Package 'eatRep'

July 22, 2025

Type Package

Title Educational Assessment Tools for Replication Methods

Version 0.15.2

Depends R (>= 4.1), survey (>= 4.1-1), BIFIEsurvey, progress, lavaan (>= 0.6-7), parallel

Imports Hmisc, fmsb, mice (>= 2.46), boot, car, reshape2, plyr, combinat, miceadds, tidyr, EffectLiteR, estimatr, eatTools (>= 0.7.8), eatGADS (>= 0.20.0), janitor, msm, lme4, utils, methods, checkmate, lifecycle, dplyr (>= 1.0.0), future

Description
Replication methods to compute some basic statistic operations (means, standard deviations, frequency tables, percentiles, mean comparisons using weighted effect coding, generalized linear models,
and linear multilevel models) in complex survey designs comprising multiple imputed or nested imputed
variables and/or a clustered sampling structure which both deserve special procedures at least in estimating standard errors. See the package documentation for a more detailed description along with references.

License GPL (>= 2)

Encoding UTF-8

URL https://github.com/weirichs/eatRep, https://weirichs.github.io/eatRep/

LazyLoad yes

LazyData yes

NeedsCompilation no

Suggests weights, testthat, knitr, rmarkdown, zoo

VignetteBuilder knitr

Author Sebastian Weirich [aut, cre], Martin Hecht [aut], Karoline Sachse [aut], Benjamin Becker [aut], Edna Grewers [ctb]

Maintainer Sebastian Weirich <sebastian.weirich@iqb.hu-berlin.de>

Repository CRAN

Date/Publication 2025-04-30 07:50:10 UTC

Contents

eatRep-package 2
checkLEs 5
generateRandomJk1Zones 6
lsa 6
pool.R2 8
repGlm 9
repLmer 13
repMean 15
report 24
repQuantile 26
repTable 29

Index **36**

`eatRep-package` *Statistical analyses in complex survey designs with multiple imputed data and trend estimation.*

Description

The package provide functions to computes some basic statistic operations—(adjusted) means, standard deviations, frequency tables, percentiles and generalized linear models—in complex survey designs comprising multiple imputed variables and/or a clustered sampling structure which both deserve special procedures at least in estimating standard errors. In large-scale assessments, standard errors are comprised of three components: the measurement error, the sampling error, and (if trend estimation of at least two times of measurement are involved) the linking error.

Measurement error: In complex surveys or large-scale assessments, measurement errors are taken into account by the mean of multiple imputed variables. The computation of standard errors for the mean of a multiple imputed variable (e.g. plausible values) involves the formulas provided by Rubin (1987). Computing standard errors for the mean of a nested imputed variable involves the formulas provided by Rubin (2003). Both methods are implemented in the package. The estimation of R^2 and adjusted R^2 in linear and generalized linear regression models with multiple imputed data sets is realized using the methods provided in Harel (2009).

Sampling error: Computation of sampling errors of variables which stem from a clustered design may involve replication methods like balanced repeated replicate (BRR), bootstrap or Jackknife methods. See Westat (2000), Foy, Galia & Li (2008), Rust and Rao (1996), and Wolter (1985) for details. To date, the Jackknife-1 (JK1), Jackknife-2 (JK2) and the Balanced Repeated Replicates (BRR; optionally with Fay's method) procedures are supported.

Linking error: Lastly, standard errors for trend estimates may involve incorporating linking errors to account for potential differential item functioning or item parameter drift. `eatRep` allows

to account for linking error when computing standard errors for trend estimates. Standard error estimation is conducted according to the operational practice in PISA, see equation 5 in Sachse & Haag (2017).

The package `eatRep` is designed to combine one or several error types which is necessary, for example, if (nested) multiple imputed data are used in clustered designs. Considering the structure is relevant especially for the estimation of standard errors. The estimation of national trends requires a sequential analysis for both measurements and a comparison of estimates between them.

Technically, `eatRep` is a wrapper for the `survey` package (Lumley, 2004). Each function in `eatRep` corresponds to a specific function in `survey` which is called repeatedly during the analysis. Hence, a nested loop is used. We use "trend replicates" in the outer loop, "imputation replicates" in the middle loop to account for multiple imputed data, and "cluster replicates" in the inner loop to account for the clustered sampling structure. While the functional principle of `survey` is based on replication of standard analyses, `eatRep` is based on replication of `survey` analyses to take multiple imputed data into account. More recent versions of the package additionally allow estimations using the `BIFIEsurvey` package instead of `survey` which provide substantial advantages in terms of speed.

For each imputed data set in each measurement, i.e. in the inner loop, the `eatRep` function first creates replicate weights based on the primary sampling unit (PSU) variable and the replication indicator variable. In the jackknife procedure, the first one is often referred to as "jackknife zone", whereas the second one is often referred to as "jackknife replicate". The number of distinct units in the PSU variable defines the number of replications which are necessary due to the clustered structure. A design object is created and the appropriate `survey` function is called. The process is repeated for each imputed dataset and the results of the analyses are pooled. The pooling procedure varies in relation to the type of variable to be pooled. For examples, means or regression coefficients are pooled according to Rubin (1987) or Rubin (2003). R^2 is pooled according to Harel (2009), using a Fisher z-transformation. Chi-square distributed values are pooled according to Thomas and Rao (1990) for clustered data and according to Enders (2010) and Allison (2002) for multiple imputed data. For trend analyses, the whole process is repeated two times (according to the two measurements) and the difference of the estimates are computed along with their pooled standard errors.

Without trend estimation, the outer loop has only one cycle (instead of two). Without multiple imputations, the middle loop has only one cycle. Without a clustered sampling structure (i.e, in a random sample), the inner loop has only one cycle. Without trend, imputation and clustered structure, no replication is performed at all. To compute simple mean estimates, for example, `eatRep` then simply calls `mean` instead of `svymean` from the `survey` package. A special case occurs with nested multiple imputation. We then have four loops in a nested structure. Hence, the corresponding analyses may take considerably computational effort.

Important note: Starting with version 0.10.0, several methods for the standard error estimation of cross level differences are implemented. Prior to version 0.10.0, the standard error for the difference between one single group (e.g., Belgium) and the total population (which is comprised of several states including Belgium) was estimated as if both groups would have been independent from each other. The standard errors, however, are biased then. Two new methods are now applicable using the argument `crossDiffSE` in `repMean` and provide unbiased standard errors—weighted effect coding (wec) and replication methods (rep); see, for example te Grotenhuis et al. (2017) and Weirich et al. (2021). The old method is still available by using `crossDiffSE = "old"`. Note that the default method now is weighted effect coding.

Second important note: Starting with version 0.13.0, function names have been changed due to

inconsistent former denomination: Function `jk2.mean` now goes under the name of `repMean`, `jk2.table` was renamed to `repTable`, `jk2.quantile` was renamed to `repQuantile`, and `jk2.glm` now goes under the name of `repGlm`. The old functions are deprecated and will be removed in further package publications. Renaming was driven by the fact that the corresponding functions now have broader range of methods than only jackknife-2.

Third important note: Starting with version 0.15.0, the reporting function `report` was deprecated due to inefficient and error-prone programming. The new reporting function `report2` has a new output format which provides an interface for the `eatPlot` package. Old functionality was roughly supplied using `report`, but if the 1:1 output of former version is requested, please use version 0.14.7.

Details

Package:	eatRep
Type:	Package
Version:	0.15.2
Date:	2025-04-29
License:	GPL(>=2)

Author(s)

Authors: Sebastian Weirich <sebastian.weirich@iqb.hu-berlin.de>, Martin Hecht <martin.hecht@hu-berlin.de>, Benjamin Becker <b.becker@iqb.hu-berlin.de>

References

Allison, P. D. (2002). Missing data. Newbury Park, CA: Sage.

Enders, C. K. (2010). Applied missing data analysis. Guilford Press.

Foy, P., Galia , J. & Li, I. (2008). Scaling the data from the TIMSS 2007 mathematics and science assessment. In J. F. Olson, M. O. Martin & I. V. S. Mullis (ed.), *TIMSS 2007 Technical Report* (S. 225–280). Chestnut Hill, MA: TIMSS & PIRLS International Study Center, Lynch School of Education, Boston College.

Harel, O. (2009): The estimation of R^2 and adjusted R^2 in incomplete data sets using multiple imputation. *Journal of Applied Statistics.* **36, 10**, 1109–1118.

Lumley, T. (2004). Analysis of complex survey samples. *Journal of Statistical Software* **9(1)**: 1–19

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys.* New York: Wiley.

Rubin, D.B. (2003): Nested multiple imputation of NMES via partially incompatible MCMC. *Statistica Neerlandica* **57, 1**, 3–18.

Rust, K., & Rao, JNK. (1996): Variance estimation for complex surveys using replication techniques. *Statistical Methods in Medical Research* **5**, 283–310.

Sachse, K. A. & Haag, N. (2017). Standard errors for national trends in international large-scale assessments in the case of cross-national differential item functioning. *Applied Measurement in Education, 30*, (2), 102-116. http://dx.doi.org/10.1080/08957347.2017.1283315

Satorra, A., & Bentler, P. M. (1994). Corrections to test statistics and standard errors in covariance structure analysis.

te Grotenhuis, M., Pelzer, B., Eisinga, R., Nieuwenhuis, R., Schmidt-Catran, A., & Konig, R. (2017). When size matters: advantages of weighted effect coding in observational studies. *International Journal of Public Health.* **62**, 163–167.

Thomas, D. R. & Rao, JNK (1990): Small-sample comparison of level and power for simple goodness-of- fit statistics under cluster sampling. JASA 82:630-636

Weirich, S., Hecht, M., Becker, B. et al. (2021). Comparing group means with the total mean in random samples, surveys, and large-scale assessments: A tutorial and software illustration. Behavior Research Methods. https://doi.org/10.3758/s13428-021-01553-1

Westat (2000). *WesVar.* Rockville, MD: Westat.

Wolter, K. M. (1985). *Introduction to variance estimation.* New York: Springer.

`checkLEs` *Checks compatibility of linking errors with GADS data bases.*

Description

This function checks if a linking error `data.frame` is compatible with multiple trend `eatGADS` data bases.

Usage

```
checkLEs(filePaths, leDF)
```

Arguments

`filePaths`	Character vectors with at least two paths to the `eatGADS` db files.
`leDF`	Linking error `data.frame`.

Details

This function inspects whether all linking error variables correspond to variables in the `eatGADS` data base and if the key variables also correspond to existing variables in the trend `eatGADS` data bases.

Value

Returns a report list.

Examples

```
trenddat1 <- system.file("extdata", "trend_gads_2010.db", package = "eatGADS")
trenddat2 <- system.file("extdata", "trend_gads_2015.db", package = "eatGADS")
trenddat3 <- system.file("extdata", "trend_gads_2020.db", package = "eatGADS")
load(system.file("extdata", "linking_error.rda", package = "eatRep"))
check1 <- checkLEs(c(trenddat1, trenddat2, trenddat3), lErr)
check2 <- checkLEs(c(trenddat1, trenddat2, trenddat3), lErr[1:14,])
```

generateRandomJk1Zones

Generates random jackknife-1 zones based on sampling units in the data set.

Description

Function adds randomly generated jackknife-1 zones to the data.

Usage

```
generateRandomJk1Zones (datL, unit, nZones, name = "randomCluster")
```

Arguments

datL	Data frame containing at least the primary sampling unit variable
unit	Variable name or column number of the primary sampling unit (i.e. student or class identifier)
nZones	integer: number of jackknife zones. Note: The umber of jackknife zones must not exceed the number of distinct sampling units
name	New name of the jackknife-zone variable in the data set

Value

The original data with an additional column of the jackknife-zone variable

Examples

```
data(lsa)

### We only consider year 2010
lsa10<- lsa[which(lsa[,"year"] == 2010),]
lsa10<- generateRandomJk1Zones(datL = lsa10, unit="idclass", nZones = 50)
```

lsa

Achievement data from two large-scale assessments of 2010 and 2015.

Description

This example data set contains fictional achievement scores of 11637 students from three countries and two times of measurement in two domains (reading and listening comprehension) in the long format. The data set contains nested multiple imputed plausible values of achievement scores as well as some demographic variables. Illustrating trend analyses, data from two fictional time points (2010 and 2015) are included.

The data set can be used for several illustration purposes. For example, if only multiple imputation should be considered (without nesting), simply use only cases from the first nest (by subsetting). If only one time of measurement should be considered (i.e., without any trend analyses), simply choose only cases from 2010 or 2015. If only reading or listening should be considered, choose the desired domain by subsetting according to the `domain` column.

Usage

```
data(lsa)
```

Format

'data.frame': 77322 obs. of 25 variables

year Year of evaluation

idstud individual student identification

idclass class identifier

wgt Total case weight

L2wgt School weight (level 2 weight)

L1wgt Student weight (level 1 weight)

jkzone jackknifing zone (jk2)

jkrep jackknife replicate

imp Number of imputation

nest Number of nest (for nested imputation only)

country The country an examinee stems from

sex student's sex

ses student's socio-economical status

mig student's migration background

domain The domain the corresponding score belongs to

score student's achievement score (corresponding to the domain reading or listening, and to the imputation 1, 2, or 3)

comp student's competence level

failMin dichotomous indicator whether the student fails to fulfill the minimal standard

passReg dichotomous indicator whether the student fulfills at least the regular standard

passOpt dichotomous indicator whether the student fulfills the optimal standard

leSore linking error of each student's achievement score

leComp linking error of each student's competence level

leFailMin linking error of each student's indicator of failing to fulfill the minimal standard

lePassReg linking error of each student's indicator of fulfilling the regular standard

lePassOpt linking error of each student's indicator of fulfilling the optimal standard

Source

Simulated data

pool.R2 *Compute R^2 in multiple imputed and nested multiple imputed data*

Description

With (nested) multiple imputations, the determination coefficient R^2 has to be computed for each imputed data set and pooled afterwards. pool.R2 provide pooling routines according to Harel (2009). The function requires that the R^2 coefficients from the multiple imputed analyses are already available.

Usage

```
pool.R2 ( r2, N, verbose = TRUE )
```

Arguments

r2	For multiple imputed data, a numeric vector of R^2 values. For nested multiple imputed data, a list of numeric vectors of R^2 values. The number of list elements must correspond to the number of nests. The number of R^2 values within each list element must be equal and must correspond to the number of imputations within each nest.
N	Optional: the sample size of each imputed data set. Only necessary if the standard error for the pooled R^2 should be computed. The structure of the N object must correspond to the structure of the r2 object. See examples for further details.
verbose	Optional: Print additional messages to console?

Value

Returns a data.frame with one or two columns which contains the pooled R^2 value and optionally it's standard error.

References

Harel, O. (2009): The estimation of R^2 and adjusted R^2 in incomplete data sets using multiple imputation. *Journal of Applied Statistics.* **36, 10**, 1109–1118.

Examples

```
# multiple imputation, assume that the regression analysis was fitted for five imputed data sets,
# resulting in five R^2 values. Assume sample sizes of 340
r2 <- c(0.12395, 0.15261, 0.16125, 0.11029, 0.1871)
Ns <- rep(340,5)
pool.R2 ( r2=r2, N=Ns)
# without standard error
pool.R2 ( r2=r2)
# nested multiple imputation
r2 <- list(nest1 = c(0.12395, 0.15261, 0.16125, 0.11029, 0.1871),
```

```
            nest2 = c(0.10603, 0.08876, 0.09248, 0.13331, 0.1114),
            nest3 = c(0.17228, 0.25203, 0.13132, 0.23331, 0.10069))
Ns <- lapply(1:3, FUN = function (x) {rep(290, 5)})
pool.R2 ( r2=r2, N=Ns)
# without standard error
pool.R2 ( r2=r2)
```

---

repGlm *Replication methods (JK1, JK2 and BRR) for linear regression models and trend estimation.*

Description

Compute generalized linear models for complex cluster designs with multiple imputed variables based on the Jackknife (JK1, JK2) or balanced repeated replicates (BRR) procedure. Conceptually, the function combines replication methods and methods for multiple imputed data. Technically, this is a wrapper for the svyglm function of the survey package.

Usage

```
repGlm(datL, ID, wgt = NULL, type = c("none", "JK2", "JK1", "BRR", "Fay"), PSU = NULL,
        repInd = NULL, repWgt = NULL, nest=NULL, imp=NULL, groups = NULL,
        group.splits = length(groups), group.delimiter = "_",
        cross.differences = FALSE, trend = NULL, linkErr = NULL, formula,
        family=gaussian, forceSingularityTreatment = FALSE,
        glmTransformation = c("none", "sdY"), doCheck = TRUE, na.rm = FALSE,
        poolMethod = c("mice", "scalar"), useWec = FALSE,
        scale = 1, rscales = 1, mse=TRUE, rho=NULL, hetero=TRUE,
        se_type = c("HC3", "HC0", "HC1", "HC2", "CR0", "CR2"),
        clusters = NULL, crossDiffSE.engine= c("lavaan", "lm"),
        stochasticGroupSizes = FALSE, verbose = TRUE, progress = TRUE,
        nCores=NULL)
```

Arguments

datL	Data frame in the long format (i.e. each line represents one ID unit in one imputation of one nest) containing all variables for analysis.
ID	Variable name or column number of student identifier (ID) variable. ID variable must not contain any missing values.
wgt	Optional: Variable name or column number of weighting variable. If no weighting variable is specified, all cases will be equally weighted.
type	Defines the replication method for cluster replicates which is to be applied. Depending on type, additional arguments must be specified (e.g., PSU and/or repInd or repWgt).
PSU	Variable name or column number of variable indicating the primary sampling unit (PSU). When a jackknife procedure is applied, the PSU is the jackknife zone variable. If NULL, no cluster structure is assumed and standard errors are computed according to a random sample.

repInd — Variable name or column number of variable indicating replicate ID. In a jackknife procedure, this is the jackknife replicate variable. If NULL, no cluster structure is assumed and standard errors are computed according to a random sample.

repWgt — Normally, replicate weights are created by repGlm directly from PSU and repInd variables. Alternatively, if replicate weights are included in the data.frame, specify the variable names or column number in the repWgt argument.

nest — Optional: name or column number of the nesting variable. Only applies in nested multiple imputed data sets.

imp — Optional: name or column number of the imputation variable. Only applies in multiple imputed data sets.

groups — Optional: vector of names or column numbers of one or more grouping variables.

group.splits — Optional: If groups are defined, group.splits optionally specifies whether analysis should be done also in the whole group or overlying groups. See examples for more details.

group.delimiter — Character string which separates the group names in the output frame.

cross.differences — Either a list of vectors, specifying the pairs of levels for which cross-level differences should be computed. Alternatively, if TRUE, cross-level differences for all pairs of levels are computed. If FALSE, no cross-level differences are computed. (see examples 2a, 3, and 4 in the help file of the repMean function)

trend — Optional: name or column number of the trend variable which contains the measurement time of the survey. Note: Levels of all grouping variables and predictors must be equal in all 'sub populations' partitioned by the discrete trend variable. repGlm computes differences for all pairwise contrasts defined by trend variable levels. or three measurement occasions, i.e. 2010, 2015, and 2020, contrasts (i.e. trends) are computed for 2010 vs. 2015, 2010 vs. 2020, and 2015 vs. 2020.

linkErr — Optional: name or column number of the linking error variable. If NULL, a linking error of 0 will be assumed in trend estimation.

formula — Model formula, see help page of glm for details.

family — A description of the error distribution and link function to be used in the model. See help page of glm for details.

forceSingularityTreatment — Logical: Forces the function to use the workaround to handle singularities in regression models.

glmTransformation — Optional: Allows for transformation of parameters from linear regression and logistic regression before pooling. Useful to compare parameters from different glm models, see Mood (2010). Note: This argument applies only if forceSingularityTreatment is set to 'TRUE'.

doCheck — Logical: Check the data for consistency before analysis? If TRUE groups with insufficient data are excluded from analysis to prevent subsequent functions from crashing.

na.rm	Logical: Should cases with missing values be dropped?
poolMethod	Which pooling method should be used? The "mice" method is recommended.
useWec	Logical: use weighted effect coding?
scale	scaling constant for variance, for details, see help page of svrepdesign from the survey package
rscales	scaling constant for variance, for details, see help page of svrepdesign from the survey package
mse	Logical: If TRUE, compute variances based on sum of squares around the point estimate, rather than the mean of the replicates. See help page of svrepdesign from the survey package for further details.
rho	Shrinkage factor for weights in Fay's method. See help page of svrepdesign from the survey package for further details.
hetero	Logical: Assume heteroscedastic variance for weighted effect coding? Only applies for random samples, i.e. if no replication analyses are executed.
se_type	The sort of standard error sought for cross level differences. Only applies if crossDiffSE == "wec" and hetero == TRUE and crossDiffSE.engine == "lm". See the help page of lm_robust from the estimatr package for further details.
clusters	Optional: Variable name or column number of cluster variable. Only necessary if weighted effecting coding should be performed using heteroscedastic variances. See the help page of lm_robust from the estimatr package for further details.
crossDiffSE.engine	Optional: Sort of estimator which should be used for standard error estimation in weighted effect coding regression. Only applies if useWec == TRUE. To date, only lavaan allows for stochastic group sizes.
stochasticGroupSizes	Logical: Assume stochastic group sizes for using weighted effect coding regression with categorical predictors? Note: To date, only lavaan allows for stochastic group sizes. Stochastic group sizes cannot be assumed if any replication method (jackknife, BRR) is applied.
verbose	Logical: Show analysis information on console?
progress	Logical: Show progress bar on console?
nCores	integer (default: NULL), number of cores to use for parallel processing, if engine = "survey". If NULL, single core processing is used.

Details

Function first creates replicate weights based on PSU and repInd variables according to JK2 or BRR procedure. According to multiple imputed data sets, a workbook with several analyses is created. The function afterwards serves as a wrapper for svyglm implemented in the survey package. The results of the several analyses are then pooled according to Rubin's rule, which is adapted for nested imputations if the nest argument implies a nested structure.

Value

A list of data frames in the long format. The output can be summarized using the `report` function. The first element of the list is a list with either one (no trend analyses) or two (trend analyses) data frames with at least six columns each. For each subpopulation denoted by the `groups` statement, each dependent variable, each parameter and each coefficient the corresponding value is given.

`group`	Denotes the group an analysis belongs to. If no groups were specified and/or analysis for the whole sample were requested, the value of 'group' is 'wholeGroup'.
`depVar`	Denotes the name of the dependent variable in the analysis.
`modus`	Denotes the mode of the analysis. For example, if a JK2 analysis without sampling weights was conducted, 'modus' takes the value 'jk2.unweighted'. If a analysis without any replicates but with sampling weights was conducted, 'modus' takes the value 'weighted'.
`parameter`	Denotes the parameter of the regression model for which the corresponding value is given further. Amongst others, the 'parameter' column takes the values '(Intercept)' and 'gendermale' if 'gender' was the dependent variable, for instance. See example 1 for further details.
`coefficient`	Denotes the coefficient for which the corresponding value is given further. Takes the values 'est' (estimate) and 'se' (standard error of the estimate).
`value`	The value of the parameter estimate in the corresponding group.

If groups were specified, further columns which are denoted by the group names are added to the data frame.

References

te Grotenhuis, M., Pelzer, B., Eisinga, R., Nieuwenhuis, R., Schmidt-Catran, A., & Konig, R. (2017). When size matters: advantages of weighted effect coding in observational studies. *International Journal of Public Health.* **62**, 163–167.

Examples

```
### load example data (long format)
data(lsa)
### use only the first nest
bt         <- lsa[which(lsa[,"nest"] == 1),]
### use only data from 2010
bt2010     <- bt[which(bt[,"year"] == 2010),]
## use only reading data
bt2010read <- bt2010[which(bt2010[,"domain"] == "reading"),]

### Example 1: Computes linear regression from reading score on gender separately
### for each country. Assume no nested structure.
mod1 <- repGlm(datL = bt2010read, ID = "idstud", wgt = "wgt", type = "jk2",
        PSU = "jkzone", repInd = "jkrep", imp = "imp", groups = "country",
        formula = score~sex, family ="gaussian")
res1 <- report(mod1, printGlm = TRUE)
```

```
### Example 2: Computes log linear regression from pass/fail on ses and gender
### separately for each country in a nested structure. Assuming equally weighted
### cases by omitting "wgt" argument
dat  <- lsa[intersect(which(lsa[,"year"] == 2010), which(lsa[,"domain"] == "reading")),]
mod2 <- repGlm(datL = dat, ID = "idstud", type = "JK2",  PSU = "jkzone",
        repInd = "jkrep", imp = "imp", nest="nest", groups = "country",
        formula = passReg~sex*ses, family = quasibinomial(link="logit"))
res2 <- report(mod2, printGlm = TRUE)

### Example 3: Like example 1, but without any replication methods
### trend estimation (without linking error) and nested imputation
dat  <- lsa[which(lsa[,"domain"] == "reading"),]
mod3 <- repGlm(datL = dat, ID = "idstud", wgt = "wgt", imp = "imp", nest = "nest",
        groups = "country",  formula = score~sex, trend = "year")
res3 <- report(mod3, printGlm = TRUE)
```

`repLmer` *Replication methods (JK1 and JK2) for multilevel linear regression models and trend estimation.*

Description

Compute multilevel linear models for complex cluster designs with multiple imputed variables based on the Jackknife (JK1, JK2) procedure. Conceptually, the function combines replication methods and methods for multiple imputed data. Technically, this is a wrapper for the `BIFIE.twolevelreg` function of the `BIFIEsurvey` package. `repLmer` only adds functionality for trend estimation. Please note that the function is not suitable for logistic logit/probit models.

Usage

```
repLmer(datL, ID, wgt = NULL, L1wgt=NULL, L2wgt=NULL, type = c("JK2", "JK1"),
            PSU = NULL, repInd = NULL, jkfac = NULL, rho = NULL, imp=NULL,
          group = NULL, trend = NULL, dependent, formula.fixed, formula.random,
           doCheck = TRUE, na.rm = FALSE, clusters, verbose = TRUE)
```

Arguments

`datL`	Data frame in the long format (i.e. each line represents one ID unit in one imputation of one nest) containing all variables for analysis.
`ID`	Variable name or column number of student identifier (ID) variable. ID variable must not contain any missing values.
`wgt`	Optional: Variable name or column number of case weighting variable. If no weighting variable is specified, all cases will be equally weighted.
`L1wgt`	Name of Level 1 weight variable. This is optional. If it is not provided, `L1wgt` is calculated from the total weight (i.e., `wgt`) and `L2wgt`.
`L2wgt`	Name of Level 2 weight variable

type	Defines the replication method for cluster replicates which is to be applied. Depending on type, additional arguments must be specified (e.g., PSU and/or repInd or repWgt).
PSU	Variable name or column number of variable indicating the primary sampling unit (PSU). When a jackknife procedure is applied, the PSU is the jackknife zone variable. If NULL, no cluster structure is assumed and standard errors are computed according to a random sample.
repInd	Variable name or column number of variable indicating replicate ID. In a jackknife procedure, this is the jackknife replicate variable. If NULL, no cluster structure is assumed and standard errors are computed according to a random sample.
jkfac	Argument is passed to BIFIE.data.jack and specifies the factor for multiplying jackknife replicate weights.
rho	Fay factor for statistical inference. The argument is passed to the fayfac argument of the BIFIE.data.jack function from the BIFIEsurvey package. See the corresponding help page for further details. For convenience, if rho = NULL (the default) and type = "JK1", BIFIE.data.jack is called with jktype="JK_GROUP" and fayfac = rho, where $\rho = (N_{cluster} - 1) \times N_{cluster}^{-1}$
imp	Name or column number of the imputation variable.
group	Optional: column number or name of one grouping variable. Note: in contrast to repMean, only one grouping variable can be specified.
trend	Optional: name or column number of the trend variable which contains the measurement time of the survey. repLmer computes differences for all pairwise contrasts defined by trend variable levels. or three measurement occasions, i.e. 2010, 2015, and 2020, contrasts (i.e. trends) are computed for 2010 vs. 2015, 2010 vs. 2020, and 2015 vs. 2020.
dependent	Name or column number of the dependent variable
formula.fixed	An R formula for fixed effects
formula.random	An R formula for random effects
doCheck	Logical: Check the data for consistency before analysis? If TRUE groups with insufficient data are excluded from analysis to prevent subsequent functions from crashing.
na.rm	Logical: Should cases with missing values be dropped?
clusters	Variable name or column number of cluster variable.
verbose	Logical: Show analysis information on console?

Value

A list of data frames in the long format. The output can be summarized using the report function. The first element of the list is a list with either one (no trend analyses) or two (trend analyses) data frames with at least six columns each. For each subpopulation denoted by the groups statement, each dependent variable, each parameter and each coefficient the corresponding value is given.

group	Denotes the group an analysis belongs to. If no groups were specified and/or analysis for the whole sample were requested, the value of 'group' is 'wholeGroup'.

depVar	Denotes the name of the dependent variable in the analysis.
modus	Denotes the mode of the analysis. For example, if a JK2 analysis without sampling weights was conducted, 'modus' takes the value 'jk2.unweighted'. If a analysis without any replicates but with sampling weights was conducted, 'modus' takes the value 'weighted'.
parameter	Denotes the parameter of the regression model for which the corresponding value is given further. Amongst others, the 'parameter' column takes the values '(Intercept)' and 'gendermale' if 'gender' was the dependent variable, for instance. See example 1 for further details.
coefficient	Denotes the coefficient for which the corresponding value is given further. Takes the values 'est' (estimate) and 'se' (standard error of the estimate).
value	The value of the parameter estimate in the corresponding group.

If groups were specified, further columns which are denoted by the group names are added to the data frame.

Examples

```
### load example data (long format)
data(lsa)
### use only the first nest, use only reading
btRead <- subset(lsa, nest==1 & domain=="reading")


### random intercept model with groups
mod1 <- repLmer(datL = btRead, ID = "idstud", wgt = "wgt", L1wgt="L1wgt", L2wgt="L2wgt",
        type = "jk2", PSU = "jkzone", repInd = "jkrep", imp = "imp",trend="year",
        group="country", dependent="score", formula.fixed = ~as.factor(sex)+mig,
        formula.random=~1, clusters="idclass")
res1 <- report(mod1)

### random slope without groups and without trend
mod2 <- repLmer(datL = subset(btRead, country=="countryA" & year== 2010),
        ID = "idstud", wgt = "wgt", L1wgt="L1wgt", L2wgt="L2wgt", type = "jk2",
        PSU = "jkzone", repInd = "jkrep", imp = "imp", dependent="score",
        formula.fixed = ~as.factor(sex)*mig, formula.random=~mig, clusters="idclass")
res2 <- report(mod2)
```

---

repMean *Replication methods (JK1, JK2 and BRR) for descriptive statistics.*

Description

Compute totals, means, adjusted means, mean differences, variances and standard deviations with standard errors in random or clustered or complex samples. Variance estimation in complex cluster designs based on Jackknife (JK1, JK2) or Balanced Repeated Replicates (BRR) procedure. Moreover, analyses can be customized for multiple or nested imputed variables, applying the combination rules of Rubin (1987) for imputed data and Rubin (2003) for nested imputed data. Conceptually, the

function combines replication methods and methods for multiple imputed data. Trend estimation as usual in large-scale assessments is supported as well. Technically, this is a wrapper for the svymean and svyvar functions of the survey package.

Usage

```
repMean (datL, ID, wgt = NULL, type = c("none", "JK2", "JK1", "BRR", "Fay"), PSU = NULL,
      repInd = NULL, jkfac=NULL, repWgt = NULL, nest=NULL, imp=NULL, groups = NULL,
         group.splits = length(groups), group.differences.by = NULL,
         cross.differences = FALSE, crossDiffSE = c("wec", "rep","old"),
       adjust = NULL, useEffectLiteR = FALSE, nBoot = 100, group.delimiter = "_",
         trend = NULL, linkErr = NULL, dependent, na.rm = FALSE, doCheck = TRUE,
         engine = c("survey", "BIFIEsurvey"), scale = 1, rscales = 1, mse=TRUE,
      rho=NULL, hetero=TRUE, se_type = c("HC3", "HC0", "HC1", "HC2", "CR0", "CR2"),
         clusters = NULL, crossDiffSE.engine= c("lavaan", "lm"),
      stochasticGroupSizes = FALSE, verbose = TRUE, progress = TRUE, nCores=NULL)
```

Arguments

datL	Data frame in the long format (i.e. each line represents one ID unit in one imputation of one nest) containing all variables for analysis.
ID	Variable name or column number of student identifier (ID) variable. ID variable must not contain any missing values.
wgt	Optional: Variable name or column number of weighting variable. If no weighting variable is specified, all cases will be equally weighted.
type	Defines the replication method for cluster replicates which is to be applied. Depending on type, additional arguments must be specified (e.g., PSU and/or repInd or repWgt).
PSU	Variable name or column number of variable indicating the primary sampling unit (PSU). When a jackknife procedure is applied, the PSU is the jackknife zone variable. If NULL, no cluster structure is assumed and standard errors are computed according to a random sample.
repInd	Variable name or column number of variable indicating replicate ID. In a jackknife procedure, this is the jackknife replicate variable. If NULL, no cluster structure is assumed and standard errors are computed according to a random sample.
jkfac	Only applies if engine = "BIFIEsurvey". Argument is passed to BIFIE.data.jack and specifies the factor for multiplying jackknife replicate weights.
repWgt	Normally, replicate weights are created by repMean directly from PSU and repInd variables. Alternatively, if replicate weights are included in the data.frame, specify the variable names or column number in the repWgt argument.
nest	Optional: name or column number of the nesting variable. Only applies in nested multiple imputed data sets.
imp	Optional: name or column number of the imputation variable. Only applies in multiple imputed data sets.
groups	Optional: vector of names or column numbers of one or more grouping variables.

`group.splits` Optional: If groups are defined, `group.splits` optionally specifies whether analysis should be done also in the whole group or overlying groups. See examples for more details.

`group.differences.by`
Optional: Specifies variable group differences should be computed for. The corresponding variable must be included in the `groups` statement. Exception: choose 'wholePop' if you want to estimate each's group difference from the overall sample mean. See examples for further details.

`cross.differences`
Either a list of vectors, specifying the pairs of levels for which cross-level differences should be computed. Alternatively, if `TRUE`, cross-level differences for all pairs of levels are computed. If `FALSE`, no cross-level differences are computed. (see example 2a, 3, and 4)

`crossDiffSE` Method for standard error estimation for cross level differences, where groups are dependent. `wec` uses weighted effect coding, `rep` uses replication methods (bootstrap or jackknife) to estimate the standard error between the total mean and group-specific means. `old` does not account for dependent groups and treat the groups as if they were independent from each other.

`adjust` Variable name or column number of variable(s) for which adjusted means should be computed. Non-numeric variables (factors) will be converted to 0/1 dichotomous variables.

`useEffectLiteR` Logical: use the `lavaan`-wrapper `EffectLiteR` to compute adjusted means? Alternatively, adjusted means are computed by applying a simple linear regression model in each group, using the variables in `adjust` as independent variables. Afterwards, the coefficients are weighted with the (weighted) means of the independent variables. Standard errors for this procedure are received using the delta method by applying an augmented variance-covariance matrix which assumes zero covariances between independent variable means and regression coefficients. We recommend to set `useEffectLiteR = TRUE` if no replication methods are applied. When replication methods are used (jackknife-1, jackknife-2, BRR), we recommend to set `useEffectLiteR = FALSE`, because otherwise the estimation is very slow.

`nBoot` Without replicates (i.e., for completely random samples), the `rep` method for standard error estimation for cross level differences needs a bootstrap. `nBoot` therefore specifies the number of bootstrap samples. This argument is only necessary, if `crossDiffSE = "rep"` *and* none of the replicate methods (JK1, JK2, or BRR) is applied. Otherwise, `nBoot` will be ignored.

`group.delimiter`
Character string which separates the group names in the output frame.

`trend` Optional: name or column number of the trend variable which contains the measurement time of the survey. Note: Levels of all grouping variables must be equal in all 'sub populations' partitioned by the discrete trend variable. `repMean` computes differences for all pairwise contrasts defined by trend variable levels. or three measurement occasions, i.e. 2010, 2015, and 2020, contrasts (i.e. trends) are computed for 2010 vs. 2015, 2010 vs. 2020, and 2015 vs. 2020.

`linkErr` Optional: Either the name or column number of the linking error variable. If `NULL`, a linking error of 0 will be assumed in trend estimation. Alternatively,

linking errors may be given as data.frame with following specifications: Two columns, named trendLevel1 and trendLevel2 which contain the levels of the trend variable. The contrasts between both values indicates which trend is meant. For only two measurement occasions, i.e. 2010 and 2015, trendLevel1 should be 2010, and trendLevel2 should be 2015. For three measurement occasions, i.e. 2010, 2015, and 2020, additional lines are necessary where trendLevel1 should be 2010, and trendLevel2 should be 2020, to mark the contrast between 2010 and 2020, and further additional lines are necessary where trendLevel1 should be 2015, and trendLevel2 should be 2020. The column depVar must include the name of the dependent variable. This string must correspond to the name of the dependent variable in the data. The column parameter indicates the parameter the linking error belongs to. Column linkingError includes the linking error value. Providing linking error in a data.frame is necessary for more than two measurement occasions. See the example 3a for further details.

dependent — Variable name or column number of the dependent variable.

na.rm — Logical: Should cases with missing values be dropped?

doCheck — Logical: Check the data for consistency before analysis? If TRUE groups with insufficient data are excluded from analysis to prevent subsequent functions from crashing.

engine — Which package should be used for estimation?

scale — scaling constant for variance, for details, see help page of svrepdesign from the survey package

rscales — scaling constant for variance, for details, see help page of svrepdesign from the survey package

mse — Logical: If TRUE, compute variances based on sum of squares around the point estimate, rather than the mean of the replicates. See help page of svrepdesign from the survey package for further details.

rho — Shrinkage factor for weights in Fay's method. If engine = "survey", argument is passed to the rho argument of the svrepdesign function from the survey package. See the corresponding help page for further details. If engine = "BIFIEsurvey", argument is passed to the fayfac argument of the BIFIE.data.jack function from the BIFIEsurvey package. See the corresponding help page for further details. For convenience, if rho = NULL (the default) and engine = "BIFIEsurvey" and type = "JK1", BIFIE.data.jack is called with jktype="JK_GROUP" and fayfac = rho, where $\rho = (N_{cluster} - 1) \times N_{cluster}^{-1}$

hetero — Logical: Assume heteroscedastic variance for weighted effect coding?

se_type — The sort of standard error sought for cross level differences. Only applies if crossDiffSE == "wec" and hetero == TRUE and crossDiffSE.engine == "lm". See the help page of lm_robust from the estimatr package for further details.

clusters — Optional: Variable name or column number of cluster variable. Only necessary if weighted effecting coding should be performed using heteroscedastic variances. See the help page of lm_robust from the estimatr package for further details.

crossDiffSE.engine
: Software implementation used for estimating cross-level differences. Choices are either "lavaan" (required if stochasticGroupSites == "TRUE") or R function lm. "lavaan" is the default.

stochasticGroupSizes
: Logical: Assume stochastic group sizes for using weighted effect coding in cross-level differences? Note: To date, only crossDiffSE.engine = "lavaan" allows for stochastic group sizes. Stochastic group sizes are not yet implemented for any replication method (jackknife, BRR).

verbose
: Logical: Show analysis information on console?

progress
: Logical: Show progress bar on console?

nCores
: integer (default: NULL), number of cores to use for parallel processing, if engine = "survey". If NULL, single core processing is used.

Details

Function first creates replicate weights based on PSU and repInd variables (if defined) according to JK2 or BRR procedure as implemented in WesVar. According to multiple imputed data sets, a workbook with several analyses is created. The function afterwards serves as a wrapper for svymean called by svyby implemented in the 'survey' package. The results of the several analyses are then pooled according to Rubin's rule.

Value

A list of data frames in the long format. The output can be summarized using the report function. The first element of the list is a list with either one (no trend analyses) or two (trend analyses) data frames with at least six columns each. For each subpopulation denoted by the groups statement, each parameter (i.e., mean, variance, or group differences) and each coefficient (i.e., the estimate and the corresponding standard error) the corresponding value is given.

group
: Denotes the group an analysis belongs to. If no groups were specified and/or analysis for the whole sample were requested, the value of 'group' is 'wholeGroup'.

depVar
: Denotes the name of the dependent variable in the analysis.

modus
: Denotes the mode of the analysis. For example, if a JK2 analysis without sampling weights was conducted, 'modus' takes the value 'jk2.unweighted'. If a analysis without any replicates but with sampling weights was conducted, 'modus' takes the value 'weighted'.

parameter
: Denotes the parameter of the regression model for which the corresponding value is given further. Amongst others, the 'parameter' column takes the values 'mean', 'sd', 'var' and 'meanGroupDiff' if group differences were requested.

coefficient
: Denotes the coefficient for which the corresponding value is given further. Takes the values 'est' (estimate) and 'se' (standard error of the estimate).

value
: The value of the parameter estimate in the corresponding group.

If groups were specified, further columns which are denoted by the group names are added to the data frame.

References

te Grotenhuis, M., Pelzer, B., Eisinga, R., Nieuwenhuis, R., Schmidt-Catran, A., & Konig, R. (2017). When size matters: advantages of weighted effect coding in observational studies. *International Journal of Public Health.* **62**, 163–167.

Sachse, K. A. & Haag, N. (2017). Standard errors for national trends in international large-scale assessments in the case of cross-national differential item functioning. *Applied Measurement in Education, 30*, (2), 102-116. http://dx.doi.org/10.1080/08957347.2017.1283315

Weirich, S., Hecht, M., Becker, B. et al. Comparing group means with the total mean in random samples, surveys, and large-scale assessments: A tutorial and software illustration. Behav Res (2021). https://doi.org/10.3758/s13428-021-01553-1

Examples

```
data(lsa)

### Example 1: only means, SD and variances for each country
### We only consider domain 'reading'
rd     <- lsa[which(lsa[,"domain"] == "reading"),]

### We only consider the first "nest".
rdN1   <- rd[which(rd[,"nest"] == 1),]

### First, we only consider year 2010
rdN1y10<- rdN1[which(rdN1[,"year"] == 2010),]


### mean estimation
means1 <- repMean(datL = rdN1y10, ID="idstud", wgt="wgt", type = "JK2", PSU = "jkzone",
          repInd = "jkrep", imp="imp", groups = "country", dependent = "score",
          na.rm=FALSE, doCheck=TRUE, engine = "BIFIEsurvey")
### reporting function: the function does not know which content domain is being considered,
### so the user may add new columns in the output using the 'add' argument
res1   <- report(means1, add = list(domain = "reading"))

### Example 1a: Additionally to example 1, we decide to estimate whether
### each country's mean differ significantly from the overall mean as well
### as from the individual means of the other contries
means1a<- repMean(datL = rdN1y10, ID="idstud", wgt="wgt", type = "JK2", PSU = "jkzone",
          repInd = "jkrep", imp="imp", groups = "country", group.splits = 0:1,
          group.differences.by = "country", cross.differences = TRUE,
          dependent = "score", na.rm=FALSE, doCheck=TRUE, hetero=FALSE)
res1a  <- report(means1a, add = list(domain = "reading"))

### See that the means of all countries significantly differ from the overall mean.
print(res1a[intersect(which(res1a[,"comparison"] == "crossDiff"),
      which(res1a[,"parameter"] == "mean")),], digits = 3)

### Example 2: Sex differences by country. Assume equally weighted cases by omitting
### 'wgt' argument.
means2 <- repMean(datL = rdN1y10, ID="idstud", type = "JK2", PSU = "jkzone",
          repInd = "jkrep", imp="imp", groups = c("country", "sex"), group.splits = 0:2,
```

```
          group.differences.by="sex", dependent = "score", na.rm=FALSE, doCheck=TRUE,
          cross.differences =TRUE, crossDiffSE.engine= "lm")
res2   <- report(means2,add = list(domain = "reading"))

### Example 2a: Additionally to example 2, we decide to estimate whether
### each country's mean differ significantly from the overall mean. (Note: by default,
### such cross level differences are estimated using 'weighted effect coding'. Use the
### 'crossDiffSE' argument to choose alternative methods.) Moreover, we estimate whether
### each country's sex difference differ significantly from the sex difference in the
### whole population.
means2a<- repMean(datL = rdN1y10, ID="idstud", wgt="wgt", type = "JK2", PSU = "jkzone",
          repInd = "jkrep", imp="imp", groups = c("country", "sex"), group.splits = 0:2,
          group.differences.by="sex", cross.differences = list(c(0,1), c(0,2)),
          dependent = "score", na.rm=FALSE, doCheck=TRUE,
          crossDiffSE.engine= "lm", clusters = "idclass")
res2a  <- report(means2a,add = list(domain = "reading"))

### Third example: like example 2a, but using nested imputations of dependent variable,
### and additionally estimating trend: use 'rd' instead of 'rdN1y10'
### assume equally weighted cases by omitting 'wgt' argument
### ignoring jackknife by omitting 'type', 'PSU' and 'repInd' argument
means3T<- repMean(datL = rd, ID="idstud", imp="imp", nest="nest",
          groups = c("country", "sex"), group.splits = 0:2, group.differences.by="sex",
          cross.differences = list(c(0,1), c(0,2)), dependent = "score", na.rm=FALSE,
          doCheck=TRUE, trend = "year", linkErr = "leScore",
          crossDiffSE = "wec", crossDiffSE.engine= "lavaan")
res3T  <- report(means3T, add = list(domain = "reading"))

### Example 3a: like example 3, but providing linking errors in an additional data.frame
### This is optional for two measurement occasions but mandatory if the analysis contains
### more than two measurement occasions
linkErr<- data.frame ( trendLevel1 = 2010, trendLevel2 = 2015,  depVar = "score",
          parameter = "mean", unique(lsa[,c("domain", "leScore")]),
          stringsAsFactors = FALSE)
colnames(linkErr) <- car::recode(colnames(linkErr), "'leScore'='linkingError'")
### note that the linking errors for the specified domain have to be chosen via
### subsetting
means3a<- repMean(datL = rd, ID="idstud", imp="imp", nest="nest",
          groups = c("country", "sex"),
          group.splits = 0:2, group.differences.by="sex",
          cross.differences = list(c(0,1), c(0,2)),
          dependent = "score", na.rm=FALSE, doCheck=TRUE, trend = "year",
          linkErr = linkErr[which(linkErr[,"domain"] == "reading"),],
          crossDiffSE = "wec", crossDiffSE.engine= "lavaan")
res3a  <- report(means3a, add = list(domain = "reading"))

### Fourth example: using a loop do analyse 'reading' and 'listening' comprehension
### in one function call. Again with group and cross differences and trends, and
### trend differences
### we use weights but omit jackknife analysis by omitting 'type', 'PSU' and 'repInd'
### argument
means4T<- by ( data = lsa, INDICES = lsa[,"domain"], FUN = function (sub.dat) {
          repMean(datL = sub.dat, ID="idstud", wgt="wgt", imp="imp", nest="nest",
```

```
                groups = c("country", "sex"), group.splits = 0:2,
                group.differences.by="sex",
                cross.differences = list(c(0,1), c(0,2)), dependent = "score",
                na.rm=FALSE, doCheck=TRUE,
                trend = "year", linkErr = "leScore", crossDiffSE.engine= "lm") })
ret4T  <- do.call("rbind", lapply(names(means4T), FUN = function ( domain ) {
          report(means4T[[domain]], add = list(domain = domain))}))

### Fifth example: compute adjusted means, also with trend estimation
### Note: all covariates must be numeric or 0/1 dichotomous
rdN1[,"mignum"] <- as.numeric(rdN1[,"mig"])
rdN1[,"sexnum"] <- car::recode(rdN1[,"sex"], "'male'=0; 'female'=1", as.numeric=TRUE,
                   as.factor=FALSE)
means5 <- repMean(datL = rdN1, ID="idstud", wgt="wgt", type = "JK2", PSU = "jkzone",
          repInd = "jkrep", imp="imp", groups = "country",
          adjust = c("sexnum", "ses", "mignum"), useEffectLiteR = FALSE,
          dependent = "score", na.rm=FALSE, doCheck=TRUE, trend = "year",
          linkErr = "leScore")
res5   <- report(means5, add = list(domain = "reading"))

## Not run:
#############################################################################################
#   Example 6: R code for running the PISA 2015 science example to compare group means    #
#                   with the total mean using weighted effect coding                      #
#############################################################################################

# Warning: large PISA data set requires at least 16 GB free working memory (RAM):

### define necessary directories (note: writing permissions required)
folder <- tempdir()

### download PISA 2015 zipped student questionnaire data (420 MB) to a folder with
### writing permissions
download.file(url = "https://webfs.oecd.org/pisa/PUF_SPSS_COMBINED_CMB_STU_QQQ.zip",
         destfile = file.path(folder, "pisa2015.zip"))

### unzip PISA 2015 student questionnaire data (1.5 GB) to temporary folder
zip::unzip(zipfile = file.path(folder, "pisa2015.zip"), files= "CY6_MS_CMB_STU_QQQ.sav",
     exdir=folder)

### read data
pisa <- foreign::read.spss(file.path (folder, "CY6_MS_CMB_STU_QQQ.sav"),
        to.data.frame=TRUE, use.value.labels = FALSE, use.missings = TRUE)

# dependent variables
measure.vars <- paste0("PV", 1:10, "SCIE")

### choose desired variables and reshape into the long format
#              'CNTSTUID' = individual student identifier
#                   'CNT' = country identifier
#                 'SENWT' = senate weight (assume a population of 5000 in each country)
#              'W_FSTUWT' = final student weight
#                  'OECD' = dummy variable indicating which country is part of the OECD
```

```
#   'W_FSTURWT' (1 to 80) = balanced repeated replicate weights
# 'PV1SCIE' to 'PV10SCIE' = 10 plausible values of (latent) science performance
pisaLong <- reshape2::melt(pisa, id.vars = c("CNTSTUID", "CNT", "SENWT", "W_FSTUWT",
            "OECD", paste0("W_FSTURWT", 1:80)),
            measure.vars = measure.vars, value.name = "value", variable.name="imp",
            na.rm=TRUE)

### choose OECD countries
oecd <- pisaLong[which(pisaLong[,"OECD"] == 1),]

### analyze data
### analysis takes approximately 30 minutes on an Intel i5-6500 machine with 32 GB RAM
means   <- repMean( datL = oecd,         # data.frame in the long format
    ID                  = "CNTSTUID",   # student identifier
    dependent           = "value",      # the dependent variable in the data
    groups              = "CNT",        # the grouping variable
    wgt                 = "SENWT",      # (optional) weighting variable. We use senate
                                        # weights (assume a population of 5000 in each
                                        # country)
    type                = "Fay",        # type of replication method. Corresponding to
                                        # the PISA sampling method, we use "Fay"
    rho                 = 0.5,          # shrinkage factor for weights in Fay's method
    scale               = NULL,         # scaling constant for variance, set to NULL
                                        # according to PISA's sampling method
    rscales             = NULL,         # scaling constant for variance, set to NULL
                                        # according to PISA's sampling method
    repWgt              = paste0("W_FSTURWT", 1:80), # the replicate weights,
                                                     # provided by the OECD
    imp                 = "imp",        # the imputation variable
    mse                 = FALSE,        # if TRUE, compute variances based on sum of
                                        # squares around the point estimate, rather
                                        # than the mean of the replicates.
    group.splits        = 0:1,          # defining the 'levels' for which means should
                                        # be computed. 0:1 implies that means for the
                                        # whole sample (level 0) as well as for groups
                                        # (level 1) are computed
    cross.differences   = TRUE,         # defines whether (and which) cross level mean
                                        # differences should be computed. TRUE means
                                        # that all cross level mean differences are
                                        # computed
    crossDiffSE         = "wec",        # method for standard errors of mean
                                        # differences
    crossDiffSE.engine  = "lm",         # software implementation for standard
                                        # errors of mean differences
    hetero              = TRUE,         # assume heteroscedastic group variances
    stochasticGroupSizes = FALSE        # assume fixed group sizes
                  )

### call a reporting function to generate user-friendly output
results <- report(means, exclude = c("Ncases", "NcasesValid", "var", "sd"))

## End(Not run)
```

report *Reporting functions for* repMean, repTable, repQuantile, *and* repGlm

Description

Summarizes the output of the four main functions repMean, repTable, repQuantile, and repGlm, and provides a single data.frame with all results.

Usage

```
report(repFunOut, trendDiffs = deprecated(), add=list(),
       exclude = c("NcasesValid", "var"),
       printGlm = FALSE, round = TRUE, digits = 3, printDeviance = FALSE,
       printSE_correction = FALSE)
report2(repFunOut, add=list(), exclude = c("NcasesValid", "var"), printGlm = FALSE,
     round = TRUE, digits = 3, printDeviance = FALSE, printSE_correction = FALSE)
```

Arguments

repFunOut	output of one of the four eatRep-functions.
trendDiffs	deprecated. In earlier versions of the package, this argument was used to determine differences in trends. As differences in trends are equivalent to the trend of differences (no matter whether group or cross-level differences), the argument was deprecated. If the user specifies group or cross-level difference along with trends, trends of differences are computed as well.
add	Optional: additional columns for output. See examples of the jk2-functions
exclude	Which parameters should be excluded from reporting?
printGlm	Only relevant for repGlm: print summary on console?
round	Logical: should the results be rounded to a limited number of digits?
digits	How many digits should be used for rounding?
printDeviance	Only relevant for repGlm when other than the identical function is used as link function, and if printGlm is TRUE. Should the deviance information printed additionally? Note: To print deviance information, the argument poolMethod of the repGlm function must be set to "scalar".
printSE_correction	Logical: Print the differences of original SEs of cross differences (method "old") and SEs obtained by the "wec" or "rep" method.

Value

report and report2 differ in the output which is returned. The output of report2 is optimized for further processing, i.e. drawing plots by means of the eatPlot package. For report, the output is a data frame with at least nine columns.

group	Denotes the group an analysis belongs to. If no groups were specified and/or analysis for the whole sample were requested, the value of 'group' is 'wholeGroup'.
depVar	Denotes the name of the dependent variable in the analysis.
modus	Denotes the mode of the analysis. For example, if a JK2 regression analysis was conducted, 'modus' takes the value 'JK2.glm'. If a mean analysis without any replicates was conducted, 'modus' takes the value 'CONV.mean'.
comparison	Denotes whether group mean comparisons or cross-level comparisons were conducted. Without any comparisons, 'comparison' takes the value 'NA'
parameter	Denotes the parameter of the corresponding analysis. If regression analysis was applied, the regression parameter is given. Amongst others, the 'parameter' column takes the values '(Intercept)' and 'gendermale' if 'gender' was the independent variable, for instance. If mean analysis was applied, the 'parameter' column takes the values 'mean', 'sd', 'var', or 'Nvalid'. See the examples of repMean,repTable, repQuantile, or repGlm for further details.
depVar	Denotes the name of the dependent variable (only if repGlm was called before)
est	Denotes the estimate of the corresponding analysis.
se	Denotes the standard error of the corresponding estimate.
p	Denotes the p value of the estimate.

For report2, the output is a list with four data.frames. The first data.frame plain summarizes the results in human-readable form. The data.frames 2 to 4 (comparisons, group, estimate) are redundant to plain and contain the results in a technical presentation suitable for further processing in eatPlot.

plain	The complete results in human-readable form.
comparison	An allocation table that indicates which comparison (group comparison or cross-level comparison) relates to which groups.
group	A table that assigns an ID to each analysis unit. This makes it easier later on to read from the output which comparison relates to which groups. This simplifies the assignment, especially when comparing comparisons (i.e., cross-level differences of group differences).
estimate	The results of the analyses, assigned to their IDs.

Author(s)

Benjamin Becker, Sebastian Weirich

Examples

```
### see examples of the eatRep main functions.
```

repQuantile *Replication methods (JK1, JK2 and BRR) for quantiles and trend estimation.*

Description

Compute quantiles with standard errors for complex cluster designs with multiple imputed variables (e.g. plausible values) based on Jackknife (JK1, JK2) or balanced repeated replicates (BRR) procedure. Conceptually, the function combines replication methods and methods for multiple imputed data. Technically, this is a wrapper for the `svyquantile()` function of the survey package.

Usage

```
repQuantile(datL, ID, wgt = NULL, type = c("none", "JK2", "JK1", "BRR", "Fay"),
            PSU = NULL, repInd = NULL, repWgt = NULL, nest=NULL, imp=NULL,
          groups = NULL, group.splits = length(groups), cross.differences = FALSE,
            group.delimiter = "_", trend = NULL, linkErr = NULL, dependent,
            probs = c(0.25, 0.50, 0.75),  na.rm = FALSE, nBoot = NULL,
            bootMethod = c("wSampling","wQuantiles") , doCheck = TRUE,
            scale = 1, rscales = 1, mse=TRUE,
            rho=NULL, verbose = TRUE, progress = TRUE)
```

Arguments

datL	Data frame in the long format (i.e. each line represents one ID unit in one imputation of one nest) containing all variables for analysis.
ID	Variable name or column number of student identifier (ID) variable. ID variable must not contain any missing values.
wgt	Optional: Variable name or column number of weighting variable. If no weighting variable is specified, all cases will be equally weighted.
type	Defines the replication method for cluster replicates which is to be applied. Depending on type, additional arguments must be specified (e.g., PSU and/or repInd or repWgt).
PSU	Variable name or column number of variable indicating the primary sampling unit (PSU). When a jackknife procedure is applied, the PSU is the jackknife zone variable. If NULL, no cluster structure is assumed and standard errors are computed according to a random sample.
repInd	Variable name or column number of variable indicating replicate ID. In a jackknife procedure, this is the jackknife replicate variable. If NULL, no cluster structure is assumed and standard errors are computed according to a random sample.
repWgt	Normally, replicate weights are created by repQuantile directly from PSU and repInd variables. Alternatively, if replicate weights are included in the data.frame, specify the variable names or column number in the repWgt argument.
nest	Optional: name or column number of the nesting variable. Only applies in nested multiple imputed data sets.

imp — Optional: name or column number of the imputation variable. Only applies in multiple imputed data sets.

groups — Optional: vector of names or column numbers of one or more grouping variables.

group.splits — Optional: If groups are defined, group.splits optionally specifies whether analysis should be done also in the whole group or overlying groups. See examples for more details.

cross.differences — Either a list of vectors, specifying the pairs of levels for which cross-level differences should be computed. Alternatively, if TRUE, cross-level differences for all pairs of levels are computed. If FALSE, no cross-level differences are computed. (see examples 2a, 3, and 4 in the help file of the repMean function)

group.delimiter — Character string which separates the group names in the output frame.

trend — Optional: name or column number of the trend variable which contains the measurement time of the survey. Note: Levels of all grouping variables must be equal in all 'sub populations' partitioned by the discrete trend variable. repQuantile computes differences for all pairwise contrasts defined by trend variable levels. or three measurement occasions, i.e. 2010, 2015, and 2020, contrasts (i.e. trends) are computed for 2010 vs. 2015, 2010 vs. 2020, and 2015 vs. 2020.

linkErr — Optional: Either the name or column number of the linking error variable. If NULL, a linking error of 0 will be assumed in trend estimation. Alternatively, linking errors may be given as data.frame with following specifications: Two columns, named trendLevel1 and trendLevel2 which contain the levels of the trend variable. The contrasts between both values indicates which trend is meant. For only two measurement occasions, i.e. 2010 and 2015, trendLevel1 should be 2010, and trendLevel2 should be 2015. For three measurement occasions, i.e. 2010, 2015, and 2020, additional lines are necessary where trendLevel1 should be 2010, and trendLevel2 should be 2020, to mark the contrast between 2010 and 2020, and further additional lines are necessary where trendLevel1 should be 2015, and trendLevel2 should be 2020. The column depVar must include the name of the dependent variable. This string must correspond to the name of the dependent variable in the data. The column parameter indicates the parameter the linking error belongs to. Column linkingError includes the linking error value. Providing linking error in a data.frame is necessary for more than two measurement occasions.

dependent — Variable name or column number of the dependent variable.

probs — Numeric vector with probabilities for which to compute quantiles.

na.rm — Logical: Should cases with missing values be dropped?

nBoot — Optional: Without replicates, standard error cannot be computed in a weighted sample. Alternatively, standard errors may be computed using the boot package. nBoot therefore specifies the number of bootstrap samples. If not specified, no standard errors will be given. In analyses containing replicates or samples without specifying person weights, nBoot will be ignored.

bootMethod — Optional: If standard error are computed in a bootstrap, two possible methods may be applied. wSampling requests the function to draw nBoot weighted

	bootstrap samples for which unweighted quantiles are computed. `wQuantiles` requests the function to draw `nBoot` unweighted bootstrap samples for which weighted quantiles are computed.
`doCheck`	Logical: Check the data for consistency before analysis? If `TRUE` groups with insufficient data are excluded from analysis to prevent subsequent functions from crashing.
`scale`	scaling constant for variance, for details, see help page of `svrepdesign` from the `survey` package
`rscales`	scaling constant for variance, for details, see help page of `svrepdesign` from the `survey` package
`mse`	Logical: If `TRUE`, compute variances based on sum of squares around the point estimate, rather than the mean of the replicates. See help page of `svrepdesign` from the `survey` package for further details.
`rho`	Shrinkage factor for weights in Fay's method. See help page of `svrepdesign` from the `survey` package for further details.
`verbose`	Logical: Show analysis information on console?
`progress`	Logical: Show progress bar on console?

Details

Function first creates replicate weights based on PSU and repInd variables according to JK2 or BRR procedure implemented in WesVar. According to multiple imputed data sets, a workbook with several analyses is created. The function afterwards serves as a wrapper for `svyquantile` called by `svyby` implemented in the `survey` package. The results of the several analyses are then pooled according to Rubins rule, which is adapted for nested imputations if the `dependent` argument implies a nested structure.

Value

A list of data frames in the long format. The output can be summarized using the `report` function. The first element of the list is a list with either one (no trend analyses) or two (trend analyses) data frames with at least six columns each. For each subpopulation denoted by the `groups` statement, each dependent variable, each parameter (i.e., the values of the corresponding categories of the dependent variable) and each coefficient (i.e., the estimate and the corresponding standard error) the corresponding value is given.

`group`	Denotes the group an analysis belongs to. If no groups were specified and/or analysis for the whole sample were requested, the value of 'group' is 'wholeGroup'.
`depVar`	Denotes the name of the dependent variable in the analysis.
`modus`	Denotes the mode of the analysis. For example, if a JK2 analysis without sampling weights was conducted, 'modus' takes the value 'jk2.unweighted'. If a analysis without any replicates but with sampling weights was conducted, 'modus' takes the value 'weighted'.
`parameter`	Denotes the parameter of the regression model for which the corresponding value is given further. For frequency tables, this is the value of the category of the dependent variable which relative frequency is given further.

coefficient	Denotes the coefficient for which the corresponding value is given further. Takes the values 'est' (estimate) and 'se' (standard error of the estimate).
value	The value of the parameter, i.e. the relative frequency or its standard error.

If groups were specified, further columns which are denoted by the group names are added to the data frame.

Examples

```
data(lsa)
### Example 1: only means, SD and variances for each country
### We only consider domain 'reading'
rd     <- lsa[which(lsa[,"domain"] == "reading"),]

### We only consider the first "nest".
rdN1   <- rd[which(rd[,"nest"] == 1),]

### First, we only consider year 2010
rdN1y10<- rdN1[which(rdN1[,"year"] == 2010),]

### First example: Computes percentile in a nested data structure for reading
### scores conditionally on country and for the whole group
perzent   <- repQuantile(datL = rd, ID = "idstud", wgt = "wgt", type = "JK2",
             PSU = "jkzone", repInd = "jkrep", imp = "imp", nest="nest",
             groups = "country", group.splits = c(0:1), dependent = "score",
             probs = seq(0.1,0.9,0.2) )
res       <- report(perzent, add = list(domain = "reading"))

### Second example: Computes percentile for reading scores conditionally on country,
### use 100 bootstrap samples, assume no nested structure
perzent2  <- repQuantile(datL = rdN1y10, ID = "idstud", wgt = "wgt",
             imp = "imp", groups = "country", dependent = "score",
             probs = seq(0.1,0.9,0.2), nBoot = 100 )
res2      <- report(perzent, add = list(domain = "reading"))
```

---

repTable *JK1, JK2 and BRR for frequency tables and trend estimation.*

Description

Compute frequency tables for categorical variables (e.g. factors: dichotomous or polytomous) in complex cluster designs. Estimation of standard errors optionally takes the clustered structure and multiple imputed variables into account. To date, Jackknife-1 (JK1), Jackknife-2 (JK2) and Balanced repeated replicate (BRR) methods are implemented to account for clustered designs. Procedures of Rubin (1987) and Rubin (2003) are implemented to account for multiple imputed data and nested imputed data, if necessary. Conceptually, the function combines replication and imputation methods. Technically, this is a wrapper for the svymean function of the survey package.

Usage

```
repTable(datL, ID, wgt = NULL, type = c("none", "JK2", "JK1", "BRR", "Fay"), PSU = NULL,
       repInd = NULL, jkfac=NULL, repWgt = NULL, nest=NULL, imp=NULL, groups = NULL,
          group.splits = length(groups), group.differences.by = NULL,
          cross.differences = FALSE, crossDiffSE = c("wec", "rep","old"),
         nBoot = 100, chiSquare = FALSE, correct = TRUE, group.delimiter = "_",
       trend = NULL, linkErr = NULL, dependent, separate.missing.indicator = FALSE,
         na.rm=FALSE, expected.values = NULL, doCheck = TRUE, forceTable = FALSE,
          engine = c("survey", "BIFIEsurvey"), scale = 1, rscales = 1, mse=TRUE,
          rho=NULL, verbose = TRUE, progress = TRUE, nCores=NULL )
```

Arguments

datL	Data frame in the long format (i.e. each line represents one ID unit in one imputation of one nest) containing all variables for analysis.
ID	Variable name or column number of student identifier (ID) variable. ID variable must not contain any missing values.
wgt	Optional: Variable name or column number of weighting variable. If no weighting variable is specified, all cases will be equally weighted.
type	Defines the replication method for cluster replicates which is to be applied. Depending on type, additional arguments must be specified (e.g., PSU and/or repInd or repWgt).
PSU	Variable name or column number of variable indicating the primary sampling unit (PSU). When a jackknife procedure is applied, the PSU is the jackknife zone variable. If NULL, no cluster structure is assumed and standard errors are computed according to a random sample.
repInd	Variable name or column number of variable indicating replicate ID. In a jackknife procedure, this is the jackknife replicate variable. If NULL, no cluster structure is assumed and standard errors are computed according to a random sample.
jkfac	Only applies if engine = "BIFIEsurvey". Argument is passed to BIFIE.data.jack and specifies the factor for multiplying jackknife replicate weights.
repWgt	Normally, replicate weights are created by repTable directly from PSU and repInd variables. Alternatively, if replicate weights are included in the data.frame, specify the variable names or column number in the repWgt argument.
nest	Optional: name or column number of the nesting variable. Only applies in nested multiple imputed data sets.
imp	Optional: name or column number of the imputation variable. Only applies in multiple imputed data sets.
groups	Optional: vector of names or column numbers of one or more grouping variables.
group.splits	Optional: If groups are defined, group.splits optionally specifies whether analysis should be done also in the whole group or overlying groups. See examples for more details.

group.differences.by
: Optional: Specifies one grouping variable for which a chi-square test should be applied. The corresponding variable must be included in the groups statement. If specified, the distribution of the dependent variable is compared between the groups. See examples for further details.

cross.differences
: Either a list of vectors, specifying the pairs of levels for which cross-level differences should be computed. Alternatively, if TRUE, cross-level differences for all pairs of levels are computed. If FALSE, no cross-level differences are computed. (see examples 2a, 3, and 4 in the help file of the repMean function)

crossDiffSE
: Method for standard error estimation for cross level differences, where groups are dependent. wec uses weighted effect coding, rep uses replication methods (bootstrap or jackknife) to estimate the standard error between the total mean and group-specific means. old does not account for dependent groups and treat the groups as if they were independent from each other.

nBoot
: Without replicates (i.e., for completely random samples), the rep method for standard error estimation for cross level differences needs a bootstrap. nBoot therefore specifies the number of bootstrap samples. This argument is only necessary, if crossDiffSE = "rep" *and* none of the replicate methods (JK1, JK2, or BRR) is applied. Otherwise, nBoot will be ignored.

chiSquare
: Logical. Applies only if group.differences.by was specified. Defines whether group differences should be represented in a chi square test or in (mean) differences of each group's relative frequency. Note: To date, chi square test is not available for engine = "BIFIEsurvey".

correct
: Logical. Applies only if 'group.differences.by' is requested without cluster replicates. A logical indicating whether to apply continuity correction when computing the test statistic for 2 by 2 tables. See help page of 'chisq.test' for further details.

group.delimiter
: Character string which separates the group names in the output frame.

trend
: Optional: name or column number of the trend variable which contains the measurement time of the survey. Note: Levels of all grouping variables must be equal in all 'sub populations' partitioned by the discrete trend variable. repTable computes differences for all pairwise contrasts defined by trend variable levels. or three measurement occasions, i.e. 2010, 2015, and 2020, contrasts (i.e. trends) are computed for 2010 vs. 2015, 2010 vs. 2020, and 2015 vs. 2020.

linkErr
: Optional: Either the name or column number of the linking error variable. If NULL, a linking error of 0 will be assumed in trend estimation. Alternatively, linking errors may be given as data.frame with following specifications: Two columns, named trendLevel1 and trendLevel2 which contain the levels of the trend variable. The contrasts between both values indicates which trend is meant. For only two measurement occasions, i.e. 2010 and 2015, trendLevel1 should be 2010, and trendLevel2 should be 2015. For three measurement occasions, i.e. 2010, 2015, and 2020, additional lines are necessary where trendLevel1 should be 2010, and trendLevel2 should be 2020, to mark the contrast between 2010 and 2020, and further additional lines are necessary where trendLevel1 should be 2015, and trendLevel2 should be 2020.

The column depVar must include the name of the dependent variable. This string must correspond to the name of the dependent variable in the data. The column parameter indicates the parameter the linking error belongs to. Column linkingError includes the linking error value. Providing linking error in a data.frame is necessary for more than two measurement occasions. See the fourth example below for further details.

dependent — Variable name or column number of the dependent variable.

separate.missing.indicator — Logical. Should frequencies of missings in dependent variable be integrated? Note: That is only useful if missing occur as NA. If the dependent variable is coded as character, for example 'male', 'female', 'missing', separate missing indicator is not necessary.

na.rm — Logical: Should cases with missing values be dropped?

expected.values — Optional. A vector of values expected in dependent variable. Recommend to left this argument empty.

doCheck — Logical: Check the data for consistency before analysis? If TRUE groups with insufficient data are excluded from analysis to prevent subsequent functions from crashing.

forceTable — Logical: Function decides internally whether the table or the mean function of survey is called. If the mean function is called, the polytomous dependent variable is converted to dichotomous indicator variables. If mean is called, group differences for each category of the polytomous dependent variable can be computed. If table is called, a chi square statistic may be computed. The argument allows to force the function either to call mean or table.

engine — Which package should be used for estimation?

scale — scaling constant for variance, for details, see help page of svrepdesign from the survey package

rscales — scaling constant for variance, for details, see help page of svrepdesign from the survey package

mse — Logical: If TRUE, compute variances based on sum of squares around the point estimate, rather than the mean of the replicates. See help page of svrepdesign from the survey package for further details.

rho — Shrinkage factor for weights in Fay's method. If engine = "survey", argument is passed to the rho argument of the svrepdesign function from the survey package. See the corresponding help page for further details. If engine = "BIFIEsurvey", argument is passed to the fayfac argument of the BIFIE.data.jack function from the BIFIEsurvey package. See the corresponding help page for further details. For convenience, if rho = NULL (the default) and engine = "BIFIEsurvey" and type = "JK1", BIFIE.data.jack is called with jktype="JK_GROUP" and fayfac = rho, where $\rho = (N_{cluster} - 1) \times N_{cluster}^{-1}$

verbose — Logical: Show analysis information on console?

progress — Logical: Show progress bar on console?

nCores — integer (default: NULL), number of cores to use for parallel processing, if engine = "survey". If NULL, single core processing is used.

Details

Function first creates replicate weights based on PSU and repInd variables according to JK2 procedure implemented in WesVar. According to multiple imputed data sets, a workbook with several analyses is created. The function afterwards serves as a wrapper for `svymean` called by `svyby` implemented in the `survey` package. Relative frequencies of the categories of the dependent variable are computed by the means of the dichotomous indicators (e.g. dummy variables) of each category. The results of the several analyses are then pooled according to Rubin's rule, which is adapted for nested imputations if the `dependent` argument implies a nested structure.

Value

A list of data frames in the long format. The output can be summarized using the `report` function. The first element of the list is a list with either one (no trend analyses) or two (trend analyses) data frames with at least six columns each. For each subpopulation denoted by the `groups` statement, each dependent variable, each parameter (i.e., the values of the corresponding categories of the dependent variable) and each coefficient (i.e., the estimate and the corresponding standard error) the corresponding value is given.

`group`	Denotes the group an analysis belongs to. If no groups were specified and/or analysis for the whole sample were requested, the value of 'group' is 'wholeGroup'.
`depVar`	Denotes the name of the dependent variable in the analysis.
`modus`	Denotes the mode of the analysis. For example, if a JK2 analysis without sampling weights was conducted, 'modus' takes the value 'jk2.unweighted'. If a analysis without any replicates but with sampling weights was conducted, 'modus' takes the value 'weighted'.
`parameter`	Denotes the parameter of the regression model for which the corresponding value is given further. For frequency tables, this is the value of the category of the dependent variable which relative frequency is given further.
`coefficient`	Denotes the coefficient for which the corresponding value is given further. Takes the values 'est' (estimate) and 'se' (standard error of the estimate).
`value`	The value of the parameter, i.e. the relative frequency or its standard error.

If groups were specified, further columns which are denoted by the group names are added to the data frame.

References

Rubin, D.B. (2003): Nested multiple imputation of NMES via partially incompatible MCMC. *Statistica Neerlandica* **57, 1**, 3–18.

Examples

```
data(lsa)

### Example 1: only means, SD and variances for each country
### subsetting: We only consider domain 'reading'
rd     <- lsa[which(lsa[,"domain"] == "reading"),]
```

```
### We only consider the first "nest".
rdN1   <- rd[which(rd[,"nest"] == 1),]

### First, we only consider year 2010
rdN1y10<- rdN1[which(rdN1[,"year"] == 2010),]


### First example: Computes frequencies of polytomous competence levels (1, 2, 3, 4, 5)
### conditionally on country, using a chi-square test to decide whether the distribution
### varies between countries (it's an overall test, i.e. with three groups, df1=8).
freq.tab1 <- repTable(datL = rdN1y10, ID = "idstud", wgt = "wgt", imp="imp",
             type = "JK2", PSU = "jkzone", repInd = "jkrep", groups = "country",
             group.differences.by = "country", dependent = "comp", chiSquare = TRUE)
res1      <- report(freq.tab1, add = list ( domain = "reading" ))

### Second example: Computes frequencies of polytomous competence levels (1, 2, 3, 4, 5)
### conditionally on country. Now we test whether the frequency of each single category
### differs between pairs of countries (it's not an overall test ... repTable now
### calls repMean internally, using dummy variables)
freq.tab2 <- repTable(datL = rdN1y10, ID = "idstud", wgt = "wgt", imp="imp",
             type = "JK2", PSU = "jkzone", repInd = "jkrep", groups = "country",
             group.differences.by = "country", dependent = "comp", chiSquare = FALSE)
res2      <- report(freq.tab2, add = list ( domain = "reading" ))

### Third example: trend estimation and nested imputation and 'by' loop
### (to date, only crossDiffSE = "old" works)
freq.tab3 <- by ( data = lsa, INDICES = lsa[,"domain"], FUN = function (subdat) {
             repTable(datL = subdat, ID = "idstud", wgt = "wgt", imp="imp",
                 nest = "nest", type = "JK2", PSU = "jkzone", repInd = "jkrep",
                 groups = "country", group.differences.by = "country",
                 group.splits = 0:1, cross.differences = TRUE, crossDiffSE = "old",
                 dependent = "comp", chiSquare = FALSE, trend = "year",
                 linkErr = "leComp") })
res3      <- do.call("rbind", lapply(names(freq.tab3), FUN = function (domain) {
             report(freq.tab3[[domain]], add = list ( domain = domain )) }))

### Fourth example: similar to example 3. trend estimation using a linking
### error data.frame
linkErrs  <- data.frame ( trendLevel1 = 2010, trendLevel2 = 2015,  depVar = "comp",
             unique(lsa[,c("domain", "comp", "leComp")]), stringsAsFactors = FALSE)
colnames(linkErrs) <- car::recode(colnames(linkErrs),
                      "'comp'='parameter'; 'leComp'='linkingError'")
freq.tab4 <- by ( data = lsa, INDICES = lsa[,"domain"], FUN = function (subdat) {
             repTable(datL = subdat, ID = "idstud", wgt = "wgt", type="none",
                 imp="imp", nest = "nest", groups = "country",
                 group.differences.by = "country", group.splits = 0:1,
                 cross.differences = FALSE, dependent = "comp", chiSquare = FALSE,
                 trend = "year",
                 linkErr = linkErrs[which(linkErrs[,"domain"] == subdat[1,"domain"]),])
             })
res4      <- do.call("rbind", lapply(names(freq.tab4), FUN = function (domain) {
             report(freq.tab4[[domain]], add = list ( domain = domain ))  }))
```

```
### Fifth example: minimal example for three measurement occasions
### borrow data from the eatGADS package
trenddat1 <- system.file("extdata", "trend_gads_2010.db", package = "eatGADS")
trenddat2 <- system.file("extdata", "trend_gads_2015.db", package = "eatGADS")
trenddat3 <- system.file("extdata", "trend_gads_2020.db", package = "eatGADS")
trenddat  <- eatGADS::getTrendGADS(filePaths = c(trenddat1, trenddat2, trenddat3),
             years = c(2010, 2015, 2020), fast=FALSE)
dat       <- eatGADS::extractData(trenddat)
### use template linking Error Object
load(system.file("extdata", "linking_error.rda", package = "eatRep"))
### check consistency of data and linking error object
check1 <- checkLEs(c(trenddat1, trenddat2, trenddat3), lErr)
### Analysis for reading comprehension
freq.tab5 <- repTable(datL = dat[which(dat[,"dimension"] == "reading"),],
             ID = "idstud", type="none", imp="imp", dependent = "traitLevel",
             chiSquare = FALSE, trend = "year",
             linkErr = lErr[which(lErr[,"domain"] == "reading"),])
res5      <- report(freq.tab5, add = list ( domain = "reading" ))
res5A     <- report2(freq.tab5, add = list ( domain = "reading" ))
```

Index

∗ **datasets**
lsa, 6
∗ **package**
eatRep-package, 2

BIFIE.data.jack, *14*, *16*, *18*, *30*, *32*
BIFIE.twolevelreg, *13*

checkLEs, 5

eatRep-package, 2

generateRandomJk1Zones, 6

jk2.glm *(*repGlm*)*, 9
jk2.mean *(*repMean*)*, 15
jk2.quantile *(*repQuantile*)*, 26
jk2.table *(*repTable*)*, 29

lm, *19*
lm_robust, *11*, *18*
lsa, 6

pool.R2, 8

repGlm, *4*, 9, *24, 25*
repLmer, 13
repMean, *3, 4*, 15, *24, 25*, *27*
report, *4*, 24
report2, *4*
report2 *(*report*)*, 24
repQuantile, *4*, *24, 25*, 26
repTable, *4*, *24, 25*, 29

svrepdesign, *11*, *18*, *28*, *32*
svyby, *19*, *28*, *33*
svyglm, *9*, *11*
svymean, *16*, *19*, *29*, *33*
svyquantile, *28*
svyvar, *16*