changed to `TRUE`. The function provides two-tailed *t*-test and *p*-values for the regression coefficients.

Value

If `save.output = FALSE`, a list containing the estimates and analysis information. If `save.output = TRUE` (default), an MS Excel (`.xlsx`) file (which can be opened in any spreadsheet program), as specified with the full path in the `output.file`. If the argument is missing, an Excel file with the generic file name "Analysis.xlsx" will be saved in the working directory (`getwd()`). The workbook contains four spreadsheets. The first one ("Estimates") contains a table with the results by country and the final part of the table contains averaged results from all countries' statistics. The following columns can be found in the table, depending on the specification of the analysis:

- <Country ID> - a column containing the names of the countries in the file for which statistics are computed. The exact column header will depend on the country identifier used in the particular study.
- <Split variable 1>, <Split variable 2>... - columns containing the categories by which the statistics were split by. The exact names will depend on the variables in `split.vars`.
- n_Cases - the number of cases in the sample used to compute the statistics.
- Sum_<Weight variable> - the estimated population number of elements per group after applying the weights. The actual name of the weight variable will depend on the weight variable used in the analysis.
- Sum_<Weight variable>_SE - the standard error of the the estimated population number of elements per group. The actual name of the weight variable will depend on the weight variable used in the analysis.
- Percentages_<Last split variable> - the percentages of respondents (population estimates) per groups defined by the splitting variables in `split.vars`. The percentages will be for the last splitting variable which defines the final groups.
- Percentages_<Last split variable>_SE - the standard errors of the percentages from above.
- Variable - the variable names (background/contextual or PV root names, or contrast coded variable names).
- Coefficients - the regression coefficients (intercept and slopes).
- Coefficients_SE - the standard error of the regression coefficients (intercepts and slopes) for each independent variable (background/contextual or PV root names, or contrast coded variable names) in the model.
- Coefficients_SVR - the sampling variance component for the regression coefficients if root PVs are specified either as dependent or independent variables.
- Coefficients_<root PV>_MVR - the measurement variance component for the regression coefficients if root PVs are specified either as dependent or independent variables.
- t_value - the *t*-test value for the regression coefficients.
- p_value - the *p*-value for the regression coefficients.

When interaction terms are included, the cells with the interactions in the `Variables` column will contain the names of the two variables in each of the interaction terms, divided by colon, e.g. `ASBGSSB:ASBGHRL`.

The second sheet contains the model statistics:

- <Country ID> - a column containing the names of the countries in the file for which statistics are computed. The exact column header will depend on the country identifier used in the particular study.
- <Split variable 1>, <Split variable 2>... - columns containing the categories by which the statistics were split by. The exact names will depend on the variables in `split.vars`.
- Statistic - a column containing the R-Squared, adjusted R-Squared, F-Statistic and degrees of freedom estimates.
- Estimate - the numerical estimates for each of the above.
- Estimate_SE - the standard errors of the estimates from above.
- Estimate_SVR - the sampling variance component if PVs were included in the model.
- Estimate_MVR - the measurement variance component if PVs were included in the model.
- t_value - the *t*-test value for the regression coefficients, value only for the F-Statistic is provided.
- p_value - the *p*-value for the regression coefficients, value only for the F-Statistic is provided.

The third sheet contains some additional information related to the analysis per country in columns:

- DATA - used `data.file` or `data.object`.
- STUDY - which study the data comes from.
- CYCLE - which cycle of the study the data comes from.
- WEIGHT - which weight variable was used.
- DESIGN - which resampling technique was used (JRR or BRR).
- SHORTCUT - logical, whether the shortcut method was used.
- NREPS - how many replication weights were used.
- ANALYSIS_DATE - on which date the analysis was performed.
- START_TIME - at what time the analysis started.
- END_TIME - at what time the analysis finished.
- DURATION - how long the analysis took in hours, minutes, seconds and milliseconds.

The fourth sheet contains the call to the function with values for all parameters as it was executed. This is useful if the analysis needs to be replicated later.

References

LaRoche, S., Joncas, M., & Foy, P. (2016). Sample Design in TIMSS 2015. In M. O. Martin, I. V. S. Mullis, & M. Hooper (Eds.), *Methods and Procedures in TIMSS 2015* (pp. 3.1-3.37). Chestnut Hill, MA: TIMSS & PIRLS International Study Center. LaRoche, S., Joncas, M., & Foy, P. (2017). Sample Design in PIRLS 2016. In M. O. Martin, I. V. S. Mullis, & M. Hooper (Eds.), *Methods and Procedures in PIRLS 2016* (pp. 3.1-3.34). Chestnut Hill, MA: Lynch School of Education, Boston College. UCLA: Statistical Consulting Group. 2020. "R LIBRARY CONTRAST CODING SYSTEMS FOR CATEGORICAL VARIABLES." *IDRE Stats - Statistical Consulting Web Resources*. Retrieved June 16, 2020 (https://stats.idre.ucla.edu/r/library/r-library-contrast-coding-systems-for-categorical-variables/).

See Also

lsa.convert.data

Examples

```
# Compute linear regression coefficients with the complex student background scale "Student
# Sense of School Belonging/SCL" as dependent variable, and "Home Educational Resources/SCL"
# and "Students Value Science/SCL" as independent variables, by sex of students in TIMSS 2015
# grade 8 using data file, omit missing from the splitting variable (female and male as answered
# by the students), without shortcut, and open the output after the computations are done
## Not run:
lsa.lin.reg(data.file = "C:/temp/test.RData", split.vars = "BSBG01", bckg.dep.var = "BSBGSSB",
bckg.indep.cont.vars = c("BSBGHER", "BSBGSVS"))

## End(Not run)

# Compute linear regression coefficients with the set of PVs on overall mathematics achievement
# as dependent variable, and "Home Educational Resources/SCL" and "Students Value Science/SCL"
# independent variables, by sex of students in TIMSS 2015 grade 8 using data file, omit missing
# from the splitting variable (female and male as answered by the students), with shortcut, and
# without opening the output after the computations are done
## Not run:
lsa.lin.reg(data.file = "C:/temp/test.RData", split.vars = "BSBG01", PV.root.dep = "BSMMAT",
bckg.indep.cont.vars = c("BSBGHER", "BSBGSVS"), shortcut = TRUE, open.output = FALSE)

## End(Not run)

# Same as above, standardizing the coefficients
## Not run:
lsa.lin.reg(data.file = "C:/temp/test.RData", split.vars = "BSBG01", PV.root.dep = "BSMMAT",
bckg.indep.cont.vars = c("BSBGHER", "BSBGSVS"), standardize = TRUE, shortcut = TRUE,
open.output = FALSE)

## End(Not run)

# Compute linear regression with contrast coded categorical variables, using student sex as
# splitting variable, the set of five PVs on overall mathematics achievement as dependent
# variable, and the frequency of speaking the language of test at home and the number of
# books at home as contrast (dummy and simple) coded variables where the second and the third
# categories, respectively, are the reference, without shortcut, saving the output in the home
# directory and opening it after the computations are done
## Not run:
lsa.lin.reg(data.object = merged.TIMSS.2015, split.vars = "BSBG01", PV.root.dep = "BSMMAT",
bckg.indep.cat.vars = c("BSBG03", "BSBG04"), bckg.cat.contrasts = c("dummy", "simple"),
bckg.ref.cats = c(2, 3))

## End(Not run)

# Compute linear regression with interaction terms using PIRLS 2016 student data.
## Not run:
lsa.lin.reg(data.file = "C:/temp/test.RData", bckg.dep.var = "ASBGSB",
bckg.indep.cont.vars = c("ASBGSSB", "ASBGHRL"), bckg.indep.cat.vars = "ASBG01",
```

```
interactions = list(c("ASBG01", "ASBGSSB"), c("ASBGHRL", "ASBGSSB")))

## End(Not run)
```

`lsa.merge.data` *Merge study data from different countries and/or respondents*

Description

`lsa.merge.data` combines data from different countries and/or different respondents (e.g. students and teachers, or students and schools).

Usage

```
lsa.merge.data(inp.folder, file.types, ISO, out.file)
```

Arguments

`inp.folder`	Folder containing the data sets. The data sets must be `.RData`, produced by `lsa.convert.data`. See details.
`file.types`	What file types (i.e. respondents) shall be merged? See details.
`ISO`	Vector containing character ISO codes of the countries' data files to include in the merged file. See details.
`out.file`	Full path to the file the data shall be stored in. The object stored in the file will have the same name. See details.

Details

The function merges files from studies where the files are per country and respondent type (e.g. student, school, teacher). That is, all studies except PISA.

The `inp.folder` specifies the path to the folder containing the `.RData` files produced by `lsa.convert.data`. The folder must contain only files for a single study, single cycle and single population (e.g. TIMSS 2015, grade 4). All files in the input folder must be exported with the same option (`TRUE` or `FALSE`) of the `missing.to.NA` argument of the `lsa.convert.data` function. If input folder is not provided to the argument, the working folder (`getwd()`) will be used.

The `file.types` is a list of the respondent types as component names and their variables as elements to be merged. The file type names are three-character codes, the first three characters of the corresponding file names. The elements are vectors of upper case variable names, `NULL` takes all variables in the corresponding file. For example, in TIMSS `asg` will merge only student-level data from grade 4, `c(asg, atg)` will merge the student-level and teacher-level data from grade 4, `c(bsg, btm)` will merge student-level and mathematics teacher-level data from grade 8. If a merge is not possible by the study design, the function will stop with an error. See the examples.

The `ISO` is a character vector specifying the countries whose data shall be merged. The elements of the vector are the fourth, fifth and sixth characters in the file names. For example, `c("aus", "swe", "svn")` will merge the data from Australia, Sweden and Slovenia for the file types specified in

file.types. If file for specific country does not exist in the inp.folder, a warning will be issued. If missing, the files for all countries in the folder will be merged for the specified file.types.

The out.file must contain full path (including the .RData extension, if missing, it will be added) to the output file (i.e. the file containing merged data). The file contains object with the same name and has a class extension lsa.data. It has additional attribute file.type showing data from which respondents is available after the merging has been done. For example, merging the student-level data with teacher-level data in TIMSS grade 4 will assign "std.bckg.tch.bckg" to this attribute. The object has two additional attributes: study name (study) and study cycle (cycle). The object in the .RData file is keyed on the country ID variable. If output folder is not provided, the merged file will be saved in the working folder (getwd()) as merged_data.RData.

Value

.RData data file containing an object with class lsa.data, an extension of the data.table class. The data.table object has the same name as the .RData file it is saved in. The object contains the data from different respondents and/or countries merged and has additional attributes: study name (study), study cycle (cycle), and respondent file type (file.type). Each variable has its own additional attributes: its own label attached to it, if it existed in the source SPSS file. If the missing.to.NA in the source file was set to TRUE, each variable has an attribute missings, containing the user-defined missing values.

See Also

lsa.convert.data

Examples

```
# Merge TIMSS 2015 grade 4 student and teacher variables for Australia, Chinese Taipei and
# Slovenia taking all variables in both files
## Not run:
lsa.merge.data(inp.folder = "C:/Data", file.types = list(asg = NULL, atg = NULL),
ISO = c("aus", "twn", "svnn"), out.file = "C:/Merged/Merged.RData")

## End(Not run)

# Same as the above, taking just few variables from each file
## Not run:
lsa.merge.data(inp.folder = "C:/Data",
file.types = list(asg = c("ASBG01", "ASBG02A", "ASBG02B"),
atg = c("ATBG01", "ATBG02", "ATBG03")), ISO = c("aus", "twn", "svnn"),
out.file = "C:/Merged/Merged.RData")

## End(Not run)
```

`lsa.pcts.means` *Compute percentages of respondents in groups and/or means (arithmetic average, median or mode) on continuous variables within specified groups*

Description

`lsa.pcts.means` computes percentages of respondents within groups defined by one or more variables and the means for one or more variables.

Usage

```
lsa.pcts.means(
  data.file,
  data.object,
  split.vars,
  bckg.avg.vars,
  PV.root.avg,
  central.tendency,
  weight.var,
  include.missing = FALSE,
  shortcut = FALSE,
  graphs = FALSE,
  save.output = TRUE,
  output.file,
  open.output = TRUE
)
```

Arguments

`data.file`	The file containing `lsa.data` object. Either this or `data.object` shall be specified, but not both. See details.
`data.object`	The object in the memory containing `lsa.data` object. Either this or `data.file` shall be specified, but not both. See details.
`split.vars`	Categorical variable(s) to split the results by. If no split variables are provided, the results will be for the overall countries' populations. If one or more variables are provided, the results will be split by all but the last variable and the percentages of respondents will be computed by the unique values of the last splitting variable.
`bckg.avg.vars`	Name(s) of continuous background or contextual variable(s) to compute the means for. The results will be computed by all groups specified by the splitting variables. See details.
`PV.root.avg`	The root name(s) for the set(s) of plausible values. See details.
`central.tendency`	Which measure of central tendency shall be computed - `mean` (default) `median` or `mode`. See details.

`weight.var` The name of the variable containing the weights. If no name of a weight variable is provided, the function will automatically select the default weight variable for the provided data, depending on the respondent type.

`include.missing` Logical, shall the missing values of the splitting variables be included as categories to split by and all statistics produced for them? The default (`FALSE`) takes all cases on the splitting variables without missing values before computing any statistics. See details.

`shortcut` Logical, shall the "shortcut" method for IEA TIMSS, TIMSS Advanced, TIMSS Numeracy, eTIMSS PSI, PIRLS, ePIRLS, PIRLS Literacy and RLII be applied? The default (`FALSE`) applies the "full" design when computing the variance components and the standard errors of the estimates.

`graphs` Logical, shall graphs be produced? Default is `FALSE`. See details.

`save.output` Logical, shall the output be saved in MS Excel file (default) or not (printed to the console or assigned to an object).

`output.file` If `save.output = TRUE` (default), full path to the output file including the file name. If omitted, a file with a default file name "Analysis.xlsx" will be written to the working directory (`getwd()`). Ignored if `save.output = FALSE`.

`open.output` Logical, shall the output be open after it has been written? The default (`TRUE`) opens the output in the default spreadsheet program installed on the computer. Ignored if `save.output = FALSE`.

Details

The function computes percentages of respondents specified by the categories of splitting variables. The percentages are computed within the groups specified by the last splitting variable. If a continuous variable(s) are provided (background or sets of plausible values), their means (as arithmetic means, medians or modes) will be computed by groups defined by one or more splitting variables. If no splitting variables are added, the results will be computed only by country.

Either `data.file` or `data.object` shall be provided as source of data. If both of them are provided, the function will stop with an error message.

Multiple continuous background variables can be provided to compute their means (as arithmetic means, medians or modes). Please note that in this case the results will slightly differ compared to using each of the same background continuous variables in separate analyses. This is because the cases with the missing values on `bckg.avg.vars` are removed in advance and the more variables are provided to `bckg.avg.vars`, the more cases are likely to be removed.

Computation of means involving plausible values requires providing a root of the plausible values names in `PV.root.avg`. All studies (except CivED, TEDS-M, SITES, TALIS and TALIS Starting Strong Survey) have a set of PVs per construct (e.g. in TIMSS five for overall mathematics, five for algebra, five for geometry, etc.). In some studies (say TIMSS and PIRLS) the names of the PVs in a set always start with character string and end with sequential number of the PV. For example, the names of the set of PVs for overall mathematics in TIMSS are BSMMAT01, BSMMAT02, BSMMAT03, BSMMAT04 and BSMMAT05. The root of the PVs for this set to be added to `PV.root.avg` will be "BSMMAT". The function will automatically find all the variables in this set of PVs and include them in the analysis. In other studies like OECD PISA and IEA ICCS and ICILS the sequential number of each PV is included in the middle of the name. For example, in

ICCS the names of the set of PVs are PV1CIV, PV2CIV, PV3CIV, PV4CIV and PV5CIV. The root PV name has to be specified in PV.root.avg as "PV#CIV". More than one set of PVs can be added. Note, however, that providing continuous variable(s) for the bckg.avg.vars argument and root PV for the PV.root.avg argument will affect the results for the PVs because the cases with missing on bckg.avg.vars will be removed and this will also affect the results from the PVs. On the other hand, using more than one set of PVs at the same time should not affect the results on any PV estimates because PVs shall not have any missing values.

If no variables are specified for bckg.avg.vars, and no PV root names for PV.root.avg, the output will contain only percentages of cases in groups specified by the splitting variables, if any. If they are, their means will be computed either as arithmetic means, medians or modes. This can be controlled by setting the central.tendency argument to mean (default), median or mode. Note that if central.tendency = "mode" and the variables passed to bckg.avg.vars or the sets of PVs passed to PV.root.avg have more than one mode, the value for the lowest value will be included in the output. As a conseequence, the standard errors may be inflated.

If include.missing = FALSE (default), all cases with missing values on the splitting variables will be removed and only cases with valid values will be retained in the statistics. Note that the data from the studies can be exported in two different ways: (1) setting all user-defined missing values to NA; and (2) importing all user-defined missing values as valid ones and adding their codes in an additional attribute to each variable. If the include.missing is set to FALSE (default) and the data used is exported using option (2), the output will remove all values from the variable matching the values in its missings attribute. Otherwise, it will include them as valid values and compute statistics for them.

The shortcut argument is valid only for TIMSS, eTIMSS PSI, TIMSS Advanced, TIMSS Numeracy, PIRLS, ePIRLS, PIRLS Literacy and RLII. Previously, in computing the standard errors, these studies were using 75 replicates because one of the schools in the 75 JK zones had its weights doubled and the other one has been taken out. Since TIMSS 2015 and PIRLS 2016 the studies use 150 replicates and in each JK zone once a school has its weights doubled and once taken out, i.e. the computations are done twice for each zone. For more details see Foy & LaRoche (2016) and Foy & LaRoche (2017). If replication of the tables and figures is needed, the shortcut argument has to be changed to TRUE.

If graphs = TRUE, the function will produce graphs. If only split.vars are specified, bar plots of percentages of respondents (population estimates) per group will be produced with error bars (95% confidence) for these percentages. If bckg.avg.vars and/or PV.root.avg are specified, plots with 95% confidence intervals of the averages (means, medians or modes) will be produced for each average analysis variable. All plots are produced per country. If bckg.avg.vars and/or PV.root.avg are specified, but no split.vars at the end there will be plots for each of the analysis average variables for all countries together.

Value

If save.output = FALSE, a list containing the estimates and analysis information. If graphs = TRUE, the plots will be added to the list of estimates.

If save.output = TRUE (default), an MS Excel (.xlsx) file (which can be opened in any spreadsheet program), as specified with the full path in the output.file. If the argument is missing, an Excel file with the generic file name "Analysis.xlsx" will be saved in the working directory (getwd()). The workbook contains three spreadsheets. The first one ("Estimates") contains a table with the results

by country and the final part of the table contains averaged results from all countries' statistics. The following columns can be found in the table, depending on the specification of the analysis:

- <Country ID> - a column containing the names of the countries in the file for which statistics are computed. The exact column header will depend on the country identifier used in the particular study.
- <Split variable 1>, <Split variable 2>... - columns containing the categories by which the statistics were split by. The exact names will depend on the variables in `split.vars`.
- n_Cases - the number of cases in the sample used to compute the statistics.
- Sum_<Weight variable> - the estimated population number of elements per group after applying the weights. The actual name of the weight variable will depend on the weight variable used in the analysis.
- Sum_<Weight variable>_SE - the standard error of the the estimated population number of elements per group. The actual name of the weight variable will depend on the weight variable used in the analysis.
- Percentages_<Last split variable> - the percentages of respondents (population estimates) per groups defined by the splitting variables in `split.vars`. The percentages will be for the last splitting variable which defines the final groups.
- Percentages_<Last split variable>_SE - the standard errors of the percentages from above.
- Mean_<Background variable> - returned if `central.tendency = "mean"`, the arithmetic average of the continuous <Background variable> specified in `bckg.avg.vars`. There will be one column with the arithmetic average estimate for each variable specified in `bckg.avg.vars`.
- Mean_<Background variable>_SE - returned if `central.tendency = "mean"`, the standard error of the arithmetic average of the continuous <Background variable> specified in `bckg.avg.vars`. There will be one column with the SE of the average estimate for each variable specified in `bckg.avg.vars`.
- Variance_<Background variable> - returned if `central.tendency = "mean"`, the variance for the continuous <Background variable> specified in `bckg.avg.vars`. There will be one column with the variance estimate for each variable specified in `bckg.avg.vars`.
- Variance_<Background variable>_SE - returned if `central.tendency = "mean"`, the error of the variance for the continuous <Background variable> specified in `bckg.avg.vars`. There will be one column with the error of the variance estimate for each variable specified in `bckg.avg.vars`.
- SD_<Background variable> - returned if `central.tendency = "mean"`, the standard deviation for the continuous <Background variable> specified in `bckg.avg.vars`. There will be one column with the standard deviation estimate for each variable specified in `bckg.avg.vars`.
- SD_<Background variable>_SE - returned if `central.tendency = "mean"`, the error of the standard deviation for the continuous <Background variable> specified in `bckg.avg.vars`. There will be one column with the error of the standard deviation estimate for each variable specified in `bckg.avg.vars`.
- Median_<Background variable> - returned if `central.tendency = "median"`, the median of the continuous <Background variable> specified in `bckg.avg.vars`. There will be one column with the median estimate for each variable specified in `bckg.avg.vars`.
- Median_<Background variable>_SE - returned if `central.tendency = "median"`, the standard error of the median of the continuous <Background variable> specified in `bckg.avg.vars`.

There will be one column with the SE of the median estimate for each variable specified in bckg.avg.vars.

- MAD_<Background variable> - returned if central.tendency = "median", the Median Absolute Deviation (MAD) for the continuous <Background variable> specified in bckg.avg.vars. There will be one column with the MAD estimate for each variable specified in bckg.avg.vars.
- MAD_<Background variable>_SE - returned if central.tendency = "median", the standard error of MAD for the continuous <Background variable> specified in bckg.avg.vars. There will be one column with the MAD SE estimate for each variable specified in bckg.avg.vars.
- Mode_<Background variable> - returned if central.tendency = "mode", the mode of the continuous <Background variable> specified in bckg.avg.vars. There will be one column with the mode estimate for each variable specified in bckg.avg.vars.
- Mode_<Background variable>_SE - returned if central.tendency = "mode", the standard error of the mode of the continuous <Background variable> specified in bckg.avg.vars. There will be one column with the SE of the mode estimate for each variable specified in bckg.avg.vars.
- Percent_Missings_<Background variable> - the percentage of missing values for the <Background variable> specified in bckg.avg.vars. There will be one column with the percentage of missing values for each variable specified in bckg.avg.vars.
- Mean_<root PV> - returned if central.tendency = "mean", the arithmetic average of the PVs with the same <root PV> specified in PV.root.avg. There will be one column with the arithmetic average estimate for each set of PVs specified in PV.root.avg.
- Mean_<root PV>_SE - returned if central.tendency = "mean", the standard error of the arithmetic average of the PVs with the same <root PV> specified in PV.root.avg. There will be one column with the standard error of arithmetic average estimate for each set of PVs specified in PV.root.avg.
- Mean_<root PV>_SVR - returned if central.tendency = "mean", the sampling variance component for the arithmetic average of the PVs with the same <root PV> specified in PV.root.avg. There will be one column with the sampling variance component for the arithmetic average estimate for each set of PVs specified in PV.root.avg.
- Mean_<root PV>_MVR - returned if central.tendency = "mean", the measurement variance component for the arithmetic average of the PVs with the same <root PV> specified in PV.root.avg. There will be one column with the measurement variance component for the arithmetic average estimate for each set of PVs specified in PV.root.avg.
- Variance_<root PV> - returned if central.tendency = "mean", the total variance of the PVs with the same <root PV> specified in PV.root.avg. There will be one column with the total variance of each set of PVs specified in PV.root.avg.
- Variance_<root PV>_SE - returned if central.tendency = "mean", the standard error of the total variance of the PVs with the same <root PV> specified in PV.root.avg. There will be one column with the standard error of the total variance of each set of PVs specified in PV.root.avg.
- Variance_<root PV>_SVR - returned if central.tendency = "mean", the sampling component of the variance of the PVs with the same <root PV> specified in PV.root.avg. There will be one column with the sampling component of the variance of each set of PVs specified in PV.root.avg.

- Variance_<root PV>_MVR - returned if `central.tendency = "mean"`, the measurement component of the variance of the PVs with the same <root PV> specified in `PV.root.avg`. There will be one column with the measurement component of the variance of each set of PVs specified in `PV.root.avg`.
- SD_<root PV> - returned if `central.tendency = "mean"`, the standard deviation of the PVs with the same <root PV> specified in `PV.root.avg`. There will be one column with the standard deviation of each set of PVs specified in `PV.root.avg`.
- SD_<root PV>_SE - returned if `central.tendency = "mean"`, the standard error of the standard deviation of the PVs with the same <root PV> specified in `PV.root.avg`. There will be one column with the standard error of the standard deviation of each set of PVs specified in `PV.root.avg`.
- SD_<root PV>_SVR - returned if `central.tendency = "mean"`, the sampling component of the standard deviation of the PVs with the same <root PV> specified in `PV.root.avg`. There will be one column with the sampling component of the standard deviation of each set of PVs specified in `PV.root.avg`.
- SD_<root PV>_MVR - returned if `central.tendency = "mean"`, the measurement component of the standard deviation of the PVs with the same <root PV> specified in `PV.root.avg`. There will be one column with the measurement component of the standard deviation of each set of PVs specified in `PV.root.avg`.
- Median_<root PV> - returned if `central.tendency = "median"`, the median of the PVs with the same <root PV> specified in `PV.root.avg`. There will be one column with the median estimate for each set of PVs specified in `PV.root.avg`.
- Median_<root PV>_SE - returned if `central.tendency = "median"`, the standard error of the median of the PVs with the same <root PV> specified in `PV.root.avg`. There will be one column with the standard error of median estimate for each set of PVs specified in `PV.root.avg`.
- MAD_<root PV> - returned if `central.tendency = "median"`, the Median Absolute Deviation (MAD) for a set of PVs specified in `PV.root.avg`. There will be one column with the MAD estimate for each set of PVs specified in `PV.root.avg`.
- MAD_<root PV>_SE - returned if `central.tendency = "median"`, the standard error of MAD for a set of PVs specified in `PV.root.avg`. There will be one column with the SE estimate of the MAD for each set of PVs specified in `PV.root.avg`.
- Mode_<root PV> - returned if `central.tendency = "mode"`, the mode of the PVs with the same <root PV> specified in `PV.root.avg`. There will be one column with the mode estimate for each set of PVs specified in `PV.root.avg`.
- Mode_<root PV>_SE - returned if `central.tendency = "mode"`, the standard error of the mode of the PVs with the same <root PV> specified in `PV.root.avg`. There will be one column with the standard error of mode estimate for each set of PVs specified in `PV.root.avg`.
- Percent_Missings_<root PV> - the percentage of missing values for the <root PV> specified in `PV.root.avg`. There will be one column with the percentage of missing values for each set of PVs specified in `PV.root.avg`.

The second sheet contains some additional information related to the analysis per country in the following columns:

- DATA - used `data.file` or `data.object`.
- STUDY - which study the data comes from.

- CYCLE - which cycle of the study the data comes from.
- WEIGHT - which weight variable was used.
- DESIGN - which resampling technique was used (JRR or BRR).
- SHORTCUT - logical, whether the shortcut method was used.
- NREPS - how many replication weights were used.
- ANALYSIS_DATE - on which date the analysis was performed.
- START_TIME - at what time the analysis started.
- END_TIME - at what time the analysis finished.
- DURATION - how long the analysis took in hours, minutes, seconds and milliseconds.

The third sheet contains the call to the function with values for all parameters as it was executed. This is useful if the analysis needs to be replicated later.

If `graphs = TRUE` there will be an additional "Graphs" sheet containing all plots.

If any warnings resulting from the computations are issued, these will be included in an additional "Warnings" sheet in the workbook as well.

References

LaRoche, S., Joncas, M., & Foy, P. (2016). Sample Design in TIMSS 2015. In M. O. Martin, I. V. S. Mullis, & M. Hooper (Eds.), *Methods and Procedures in TIMSS 2015* (pp. 3.1-3.37). Chestnut Hill, MA: TIMSS & PIRLS International Study Center. LaRoche, S., Joncas, M., & Foy, P. (2017). Sample Design in PIRLS 2016. In M. O. Martin, I. V. S. Mullis, & M. Hooper (Eds.), *Methods and Procedures in PIRLS 2016* (pp. 3.1-3.34). Chestnut Hill, MA: Lynch School of Education, Boston College.

See Also

`lsa.convert.data`

Examples

```
# Compute percentages of female and male students in TIMSS 2015 grade 8 using data file, omit
# missing from the splitting variable (female and male as answered by the students), without
# shortcut, and open the output after the computations are done
## Not run:
lsa.pcts.means(data.file = "C:/Data/TIMSS_2015_G8_Student_Miss_to_NA.RData",
split.vars = "BSBG01", include.missing = FALSE,
output.file = "C:/temp/test.xlsx", open.output = TRUE)

## End(Not run)

# Compute the arithmetic average of the complex background scale "Students like learning
# mathematics" by student sex and frequency of using computer or tablet at home using TIMSS
# 2015 grade 8 data loaded in memory, using the shortcut, include the missing values in
# the splitting variables, and use the senate weights
## Not run:
lsa.pcts.means(data.object = T15_G8_student_data, split.vars = c("BSBG01", "BSBG13A"),
bckg.avg.vars = "BSBGSLM", weight.var = "SENWGT", include.missing = FALSE, shortcut = TRUE,
```

```
output.file = "C:/temp/test.xlsx", open.output = TRUE)

## End(Not run)

# Repeat the analysis from above, adding a second continuous variable to compute the arithmetic
# average for, the "Students Like Learning Science" complex scale
## Not run:
lsa.pcts.means(data.object = T15_G8_student_data, split.vars = c("BSBG01", "BSBG13A"),
bckg.avg.vars = c("BSBGSLM", "BSBGSLS"), weight.var = "SENWGT", include.missing = FALSE,
shortcut = TRUE, output.file = "C:/temp/test.xlsx", open.output = TRUE)

## End(Not run)

# Compute the arithmetic average of student overall reading achievement scores
# (i.e. using a set of PVs), using PIRLS 2016 student data file, split the output by student
# sex, use the full design, include the missing values od the splitting variable
# (i.e. student sex), and do not open the output after the computations are finished
## Not run:
lsa.pcts.means(data.file = "C:/Data/PIRLS_2016_Student_Miss_to_NA.RData", split.vars = "ASBG01",
PV.root.avg = "ASRREA", include.missing = TRUE,
output.file = "C:/temp/test.xlsx", open.output = FALSE)

## End(Not run)

# Same as above, this time compute the median instead of the arithmetic average
## Not run:
lsa.pcts.means(data.file = "C:/Data/PIRLS_2016_Student_Miss_to_NA.RData", split.vars = "ASBG01",
PV.root.avg = "ASRREA", include.missing = TRUE,
central.tendency = "median",
output.file = "C:/temp/test.xlsx", open.output = FALSE)

## End(Not run)
```

`lsa.prctls` *Compute percentiles of continuous variables within groups*

Description

`lsa.prctls` computes percentiles of continuous variables within groups defined by one or more variables.

Usage

```
lsa.prctls(
  data.file,
  data.object,
  split.vars,
  bckg.prctls.vars,
  PV.root.prctls,
```

```
  prctls = c(5, 25, 50, 75, 95),
  weight.var,
  include.missing = FALSE,
  shortcut = FALSE,
  graphs = FALSE,
  save.output = TRUE,
  output.file,
  open.output = TRUE
)
```

Arguments

`data.file`	The file containing `lsa.data` object. Either this or `data.object` shall be specified, but not both. See details.
`data.object`	The object in the memory containing `lsa.data` object. Either this or `data.file` shall be specified, but not both. See details.
`split.vars`	Categorical variable(s) to split the results by. If no split variables are provided, the results will be for the overall countries' populations. If one or more variables are provided, the results will be split by all but the last variable and the percentages of respondents will be computed by the unique values of the last splitting variable.
`bckg.prctls.vars`	Name(s) of continuous background or contextual variable(s) to compute the percentiles for. The results will be computed by all groups specified by the splitting variables. See details.
`PV.root.prctls`	The root name(s) for the set(s) of plausible values. See details.
`prctls`	Vector of integers specifying the percentiles to be computed, the default is `c(5, 25, 50, 75, 95)`. See examples.
`weight.var`	The name of the variable containing the weights. If no name of a weight variable is provide, the function will automatically select the default weight variable for the provided data, depending on the respondent type.
`include.missing`	Logical, shall the missing values of the splitting variables be included as categories to split by and all statistics produced for them? The default (`FALSE`) takes all cases on the splitting variables without missing values before computing any statistics. See details.
`shortcut`	Logical, shall the "shortcut" method for IEA TIMSS, TIMSS Advanced, TIMSS Numeracy, eTIMSS PSI, PIRLS, ePIRLS, PIRLS Literacy and RLII be applied when using PVs? The default (`FALSE`) applies the "full" design when computing the variance components and the standard errors of the PV estimates.
`graphs`	Logical, shall graphs be produced? Default is `FALSE`. See details.
`save.output`	Logical, shall the output be saved in MS Excel file (default) or not (printed to the console or assigned to an object).
`output.file`	If `save.output = TRUE` (default), full path to the output file including the file name. If omitted, a file with a default file name "Analysis.xlsx" will be written to the working directory (`getwd()`). Ignored if `save.output = FALSE`.

open.output Logical, shall the output be open after it has been written? The default (TRUE) opens the output in the default spreadsheet program installed on the computer. Ignored if save.output = FALSE.

Details

Either data.file or data.object shall be provided as source of data. If both of them are provided, the function will stop with an error message.

The function computes percentiles of variables (background/contextual or sets of plausible values) by groups defined by one or more categorical variables (splitting variables). Multiple splitting variables can be added, the function will compute the percentages for all formed groups and their percentiles on the continuous variables. If no splitting variables are added, the results will be computed only by country.

Multiple continuous background variables can be provided to compute the specified percentiles for them. Please note that in this case the results will slightly differ compared to using each of the same background continuous variables in separate analyses. This is because the cases with the missing values on bckg.prctls.vars are removed in advance and the more variables are provided to bckg.prctls.vars, the more cases are likely to be removed.

Computation of percentiles involving plausible values requires providing a root of the plausible values names in PV.root.prctls. All studies (except CivED, TEDS-M, SITES, TALIS and TALIS Starting Strong Survey) have a set of PVs per construct (e.g. in TIMSS five for overall mathematics, five for algebra, five for geometry, etc.). In some studies (say TIMSS and PIRLS) the names of the PVs in a set always start with character string and end with sequential number of the PV. For example, the names of the set of PVs for overall mathematics in TIMSS are BSMMAT01, BSMMAT02, BSMMAT03, BSMMAT04 and BSMMAT05. The root of the PVs for this set to be added to PV.root.prctls will be "BSMMAT". The function will automatically find all the variables in this set of PVs and include them in the analysis. In other studies like OECD PISA and IEA ICCS and ICILS the sequential number of each PV is included in the middle of the name. For example, in ICCS the names of the set of PVs are PV1CIV, PV2CIV, PV3CIV, PV4CIV and PV5CIV. The root PV name has to be specified in PV.root.prctls as "PV#CIV". More than one set of PVs can be added. Note, however, that providing continuous variable(s) for the bckg.prctls.vars argument and root PV for the PV.root.prctls argument will affect the results for the PVs because the cases with missing on bckg.prctls.vars will be removed and this will also affect the results from the PVs. On the other hand, using more than one set of PVs at the same time should not affect the results on any PV estimates because PVs shall not have any missing values.

If include.missing = FALSE (default), all cases with missing values on the splitting variables will be removed and only cases with valid values will be retained in the statistics. Note that the data from the studies can be exported in two different ways using the lsa.convert.data: (1) setting all user-defined missing values to NA; and (2) importing all user-defined missing values as valid ones and adding their codes in an additional attribute to each variable. If the include.missing in lsa.prctls is set to FALSE (default) and the data used is exported using option (2), the output will remove all values from the variable matching the values in its missings attribute. Otherwise, it will include them as valid values and compute statistics for them.

The shortcut argument is valid only for TIMSS, eTIMSS PSI, TIMSS Advanced, TIMSS Numeracy, PIRLS, ePIRLS, PIRLS Literacy and RLII. Previously, in computing the standard errors, these studies were using 75 replicates because one of the schools in the 75 JK zones had its weights doubled and the other one has been taken out. Since TIMSS 2015 and PIRLS 2016 the studies use

150 replicates and in each JK zone once a school has its weights doubled and once taken out, i.e. the computations are done twice for each zone. For more details see Foy & LaRoche (2016) and Foy & LaRoche (2017). If replication of the tables and figures is needed, the `shortcut` argument has to be changed to `TRUE`.

If `graphs = TRUE`, the function will produce graphs. Bar plots of percentages of respondents (population estimates) per group will be produced with error bars (95% confidence) for these percentages. Line plots for the percentiles per group defined by the `split.vars` will be created with 95% confidence intervals for the percentile values. All plots are produced per country. If no `split.vars` are specified, at the end there will be percentile plots for each of the variables specified in `bckg.prctls.vars` and/or `PV.root.prctls` for all countries together.

Value

If `save.output = FALSE`, a list containing the estimates and analysis information. If `graphs = TRUE`, the plots will be added to the list of estimates.

If `save.output = TRUE` (default), an MS Excel (`.xlsx`) file (which can be opened in any spreadsheet program), as specified with the full path in the `output.file`. If the argument is missing, an Excel file with the generic file name "Analysis.xlsx" will be saved in the working directory (`getwd()`). The workbook contains three spreadsheets. The first one ("Estimates") contains a table with the results by country and the final part of the table contains averaged results from all countries' statistics. The following columns can be found in the table, depending on the specification of the analysis:

- <Country ID> - a column containing the names of the countries in the file for which statistics are computed. The exact column header will depend on the country identifier used in the particular study.
- <Split variable 1>, <Split variable 2>... - columns containing the categories by which the statistics were split by. The exact names will depend on the variables in `split.vars`.
- n_Cases - the number of cases in the sample used to compute the statistics.
- Sum_<Weight variable> - the estimated population number of elements per group after applying the weights. The actual name of the weight variable will depend on the weight variable used in the analysis.
- Sum_<Weight variable>_SE - the standard error of the the estimated population number of elements per group. The actual name of the weight variable will depend on the weight variable used in the analysis.
- Percentages_<Last split variable> - the percentages of respondents (population estimates) per groups defined by the splitting variables in `split.vars`. The percentages will be for the last splitting variable which defines the final groups.
- Percentages_<Last split variable>_SE - the standard errors of the percentages from above.
- Prctl_<Percentile value>_<Background variable> - the percentile of the continuous <Background variable> specified in `bckg.prctls.vars`. There will be one column for each percentile estimate for each variable specified in `bckg.prctls.vars`.
- Prctl_<Percentile value>_<Background variable>_SE - the standard error of the percentile of the continuous <Background variable> specified in `bckg.prctls.vars`. There will be one column with the SE per percentile estimate for each variable specified in `bckg.prctls.vars`.
- Percent_Missings_<Background variable> - the percentage of missing values for the <Background variable> specified in `bckg.prctls.vars`. There will be one column with the percentage of missing values for each variable specified in `bckg.prctls.vars`.

- Prctl_<Percentile value>_<root PV> - the percentile of the PVs with the same <root PV> specified in `PV.root.prctls`. There will be one column per percentile value estimate for each set of PVs specified in `PV.root.prctls`.
- Prctl_<Percentile value>_<root PV>_SE - the standard error per percentile per set of PVs with the same <root PV> specified in `PV.root.prctls`. There will be one column with the standard error of estimate per percentile per set of PVs specified in `PV.root.prctls`.
- Prctl_<Percentile value>_<root PV>_SVR - the sampling variance component per percentile per set of PVs with the same <root PV> specified in `PV.root.prctls`. There will be one column with the sampling variance component percentile estimate for each set of PVs specified in `PV.root.prctls`.
- Prctl_<Percentile value>_<root PV>_MVR - the measurement variance component per percentiles per set of PVs with the same <root PV> specified in `PV.root.prctls`. There will be one column with the measurement variance component per percentile per set of PVs specified in `PV.root.prctls`.
- Percent_Missings_<root PV> - the percentage of missing values for the <root PV> specified in `PV.root.prctls`. There will be one column with the percentage of missing values for each set of PVs specified in `PV.root.prctls`.

The second sheet contains some additional information related to the analysis per country in columns:

- DATA - used `data.file` or `data.object`.
- STUDY - which study the data comes from.
- CYCLE - which cycle of the study the data comes from.
- WEIGHT - which weight variable was used.
- DESIGN - which resampling technique was used (JRR or BRR).
- SHORTCUT - logical, whether the shortcut method was used.
- NREPS - how many replication weights were used.
- ANALYSIS_DATE - on which date the analysis was performed.
- START_TIME - at what time the analysis started.
- END_TIME - at what time the analysis finished.
- DURATION - how long the analysis took in hours, minutes, seconds and milliseconds.

The third sheet contains the call to the function with values for all parameters as it was executed. This is useful if the analysis needs to be replicated later.

If `graphs = TRUE` there will be an additional "Graphs" sheet containing all plots.

If any warnings resulting from the computations are issued, these will be included in an additional "Warnings" sheet in the workbook as well.

References

LaRoche, S., Joncas, M., & Foy, P. (2016). Sample Design in TIMSS 2015. In M. O. Martin, I. V. S. Mullis, & M. Hooper (Eds.), *Methods and Procedures in TIMSS 2015* (pp. 3.1-3.37). Chestnut Hill, MA: TIMSS & PIRLS International Study Center. LaRoche, S., Joncas, M., & Foy, P. (2017). Sample Design in PIRLS 2016. In M. O. Martin, I. V. S. Mullis, & M. Hooper (Eds.), *Methods and Procedures in PIRLS 2016* (pp. 3.1-3.34). Chestnut Hill, MA: Lynch School of Education, Boston College.

See Also

lsa.convert.data

Examples

```
# Compute the 5th, 25th and 50th percentiles of the complex background scale "Students like
# learning mathematics" by student sex and frequency of using computer or tablet at home using
# TIMSS 2015 grade 8 data loaded in memory, without shortcut, exclude the cases with missing
# values in the splitting variables, and use the default (TOTWGT) weights
## Not run:
lsa.pcts.prctls(data.object = T15_G8_student_data, split.vars = c("BSBG01", "BSBG13A"),
bckg.prctls.vars = "BSBGSLM", prctls = c(5, 25, 50),
output.file = "C:/temp/test.xlsx", open.output = TRUE)

## End(Not run)

# Repeat the analysis from above, this time with shortcut, include the cases with missing
# values in the splitting variables, and use the senate weights
## Not run:
lsa.pcts.prctls(data.object = T15_G8_student_data, split.vars = c("BSBG01", "BSBG13A"),
bckg.prctls.vars = "BSBGSLM", prctls = c(5, 25, 50), weight.var = "SENWGT",
include.missing = TRUE, shortcut = TRUE, output.file = "C:/temp/test.xlsx",
open.output = TRUE)

## End(Not run)

# Repeat the analysis from above, adding a second continuous variable to compute the
# percentiles for, the "Students Like Learning Science" complex scale
## Not run:
lsa.pcts.prctls(data.object = T15_G8_student_data, split.vars = c("BSBG01", "BSBG13A"),
bckg.prctls.vars = c("BSBGSLM", "BSBGSLS"), prctls = c(5, 25, 50), weight.var = "SENWGT",
include.missing = FALSE, shortcut = TRUE,
output.file = "C:/temp/test.xlsx", open.output = TRUE)

## End(Not run)

# Compute the 5th, 25th and 50th percentiles for the student overall reading achievement
# scores (i.e. using a set of PVs), using PIRLS 2016 student data file, split the output
# by student sex, use the full design, include the missing values od the splitting variable
# (i.e. student sex), and do not open the output after the computations are finished
## Not run:
lsa.pcts.prctls(data.file = "C:/Data/PIRLS_2016_Student_Miss_to_NA.RData",
split.vars = "ASBG01", PV.root.prctls = "ASRREA", prctls = c(5, 25, 50),
include.missing = TRUE, output.file = "C:/temp/test.xlsx", open.output = FALSE)

## End(Not run)
```

---

lsa.recode.vars *Recode variables in large-scale assessments' data sets*

Description

Utility function to recode variables in objects or data sets containing objects of class `lsa.data`, taking care of user-defined missing values, if specified.

Usage

```
lsa.recode.vars(
  data.file,
  data.object,
  src.variables,
  new.variables,
  old.new,
  new.labels,
  missings.attr,
  variable.labels,
  out.file
)
```

Arguments

`data.file`	Full path to the `.RData` file containing `lsa.data` object. Either this or `data.object` shall be specified, but not both. See details.
`data.object`	The object in the memory containing `lsa.data` object. Either this or `data.file` shall be specified, but not both. See details.
`src.variables`	Names of the source variables with the same class whose values shall be recoded. See details.
`new.variables`	Optional, vector of variable names to be created with the recoded values with the same length as `src.variables`. If missing, the `src.variables` will be overwritten.
`old.new`	String with the recoding instructions matching the length of the factor levels (or unique values in case of numeric or character variables) in the variables. See details and examples.
`new.labels`	The new labels if the `src.variables` variables are of class `factor` or labels to be assigned to the recoded values (i.e. turning variables of class `numeric` or `character` into factors) with the same length as the new desired values. See details.
`missings.attr`	Optional, list of character vectors to assign user-defined missing values for each recoded variable. See details and examples.
`variable.labels`	Optional, string vector with the new variable labels to be assigned. See details.
`out.file`	Full path to the `.RData` file to be written. If missing, the object will be written to memory. See examples.

Details

Before recoding variables of interest, it is worth running the `lsa.vars.dict` to check their properties.

Either data.file or data.object shall be provided as source of data. If both of them are provided, the function will stop with an error message.

The variable names passed to src.variables must be with the same class and structure, i.e. same number of levels and same labels in case of factor variables, or the same unique values in case of numeric or character variables. If the classes differ, the function will stop with an error. If the unique values and/or labels differ, the function would execute the recodings, but will drop a warning.

The new.variables is optional. If provided, the recoded values will be saved under the provided new variable names and the src.variables will remain unchanged. If missing, the variables passed in src.variables will be overwritten. Note that the number of names passed to src.variables and new.variables must be the same.

The old.new (old values to new values) is the recoding scheme to be evaluated and executed provided as a characters string in the form of "1=1;2=1;3=2;4=3". In this example it means "recode 1 into 1, 2 into one, 3 into 2, and 4 into 3". Note that all available values have to be included in the recoding statement, even if they are not to be changed. In this example, if we omit recoding 1 into 1, 1 will be set to NA during the recoding. This recoding definition works with factor and numeric variables. For character variables the individual values have to be defined in full, e.g. "'No time'='30 minutes or less';'30 minutes or less'='30 minutes or less';'More than 30 minutes'='More than 30 minutes';'Omitted or invalid'='Omitted or invalid'" because these cannot be reliably referred to by position (as for factors) or actual number (as for numeric).

The new.labels assigns new labels to factor variables. Their length must be the same as for the newly recoded values. If the variables passed to src.variabes are character or numeric, and new.labels are provided, the recoded variables will be converted to factors. If, on the other hand, the src.variables are factors and no new.labels are provided, the variables will be converted to numeric.

Note that the lsa.convert.data has two options: keep the user-defined missing values (missing.to.NA = FALSE) and set the user-defined missing values to NA (missing.to.NA = TRUE). The former option will provide an attribute with user-defined missing values attached to each variable they have been defined for, the latter will not (i.e. will assign all user-defined missing values to NA). In case variables from data converted with the former option are recoded, user-defined missing values have to be supplied to missings.attr, otherwise (if all available values are recoded) the user-defined missing values will appear as valid codes. Not recoding the user-defined missing codes available in the data will automatically set them to NA. In either case, the function will drop a warning. On the other hand, if the data was exported with missing.to.NA = TRUE, there will be no attributes with user-defined missing codes and omitting missings.attr will issue no warning. User-defined missing codes can, however, be added in this case too, if necessary. The missings.attr has to be provided as a list where each component is a vector with the values for the missing codes. See the examples.

The variable.labels argument provides the variable labels to be assigned to the recoded variables. If omitted and new.variables are provided the newly created variables will have no variable labels. If provided, and new.variables are not provided, they will be ignored. If full path to .RData file is provided to out.file, the data.set will be written to that file. If no, the data will remain in the memory.

Value

A `lsa.data` object in memory (if `out.file` is missing) or `.RData` file containing `lsa.data` object with the recoded values for the specified variables. In addition, the function will print tables for the specified variables before and after recoding them to check if all recodings were done as intended. In addition, it will print warnings if different issues have been encountered.

See Also

`lsa.convert.data`, `lsa.vars.dict`

Examples

```
# Recode PIRLS 2016 student variables "ASBG10A" (How much time do you spend using a computer or
# tablet to do these activities for your schoolwork on a normal school day? Finding and reading
# information) and "ASBG10B" (How much time do you spend using a computer or tablet to do these
# activities for your schoolwork on a normal school day? Preparing reports and presentations).
# Both variables are factors, the original valid values (1 - "No time",
# 2 - "30 minutes or less", 3 - "More than 30 minutes") are recoded to
# 1 - "No time or 30 minutes" and 2 - "More than 30 minutes", collapsing the first two.
# The missing value "Omitted or invalid" which originally appears as 4th has to be recoded to
# 3rd. The "Omitted or invalid" is assigned as user-defined missing value for both variables.
# The data is saved on disk as a new data set.
## Not run:
lsa.recode.vars(data.file = "C:/temp/test.RData", src.variables = c("ASBG10A", "ASBG10B"),
new.variables = c("ASBG10A_rec", "ASBG10B_rec"),
variable.labels = c("Recoded ASBG10A", "Recoded ASBG10B"),
old.new = "1=1;2=1;3=2;4=3",
new.labels = c("No time or 30 minutes", "More than 30 minutes", "Omitted or invalid"),
missings.attr = list("Omitted or invalid", "Omitted or invalid"),
out.file = "C:/temp/test_new.RData")

## End(Not run)

# Similar to the above, recode PIRLS 2016 student variables "ASBG10A" and "ASBG10B", this time
# leaving the original categories (1 - "No time", 2 - "30 minutes or less",
# 3 - "More than 30 minutes") as they are, but changing the user-defined missing values
# definition (1 - "No time" becomes user-defined missing).
# The recoded data remains in the memory.
## Not run:
lsa.recode.vars(data.file = "C:/temp/test.RData", src.variables = c("ASBG10A", "ASBG10B"),
new.variables = c("ASBG10A_rec", "ASBG10B_rec"),
variable.labels = c("Recoded ASBG10A", "Recoded ASBG10B"), old.new = "1=1;2=2;3=3;4=4",
new.labels = c("No time", "30 minutes or less", "More than 30 minutes", "Omitted or invalid"),
missings.attr = list(c("No time", "Omitted or invalid"), c("No time", "Omitted or invalid")))

## End(Not run)

# Similar to the first example, this time overwriting the original variables. The first valid
# value (1 - "No time") is set to NA (note that no new value and factor level is provided for
# it in "new.labels"), the rest of the values are redefined, so the factor starts from 1,
# as it always does in R.
## Not run:
```

```
lsa.recode.vars(data.file = "C:/temp/test.RData", src.variables = c("ASBG10A", "ASBG10B"),
variable.labels = c("Recoded ASBG10A", "Recoded ASBG10B"), old.new = "2=1;3=2;4=3",
new.labels = c("30 minutes or less", "More than 30 minutes", "Omitted or invalid"),
missings.attr = list("Omitted or invalid"),
out.file = "C:/temp/test_new.RData")

## End(Not run)
# The databases rarely contain character variables and the numeric variables have too many
# unique values to be recoded using the function. The following two examples are just for
# demonstration purpose on how to recode character and numeric variables.

# Convert the "ASBG04" (number of books at home) from ePIRLS 2016 to numeric and recode the
# values of the new variable, collapsing the first two and the last two valid values.
# The data remains in the memory.
## Not run:
load("/tmp/test.RData")
test[ , ASBG04NUM := as.numeric(ASBG04)]
table(test[ , ASBG04NUM])
lsa.recode.vars(data.object = test, src.variables = "ASBG04NUM",
old.new = "1=1;2=1;3=2;4=3;5=3;6=4",
missings.attr = list("Omitted or invalid" = 4))

# Similar to the above, this time converting "ASBG03" to character, collapsing its categories
# of frequency of using the test language at home to two ("Always or almost always" and
# "Sometimes or never").
\dontrun{
load("/tmp/test.RData")
test[ , ASBG03CHAR := as.character(ASBG03)]
table(test[ , ASBG03CHAR])
# Add the lines together to be able to run the following
lsa.recode.vars(data.object = test, src.variables = "ASBG03CHAR",
old.new = "'I always speak <language of test> at home'='Always or almost always';
'I almost always speak <language of test> at home'='Always or almost always';
'I sometimes speak <language of test> and sometimes speak another language at home'=
'Sometimes or never';'I never speak <language of test> at home'='Sometimes or never';
'Omitted or invalid'='Omitted or invalid'",
missings.attr = list("Omitted or invalid"))
}


## End(Not run)
```

`lsa.vars.dict` *Produce dictionary for large-scale assessments data variables*

Description

Utility function to display dictionaries of variables from data sets containing objects of class `lsa.data`.

Usage

```
lsa.vars.dict(
  data.file,
  data.object,
  var.names,
  out.file,
  open.out.file = FALSE
)
```

Arguments

`data.file`	Full path to the `.RData` file containing `lsa.data` object. Either this or `data.object` shall be specified, but not both. See details.
`data.object`	The object in the memory containing `lsa.data` object. Either this or `data.file` shall be specified, but not both. See details.
`var.names`	Vector of variable names whose dictionaries shall be produced. See details.
`out.file`	Optional, full path to a `.txt` file where the dictionaries shall be saved, if needed. See details.
`open.out.file`	Optional, if file path is provided to `out.file` shall the produced file be open after the file is written?

Details

Either `data.file` or `data.object` shall be provided as source of data. If both of them are provided, the function will stop with an error message.

If `var.names` are not provided, then the function will produce dictionaries for all variables in the file/object.

The function will print the dictionaries on the screen. If these need to be saved to a file for further reference as well, a full path to the `.txt` file shall be provided. If the file exists, it will be overwritten. If the file name is provided to `out.file` and `open.out.file = TRUE`, it will be automatically open in the default text editor after being written.

Value

The dictionaries for the variables in `var.names` will be printed as tables on the screen. For each variable the dictionaries contain the variable name, the variable class, the variable label, unique variable values (see below) and the user-defined missing values (if any).

The unique values' representation will depend on the variable class. If the variable is a factor, the factor levels will be displayed. If the variable is numeric or character, the unique values will be printed up to the sixth one.

The user-defined missing values for factor variables will be as text strings. For the numeric variables these will be integers, followed by their labels in brackets.

If a full file path is provided to the `out.file`, the same output will be written to a `.txt` file with a text on top which data file/object was used.

See Also

lsa.convert.data, lsa.recode.vars

Examples

```
# Display and write to file the dictionaries for multiple factor and numeric variables using
# PIRLS 2016 file with teacher and student data from several countries and open the file after
# it has been written to the disk.
## Not run:
lsa.vars.dict(data.file = "C:/temp/test.RData", var.names = c("ASBG10A", "ASBG10B", "ASBG05A",
"ASBG05B", "ASBG05C", "ASBG05D", "ASBG05E", "ASBG05F", "ASBG05G", "ASBG05H", "ASBG06",
"ASBG07A", "ASBG07B", "ASBG08", "ATBG05BA", "ATBG05BB", "ATBG05BC", "ATBG05BD"),
out.file = "C:/temp/dict.txt", open.out.file = TRUE)

## End(Not run)

## Not run:
lsa.vars.dict(data.object = test, var.names = c("ASBG10A", "ASBG10B", "ASBG05A", "ASBG05B",
"ASBG05C", "ASBG05D", "ASBG05E", "ASBG05F", "ASBG05G", "ASBG05H","ASBG06", "ASBG07A",
"ASBG07B", "ASBG08", "ATBG05BA", "ATBG05BB", "ATBG05BC", "ATBG05BD"),
out.file = "C:/temp/dict.txt", open.out.file = TRUE)

## End(Not run)
```

RALSA *R Analyzer for Large-Scale Assessments (RALSA)*

Description

The RALSA package provides functionality for analyzing data from large-scale assessments and surveys which use complex sampling and assessment design. Such (international) assessments and surveys are TIMSS, PIRLS and PISA, for example.

The sampling (complex sampling design) in large-scale assessments and surveys is multistage with probability proportional to the size of the (primary) sampling units (usually schools), i.e. with unequal probabilities of selection. Thus, all weights assigned to the individual respondents reflect these unequal probabilities. This is quite different from the usual simple or systematic random sampling. Different modifications of Jackknife Repeated Replication (JRR, with full or half replication) or Balanced Repeated Replication (BRR) are used in different studies to compute the standard errors of the population estimates. The proficiency test scores (complex assessment design) is applied to cope with practical issues. No respondent takes all test items, but the items are distributed across multiple test item blocks and the blocks are rotated across multiple assessment booklets, each respondent taking one booklet only. As a consequence, no respondent receives a single test score, but five (or even 10) separate test scores (called "plausible values" or PVs) resulting from multiple imputation technique where the missing by design responses are imputed. As a consequence of the complex sampling and assessment designs, each estimate has to be computed with each JRR or BRR weight and each PV (this can take up to 781 computations per estimate per group per

country, depending on the study), then summarized to compute the final estimate, its sampling and imputation variance, and the final standard error.

RALSA provides data preparation and analysis functions which take into account the complex sampling and assessment design of the studies. Each study has its a different implementation of the complex sampling and assessment designs and RALSA handles these and implements the corresponding computational procedure.

Studies

Currently, RALSA works with data from **all cycles** of the the following studies:

- IEA CivED;
- IEA ICCS;
- IEA ICILS;
- IEA RLII;
- IEA PIRLS (including PIRLS Literacy and ePIRLS);
- IEA TIMSS (including TIMSS Numeracy, eTIMSS);
- IEA TiPi (TIMSS and PIRLS joint study);
- IEA TIMSS Advanced;
- IEA SITES;
- IEA TEDS-M;
- IEA REDS;
- OECD PISA;
- OECD PISA for Development;
- OECD TALIS; and
- OECD TALIS Starting Strong Survey (a.k.a. TALIS 3S).

More studies (national international) will be added in future.

Functions

Currently, RALSA provides the following functionality:

- Data preparation functions - prepare data for analysis
  - `lsa.convert.data` The studies provide their data in SPSS and SAS format. In addition, PISA cycles prior to 2015 provide the data in `.TXT` format, along with their SPSS and SAS import syntax files. This function takes the originally provided SPSS data (or `.TXT`, along with the import syntaxes) files and converts them into native `.RData` files. It also adds variable labels, user-defined missing codes (if requested) and identifiers of the study, cycle, and respondent types (i.e. student, parent, teacher, school principal).
  - `lsa.select.countries.PISA` Utility function to select countries' data from a converted PISA file and save them as a new file or assign them to an object in memory. This makes it more convenient to work with PISA data files which contain data from all countries per respondent type.

  - `lsa.merge.data` The studies provide data from different respondents (i.e. student, parent, teacher, school principal) which are sampled hierarchically (e.g. students are nested in classes, classes are nested in schools and taught by teachers) and linked between each other. The files in the databases are provided separately per country and respondent type. This function merges data sets from different respondents and/or countries assuring the links between the different types of respondents (i.e. linking students only to principals' data only for their school and to the teachers who teach them). This function merges data for all studies, except for PISA where the structure of the files does not allow (for now) merging data from different respondent types.
  - `lsa.vars.dict` Prints and/or saves variable dictionaries in a file. Convenient when need to know the structure of the variables of interest.
  - `lsa.data.diag` Helper function for quick frequency (for categorical variables) and descriptive (continuous variables) tables (weighted or unweighted). These can serve for initial exploration of the data and elaborating hypotheses. Not intended for actual analysis.
  - `lsa.recode.vars` Recodes variables from large-scale assessments taking care of the user-defined missing values. Convenient for collapsing categories or changing their order.
- Analysis functions - estimates are on population level, taking into account the complex sampling and assessment design
  - `lsa.pcts.means` Computes percentages of respondents and means (arithmetic average, median or mode) for continuous variables within groups
  - `lsa.prctls` Computes percentiles of continuous variables within groups
  - `lsa.bench` Computes percentages of respondents reaching or surpassing benchmarks of achievement
  - `lsa.crosstabs` Crosstabulations with Rao-Scott first- and second-order chi-square adjustments
  - `lsa.corr` Computes correlations between variables (Pearson or Spearman)
  - `lsa.lin.reg` Computes linear regression with or without contrast coding of categorical variables
  - `lsa.bin.log.reg` Computes binary logistic regression with or without contrast coding of categorical variables

The `lsa.pcts.means`, `lsa.prctls`, `lsa.bench` and `lsa.crosstabs` also have the option to produce graphs from the estimates.

More studies and analysis types will be added in future, and the existing ones will be updated, adding more features.

RALSA also has a Graphical User Interface (GUI) for the less technical users. The GUI incorporates all aspects of the data preparation and analysis functions.

References

Here are the two articles presenting the package and it's technical details:

Mirazchiyski, P.V. (2021). RALSA: The R analyzer for large-scale assessments. *Large-scale Assess Educ 9*(21), 1-24. https://doi.org/10.1186/s40536-021-00114-4

Mirazchiyski, P. V. (2021). RALSA: Design and Implementation. *Psych, 3*(2), 233-248. https://doi.org/10.3390/psych302001

Here is a list of selected references related to some of the studies' design, relevant to their latest cycles:

Foy, P., & LaRoche, S. (2017). Estimating Standard Errors in the PIRLS 2016 Results. In M. O. Martin, I. V. S. Mullis, & M. Hooper (Eds.), *Methods and Procedures in PIRLS 2016* (p. 4.1-4.22). Lynch School of Education, Boston College.

Foy, P., & Yin, L. (2016). TIMSS 2015 Achievement Scaling Methodology. In M. O. Martin, I. V. S. Mullis, & M. Hooper (Eds.), *Methods and Procedures in TIMSS 2015* (p. 13.1-13.62). TIMSS & PIRLS International Study Center.

LaRoche, S., Joncas, M., & Foy, P. (2016). Sample Design in TIMSS 2015. In M. O. Martin, I. V. S. Mullis, & M. Hooper (Eds.), *Methods and Procedures in TIMSS 2015* (p. 3.1-3.37). TIMSS & PIRLS International Study Center.

OECD. (in press). *PISA 2018 Technical Report*. OECD.

Rutkowski, L., Gonzalez, E., Joncas, M., & von Davier, M. (2010). International Large-Scale Assessment Data: Issues in Secondary Analysis and Reporting. *Educational Researcher, 39*(2), 142-151.

Rutkowski, L., Rutkowski, D., & von Davier, M. (2014). A Brief Introduction to Modern International Large-Scale Assessment. In L. Rutkowski, M. von Davier, & D. Rutkowski (Eds.), *Handbook of International Large-Scale Assessments: Background, Technical Issues, and Methods of Data Analysis* (pp. 3-10). CRC Press.

`ralsaGUI` *Start RALSA's Graphical User Interface (GUI)*

Description

Starts the GUI. The GUI contains elements for each parameter of each function (data preparation or analysis).

Usage

```
ralsaGUI()
```

Index