Package ‘eatRep’

October 21, 2014

Type Package

Title eatRep

Version 0.6.6

Depends R (>= 2.14.0), survey, stringr

Imports Hmisc, fmsb, mice, boot, car, reshape2, plyr, combinat

Date 2014-09-04

Author Sebastian Weirich, Martin Hecht

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

Description Compute descriptives and regression models for complex survey designs with multiple imputed data

License GPL (>= 2)

LazyLoad yes

LazyData yes

Repository R-Forge

Repository/R-Forge/Project eat

Repository/R-Forge/Revision 428

Repository/R-Forge/DateTimeStamp 2014-09-04 15:33:07

Date/Publication 2014-09-04 15:33:07

R topics documented:

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

Description

Computes some basic statistic operations (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.

For example, computing 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).

Moreover, computing standard errors for the mean of a variable which stems from a clustered design may involve replication methods like balanced repeated replicate (BRR), bootstrap or Jackknife methods. See Weststat (2000), Foy, Galia & Li (2008), Rust and Rao (1996), and Wolter (1985) for details. To date, only the Jackknife-2 (JK2) and the Balanced Repeated Replicates (BRR) procedures are supported.

The package `eatRep` is designed to combine both methods which is necessary if (nested) multiple imputed data are used in clustered designs. Considering the structure is relevant especially for the estimation of standard errors.

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 “imputation replicates” in the outer 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.

For each imputed data set, 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 define 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 according to Rubin (1987) or Rubin (2003).

Without multiple imputations, the outer loop has only one cycle. Without a clustered sampling structure (i.e, in a random sample), the inner loop has only one cycle. Without both, 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 three loops in a nested structure. Hence, the corresponding analyses may take considerably computational effort.

Details

Package:	eatRep
Type:	Package

Version:	0.6.6
Date:	2014-09-04
License:	GPL(>=2)

Author(s)

Author/maintainer: Sebastian Weirich <sebastian.weirich@iqb.hu-berlin.de>

References

Foy, P., Galia , J. & Li, I. (2008). Scaling the data from the TIMSS 2007 mathematics and science asssessment. 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.

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

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

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

dG *Display results of jk2.glm analyses.*

Description

This is an easy function only addressed to display results of jk2.glm analyses in an abbreviated manner.

Usage

```
dG ( object , analyses = NULL )
```

Arguments

`object`	A results object created by jk2.glm().
`analyses`	Optionally: a numeric vector auf analyses which should be displayed.

Value

A simply data frame, not intended for further processing.

Author(s)

Sebastian Weirich

Examples

```
### see examples in the vignette and in the help page of jk2.glm()
```

---

dM *Display results of jk2.mean analyses.*

Description

This is an easy function only addressed to display results of jk2.mean analyses in an abbreviated manner.

Usage

```
dM ( object, omitTerms = c("mean","sd","var", "Ncases","NcasesValid",
     "meanGroupDiff", "se","est") )
```

Arguments

object	A results object created by jk2.mean().
omitTerms	Optionally: character vector of terms you do not want to see on console.

Value

A simply data frame, not intended for further processing.

Author(s)

Sebastian Weirich

Examples

```
### see examples in the vignette and in the help page of jk2.glm()
```

dQ *Display results of jk2.quantile analyses.*

Description

This is an easy function only addressed to display results of jk2.quantile analyses in an abbreviated manner.

Usage

```
dQ ( object, seOmit = FALSE)
```

Arguments

`object`	A results object created by jk2.quantile().
`seOmit`	Optionally: omit displaying standard errors?

Value

A simply data frame, not intended for further processing.

Author(s)

Sebastian Weirich

Examples

```
### see examples in the vignette.
```

dT *Display results of jk2.table analyses.*

Description

This is an easy function only addressed to display results of jk2.table analyses in an abbreviated manner.

Usage

```
dT ( object, reshapeFormula = depVar + group ~ parameter + coefficient, seOmit = FALSE)
```

Arguments

`object`	A results object created by jk2.table().
`reshapeFormula`	Optionally: specify the formula used for reshaping.
`seOmit`	Optionally: omit displaying stand errors?

Value

A simply data frame, not intended for further processing.

Author(s)

Sebastian Weirich

Examples

```
### see examples in the vignette.
```

`hisei` *Simulated imputed data in a large-scale assessment context*

Description

This data set contains fictional scores of 4619 students of 3 countries in a long format. Each row corresponds to one measure of one student in one imputation.

Usage

```
data(hisei)
```

Format

'data.frame': 23095 obs. of 9 variables

idstud Student identifier

wgtSTUD variable of individual student weights

sex Examinee's sex

country Country where the examinee is from.

JKZone jackknifing zone

JKrep replicate ID

income Mean month income.

hisei Indicator of the highest socio-economical income, divided into five distinct groups.

imputation The number of the plausible value whose value is given in the 'score' column.

Source

Simulated data

`jk2.glm` *JK2 for linear regression models.*

Description

Compute generalized linear models for complex cluster designs with multiple imputed variables based on the Jackknife (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

```
jk2.glm(datL, ID, wgt = NULL, type = c("JK2", "BRR"), PSU = NULL, repInd = NULL,
        nest=NULL, imp=NULL, groups = NULL, group.splits = length(groups),
        group.delimiter = "_", formula, family=gaussian,
        forceSingularityTreatment = FALSE, doCheck = TRUE, na.rm = FALSE )
```

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. Without cluster replicates (i.e., if `PSU` and/or `repInd` is NULL, `type` will be ignored.

`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.

`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.

`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.

`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?

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 `dependent` argument implies a nested structure.

Value

A data frame in the long format with at least six columns. 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.

Author(s)

Sebastian Weirich

Examples

```
data(reading)
### Example 1: Computes linear regression from reading score on gender separately for each
### country. Assume no nested structure.
```

```
readingNest1 <- reading[which(reading[,"nest"] == 1),]
mod1  <- jk2.glm(datL = readingNest1, ID = "idstud", wgt = "wgtSTUD", type = "JK2",
         PSU = "JKZone", repInd = "JKrep", imp = "imputation",
         groups = "country",  formula = score~sex)
dG(mod1)

### Example 2: Computes log linear regression from pass/fail on hisei and gender
### separately for each country in a nested structure
mod1  <- jk2.glm(datL = reading, ID = "idstud", wgt = "wgtSTUD", type = "JK2",
         PSU = "JKZone", repInd = "JKrep", imp = "imputation", nest="nest",
         groups = "country",  formula = passed~sex*income, family = binomial(link="logit"))
dG(mod1)
```

jk2.mean *JK2 for mean estimates.*

Description

Compute totals, means, variances and standard deviations with standard errors for complex cluster designs with multiple imputed variables (e.g. plausible values) based on Jackknife (JK2) of Balanced Repeated Replicates (BRR) procedure. Conceptually, the function combines replication methods and methods for multiple imputed data. Nested imputations of the dependent variable(s) are supported as well. Technically, this is a wrapper for the `svymean()` and `svyvar()` functions of the `survey` package.

Usage

```
jk2.mean (datL, ID, wgt = NULL, type = c("JK2", "BRR"), PSU = NULL, repInd = NULL,
          nest=NULL, imp=NULL, groups = NULL, group.splits = length(groups),
            group.differences.by = NULL, group.delimiter = "_", dependent, na.rm = FALSE,
            doCheck = 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. Without cluster replicates (i.e., if `PSU` and/or `repInd` is NULL, `type` will be ignored.

`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.

`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.

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

`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.

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 data frame in the long format with at least six columns. 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.

Author(s)

Sebastian Weirich

Examples

```
data(reading)
### First example: only means, SD and variances for each country
### We only consider the first "nest"
readingNest1 <- reading[which(reading[,"nest"] == 1),]
means        <- jk2.mean(datL = readingNest1, ID="idstud", wgt="wgtSTUD", type = "JK2",
                PSU = "JKZone", repInd = "JKrep", imp="imputation", groups = "country",
                dependent = "score", na.rm=FALSE, doCheck=TRUE)
dM(means)

### Second example: Sex differences by country. Assume equally weighted cases by omitting
### wgt argument.
means        <- jk2.mean(datL = readingNest1, ID="idstud", type = "JK2",
                PSU = "JKZone", repInd = "JKrep", imp="imputation", groups = c("country", "sex"),
                group.differences.by="sex", dependent = "score", na.rm=FALSE, doCheck=TRUE)

### Third example: Nested imputations of dependent variable
### First split the income in above and below 2000
reading[,"incomeHL"] <- ifelse(reading[,"income"]>2000,1,0)
### Assuming 2 nests, i.e. variable "income" with 2 imputations, and three plausible
### values for each imputation
means  <- jk2.mean(datL = reading, ID="idstud", wgt="wgtSTUD", type = "JK2", PSU = "JKZone",
          repInd = "JKrep", nest="nest",imp="imputation", groups = c("country", "incomeHL"),
          dependent = "score", na.rm=FALSE, doCheck=TRUE)

### Fourth example: Assume a completely random sample (e.g. no cluster structure and no weights),
### and give descriptives for subpopulations and the overlying populations
means  <- jk2.mean(datL = readingNest1, ID=1, imp="imputation", groups = c("country", "sex"),
          group.splits = c(0:2), dependent = "score", na.rm=FALSE, doCheck=TRUE)
```

`jk2.quantile` *JK2 method for quantiles.*

Description

Compute quantiles with standard errors for complex cluster designs with multiple imputed variables (e.g. plausible values) based on Jackknife (JK2) 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

```
jk2.quantile(datL, ID, wgt = NULL, type = c("JK2", "BRR"), PSU = NULL, repInd = NULL,
             nest=NULL, imp=NULL, groups = NULL, group.splits = length(groups),
             group.delimiter = "_", dependent, probs = seq(0, 1, 0.25),  na.rm = FALSE,
             nBoot = NULL, bootMethod = c("wSampling","wQuantiles") , doCheck = 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. Without cluster replicates (i.e., if `PSU` and/or `repInd` is NULL, `type` will be ignored.

`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.

`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.

`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.

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 data frame in the long format. 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.

Author(s)

Sebastian Weirich

Examples

```
data(reading)
### First example: Computes percentile in a nested data structure for reading
### scores conditionally on country and for the whole group
perzent   <- jk2.quantile(datL = reading, ID = "idstud", wgt = "wgtSTUD", type = "JK2",
             PSU = "JKZone", repInd = "JKrep", imp = "imputation", nest="nest",
             groups = "country", group.splits = c(0:1), dependent = "score",
             probs = seq(0.1,0.9,0.2) )

### Second example: Computes percentile for reading scores conditionally on country,
### use 100 bootstrap samples, assume no nested structure
readingNest1 <- reading[which(reading[,"nest"] == 1),]
perzent   <- jk2.quantile(datL = readingNest1, ID = "idstud", wgt = "wgtSTUD",
             imp = "imputation", groups = "country", dependent = "score",
             probs = seq(0.1,0.9,0.2), nBoot = 100 )
```

jk2.table *JK2 for frequency tables.*

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, only the Jackknife-2 procedure (JK2) is 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. Conceptually, the function combines replication and imputation methods. Technically, this is a wrapper for the svymean() function of the survey package.

Usage

```
jk2.table(datL, ID, wgt = NULL, type = c("JK2", "BRR"), PSU = NULL, repInd = NULL,
          nest=NULL, imp=NULL, groups = NULL, group.splits = length(groups),
          group.delimiter = "_", dependent , separate.missing.indicator = FALSE,
          na.rm=FALSE, expected.values = NULL, doCheck = 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. Without cluster replicates (i.e., if `PSU` and/or `repInd` is NULL, `type` will be ignored.

`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.

`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.

`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 auf 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.

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 data frame in the long format. 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.

Author(s)

Sebastian Weirich

References

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

Examples

```
data(hisei)
### First example: Computes frequencies of the hisei group conditionally on country
freq.tab  <- jk2.table(datL = hisei, ID = "idstud", wgt = "wgtSTUD",
             imp="imputation", type = "JK2", PSU = "JKZone", repInd = "JKrep",
             groups = "country", dependent = "hisei")

### Second example: Computes frequencies of passed and failed conditionally on the
### groups of sex and country. Assuming no cluster but a nested imputation structure.
freq.tab  <- jk2.table(datL = reading, ID = "idstud", wgt = "wgtSTUD",
             nest="nest", imp="imputation", groups = c("country","sex"), dependent = "passed")
```

`reading` *Simulated nested imputed data for a fictional reading achievement test*

Description

This data set contains fictional achievement scores of 4619 students of 3 countries in a long format. The data corresponds to a nested multiple imputed structure. Each row corresponds to one measure (e.g. a plausible value) of one student in one nest.

Usage

```
data(reading)
```

Format

'data.frame': 27714 obs. of 11 variables

idstud Student identifier

wgtSTUD variable of individual student weights

sex Examinee's sex

country Country where the examinee is from.

JKZone jackknifing zone

JKrep replicate ID

income Mean month income.

imputation The number of the plausible value whose value is given in the 'score' column.

nest The number of the nest the value in the 'score' column belongs to.

score The value of the plausible value.

passed An indicator whether the value is above a certain threshold, i.e. whether the individual passed or failed the test.

Source

Simulated data

Index