Package ‘synthpop’

May 9, 2026

Type Package

Title Generating Synthetic Versions of Sensitive Microdata for Statistical Disclosure Control

Version 1.9-2

Date 2025-07-12

Description A tool for producing synthetic versions of microdata containing confidential information so that they are safe to be released to users for exploratory analysis. The key objective of generating synthetic data is to replace sensitive original values with synthetic ones causing minimal distortion of the statistical information contained in the data set. Variables, which can be categorical or continuous, are synthesised one-by-one using sequential modelling. Replacements are generated by drawing from conditional distributions fitted to the original data using parametric or classification and regression trees models. Data are synthesised via the function syn() which can be largely automated, if default settings are used, or with methods defined by the user. Optional parameters can be used to influence the disclosure risk and the analytical quality of the synthesised data. For a description of the implemented method see Nowok, Raab and Dibben (2016) <doi:10.18637/jss.v074.i11>. Functions to assess identity and attribute disclosure for the original and for the synthetic data are included in the package, and their use is illustrated in a vignette on disclosure (Practical Privacy Metrics for Synthetic Data).

License GPL-2 | GPL-3

URL <https://www.synthpop.org.uk/>

Imports lattice, MASS, methods, nnet, ggplot2, graphics, stats, utils, rpart, party, foreign, plyr, proto, polspline, randomForest, ranger, classInt, mipfp, survival, stringr, rmutil, broman, forcats

Encoding UTF-8

LazyData yes

NeedsCompilation no

Author Beata Nowok [aut, cre],
Gillian M Raab [aut],
Chris Dibben [ctb],
Joshua Snoke [ctb],
Caspar van Lissa [ctb],
Lotte Pater [ctb]

Maintainer Beata Nowok <beata.nowok@gmail.com>

Repository CRAN

Date/Publication 2025-07-12 22:00:02 UTC

Contents

synthpop-package . . . 3
codebook.syn . . . 4
compare . . . 5
compare.fit.synds . . . 6
compare.synds . . . 9
disclosure . . . 11
glm.synds, lm.synds . . . 16
mergelevels.syn . . . 17
multi.compare . . . 19
multi.disclosure . . . 20
multinom.synds . . . 24
numtocat.syn . . . 25
polr.synds . . . 26
read.obs . . . 28
replicated.uniques . . . 29
SD2011 . . . 30
sdc . . . 32
summary.fit.synds . . . 33
summary.synds . . . 36
syn . . . 37
syn.bag . . . 44
syn.catall . . . 45
syn.ctree, syn.cart . . . 47
syn.ipf . . . 49
syn.lognorm, syn.sqrtnorm, syn.cubertnorm . . . 51
syn.logreg . . . 52
syn.nested . . . 53
syn.norm . . . 54
syn.normrank . . . 55
syn.passive . . . 56
syn.pmm . . . 57
syn.polr . . . 59
syn.polyreg . . . 60
syn.ranger . . . 61
syn.rf . . . 62
syn.sample . . . 63
syn.satcat . . . 64
syn.smooth . . . 65
syn.survctree . . . 66
synorig.compare . . . 68
utility.gen . . . 69

utility.tab 74
utility.tables 78
write.syn 83

Index **85**

`synthpop-package` *Generating synthetic versions of sensitive microdata for statistical disclosure control*

Description

Generate synthetic versions of a data set using parametric or CART methods.

Details

Package:	synthpop
Type:	Package
Version:	1.9-2
Date:	2025-07-12
License:	GPL-2 | GPL-3

Synthetic data are generated from the original (observed) data by the function `syn`. The package includes also tools to compare synthetic data with the observed data (`compare.synds`) and to fit (generalized) linear model to synthetic data (`lm.synds`, `glm.synds`) and compare the estimates with those for the observed data (`compare.fit.synds`). More extensive documentation on how to create synthetic data, with illustrative examples, is provided in the package vignette **synthpop**. Since that vignette was written more methods have been added to synthpop, including mthods for categorical variables based on log-linear models that can be made differentially private.

Now the package also includes functions to eavaluate the utility and disclosure risk of synthetic data. For details see the vignettes **utility** and **disclosure**. You can access all the vignettes via the index link at the bottom of this help page (`synthpop-package`)

Author(s)

Beata Nowok, Gillian M Raab, and Chris Dibben

References

Elliot, M. (2014) Final report on the disclosure risk associated with the synthetic data produced by the SYLLS team. Report 2015-2, Cathie Marsh Centre for Census and Survey Research (CCSR).

Nowok, B. Utility of synthetic microdata generated using tree-based methods (2015) Paper presented at the Privacy in Statistical Databases Conference 2016; Dubrovnik, Croatia, 14-16 September 2016 .

Nowok, B., Raab, G.M and Dibben, C. (2016). synthpop: Bespoke creation of synthetic data in R. **Journal of Statistical Software**, **74**(11), 1-26. doi:10.18637/jss.v074.i11.

Raab, G.M., Nowok, B., and Dibben, C. (2016) Practical data synthesis for large samples **Journal of Privacy and Confidentiality**, **7**(3):67-97. doi:10.29012/jpc.v7i3.407.

Raab, G.M., Nowok, B., and Dibben, C. (2016) Guidelines for producing useful synthetic data doi:10.48550/arXiv.1712.04078 An earlier version was presented at the Privacy in Statistical Databases Conference 2016; Dubrovnik, Croatia, 14-16 September 2016

Nowok, B., Raab, G.M. and Dibben, C. (2017) Providing bespoke synthetic data for the UK Longitudinal Studies and other sensitive data with the synthpop package for R **Statistical Journal of the IAOS, 33(3):785-796**. doi:10.3233/SJI150153.

Raab, G.M., Nowok, B., and Dibben, C. (2021) Assessing, visualizing and improving the utility of synthetic data. Available atdoi:10.48550/arXiv.2109.12717. An earlier version was presented at the Joint UNECE/Eurostat expert meeting on statistical data confidentiality; Poznan, Poland, 1-3 December 2021.

Raab, G.M. (2022) Utility and Disclosure Risk for Differentially Private Synthetic Categorical Data, Chapter in Privacy in Statistical Databases 2022. Published in Springer Series Lecture notes in Computer Science. Also available at doi:10.48550/arXiv.2206.01362.

Raab, G.M., Nowok, B., and Dibben, C. (2024) Practical privacy metrics for synthetic data, Vignette in synthpop package. Also available at doi:10.48550/arXiv.2406.16826.

Raab, G.M. (2024) Privacy risk from synthetic data: practical proposals. Chapter in Privacy in Statistical databases 2024. published in Springer Series Lecture notes in Computer Science. Also available at doi:10.48550/arXiv.2409.04257.

`codebook.syn` *Makes a codebook from a data frame*

Description

Describes features of variables in a data frame relevant for synthesis.

Usage

```
codebook.syn(data, maxlevs = 3)
```

Arguments

`data`	a data frame with a data set to be synthesised.
`maxlevs`	the number of factor levels above which separate tables with all labels are returned as part of `labs` component.

Value

A list with two components.

`tab` - a data frame with the following information about each variable:

`name`	variable name
`class`	class of variable

nmiss	number of missing values (NA)
perctmiss	percentage of missing values
ndistinct	number of distinct values (excluding missing values)
details	range for numeric variables, maximum length for character variables, labels for factors with <= maxlevs levels

labs - a list of extra tables with labels for each factor with number of levels greater than maxlevs.

Examples

```
codebook.syn(SD2011)
```

---

compare *Comparison of synthesised and observed data*

Description

A generic function for comparison of synthesised and observed data. The function invokes particular methods which depend on the class of the first argument.

Usage

```
compare(object, data, ...)
```

Arguments

object	a synthetic data object of class synds or fit.synds.
data	an original observed data set.
...	additional arguments specific to a method.

Details

Compare methods facilitate quality assessment of synthetic data by comapring them with the original observed data sets. The data themselves (for class synds) or models fitted to them (for class fit.synds) are compared.

Value

The value returned by compare depends on the class of its argument. See the documentation of the particular methods for details.

See Also

compare.synds, compare.fit.synds

`compare.fit.synds` *Compare model estimates based on synthesised and observed data*

Description

The same model that was used for the synthesised data set is fitted to the observed data set. The coefficients with confidence intervals for the observed data is plotted together with their estimates from synthetic data. When more than one synthetic data set has been generated (`object$m>1`) combining rules are applied. Analysis-specific utility measures are used to evaluate differences between synthetic and observed data.

Usage

```
## S3 method for class 'fit.synds'
compare(object, data, plot = "Z",
  print.coef = FALSE, return.plot = TRUE, plot.intercept = FALSE,
  lwd = 1, lty = 1, lcol = c("#1A3C5A","#4187BF"),
  dodge.height = .5, point.size = 2.5,
  population.inference = FALSE, ci.level = 0.95, ...)

## S3 method for class 'compare.fit.synds'
print(x, print.coef = x$print.coef, ...)
```

Arguments

`object`	an object of type `fit.synds` created by fitting a model to synthesised data set using function `glm.synds` or `lm.synds`.
`data`	an original observed data set.
`plot`	values to be plotted: `"Z"` (Z scores) or `"coef"` (coefficients).
`print.coef`	a logical value determining whether tables of estimates for the original and synthetic data should be printed.
`return.plot`	a logical value indicating whether a confidence interval plot should be returned.
`plot.intercept`	a logical value indicating whether estimates for intercept should be plotted.
`lwd`	the line type.
`lty`	the line width.
`lcol`	line colours.
`dodge.height`	size of vertical shifts for confidence intervals to prevent overlapping.
`point.size`	size of plotting symbols used to plot point estimates of coefficients.
`population.inference`	a logical value indicating whether intervals for inference to population quantities, as decribed by Karr et al. (2006), should be calculated and plotted. This option suppresses the lack-of-fit test and the standardised differences since these are based on differences standardised by the original interval widths.

ci.level	Confidence interval coverage as a proportion.
...	additional parameters passed to ggplot.
x	an object of class compare.fit.synds.

Details

This function can be used to evaluate whether the method used for synthesis is appropriate for the fitted model. If this is the case the estimates from the synthetic data of what would be expected from the original data xpct(Beta) xpct(Z) should not differ from the estimates from the observed data (Beta and Z) by more than would be expected from the standard errors (se(Beta) and se(Z)). For more details see the vignette on inference.

Value

An object of class compare.fit.synds which is a list with the following components:

call	the original call to fit the model to the synthesised data set.
coef.obs	a data frame including estimates based on the observed data: coefficients (Beta), their standard errors (se(Beta)) and Z scores (Z).
coef.syn	a data frame including (combined) estimates based on the synthesised data: point estimates of observed data coefficients (B.syn), standard errors of those estimates (se(B.syn)), estimates of the observed standard errors (se(Beta).syn), Z scores estimates (Z.syn) and their standard errors (se(Z.syn)). Note that se(B.syn) and se(Z.syn) give the standard errors of the mean of the m syntheses and can be made very small by increasing m (see the vignette on inference for more details).
coef.diff	a data frame containing standardized differences between the coefficients estimated from the original data and those calculated from the combined synthetic data. The difference is standardized by dividing by the estimated standard error of the fit from the original. The corresponding p-values are calculated from a standard Normal distribution and represent the probability of achieving differences as large as those found if the model use for synthesis is compatible with the model that generated the original data.
mean.abs.std.diff	Mean absolute standardized difference (over all coefficients).
ci.overlap	a data frame containing the percentage of overlap between the estimated synthetic confidence intervals and the original sample confidence intervals for each parameter. When population.inference = TRUE overlaps are calculated as suggested by Karr et al. (2006). Otherwise a simpler overlap measure with intervals of equal length is calculated.
mean.ci.overlap	Mean confidence interval overlap (over all coefficients).
lack.of.fit	lack-of-fit measure from all m synthetic data sets combined, calculated as follows, when object$incomplete = FALSE. The vector of mean differences (diff) between the coefficients calculated from the synthetic and original data provides a standardised lack-of-fit = t(diff) %*% V^(-1) t(diff), where %*% represents the matrix product and V^(-1) is the inverse of the variance-covariance

matrix for the mean coefficients from the original data. If the model used to synthesize the data is correct this quantity, which is a Mahalanobis distance measure, will follow a chi-squared distribution with degrees of freedom, and thus expectation, equal to the number of parameters (p) in the fitted model. When object$incomplete = TRUE the variance-covariance matrix of the coefficients is estimated from the differences between the m estimates and the lack-of-fit statistic follows a Hotelling's T*2 distribution and the lack-of-fit statistic is referred to an F(p, m - p).

lof.pvalue
: p-value for the combined lack-of-fit test of the NULL hypothesis that the method used for synthesis retains all relationships between variables that influence the parameters of the fit.

ci.plot
: ggplot of the the coefficients with confidence intervals for models based on observed and synthetic data. If return.plot was set to FALSE then ci.plot is NULL.

print.coef
: a logical value determining whether tables of estimates for the original and synthetic data should be printed.

m
: the number of synthetic versions of the original (observed) data.

ncoef
: the number of coefficients in the fitted model (including an intercept).

incomplete
: whether methods for incomplete synthesis due to Reiter (2003) have been used in calculations.

population.inference
: whether intervals as decribed by Karr et al. (2016) have been calculated.

References

Karr, A., Kohnen, C.N., Oganian, A., Reiter, J.P. and Sanil, A.P. (2006). A framework for evaluating the utility of data altered to protect confidentiality. *The American Statistician*, **60**(3), 224-232.

Nowok, B., Raab, G.M and Dibben, C. (2016). synthpop: Bespoke creation of synthetic data in R. *Journal of Statistical Software*, **74**(11), 1-26. doi:10.18637/jss.v074.i11.

Reiter, J.P. (2003) Inference for partially synthetic, public use microdata sets. *Survey Methodology*, **29**, 181-188.

See Also

summary.fit.synds

Examples

```
ods <- SD2011[,c("sex","age","edu","smoke")]
s1 <- syn(ods, m = 3)
f1 <- glm.synds(smoke ~ sex + age + edu, data = s1, family = "binomial")
compare(f1, SD2011)
compare(f1, SD2011, print.coef = TRUE, plot = "coef")
```

compare.synds *Compare univariate distributions of synthesised and observed data*

Description

Compare synthesised data set with the original (observed) data set using percent frequency tables and histograms. When more than one synthetic data set has been generated (`object$m > 1`), by default pooled synthetic data are used for comparison.

This function can be also used with synthetic data NOT created by `syn()`, but then an additional parameter `cont.na` might need to be provided.

Usage

```
## S3 method for class 'synds'
compare(object, data, vars = NULL,
        msel = NULL, stat = "percents", breaks = 20, ngroups =5,
        nrow = 2, ncol = 2, rel.size.x = 1,
        utility.stats = c("pMSE", "S_pMSE", "df"),
        utility.for.plot = "S_pMSE",
        cols = c("#1A3C5A","#4187BF"),
        plot = TRUE, table = FALSE,
        print.flag = TRUE, ...)

## S3 method for class 'data.frame'
compare(object, data, vars = NULL, cont.na = NULL,
        msel = NULL, stat = "percents", breaks = 20,ngroups = 5,
        nrow = 2, ncol = 2, rel.size.x = 1,
        utility.stats = c("pMSE", "S_pMSE", "df"),
        utility.for.plot = "S_pMSE",
        cols = c("#1A3C5A","#4187BF"),
        plot = TRUE, table = FALSE,
        print.flag = TRUE, compare.synorig = TRUE, ...)

## S3 method for class 'list'
compare(object, data, vars = NULL, cont.na = NULL,
        msel = NULL, stat = "percents", breaks = 20,ngroups = 5,
        nrow = 2, ncol = 2, rel.size.x = 1,
        utility.stats = c("pMSE", "S_pMSE", "df"),
        utility.for.plot = "S_pMSE",
        cols = c("#1A3C5A","#4187BF"),
        plot = TRUE, table = FALSE,
        print.flag = TRUE, compare.synorig = TRUE, ...)

## S3 method for class 'compare.synds'
print(x, ...)
```

Arguments

`object`	an object of class `synds`, which stands for 'synthesised data set'. It is typically created by function `syn()` and it includes `object$m` synthesised data set(s) as `object$syn`. Alternatively, when data are synthesised not using `syn()`, it can be a data frame with a synthetic data set or a list of data frames with synthetic data sets, all created from the same original data with the same variables and the same method.
`data`	an original (observed) data set.
`vars`	variables to be compared. If `vars` is `NULL` (the default) all synthesised variables are compared.
`cont.na`	a named list of codes for missing values for continuous variables if different from the `R` missing data code `NA`. The names of the list elements must correspond to the variables names for which the missing data codes need to be specified.
`msel`	index or indices of synthetic data copies for which a comparison is to be made. If `NULL` pooled synthetic data copies are compared with the original data.
`stat`	determines whether tables and plots present percentages `stat = "percents"`, the default, or counts `stat = "counts"`. If `m > 1` and `msel = NULL` average counts for synthetic data are derived.
`breaks`	the number of cells for the histogram.
`ngroups`	the number of groups used to categorise numeric variables when calculating the one-way utility measures.
`nrow`	the number of rows for the plotting area.
`ncol`	the number of columns for the plotting area.
`rel.size.x`	a number representing the relative size of x-axis labels.
`utility.stats`	a single string or a vector of strings that determines which utility measures to print. Must be a selection from: `"VW"`, `"FT"`,`"JSD"`, `"SPECKS"`, `"WMabsDD"`, `"U"`, `"G"`, `"pMSE"`, `"PO50"`, `"MabsDD"`, `"dBhatt"`, `"S_VW"`, `"S_FT"`, `"S_JSD"`, `"S_WMabsDD"`, `"S_G"`, `"S_pMSE"`, `"df"`. If `utility.stats = "all"`, all of these will be printed. For more information see the details section for `utility.tab`.
`utility.for.plot`	a single string that determines which utility measure to print in facet labels of the plot. Set to `NULL` to print variable names only.
`cols`	bar colors.
`plot`	a logical value with default set to `TRUE` indicating whether plots should be produced.
`table`	a logical value with default set to `FALSE` indicating whether tables should be printed.
`print.flag`	a logical value with default set to `TRUE` indicating whether a message should be printed as the statistic for each variable are calculated.
`compare.synorig`	a logical value to determine if the functions `synorig.compare()` should be used to check that data sets can be compared. Used when the synthetic data are supplied as a data.frame or a list when default set to TRUE.
`...`	additional parameters.
`x`	an object of class `compare.synds`.

Details

Missing data categories for numeric variables are plotted on the same plot as non-missing values. They are indicated by `miss.` suffix.

Numeric variables with fewer than 6 distinct values are changed to factors in order to make plots more readable.

Value

An object of class `compare.synds` which is a list including a list of comparative frequency tables (`tables`) and a ggplot object (`plots`) with bar charts/histograms. If multiple plots are produced they and their corresponding frequency tables are stored as a list.

References

Nowok, B., Raab, G.M and Dibben, C. (2016). synthpop: Bespoke creation of synthetic data in R. *Journal of Statistical Software*, **74**(11), 1-26. doi:10.18637/jss.v074.i11.

See Also

`multi.compare`

Examples

```
ods <- SD2011[ , c("sex", "age", "edu", "marital", "ls", "income")]
s1  <- syn(ods, cont.na = list(income = -8))

### synthetic data provided as a 'synds' object
compare(s1, ods, vars = "ls")
compare(s1, ods, vars = "income", stat = "counts",
        table = TRUE, breaks = 10)

### synthetic data provided as 'data.frame'
compare(s1$syn, ods, vars = "ls")
compare(s1$syn, ods, vars = "income", cont.na = list(income = -8),
        stat = "counts", table = TRUE, breaks = 10)
```

`disclosure` *Disclosure measures*

Description

Calculates disclosure measures for synthetic data. NOTE: The other function that calculates disclosure results for multiple targets has been renamed as multi.disclosure from disclosure.summary.

Usage

```
## S3 method for class 'synds'
disclosure(object, data, keys , target , print.flag = TRUE,
           denom_lim = 5, exclude_ov_denom_lim = FALSE, not.targetlev = NULL,
           usetargetNA = TRUE, usekeysNA = TRUE,
          exclude.keys =NULL, exclude.keylevs = NULL, exclude.targetlevs = NULL,
           ngroups_target = NULL, ngroups_keys = NULL,
           thresh_1way = c(50, 90),thresh_2way = c(4, 80),
           digits = 2, to.print =c("short"),...)

## S3 method for class 'data.frame'
disclosure(object, data,cont.na = NULL, keys , target , print.flag = TRUE,
           denom_lim = 5, exclude_ov_denom_lim = FALSE,
           not.targetlev = NULL,
           usetargetNA = TRUE, usekeysNA = TRUE,
          exclude.keys =NULL, exclude.keylevs = NULL, exclude.targetlevs = NULL,
           ngroups_target = NULL, ngroups_keys = NULL,
           thresh_1way = c(50, 90),thresh_2way = c(4, 80),
           digits = 2, to.print =c("short"), compare.synorig = TRUE, ...)

## S3 method for class 'list'
disclosure(object, data,cont.na = NULL, keys , target , print.flag = TRUE,
           denom_lim = 5, exclude_ov_denom_lim = FALSE,
           not.targetlev = NULL,
           usetargetNA = TRUE, usekeysNA = TRUE,
          exclude.keys =NULL, exclude.keylevs = NULL, exclude.targetlevs = NULL,
           ngroups_target = NULL, ngroups_keys = NULL,
           thresh_1way = c(50, 90),thresh_2way = c(4, 80),
           digits = 2, to.print =c("short"), compare.synorig = TRUE, ...)

## S3 method for class 'disclosure'
print(x,  to.print =NULL, digits = NULL, ...)
```

Arguments

object	an object of class synds, which stands for 'synthesised data set'. It is typically created by function syn() and it includes object$m synthesised data set(s) as object$syn. This a single data set when object$m = 1 or a list of length object$m when object$m > 1. Alternatively, when data are synthesised not using syn(), it can be a data frame with a synthetic data set or a list of data frames with synthetic data sets, all created from the same original data with the same variables.
data	the original (observed) data set.
cont.na	For data NOT supplied as a synthetic data object created by synthpop, this gives special values for continuous variables as described in the documentation for the function syn.
keys	vector of variable names or column numbers in data that are also present in the synthetic data to act as quasi-identifiers for identity or attribute disclosure.

`target` name of target variable for attribute disclosure.

`denom_lim` Limit to use to exclude large key-target group, see next item.

`exclude_ov_denom_lim`
logical to exclude key targetcombinations that contribute more than `denom_lim` disclosive records. These are often flagged from `thresh_2way` where the first element corresponds to `denom_lim`

`print.flag` logical value as to whether a line is printed as disclosure is calculated for each synthetic data set.

`digits` number of digits to print for disclosure measures.

`usetargetNA` determines whether NA values in target are to be used in checking for disclosure

`usekeysNA` determines whether NA values in keys are to be used in checking for disclosure.

`not.targetlev` Character variable giving level of target to be excluded from disclosure measures. Usually identified by checklev_1way.

`exclude.keys` vector of names of keys that, with the next two items will define the target and key combinations to be excluded from the calculation of disclosure measures. Often identified by checklev_2way.

`exclude.keylevs`
vector of the same length as exclude.keys that give the levels to be excluded for the corresponding key.

`exclude.targetlevs`
vector of the same length as exclude.keys that give the levels of target that will be excluded for each key and key level.

`ngroups_target` Unless set to NULL (the default) a numeric target variable will be grouped into `ngroups_target` categories.

`ngroups_keys` Unless set to NULL (the default) any numeric variable will be grouped into categories. If `ngroups_keys` is of length 1 all numeric keys will be have the same number of groups. Otherwise `ngroups_keys` needs to be the same length as keys and will give the number of groups for each key. If an element of `ngroups_keys` is zero, no grouping will be done.

`thresh_1way` A vector of two numeric values both of which meed to be exceeded for warnings about a level of the target that may be dominating the results. The first is the count of all disclosive records for this level of the target, and the second is the % of all original records for this level of the target. Default is c(50, 90), meaning a group of 50 disclosive records for this level of the target where they make up over 90% of all disclosive records.

`thresh_2way` A vector of two numeric values both of which meed to be exceeded for warnings about a level of the target that may be dominating the results. The first is the count of disclosive records for a quasi-identifier used to identify possible s that are searched for the most disclosive key-target combination. The second is the percentage of all original records for each combination examined that must be exceeded to trigger a warning. Default is c(5, 80), meaning a pairs found from key-target groups of more than 5 records where over 80% of all the original values with these key-target pairs have this level of the target.

`to.print` Vector to determine what aspects of an object of class disclosure will be printed. Must consist of one or more of the following "short", "ident", "attrib","allCAPs", "all", "check_1way", "check_2way", "exclusions". Default is "short" giving a brief summary.

`compare.synorig` a logical value to determine if the functions `synorig.compare()` should be used to check that data sets can be compared. Used when the synthetic data are supplied as a data.frame or a list when default set to TRUE.

`...` additional parameters

`x` an object of class `disclosure`.

Details

Calculates identity disclosure measures for a for a set of keys, (quasi identifiers) and attribute disclosure measures for one variable from the same set of keys considered as a target. The function `multi.disclosure` calls this function and summarises the attribute disclosure measures for multiple targets. See the vignette

Value

An object of class `disclosure` which is a list with the following components.

`call` the call that created the object.

`ident` Table of measures of identity disclosure one for each synthesis. Measures are "UiO","UiS","UiSiO" and "repU". See vignette disclosure.pdf for an explanation of these and the following measures.

`attrib` Table of measures of attribute disclosure one for each synthesis. These include "DiO","DiS","iSO","DiSCO" and "DiSDiO". The measures "DiO" and "DiS" are the percentage of the target that are disclosed from the original and synthetic data with these keys. The next measure "iSO" gives the percentage of the key combinations in the synthetic data that are present in the original - one was in which the disclosure. "DiSCO" gives the percentage of original records where the attribution to the target is correct as judged from the original. "DiSDiO" gives the % of origina; records in "DISCO" that are unique in the original data. The table also as gives the maximum and mean of the denominators for the "DiSCO" measure i.e. the distribution for every record that leads to a correct disclosure of the number of observations with the same keys and the same correct target in the synthetic data. Large denominators are often an indication that the disclosure is something that might be expected from prior knowledge of relations.

`allCAPs` Table of the following measures of correct attribution probability: "baseCAPd","CAPd", "CAPs" , "DCAP" and "TCAP"'

`check_1way` A data frame with one record per synthesis identifying the level of the target with numbers of disclosive records that are above thresholds defined by `thresh_1way`, with default value c(50,90). This means that there must be more than 50 disclosive records with this level of the target, and that 90% or more of all disclosive records must have this target. The value of most_dis_lev will

be blank if no level exceeds these thresholds. Note this level will be identified for data without excluded or missing values of keys if there are any excluded records.

`check1` The level of the target identified by check_1way ‘ or blank if none

`check_2way` A list of length number of syntheses giving details for each of the two-way combinations of target and keys where the the numbers of disclosive records are above thresholds defined by `thresh_2way`. The default value for this is c(5, 80), meaning that there must be at least 5 records with this combination of targets and keys and that 80% or more of records in the original data with this level of the key will have this level of the target. If no combinations exceed `thresh_2way` for one of the syntheses then the list element is NULL. Such disclosive combinations are often associated with a high prior probability of the target from just this level of one of the keys in the original data. Note these combinations will be identified for data without excluded or missing values of keys if there are any excluded combinations or target if any of `usekeysNA` or `usetargetNA` are FALSE.

`Nexclusions` A list of length number of syntheses with number of records excluded from attribute measures for different reasons.

`keys` as input

`digits` as input

`Norig` Number of records in data

`to.print` as input

Note

See package vignette disclosure.pdf for additional information including formal definitions of all quantities and worked examples.

References

See references in package vignette

See Also

`syn` `multi.disclosure`

Examples

```
library(synthpop)
ods <- SD2011[, c("sex", "age", "edu", "marital", "income")]
odsF <- numtocat.syn(ods, numtocat = "income", catgroups = 7, cont.na = list(income = -8))
s1 <- syn(odsF$data, method = "ctree",seed = 75, m=3, k=1000)
disc1 <- disclosure(s1, odsF$data, target = "income",
                    keys = c("sex", "age", "edu","marital"))
```

glm.synds, lm.synds *Fitting (generalized) linear models to synthetic data*

Description

Fits generalized linear models or simple linear models to the synthesised data set(s) using `glm` and `lm` function respectively.

Usage

```
glm.synds(formula, family = "binomial", data,  ...)
lm.synds(formula, data, ...)

## S3 method for class 'fit.synds'
print(x, msel = NULL, ...)
```

Arguments

`formula`	a symbolic description of the model to be estimated. A typical model has the form `response ~ predictors`. See the documentation of `glm` and `formula` for details.
`family`	a description of the error distribution and link function to be used in the model. See the documentation of `glm` and `family` for details.
`data`	an object of class `synds`, which stands for 'synthesised data set'. It is typically created by function `syn` and it includes `data$m` synthesised data set(s).
`...`	additional parameters passed to `glm` or `lm`.
`x`	an object of class `fit.synds`.
`msel`	index or indices of synthetic data copies for which coefficient estimates are to be displayed. If `NULL` (default) the combined (average) coefficient estimates are printed.

Value

The `summary` function (`summary.fit.synds`) can be used to obtain the combined results of models fitted to each of the `m` synthetic data sets.

An object of class `fit.synds`. It is a list with the following components:

`call`	the original call to `glm.synds` or `lm.synds`.
`mcoefavg`	combined (average) coefficient estimates.
`mvaravg`	combined (average) variance estimates of `mcoef`.
`analyses`	`summary.glm` or `summary.lm` object respectively or a list of `m` such objects.
`fitting.function`	function used to fit the model.
`n`	a number of cases in the original data.

k	a number of cases in the synthesised data.
proper	a logical value indicating whether synthetic data were generated using proper synthesis.
m	the number of synthetic versions of the observed data.
method	a vector of synthesising methods applied to each variable in the saved synthesised data.
incomplete	a logical value indicating whether the dependent variable in the model was not synthesised.
mcoef	a matrix of coefficients estimates from all m syntheses.
mvar	a matrix of variance estimates from all m syntheses.

See Also

glm, lm, multinom.synds, polr.synds, compare.fit.synds, summary.fit.synds

Examples

```
### Logit model
ods <- SD2011[1:1000, c("sex", "age", "edu", "marital", "ls", "smoke")]
s1 <- syn(ods, m = 3)
f1 <- glm.synds(smoke ~ sex + age + edu + marital + ls, data = s1, family = "binomial")
f1
print(f1, msel = 1:2)

### Linear model
ods <- SD2011[1:1000,c("sex", "age", "income", "marital", "depress")]
ods$income[ods$income == -8] <- NA
s2 <- syn(ods, m = 3)
f2 <- lm.synds(depress ~ sex + age + log(income) + marital, data = s2)
f2
print(f2,1:3)
```

---

mergelevels.syn *Merge levels of factors in a data frame*

Description

merges levels of selected variables in a data frame either according to minimum numbers in a category or according to user-defined rules.

Usage

```
mergelevels.syn(data, vars = NULL, newlabel = FALSE, addNA = FALSE,
                    print.flag = FALSE, minsize = 10, merge.byhand =
                    FALSE, merge.details = NULL)
```

Arguments

`data`	An observed data set before synthesis.
`vars`	a vector of names or numbers for the variables for which categories are to be merged. defaults to all factors in data when sert to NULL
`minsize`	The minimum size that of combined categories when `merge.byhand` is FALSE
`newlabel`	When merge.byhand is FALSE when `newlabel = TRUE` causes the new factor levels to be labelled with a concatenation of the labels of all the categories that have contributed. `newlabel = FALSE` labels the new category "Other"
`addNA`	Causes the NA category to be included when determining which groups are below `minsize` and also to allow NA values, designated by "NA", to be combined with others in merge.details
`print.flag`	prints tables of variables before and after recoding.
`merge.byhand`	Uses the information in `merge.byhand` to create groupings, ignoring the parameters `minsize` and `vars`
.	
`merge.details`	A named list of variable names with names giving the names of the variables that will have levels merged. Each item is a vector where the first item is the name for the new combined level and the other entries are the levels to be merged. If it exists already the levels with small counts will be added to it, otehrwise a new level will be formed.

Value

A data frame of the same size and structure as `data` with levels of selected variables merged.

See Also

sdc

Examples

```
test <- SD2011[1:20]
data.mlevs1 <- mergelevels.syn(test, vars = c(3,5,18:20),minsize = 20, addNA = TRUE,
              print.flag = TRUE, newlabel = TRUE)

mlevs <- list(agegr = c("60+", "60-64"  , "65+"),socprof = c("NEW","UNEMPLOYED","FARMER"))
data.mlevs2 <-mergelevels.syn(test, merge.byhand = TRUE, merge.details = mlevs,
           addNA=TRUE, print.flag = TRUE)
```

multi.compare *Multivariate comparison of synthesised and observed data*

Description

Graphical comparisons of a variable (var) in the synthesised data set with the original (observed) data set within subgroups defined by the variables in a vector by. var can be a factor or a continuous variable and the plots produced will depend on the class of var. The variables in by will usually be factors or variables with only a few values.

Usage

```
multi.compare(object, data, var = NULL, by = NULL, msel = NULL,
  barplot.position = "fill", cont.type = "hist", y.hist = "count",
  boxplot.point = TRUE, binwidth = NULL, ...)
```

Arguments

object	an object of class synds, which stands for 'synthesised data set'. It is typically created by function syn() and it includes object$m synthesised data set(s).
data	an original (observed) data set.
var	variable to be compared between observed and synthetic data within subgroups.
by	variables to be tabulated or cross-tabulated to form groups.
barplot.position	type of barplot. The default "fill" gives a single bar with the proportions in each group while "dodge" gives side-by-side bars with the numbers in each category.
cont.type	default "hist" gives histograms and "boxplot" gives boxplots.
y.hist	defines y scale for histograms - "count" is default; "density" gives proportions.
boxplot.point	default (TRUE) adds individual points to boxplots.
msel	numbers of synthetic data sets to be used - must be numbers in the range 1:object$m. If NULL pooled synthetic data copies are compared with the original data.
binwidth	sets width of a bin for histograms.
...	additional parameters that can be supplied to ggplot.

Value

Plots as specified above. A table of the numbers in the subgroups is printed to the R console.

Numeric variables with fewer than 6 distinct values are changed to factors in order to make plots more readable.

See Also

compare.synds, compare.fit.synds

Examples

```
### default synthesis of selected variables
vars <- c("sex", "age", "edu", "smoke")
ods  <- na.omit(SD2011[1:1000, vars])
s1 <- syn(ods)

### categorical var
multi.compare(s1, ods, var = "smoke", by = c("sex","edu"))

### numeric var
multi.compare(s1, ods, var = "age", by = c("sex"), y.hist = "density", binwidth = 5)
multi.compare(s1, ods, var = "age", by = c("sex", "edu"), cont.type = "boxplot")
```

`multi.disclosure` *Disclosure measures for multiple of target variables.*

Description

Calculates, prints and plots tables of disclosure measures for a set of target variables from a fixed set of keys to form quasi-identifiers. The calculations of disclosure measures are done by the function `disclosure` for each target.

This function can be also used with synthetic data NOT created by `syn()`, or even made anonymous by other methods such as sampling More details of the measures calculated can be found in the package vignette "Disclosure measures for Synthetic Data".

Usage

```
## S3 method for class 'synds'
multi.disclosure(object, data,
           keys , targets = NULL, print.flag = TRUE,
           denom_lim = 5, exclude_ov_denom_lim = FALSE,
           not.targetslev = NULL,
           usetargetsNA = TRUE,  usekeysNA = TRUE,
         exclude.keys = NULL, exclude.keylevs = NULL,  exclude.targetlevs = NULL,
           ngroups_targets = NULL, ngroups_keys = NULL,
           ident.meas = "repU", attrib.meas = "DiSCO",
           thresh_1way = c(50, 90),thresh_2way = c(4, 80),
           digits = 2, plot = TRUE,  ...)


## S3 method for class 'data.frame'
multi.disclosure(object, data, cont.na = NULL,
           keys , targets = NULL,  print.flag = TRUE,
           denom_lim = 5, exclude_ov_denom_lim = FALSE,
           not.targetslev = NULL,
           usetargetsNA = TRUE,  usekeysNA = TRUE,
         exclude.keys = NULL, exclude.keylevs = NULL,  exclude.targetlevs = NULL,
```

```
            ngroups_targets = NULL, ngroups_keys = NULL,
            ident.meas = "repU", attrib.meas = "DiSCO",
            thresh_1way = c(50, 90),thresh_2way = c(4, 80),
            digits = 2, plot = TRUE,  compare.synorig = TRUE,  ...)

## S3 method for class 'list'
multi.disclosure(object, data, cont.na = NULL,
             keys , targets = NULL,  print.flag = TRUE,
             denom_lim = 5, exclude_ov_denom_lim = FALSE,
            not.targetslev = NULL,
            usetargetsNA = TRUE,  usekeysNA = TRUE,
          exclude.keys = NULL, exclude.keylevs = NULL, exclude.targetlevs = NULL,
            ngroups_targets = NULL, ngroups_keys = NULL,
            ident.meas = "repU", attrib.meas = "DiSCO",
            thresh_1way = c(50, 90),thresh_2way = c(4, 80),
            digits = 2, plot = TRUE, compare.synorig = TRUE,...)


## S3 method for class 'multi.disclosure'
print(x, digits = NULL, plot = NULL, to.print =  c("ident","attrib"),
       ...)
```

Arguments

`object` an object of class `synds`, which stands for 'synthesised data set'. It is typically created by function `syn()` and it includes `object$m` synthesised data set(s) as `object$syn`. This a single data set when `object$m = 1` or a list of length `object$m` when `object$m > 1`. Alternatively, when data are synthesised not using `syn()`, it can be a data frame with a synthetic data set or a list of data frames with synthetic data sets, all created from the same original data with the same variables and the same method.

`data` the original (observed) data set.

`cont.na` For data NOT supplied as a synthetic data object created by `synthpop`, this gives special values for continuous variables as described in the documentation for the function `syn`.

`keys` a vector of strings with the names of variables to be used in combination to form a quasi identifier.

`targets` a vector of strings with the names of variables to be used as targets for the disclosure measures. Defaults to all variables in both original and synthetic data that are not in `keys`.

`denom_lim` an integer that determines the limit above which a warning to check the two way relationships for potential prior disclosure information.

`exclude_ov_denom_lim` TRUE/FALSE according to whether disclosive groups with denominators > denom_lim should be excluded from disclosure measures.

`not.targetslev` Vector of same length as targets giving level of each target to be excluded from calculating disclosure measures. Set elements for unaffected targets as blanks.

`print.flag` TRUE/FALSE to print out line as disclosure for each member of targets is calculated.

`usetargetsNA` A logical vector of the same length as `targets` that determines if `NA` values of each are to be considered disclosive. Defaults to `FAULT` for all.

`usekeysNA` A logical vector of the same length as `keys` that determines if `NA` values of each key are to be considered disclosive. Defaults to `FAULT` for all keys.

`exclude.keys` A list of same length as `targets` giving the keys for two way exclusions for the ith target. For details see documentation in `disclosure`

`exclude.keylevs`
A list of same length as `targets` giving the levels of keys for two way exclusions for the ith target. For details see documentation in `disclosure`

`exclude.targetlevs`
A list of same length as `targets` giving the levels of target for two way exclusions for the ith target. For details see documentation in `disclosure`

`ngroups_targets`
Unless set to NULL (the default) numeric target variables will be grouped into `ngroups_target` categories. If `ngroups_keys` is of length 1 all numeric targets will be have the same number of groups. Otherwise `ngroups_targets` needs to be a vector of the same length as targets and will give the number of groups for each target. If an element of `ngroups_targets` is zero, no grouping will be done.

`ngroups_keys` Unless set to NULL (the default) any numeric variable will be grouped into categories If `ngroups_keys` is of length 1 all numeric keys will be have the same number of groups. Otherwise `ngroups_keys` needs to be the same length as keys and will give the number of groups for each key. If an element of `ngroups_keys` is zero, no grouping will be done.

`ident.meas` Choice of statistics to use as a measure of identity disclosure. Must be a selection from: `"repU"` or `"UiSiO"`. See `disclosure` for explanations of measures.

`attrib.meas` Choice of statistics to use as a measure of attribute disclosure. Must be a selection from: `"DiSCO"` or `"DiSDiO"`. See `disclosure` for explanations of measures.

`thresh_1way` A vector of two numeric values both of which meed to be exceeded for warnings about a level of the target that may be dominating the results. The first is the count of all disclosive records, and the second is the % of all records for this level of the target. Default is c(50, 90), meaning a group of 50 disclosive records for this level of the target where they make up over 90% of all disclosive records.

`thresh_2way` A vector of two numeric values both of which meed to be exceeded for warnings about a level of the target that may be dominating the results. The first is the count of all disclosive records for this key-target combination and the second is the percantage of all disclosive records for this combination. Default is c(5, 80), meaning a group of more than 5 records where over 80% of all the original values with this key have this level of the target.

`digits` number of digits to print for the disclosure measures.

`plot` determines if plot will be produced when the result is printed.

`print` logical value that determines if a summary of results is to be printed.

`compare.synorig`
: a logical value to determine if the functions `synorig.compare()` should be used to check that data sets can be compared. Default set to `FALSE` except when the synthetic data are supplied as a data.frame or a list when set to TRUE.

`to.print`
: Vector of items to be printed including "ident", "attrib", both or NULL

`...`
: additional parameters

`x`
: an object of class `multi.disclosure`.

Details

Calculates measures of identity and attribution disclosure from the keys specified in `keys` with the function `disclosure`. For attribute disclosure a table with one line for each target can be printed or plotted. Details are in help file for `disclosure`.

Value

An object of class `multi.disclosure` which is a list with the following components:

`attrib.table`
: a table with the selected attribute disclosure measure (`attrib.meas`) for synthetic data and corresponding measure for the original data "CAPd" if (`attrib.meas`) is "DCAP", and "DiO" for others.

`attrib.plot`
: plot of attrib.table with labels indicating where large denominators suggest checking.

`keys`
: see above.

`ident.orig`
: value of identity disclosure `UiO` from the original data, see help file for `disclosure`.

`ident.syn`
: value of identity disclosure `ident.meas` from the synthetic data, see help file for `disclosure`.

`Norig`
: Number of records in data.

`denom_lim`
: see above.

`exclude_ov_denom_lim`
: see above.

`digits`
: see above.

`usetargetsNA`
: see above.

`usekeysNA`
: see above.

`ident.meas`
: see above.

`attrib.meas`
: see above.

`m`
: see above.

`plot`
: see above.

`output.list`
: A named list with a component for each target where each component is the output from the function `disclosure` for that target. This allows check_1way and check_2way to be examined for each target.

`call`
: R call used to create the object

References

to follow link to vignette

See Also

disclosure

Examples

```
ods <- SD2011[, c("sex", "age", "edu", "marital", "region", "income")]
s1 <- syn(ods)

### synthetic data provided as a 'data.frame' object
t1 <- multi.disclosure(s1$syn, ods,
keys = c("sex", "age", "edu"))

### synthetic data provided as a 'synds' object
t1 <- multi.disclosure(s1, ods,
keys = c("sex", "age", "edu"))
```

---

multinom.synds *Fitting multinomial models to synthetic data*

Description

Fits multinomial models to the synthesised data set(s) using the multinom function.

Usage

```
multinom.synds(formula, data,  ...)
```

Arguments

formula	a symbolic description of the model to be estimated. A typical model has the form response ~ predictors. See the documentation of multinom and formula for details.
data	an object of class synds, which stands for 'synthesised data set'. It is typically created by function syn and it includes data$m synthesised data set(s).
...	additional parameters passed to multinom.

Value

To print the results the print function (print.fit.synds) can be used. The summary function (summary.fit.synds) can be used to obtain the combined results of models fitted to each of the m synthetic data sets.

An object of class fit.synds. It is a list with the following components:

call	the original call to multinom.synds.
mcoefavg	combined (average) coefficient estimates.
mvaravg	combined (average) variance estimates of mcoef.
analyses	an object summarising the fit to each synthetic data set or a list of m such objects. Note that this is different from the object created by summary.multinom to make it compatible with other fitting methods. In particular the coefficients are vectors, not matrices.
fitting.function	function used to fit the model.
n	a number of cases in the original data.
k	a number of cases in the synthesised data.
proper	a logical value indicating whether synthetic data were generated using proper synthesis.
m	the number of synthetic versions of the observed data.
method	a vector of synthesising methods applied to each variable in the saved synthesised data.
incomplete	a logical value indicating whether the dependent variable in the model was not synthesised.
mcoef	a matrix of coefficients estimates from all m syntheses.
mvar	a matrix of variance estimates from all m syntheses.

See Also

multinom, glm.synds, polr.synds, print.fit.synds, summary.fit.synds, compare.fit.synds

Examples

```
ods <- SD2011[1:1000, c("sex", "age", "edu", "marital", "ls", "smoke")]
s1 <- syn(ods, m = 3)
f1 <- multinom.synds(edu  ~ sex + age, data = s1)
summary(f1)
print(f1, msel = 1:2)
compare(f1, SD2011)
```

---

numtocat.syn *Group numeric variables before synthesis*

Description

Selected numeric variables are grouped into factors with ranges selected from the data.

Usage

```
numtocat.syn(data, numtocat = NULL, print.flag = TRUE, cont.na = NULL,
             catgroups = 5, style.groups = "quantile")
```

Arguments

`data`	a data frame.
`numtocat`	a vector of numbers or variable names of numeric variables to be grouped into factors. If `NULL` all the numeric variables in data will be grouped.
`print.flag`	if TRUE a list of grouped variables is printed.
`cont.na`	a named list that gives the values of the named variables to be treated as separate categories, often missing values like `-8`. See the corresponding parameter of `syn()`.
`catgroups`	a single integer or a vector of integers indicating the target number of groups for the variables in numtocat in the same order as numtocat, or as their relative postions in data. The achieved number of groups may be different if, for example there are fewer than `ngroups` distinct values.
`style.groups`	parameter of the function `classInt()` that determines how the breaks used to categorise each variable are chosen. See the help file for `classInt()` for details. The default setting `"quantile"` makes groups of approximately equal size. To divide into approximately equal ranges we suggest using `"fisher"`.

Value

A list with the following components:

`data`	a data frame with the numeric variables replaced by factors grouped into ranges.
`breaks`	a named list of the breaks used to divide each numeric variable into categories.
`levels`	a named list of the levels for the categories of each numeric variable.
`orig`	a data frame with the original numeric data.
`cont.na`	a named list of the levels for the categorical version of each numeric variable.
`numtocat`	names of the variables changed to categories.
`ind`	positions in data of the variables changed to categories.

Examples

```
SD2011.cat <- numtocat.syn(SD2011, cont.na = list(income = -8 , unempdur = -8,
nofriend = -8))
summary(SD2011.cat$data)
```

`polr.synds` *Fitting ordered logistic models to synthetic data*

Description

Fits ordered logistic models to the synthesised data set(s) using the `polr` function.

Usage

```
polr.synds(formula, data, ...)
```

Arguments

formula	a symbolic description of the model to be estimated. A typical model has the form response ~ predictors. See the documentation of polr and formula for details.
data	an object of class synds, which stands for 'synthesised data set'. It is typically created by function syn and it includes data$m synthesised data set(s).
...	additional parameters passed to polr.

Value

To print the results the print function (print.fit.synds) can be used. The summary function (summary.fit.synds) can be used to obtain the combined results of models fitted to each of the m synthetic data sets.

An object of class fit.synds. It is a list with the following components:

call	the original call to polr.synds.
mcoefavg	combined (average) coefficient estimates.
mvaravg	combined (average) variance estimates of mcoef.
analyses	an object summarising the fit to each synthetic data set or a list of m such objects. Note that this is different from the object created by summary.polr to make it compatible with other fitting methods for the compare.fit.synds and summary.fit.synds functions. In particular the coefficients combine the items coefficients and zeta from summary.polr to make a combined vector of coefficients.
fitting.function	function used to fit the model.
n	a number of cases in the original data.
k	a number of cases in the synthesised data.
proper	a logical value indicating whether synthetic data were generated using proper synthesis.
m	the number of synthetic versions of the observed data.
method	a vector of synthesising methods applied to each variable in the saved synthesised data.
incomplete	a logical value indicating whether the dependent variable in the model was not synthesised.
mcoef	a matrix of coefficients estimates from all m syntheses.
mvar	a matrix of variance estimates from all m syntheses.

See Also

polr, glm.synds, multinom.synds, print.fit.synds, summary.fit.synds, compare.fit.synds

Examples

```
ods <- SD2011[1:1000, c("sex", "age", "edu", "marital", "ls", "smoke")]
s1 <- syn(ods, m = 3)
f1 <- polr.synds(edu  ~ sex + age, data = s1)
summary(f1)
print(f1, msel = 1:2)
compare(f1, SD2011)
```

`read.obs` *Importing original data sets form external files*

Description

Imports data data sets form external files into a data frame. Currently supported files include: sav (SPSS), dta (Stata), xpt (SAS), csv (comma-separated file), tab (tab-delimited file) and txt (delimited text files). For SPSS, Stata and SAS it uses functions from the `foreign` package with some adjustments where necessary.

Usage

```
read.obs(file, convert.factors = TRUE, lab.factors = FALSE,
export.lab = FALSE, ...)
```

Arguments

`file`	the name of the file (including extension) which the data are to be read from.
`convert.factors`	a logical value indicating whether variables with value labels in Stata and SPSS should be converted into R factors with those levels.
`lab.factors`	a logical value indicating whether variables with complete value labels but imported using their numeric codes (`convert.factors = FALSE`) should be converted from numeric to factor variables.
`export.lab`	a logical variable indicating whether labels from SPSS or Stata should be exported to an external file.
`...`	additional parameters passed to read functions.

Value

A data frame with an imported data set. For SPSS, Stata and SAS it has attributes with labels.

See Also

`write.syn`

replicated.uniques *Replications in synthetic data*

Description

Determines which unique units in the synthesised data set(s) have combinations of variables in the keys as follows:

1) unique in original data

2) unique in the synthetic data set(s)

3) unique in synthetic data and present,but not necessarily unique in original

4) unique in synthetic and unique in original.

For each of 3) and 4) results are returned that identify the rows in the synthetic data with each type of unique. This function is called by sdc where there are options to include each type of unique.

Usage

```
replicated.uniques(object, data, keys = names(data))

## S3 method for class 'repuniq.synds'
print(x, ...)
```

Arguments

object	an object of class synds, which stands for 'synthesised data set'. It is typically created by function syn() and it includes object$m synthesised data set(s).
data	the original observed data set.
keys	Variables to be used as quasi-identifiers to check for unique combinations.
...	additional parameters
x	an object of class repuniq.synds; a result of a call to replicated.uniques().

Value

A list of class "repuniq.synds" with the following components:

m	number of synthetic data sets in object object$m
n	number of rows in data object$n
k	number of rows in of synthetic data set(s) in object object$k
res_tab	Table or list of tables with numbers and percentages of uniques
synU.rm	A vector of length object$k TRUE/FALSE values wherea TRUE value identifies a unique in synthetic and prtesent in the original
repU.rm	A vector of length object$k TRUE/FALSE values where a TRUE value identifies a replicated unique

See Also

sdc

Examples

```
ods <- SD2011[1:1000,c("sex","age","region","edu","marital","smoke")]
s1 <- syn(ods, m = 2)
replicated.uniques(s1,ods, keys = c("sex","age","region"))
```

SD2011 *Social Diagnosis 2011 - Objective and Subjective Quality of Life in Poland*

Description

Sample of 5,000 individuals from the Social Diagnosis 2011 survey; selected variables only.

Usage

```
SD2011
```

Format

A data frame with 5,000 observations on the following 35 variables:

sex Sex

age Age of person, 2011

agegr Age group, 2011

placesize Category of the place of residence

region Region (voivodeship: a province in Poland, the highest level of administrative division in the country)

edu Highest educational qualification, 2011

eduspec Discipline of completed qualification

socprof Socio-economic status, 2011

unempdur Total duration of unemployment in the last 2 years (in months)

income Personal monthly net income

marital Marital status

mmarr Month of marriage

ymarr Year of marriage

msepdiv Month of separation/divorce

ysepdiv Year of separation/divorce

ls Perception of life as a whole

depress Depression symptoms indicator

trust View on interpersonal trust

trustfam Trust in own family members

trustneigh Trust in neighbours

sport Active engagement in some form of sport or exercise

nofriend Number of friends

smoke Smoking cigarettes

nociga Number of cigarettes smoked per day

alcabuse Drinking too much alcohol

alcsol Starting to use alcohol to cope with troubles

workab Working abroad in 2007-2011

wkabdur Total time spent on working abroad

wkabint Plans to go abroad to work in the next two years

wkabintdur Intended duration of working abroad

emcc Intended destination country

englang Knowledge of English language

height Height of person

weight Weight of person

bmi Body mass index

Note

Please note that the original variable names have been changed to make them more self-explanatory. Some variable labels have been adjusted as well.

Source

Council for Social Monitoring. Social Diagnosis 2000-2011: integrated database. http://www.diagnoza.com/index-en.html [downloaded on 13/12/2013]

References

Czapinski J. and Panek T. (Eds.) (2011). Social Diagnosis 2011. Objective and Subjective Quality of Life in Poland - full report. Contemporary Economics, Volume 5, Issue 3 (special issue) http://ce.vizja.pl/en/issues/volume/5/issue/3#art254

Examples

```
spineplot(englang ~ agegr, data = SD2011, xlab = "Age group", ylab = "Knowledge of English")
boxplot(income ~ sex, data = SD2011[SD2011$income != -8,])
```

sdc *Tools for statistical disclosure control (sdc)*

Description

Labeling, top and bottom coding, smoothing numeric data, and removing different types of unique records defined by keys from synthetic data. The function calls `replicated.uniques` to identify the rows to be excluded from the synthetic data set(s)

Usage

```
sdc(object, data,keys = NULL, prefix = NULL, suffix = NULL, label = NULL,
rm.uniques.in.orig = FALSE, rm.replicated.uniques = FALSE,
recode.vars = NULL, bottom.top.coding = NULL,
 recode.exclude = NULL, smooth.vars = NULL)
```

Arguments

`object` an object of class `synds`, which stands for 'synthesised data set'. It is typically created by function `syn()` and it includes `object$m` synthesised data set(s).

`data` the original (observed) data set.

`keys` Variables to be used as quasi-identifiers to check for unique combinations. Passed to `replicated.uniques` to exclude rows in the synthetic data.

`prefix` A character string to be added as a prefix to all variable names in the synthetic data set(s)

`suffix` A character string to be added as a suffix to all variable names in the synthetic data set(s)

`label` a single string with a label to be added to the synthetic data sets as a new variable to make it clear that the data are synthetic/fake.

`rm.uniques.in.orig` a logical value indicating whether unique replicates of key variables that are present in the orginal data set should be removed from synthetic data set(s).

`rm.replicated.uniques` a logical value indicating whether unique replicates of key variables that are also unique in the orginal data set should be removed.

`recode.vars` a single string or a vector of strings with name(s) of variable(s) to be bottom- or/and top-coded.

`bottom.top.coding` a list of two-element vectors specifing bottom and top codes for each variable in `recode.vars`. If there is no need for one of bottom or top coding `NA` should be used. If only one variable is to be recoded, codes can be given as a two-element vector.

`recode.exclude` a list specifying for each variable in `recode.vars` values to be excluded from recoding, e.g. missing data codes. If all non-missing values should be considered for recoding `NA` should be used if missing values are present. If only one variable is to be recoded, code(s) can be given as a single number or a vector.

`smooth.vars` a single string or a vector of strings with name(s) of numeric variable(s) to be smoothed (`smooth.spline` function is used).

Value

An `object` provided as an argument adjusted in accordance with the other parameters' values.

See Also

`replicated.uniques`

Examples

```
ods <- SD2011[1:1000,c("sex","age","region","edu","marital","income")]
s1 <- syn(ods, m = 2)
s1.sdc <- sdc(s1, ods, keys = c("sex","age","region"),suffix = "_synthetic",
label="false_data", rm.uniques.in.orig = TRUE,
recode.vars = c("age","income"),
bottom.top.coding = list(c(20,80),c(NA,2000)),
recode.exclude = list(NA,c(NA,-8)))
head(s1.sdc$syn[[2]])
```

---

summary.fit.synds *Inference from synthetic data*

Description

Combines the results of models fitted to each of the `m` synthetic data sets.

Usage

```
## S3 method for class 'fit.synds'
summary(object, population.inference = FALSE, msel = NULL,
  real.varcov = NULL, incomplete = NULL, ...)

## S3 method for class 'summary.fit.synds'
print(x, ...)
```

Arguments

`object` an object of class `fit.synds` created by fitting a model to synthesised data set using function `glm.synds`, `lm.synds`,`multinom.synds` or `polr.synds`.

population.inference
: a logical value indicating whether inference should be made to population quantities. If FALSE inference is made to the results that would be expected from an analysis of the original data. This option should be selected if the synthetic data are being used for exploratory analysis, but the final published results will be obtained by running code on the original confidential data. If population.inference = TRUE results would allow population inference to be made from the synthetic data. In both cases the inference will depend on the synthesising model being correct, but this can be checked by running the same analysis on the real data, see compare.fit.synds.

msel
: index or indices of the synthetic datasets (1, . . . , m), for which summaries of fitted models are to be produced. If NULL (default) only the summary of combined estimates is produced.

real.varcov
: the estimated variance-covariance matrix of the fit of the model to the original data. This parameter is used in the function compare.fit.synds which has the original data as one of its parameters.

incomplete
: Logical variable as to whether population inference for incomplete synthesis is to be used. If this is left at a NULL value it will be determined by whether the dependent variable has been synthesised. See also below as output.

...
: additional parameters.

x
: an object of class summary.fit.synds.

Details

The mean of the estimates from each of the m synthetic data sets yields asymptotically unbiased estimates of the coefficients if the observed data conform to the distribution used for synthesis. The standard errors are estimated differently depending whether inference is made for the results that we would expect to obtain from the observed data or for the parameters of the population that we assume the observed data are sampled from. The standard errors also differ according to whether synthetic data were produced using simple or proper synthesis (for details see Raab et al. (2017)).

Value

An object of class summary.fit.synds which is a list with the following components:

call
: the original call to glm.synds or lm.synds.

proper
: a logical value indicating whether synthetic data were generated using proper synthesis.

population.inference
: a logical value indicating whether inference is made to population coefficients or to the results that would be expected from an analysis of the original data (see above).

incomplete
: a logical value indicating whether the dependent variable in the model was not synthesised. It is derived in the synthpop implementation of the fitting functions (lm.synds, glm.synds, multinom.synds and polr.synds) and saved with the fitted object. When TRUE inference with population.inference = TRUE uses the method proposed by Reiter (2003) for what he terms partially synthetic data.

This method requires multiple syntheses (m > 1). If m = 1, incomplete = TRUE and population.inference = TRUE the results will be still calculated and returned with warning. This will usually give standard errors that are larger than they should be. This method can be forced by setting incomplete = TRUE as a parameter because it can also be used for complete synthesis.

fitting.function
: function used to fit the model.

m
: the number of synthetic versions of the original (observed) data.

coefficients
: a matrix with combined estimates. If inference is required to the results that would be obtained from an analysis of the original data, (population.inference = FALSE) the coefficients are given by xpct(Beta), the standard errors by xpct(se.Beta) and the corresponding Z-statistic by xpct(Z). If the synthetic data are to be used to make inferences to population quantities (population.inference = TRUE), the coefficients are given by Beta.syn, their standard errors by se.Beta.syn and the Z-statistic by Z.syn (see vignette on inference for more details).

n
: a number of cases in the original data.

k
: the number of cases in the synthesised data. Note that if k and n are not equal and population.inference = FALSE (the default), then the standard errors produced will estimate what would be expected by an analysis of the original data set of size n.

analyses
: summary.glm or summary.lm object respectively or a list of m such objects.

msel
: index or indices of synthetic data copies for which summaries of fitted models are produced. If NULL only a summary of combined estimates is produced.

References

Nowok, B., Raab, G.M and Dibben, C. (2016). synthpop: Bespoke creation of synthetic data in R. *Journal of Statistical Software*, **74**(11), 1-26. doi:10.18637/jss.v074.i11.

Raab, G.M., Nowok, B. and Dibben, C. (2017). Practical data synthesis for large samples. *Journal of Privacy and Confidentiality*, **7**(3), 67-97. Available at: https://journalprivacyconfidentiality.org/index.php/jpc/article/view/407

Reiter, J.P. (2003) Inference for partially synthetic, public use microdata sets. *Survey Methodology*, **29**, 181-188.

See Also

compare.fit.synds, summary, print

Examples

```
ods <- SD2011[1:1000,c("sex","age","edu","ls","smoke")]

### simple synthesis
s1 <- syn(ods, m = 5)
f1 <- glm.synds(smoke ~ sex + age + edu + ls, data = s1, family = "binomial")
summary(f1)
summary(f1, population.inference = TRUE)
```

```
### proper synthesis
s2 <- syn(ods, m = 5, method = "parametric", proper = TRUE)
f2 <- glm.synds(smoke ~ sex + age + edu + ls, data = s2, family = "binomial")
summary(f2)
summary(f2, population.inference = TRUE)
```

---

summary.synds *Synthetic data object summaries*

Description

Produces summaries of the synthesised variables. When more than one synthetic data set has been generated (object$m > 1), by default summaries are calculated by averaging summary values for all synthetic data copies (see msel argument).

Usage

```
## S3 method for class 'synds'
summary(object, msel = NULL, maxsum = 7,
  digits = max(3, getOption("digits")-3), ...)

## S3 method for class 'summary.synds'
print(x, ...)
```

Arguments

object	an object of class synds; a result of a call to syn.
msel	index or indices of synthetic data copies for which a summary is desired. If NULL (default) summaries are calculated by averaging summary values for all synthetic data copies.
maxsum	integer, indicating how many levels should be shown for factors.
digits	integer, used for number formatting with format.
...	additional arguments passed to summary.
x	an object of class summary.synds.

Details

See summary for more details.

Value

An object of class summary.synds, which is a list with the following components:

m	the number of synthetic versions of the original (observed) data.

msel	index or indices of synthetic data copies for which a summary is produced. If NULL summaries are calculated by averaging summary values for all synthetic data copies.
method	a vector of synthesising methods applied to each variable in the saved synthesised data.
result	a table or a list of tables (if more than one synthetic data set is selected) with summaries of synthesised variables.

References

Nowok, B., Raab, G.M and Dibben, C. (2016). synthpop: Bespoke creation of synthetic data in R. *Journal of Statistical Software*, **74**(11), 1-26. doi:10.18637/jss.v074.i11.

See Also

summary,print

Examples

```
s1 <- syn(SD2011[,c("sex","age","edu","marital")], m = 3)
summary(s1)
summary(s1, msel = c(1,3))
```

---

syn *Generating synthetic data sets*

Description

Generates synthetic version(s) of a data set. Function `syn.strata()` performs stratified synthesis.

Usage

```
syn(data, method = "cart", visit.sequence = (1:ncol(data)),
    predictor.matrix = NULL,
    m = 1, k = nrow(data), proper = FALSE,
    minnumlevels = 1, maxfaclevels = 60,
    rules = NULL, rvalues = NULL,
    cont.na = NULL, semicont = NULL,
    smoothing = NULL, event = NULL, denom = NULL,
    drop.not.used = FALSE, drop.pred.only = FALSE,
    default.method = c("normrank", "logreg", "polyreg", "polr"),
    numtocat = NULL, catgroups = rep(5, length(numtocat)),
    models = FALSE, print.flag = TRUE, seed = "sample", ...)

syn.strata(data, strata = NULL,
    minstratumsize = 10 + 10 * length(visit.sequence),
    tab.strataobs = TRUE, tab.stratasyn = FALSE,
```

```
    method = "cart", visit.sequence = (1:ncol(data)),
    predictor.matrix = NULL,
    m = 1, k = nrow(data), proper = FALSE,
    minnumlevels = 1, maxfaclevels = 60,
    rules = NULL, rvalues = NULL,
    cont.na = NULL, semicont = NULL,
    smoothing = NULL, event = NULL, denom = NULL,
    drop.not.used = FALSE, drop.pred.only = FALSE,
    default.method = c("normrank", "logreg", "polyreg", "polr"),
    numtocat = NULL, catgroups = rep(5,length(numtocat)),
    models = FALSE, print.flag = TRUE, seed = "sample", ...)

## S3 method for class 'synds'
print(x, ...)
```

Arguments

`data` a data frame or a matrix (n x p) containing the original data. Observations are in rows and variables are in columns.

`method` a single string or a vector of strings of length `ncol(data)` specifying the synthesising method to be used for each variable in the data. Order of variables is exactly the same as in `data`. If specified as a single string, the same method is used for all variables in a visit sequence unless a data type or a position in a visit sequence requires a different method. If `method` is set to `"parametric"` the default synthesising method specified by the `default.method` argument are applied. Variables that are transformations of other variables can be synthesised using a passive method that is specified as a string starting with ~ (see `syn.passive`). Variables that need not to be synthesised have the empty method `""`. By default all variables are synthesised using `"cart"` method, which is `rpart` implementation of a CART model (see `syn.cart`). See details for more information on method.

`visit.sequence` a character vector of names of variables or an integer vector of their column indices specifying the order of synthesis. The default sequence `1:ncol(data)` implies that column variables are synthesised from left to right. See details for more information.

`predictor.matrix` a square matrix of size `ncol(data)` specifying the set of column predictors to be used for each target variable in the row. Each entry has value 0 or 1. A value of 1 means that the column variable is used as a predictor for the row variable. Order of variables is exactly the same as in `data`. By default all variables that are earlier in the visit sequence are used as predictors. For the default visit sequence (`1:ncol(data)`) the default `predictor.matrix` will have values of 1 in the lower triangle. See details for more information.

`m` number of synthetic copies of the original (observed) data to be generated. The default is `m = 1`.

`k` a size of the synthetic data set (k x p), which can be smaller or greater than the size of the original data set (n x p). The default is `nrow(data)` which means that

the number of individuals in the synthesised data is the same as in the original (observed) data (k = n).

`proper` a logical value with default set to `FALSE`. If `TRUE` proper synthesis is conducted.

`minnumlevels` a minimum number of values a numeric variable should exceed to be treated as numeric during the synthesis. Numeric variables with only `minnumlevels` or fewer distinct values are changed into factors. If set to `1` (default) numeric variables are left unchanged unless they have only one non-missing value.

`maxfaclevels` a maximum number of factor levels that can be handled. It can be increased to allow the synthesis to run but too large a value may cause computational problems, especially for parametric methods.

`rules` a named list of rules for restricted values. Restricted values are those that are determined explicitly by values of other variables. The names of the list elements must correspond to the variables names for which the rules need to be specified.

`rvalues` a named list of the values corresponding to the rules specified by `rules`.

`cont.na` a named list of codes for missing values for continuous variables if different from the `R` missing data code `NA`. The names of the list elements must correspond to the variables names for which the missing data codes need to be specified.

`semicont` a named list of values at which semi-continuous variables have spikes. The names of the list elements must correspond to the names of the semi-continuous variables.

`smoothing` a single string specifying a smoothing method for all numeric variables in the data or a named list specifying a smoothing method to be used for selected variables. Available methods include: `"spline"` (recommended), `"rmean"`, `"density"`, and `""`). Smoothing can only be applied to continuous variables synthesised using `sample`, `ctree`, `cart`, `rf`, `bag`, `ranger`, `normrank`, `pmm` or `nested` method. The names of the list elements must correspond to the names of the variables whose values are to be smoothed. Smoothing is applied to the synthesised values. For more details see `syn.smooth`.

`event` a named list specifying for survival data the names of corresponding event indicators. The names of the list elements must correspond to the names of the survival variables.

`denom` a named list specifying for variables to be modelled using binomial regression the names of corresponding denominator variables. The names of the list elements must correspond to the names of the variables to be modelled using binomial regression.

`drop.not.used` a logical value. If `TRUE` (default) variables not used in synthesis are not saved in the synthesised data and are not included in the corresponding synthesis parameters.

`drop.pred.only` a logical value. If `TRUE` (default) variables not synthesised and used as predictors only are not saved in the synthesised data.

`default.method` a vector of four strings containing the default parametric synthesising methods for numerical variables, factors with two levels, unordered factors with more than two levels and ordered factors with more than two levels respectively. They are used when `method` is set to `"parametric"` or when there is an inconsistency between variable type and provided method.

numtocat	a vector of numbers or names to indicate columns of data that are to be divided into groups to allow the grouped variables to be synthesised as factors. The target number of groups for each variable is specified by catgroups. After the grouped variables have been synthesised the numeric variables are synthesised from them by the method syn.nested and are placed in the same position in the synthetic data as in the original. The grouped variables are not stored in the synthetic data. If you want to keep the categorised values you should change the relevant variables in data before running syn() with the function numtocat.syn()
catgroups	An integer or a vector of integers of the same length as numtocat giving the target number of groups into which of the numeric variables is to be categorised. The function group_var from the classInt package performs the categorisation.
models	if TRUE parameters of models fitted to the original data and used to generate the synthetic values are stored.
print.flag	if TRUE (default) synthesising history and information messages will be printed at the console. For silent computation use print.flag = FALSE.
seed	an integer to be used as an argument for the set.seed(). If no integer is provided, the default "sample" will generate one and it will be stored. To prevent generating an integer set seed to NA.
...	additional arguments to be passed to synthesising functions. See section 'Details' below for more information.
strata	a numeric vector with strata identifiers or a string vector with names of stratifying variable(s).
minstratumsize	minimum size of each stratum.
tab.strataobs	a logical value indicating whether a frequency table of the number of observations in strata in the original data set should be printed.
tab.stratasyn	a logical value indicating whether a frequency table of the number of observations in strata in the synthetic data set(s) should be printed.
x	an object of class synds; a result of a call to syn().

Details

Only variables that are in visit.sequence with corresponding non-empty method are synthesised. The only exceptions are event indicators. They are synthesised along with the corresponding time to event variables and should not be included in visit.sequence. All other variables (not in visit.sequence or in visit.sequence with a corresponding blank method) can be used as predictors. Including them in visit.sequence generates a default predictor.matrix reflecting the order of variables in the visit.sequence otherwise predictor.matrix has to be adjusted accordingly. All predictors of the variables that are not in visit.sequence or are in visit.sequence but with a blank method are removed from predictor.matrix.

Variables to be synthesised that are not synthesised yet cannot be used as predictors. Also all variables used in passive synthesis or in restricted values rules (rules) have to be synthesised before the variables they apply to.

Mismatch between data type and synthesising method stops execution and print an error message but numeric variables with number of levels less than minnumlevels are changed into factors and

methods are changed automatically, if necessary, to methods for categorical variables. Methods for variables not in a visit sequence will be changed into blank.

The built-in elementary synthesising methods defined by conditional distributions include:

ctree, cart classification and regression trees (CART), see `syn.cart`

bagging, random forests, ranger methods using ensembles of CART trees, see `syn.bag`, `syn.rf`, and `syn.ranger`

survctree classification and regression trees (CART) for duration time data (parametric methods for survival data are not implemented yet), see `syn.survctree`

norm normal linear regression, see `syn.norm`

normrank normal linear regression preserving the marginal distribution, see `syn.normrank`

lognorm, sqrtnorm, cubertnorm normal linear regression after natural logarithmic, square root and cube root transformation of a dependent variable respectively, see `syn.lognorm`

logreg logistic regression, see `syn.logreg`

polyreg unordered polytomous regression, see `syn.polyreg`

polr ordered polytomous regression, see `syn.polr`

pmm predictive mean matching, see `syn.pmm`

sample random sample from the observed data, see `syn.sample`

passive function of other synthesised data, see `syn.passive`

nested bootstrap sample within each category of the original grouping variable, see `syn.nested`

satcat bootstrap sample within each category of the crosstabulation of all the predictor variables, see `syn.satcat`

These methods use a group of variables that are synthesised together. They must always be together at the start of the visit sequence:

catall fit a saturated log-linear model, see `syn.catall`

ipf fit a log-linear model, defined by its margins, by iterative proportional fitting see `syn.ipf`

The functions corresponding to these methods are called `syn.method`, where `method` is a string with the name of a synthesising method. For instance a function corresponding to `ctree` function is called `syn.ctree`. A new synthesising method can be introduced by writing a function named `syn.newmethod` and then specifying `method` parameter of `syn()` function as `"newmethod"`.

In order to use `"nested"` sampling, `method` parameter of `syn` function has to be specified as `"nested.varname"`, where `"varname"` is the name of the grouped (less detailed) variable, the only one used in nested synthesis. A variable synthesised using `"nested"` method is excluded from synthesising other variables except when used for `"nested"` method.

Additional parameters can be passed to synthesising methods as part of the `dots` argument. They have to be named using period-separated method and parameter name (`method.parameter`). For instance, in order to set a `minbucket` (minimum number of observations in any terminal node of a CART model) for a `ctree` synthesising method, `ctree.minbucket` has to be specified. The parameters are method-specific and will be used for all variables to be synthesised using that method. See help for `syn.method` for further details about the allowed parameters for a specific method.

Value

The summary function (summary.synds) can be used to obtain a summary of the synthesised variables.

An object of class synds, which stands for 'synthesised data set'. It is a list with the following components:

- call — an original call to syn().
- m — number of synthetic versions of the original (observed) data.
- syn — a data frame (for m = 1) or a list of m data frames (for m > 1) with synthetic data set(s).
- method — a vector of synthesising methods applied to each variable in the saved synthesised data.
- visit.sequence — a vector of column indices of the visiting sequence. The indices refer to the columns in the saved synthesised data.
- predictor.matrix — a matrix specifying the set of predictors used for each variable in the saved synthesised data.
- smoothing — a vector specifying smoothing methods applied to each variable in the saved synthesised data.
- event — a vector of integers specifying for survival data the column indices for corresponding event indicators. The indices refer to the columns in the saved synthesised data.
- denom — a vector of integers specifying for variables modelled using binomial regression the column indices for corresponding denominator variables. The indices refer to the columns in the saved synthesised data.
- proper — a logical value indicating whether proper synthesis was conducted.
- n — a number of cases in the original data.
- k — a number of cases in the synthesised data.
- rules — a list of rules for restricted values applied to the synthetic data.
- rvalues — a list of the values corresponding to the rules specified by rules.
- cont.na — a list of codes for missing values for continuous variables.
- semicont — a list of values for semi-continuous variables at which they have spikes.
- drop.not.used — a logical value indicating whether variables not used in synthesis are saved in the synthesised data and corresponding synthesis parameters.
- drop.pred.only — a logical value indicating whether variables not synthesised and used as predictors only are saved in the synthesised data.
- models — if models = TRUE a named list of estimates of models fitted to the original data and used to generate the synthetic values is returned from the $fit component of each method (e.g. syn.cart()). The list is ordered by the variables position in the data, and any models used to predict missing values are appended to the list.
- seed — an integer used as a set.seed() argument.

var.lab	a vector of variable labels for data imported from SPSS using read.obs().
val.lab	a list of value labels for factors for data imported from SPSS using read.obs().
obs.vars	a vector of all variable names in the observed data set.

When syn.strata() is used there are two additional components:

strata.syn	a factor variable or a list of factor variables containing stratum values for all observation units in syn.
strata.lab	a character vector of strata labels.

Note also that when syn.strata is used most values of the items are matrices with each row corresponding to a stratum or lists with one element per stratum.

Note

See package vignette for additional information.

References

Nowok, B., Raab, G.M and Dibben, C. (2016). synthpop: Bespoke creation of synthetic data in R. *Journal of Statistical Software*, **74**(11), 1-26. doi:10.18637/jss.v074.i11.

See Also

compare.synds, summary.synds

Examples

```
### selection of variables
vars <- c("sex","age","marital","income","ls","smoke")
ods  <- SD2011[1:1000, vars]

### default synthesis
s1 <- syn(ods)
s1

### synthesis with default parametric methods
s2 <- syn(ods, method = "parametric", seed = 123)
s2$method

### multiple synthesis of selected variables with customised methods
s3 <- syn(ods, visit.sequence = c(2, 1, 4, 5), m = 2,
          method = c("logreg","sample","","normrank","ctree",""),
          ctree.minbucket = 10)
summary(s3)
summary(s3, msel = 1:2)

### adjustment to the default predictor matrix
s4.ini <- syn(data = ods, visit.sequence = c(1, 2, 5, 3),
              m = 0, drop.not.used = FALSE)
pM.cor <- s4.ini$predictor.matrix
```

```
pM.cor["marital","ls"] <- 0
s4 <- syn(data = ods, visit.sequence = c(1, 2, 5, 3),
          predictor.matrix = pM.cor)

### handling missing values in continuous variables
s5 <- syn(ods, cont.na = list(income = c(NA, -8)))

### rules for restricted values - marital status of males under 18 should be 'single'
s6 <- syn(ods, rules = list(marital = "age < 18 & sex == 'MALE'"),
          rvalues = list(marital = 'SINGLE'), method = "parametric", seed = 123)
with(s6$syn, table(marital[age < 18 & sex == 'MALE']))
### results for default parametric synthesis without the rule
with(s2$syn, table(marital[age < 18 & sex == 'MALE']))

### synthesis with ipf for all variables
s7 <- syn(ods[, 1:3], method = "ipf", numtocat = "age")

### alternatively group the numeric variable before synthesis to save
### the grouped data rather than the numeric in the synthetic data set
ods.cat <- numtocat.syn(ods, numtocat = "age", catgroups = 10)$data
s8 <- syn(ods.cat[, 1:3], method = "ipf")

### stratified synthesis
s9 <- syn.strata(ods, strata = "sex")
```

---

syn.bag *Synthesis with bagging*

Description

Generates univariate synthetic data using bagging. It uses `randomForest` function from the **randomForest** package with number of sampled predictors equal to number of all predictors.

Usage

```
syn.bag(y, x, xp, smoothing = "", proper = FALSE, ntree = 10, ...)
```

Arguments

`y`	an original data vector of length `n`.
`x`	a matrix (`n` x `p`) of original covariates.
`xp`	a matrix (`k` x `p`) of synthesised covariates.
`smoothing`	smoothing method for numeric variable. See `syn.smooth`.
`proper`	for proper synthesis (`proper = TRUE`) a model is fitted to a bootstrapped sample of the original data.
`ntree`	number of trees to grow.
`...`	additional parameters passed to `randomForest`.

Details

...

Value

A list with two components:

`res`	a vector of length `k` with synthetic values of `y`.
`fit`	the model fitted to the observed data that was used to produce synthetic values.

References

...

See Also

`syn`, `syn.rf`, `syn.cart`, `randomForest`, `syn.smooth`

---

syn.catall — *Synthesis of a group of categorical variables from a saturated model*

Description

A saturated model is fitted to a table produced by cross-tabulating all the variables.

Usage

```
syn.catall(x, k, proper = FALSE, priorn = 1, structzero = NULL,
           maxtable = 1e8, epsilon = 0,  delta = 0.05,rand = TRUE,
          noisetype = "",  ...)
```

Arguments

`x`	a data frame (`n` x `p`) of the set of original variables.
`k`	a number of rows in each synthetic data set - defaults to `n`.
`proper`	if `proper = TRUE` `x` is replaced with a bootstrap sample before synthesis, thus effectively sampling from the posterior distribution of the model, given the data.
`priorn`	the sum of the parameters of the Dirichelet prior which can be thought of as a pseudo-count giving the number of observations that inform prior knowledge about the parameters.
`structzero`	a named list of lists that defines which cells in the table are structural zeros and will remain as zeros in the synthetic data, by leaving their prior as zeros. Each element of the `structzero` list is a list that describes a set of cells in the table defined by a combination of two or more variables and a name of each such element must consist of those variable names seperated by an underscore, e.g. `sex_edu`. The length of each such element is determined by the number of variables and each component gives the variable levels (numeric or labels) that define the structural zero cells (see an example below).

`maxtable`	a number of cells in the cross-tabulation of all the variables that will trigger a severe warning.
`epsilon`	measures scale of Laplace Gaussian or Exponential noise to be added under differential privacy (DP)
`delta`	Parameter delta for Gaussian noise when this method is used to make the synthesis approximately differentially private (DP)
`rand`	for DP versions determines if multinomial noise is to be added to DP counts. If it is set to false the DP adjusted counts are simply rounded to a whole number in a manner that preserves the desired sample size (k).
`noisetype`	One of "Laplace" "Gaussian" or "Exponential" to determine the type of noise to be added that will make the synthesis DP (Laplace, Exponential) or approximately DP (Gaussian). For noisetype "Gaussian" your synthesis will fail if epsilon >1 or delta not in range 0-1.
`...`	additional parameters.

Details

When used in `syn` function the group of categorical variables with `method = "catall"` must all be together at the start of the `visit.sequence`. Subsequent variables in `visit.sequence` are then synthesised conditional on the synthesised values of the grouped variables. A saturated model is fitted to a table produced by cross-tabulating all the variables. Prior probabilities for the proportions in each cell of the table are specified from the parameters of a Dirichlet distribution with the same parameter for every cell in the table that is not a structural zero (see above). The sum of these parameters is `priorn` so that each one is $priorn/N$ where N is the number of cells in the table that are not structural zeros. The default `priorn = 1` can be thought of as equivalent to the knowledge that 1 observation would be equally likely to be in any cell that is not a structural zero. The posterior expectation, given the observed counts, for the probability of being in a cell with observed count n_i is thus $(n_i + priorn/N)/(N + priorn)$. The synthetic data are generated from a multinomial distribution with parameters given by these probabilities.

Unlike `syn.satcat`, which fits saturated conditional models, the synthesised data can include any combination of variables, except those defined by the combinations of variables in `structzero`.

NOTE that when the function is called by setting elements of method in `syn()` to `"catall"`, the parameters `priorn`, `structzero`, `maxtable`, `epsilon`, and `rand` must be supplied to `syn` as e.g. `catall.priorn`.

Value

A list with two components:

`res`	a data frame of dimension `k x p` containing the synthesised data.
`fit`	the cross-tabulation of all the original variables used.

Examples

```
ods <- SD2011[, c(1, 4, 5, 6, 2, 10, 11)]
table(ods[, c("placesize", "region")])
```

```
# Each `placesize_region` sublist:
# for each relevant level of `placesize` defined in the first element,
# the second element defines regions (variable `region`) that do not
# have places of that size.

struct.zero <- list(
  placesize_region = list(placesize = "URBAN 500,000 AND OVER",
                          region = c(2, 4, 5, 8:13, 16)),
  placesize_region = list(placesize = "URBAN 200,000-500,000",
                          region = c(3, 4, 10:11, 13)),
  placesize_region = list(placesize = "URBAN 20,000-100,000",
                          region = c(1, 3, 5, 6, 8, 9, 14:15)))

# you could use  the object struct.zero in the command below
# byt devtools checking did not like it so have added the list instead

syncatall <- syn(ods, method = c(rep("catall", 4), "ctree", "normrank", "ctree"),
                 catall.priorn = 2, catall.structzero = list(
  placesize_region = list(placesize = "URBAN 500,000 AND OVER",
                          region = c(2, 4, 5, 8:13, 16)),
  placesize_region = list(placesize = "URBAN 200,000-500,000",
                          region = c(3, 4, 10:11, 13)),
  placesize_region = list(placesize = "URBAN 20,000-100,000",
                          region = c(1, 3, 5, 6, 8, 9, 14:15))))
```

`syn.ctree, syn.cart` *Synthesis with classification and regression trees (CART)*

Description

Generates univariate synthetic data using classification and regression trees (without or with bootstrap).

Usage

```
syn.ctree(y, x, xp, smoothing = "", proper = FALSE,
          minbucket = 5, mincriterion = 0.9, ...)
syn.cart(y, x, xp, smoothing = "", proper = FALSE,
         minbucket = 5, cp = 1e-08, ...)
```

Arguments

`y`	an original data vector of length `n`.
`x`	a matrix (`n` x `p`) of original covariates.
`xp`	a matrix (`k` x `p`) of synthesised covariates.
`smoothing`	smoothing method for numeric variable. See `syn.smooth`.
`proper`	for proper synthesis (`proper = TRUE`) a CART model is fitted to a bootstrapped sample of the original data.

minbucket	the minimum number of observations in any terminal node. See rpart.control and ctree_control for details.
cp	complexity parameter. Any split that does not decrease the overall lack of fit by a factor of cp is not attempted. Small values of cp will grow large trees. See rpart.control for details.
mincriterion	1 - p-value of the test that must be exceeded for a split to be retained. Small values of mincriterion will grow large trees. See ctree_control for details.
...	additional parameters passed to ctree_control for syn.ctree and rpart.control for syn.cart.

Details

The procedure for synthesis by a CART model is as follows:

1. Fit a classification or regression tree by binary recursive partitioning.
2. For each xp find the terminal node.
3. Randomly draw a donor from the members of the node and take the observed value of y from that draw as the synthetic value.

syn.ctree uses ctree function from the **party** package and syn.cart uses rpart function from the **rpart** package. They differ, among others, in a selection of a splitting variable and a stopping rule for the splitting process.

A Guassian kernel smoothing can be applied to continuous variables by setting smoothing parameter to "density". It is recommended as a tool to decrease the disclosure risk. Increasing minbucket is another means of data protection.

CART models were suggested for generation of synthetic data by Reiter (2005) and then evaluated by Drechsler and Reiter (2011).

Value

A list with two components:

res	a vector of length k with synthetic values of y.
fit	the fitted model which is an object of class rpart.object or ctree.object that can be printed or plotted.

References

Reiter, J.P. (2005). Using CART to generate partially synthetic, public use microdata. *Journal of Official Statistics*, **21**(3), 441–462.

Drechsler, J. and Reiter, J.P. (2011). An empirical evaluation of easily implemented, nonparametric methods for generating synthetic datasets. *Computational Statistics and Data Analysis*, **55**(12), 3232–3243.

See Also

syn, syn.survctree, rpart, ctree, syn.smooth

syn.ipf *Synthesis of a group of categorical variables by iterative proportional fitting*

Description

A fit to the table is obtained from the log-linear fit that matches the numbers in the margins specified by the margin parameters.

Usage

```
syn.ipf(x, k, proper = FALSE, priorn = 1, structzero = NULL,
        gmargins = "twoway", othmargins = NULL, tol = 1e-3,
        max.its = 5000, maxtable = 1e8, print.its = FALSE,
        epsilon = 0, delta = 0.05,
        noisetype = "Laplace", ...)
```

Arguments

x	a data frame of the set of original data to be synthesised.
k	a number of rows in each synthetic data set - defaults to n.
proper	if proper = TRUE x is replaced with a bootstrap sample before synthesis, thus effectively sampling from the posterior distribution of the model, given the data.
priorn	the sum of the parameters of the Dirichlet prior which can be thought of as a pseudo-count giving the number of observations that inform prior knowledge about the parameters.
structzero	a named list of lists that defines which cells in the table are structural zeros and will remain as zeros in the synthetic data, by leaving their prior as zeros. Each element of the structzero list is a list that describes a set of cells in the table defined by a combination of two or more variables and a name of each such element must consist of those variable names seperated by an underscore, e.g. sex_edu. The length of each such element is determined by the number of variables and each component gives the variable levels (numeric or labels) that define the structural zero cells (see an example below).
gmargins	a single character to define a group of margins. At present there is "oneway" and "twoway" option that creates, respectively, all 1-way and 2-way margins from the table.
othmargins	a list of margins that will be fitted. If gmargins is not NULL othmargins will be added to them.
tol	stopping criterion for Ipfp.
max.its	maximum umber of iterations allowed for Ipfp.
maxtable	the number of cells in the cross-tabulation of all the variables that will trigger a severe warning.

print.its	if true the iterations from Ipfp will be printed on the console. Otherwise only a message as to whether the iterations have converged will be given at the end of the fitting.
epsilon	epsilon value for overall differential privacy (DP) parameter. This is implemented by dividing the privacy budget equally over all the margins used to fit the data.
delta	Parameter for epsilon-delta differential privacy (DP) parameter, when noisetype = "Gaussian".
noisetype	One of "Laplace" or "Gaussian" to determine the type of noise to be added that will make the synthesis DP (Laplace) or approximately DP (Gaussian). For noisetype "Gaussian" your synthesis will fail if epsilon >1.
...	additional parameters.

Details

When used in syn function the group of variables with method = "ipf" must all be together at the start of the visit sequence. This function is designed for categorical variables, but it can also be used for numerical variables if they are categorised by specifying them in the numtocat parameter of the main function syn. Subsequent variables in visit.sequence are then synthesised conditional on the synthesised values of the grouped variables. A fit to the table is obtained from the log-linear fit that matches the numbers in the margins specified by the margin parameters. Prior probabilities for the proportions in each cell of the table are given by a Dirichlet distribution with the same parameter for every cell in the table that is not a structural zero. The sum of these parameters is priorn. The default priorn = 1 can be thought of as equivalent to the knowledge that 1 observation would be equally likely to fall in any cell of the table. The synthetic data are generated from a multinomial distribution with parameters given by the expected posterior probabilities for each cell of the table. If the maximum likelihood estimate from the log-linear fit to cell c_i is p_i and the table has N cells that are not structural zeros then the expectation of the posterior probability for this cell is $(p_i + priorn/N^2)/(1 + priorn/N^2)$ or equivalently $(N * p_i + priorn/N)/(N + priorn/N)$.

Unlike syn.satcat, which fits saturated models from their conditional distributions, x can include any combination of variables, including those not present in the original data, except those defined by structzero.

NOTE that when the function is called by setting elements of method in syn to "ipf", the parameters priorn, structzero, gmargins, othmargins, tol, max.its, maxtable, print.its and epsilon, must be supplied to syn as e.g. ipf.priorn.

Value

A list with two components:

res	a data frame with k rows containing the synthesised data.
fit	a list made up of two lists: the margins fitted and the original data for each margin.

Examples

```
ods <- SD2011[, c(1, 4, 5, 6, 2, 10, 11)]
table(ods[, c("placesize", "region")])
```

```
# Each `placesize_region` sublist:
# for each relevant level of `placesize` defined in the first element,
# the second element defines regions (variable `region`) that do not
# have places of that size.

struct.zero <- list(
  placesize_region = list(placesize = "URBAN 500,000 AND OVER",
                          region = c(2, 4, 5, 8:13, 16)),
  placesize_region = list(placesize = "URBAN 200,000-500,000",
                          region = c(3, 4, 10:11, 13)),
  placesize_region = list(placesize = "URBAN 20,000-100,000",
                          region = c(1, 3, 5, 6, 8, 9, 14:15)))

# you could use  the object struct.zero in the command below
# byt devtools checking did not like it so have added the list instead

synipf <- syn(ods, method = c(rep("ipf", 4), "ctree", "normrank", "ctree"),
              ipf.gmargins = "twoway", ipf.othmargins = list(c(1, 2, 3)),
              ipf.priorn = 2, ipf.structzero = list(
  placesize_region = list(placesize = "URBAN 500,000 AND OVER",
                          region = c(2, 4, 5, 8:13, 16)),
  placesize_region = list(placesize = "URBAN 200,000-500,000",
                          region = c(3, 4, 10:11, 13)),
  placesize_region = list(placesize = "URBAN 20,000-100,000",
                          region = c(1, 3, 5, 6, 8, 9, 14:15))))
```

`syn.lognorm, syn.sqrtnorm, syn.cubertnorm`

Synthesis by linear regression after transformation of a dependent variable

---

Description

Generates univariate synthetic data using linear regression of an outcome variable transformed by natural logarithm (`lognorm`), square root (`sqrtnorm`) or cube root (`cubertnorm`).

Usage

```
syn.lognorm(y, x, xp, proper = FALSE, ...)
syn.sqrtnorm(y, x, xp, proper = FALSE, ...)
syn.cubertnorm(y, x, xp, proper = FALSE, ...)
```

Arguments

`y`	an original data vector of length `n`.
`x`	a matrix (`n` x `p`) of original covariates.
`xp`	a matrix (`k` x `p`) of synthesised covariates.

proper	a logical value specifying whether proper synthesis should be conducted. See details.
...	additional parameters.

Details

Generates synthetic values using the spread around the fitted linear regression line of transformed y given x. For proper synthesis first the regression coefficients are drawn from normal distribution with mean and variance from the fitted model. The synthetic values are transformed back to the original scale.

Value

A list with two components:

res	a vector of length k with synthetic values of y.
fit	a data frame with regression coefficients and error estimates.

See Also

syn, syn.norm, syn.normrank

---

syn.logreg *Synthesis by logistic regression*

Description

Generates univariate synthetic data for binary or binomial response variable using logistic regression model.

Usage

```
syn.logreg(y, x, xp, denom = NULL, denomp = NULL, proper = FALSE, ...)
```

Arguments

y	an original data vector of length n.
x	a matrix (n x p) of original covariates.
xp	a matrix (k x p) of synthesised covariates.
denom	an original denominator vector of length n for a binomial regression model.
denomp	a synthesised denominator vector of length k for a binomial regression model.
proper	a logical value specifying whether proper synthesis should be conducted. See details.
...	additional parameters.

Details

Synthesis for binary response variables by the non-Bayesian or approximate Bayesian logistic regression model. The non-Bayesian method consists of the following steps:

1. Fit a logistic regression to the original data.
2. Calculate predicted inverse logits for synthesied covariates.
3. Compare the inverse logits to a random (0,1) deviate and get synthetic values.

The Bayesian version (for proper synthesis) includes additional step before computing inverse logits, namely drawing coefficients from normal distribution with mean and variance estimated in step 1.

The method relies on the standard `glm.fit` function. Warnings from `glm.fit` are suppressed. Perfect prediction is handled by the data augmentation method.

Value

A list with two components:

`res`	a vector of length `k` with synthetic values of `y`.
`fit`	a summary of the model fitted to the observed data and used to produce synthetic values.

See Also

`syn`, `glm`

`syn.nested` *Synthesis for a variable nested within another variable.*

Description

Synthesizes one variable (`y`) from another one (`x`) when `y` is nested in the categories of `x`. A bootstrap sample is created from the original values of `y` within each category of `xp` (the synthesised values of the grouping variable).

Usage

```
syn.nested(y, x, xp, smoothing = "", cont.na = NA,  ...)
```

Arguments

`y`	an original data vector of length `n` for the nested variable.
`x`	an original data vector of length `n` for the variable within which `y` is nested.
`xp`	a vector of length `k` with synthetic values of `x`.
`smoothing`	smoothing method. See `syn.smooth`.
`cont.na`	when `y` is numeric this can be a list or a vector giving values of `y` that indicate missing values.
`...`	additional parameters.

Details

An example would be when x is a classification of occupations and y is a more detailed sub-classification. It is intended that x is a categorical (factor) variable. A warning will be issued if the original y is not nested within x. A variable synthesised by syn.nested() is automatically excluded from predicting later variables because it will provide no extra information, given its grouping variable. syn.nested() is also used for the final synthesis of variables in syn() when the option numtocat is used to synthesise numerical variables as groups.

Value

A list with two components:

res	a vector of length k with synthetic values of y.
fit	a name of the method used for synthesis ("nested").

---

syn.norm *Synthesis by linear regression*

Description

Generates univariate synthetic data using linear regression analysis.

Usage

```
syn.norm(y, x, xp, proper = FALSE, ...)
```

Arguments

y	an original data vector of length n.
x	a matrix (n x p) of original covariates.
xp	a matrix (k x p) of synthesised covariates.
proper	a logical value specifying whether proper synthesis should be conducted. See details.
...	additional parameters.

Details

Generates synthetic values using the spread around the fitted linear regression line of y given x. For proper synthesis first the regression coefficients are drawn from normal distribution with mean and variance from the fitted model.

Value

A list with two components:

res	a vector of length k with synthetic values of y.
fit	a data frame with regression coefficients and error estimates.

See Also

syn, syn.normrank, syn.lognorm

---

syn.normrank *Synthesis by normal linear regression preserving the marginal distribution*

Description

Generates univariate synthetic data using linear regression analysis and preserves the marginal distribution. Regression is carried out on Normal deviates of ranks in the original variable. Synthetic values are assigned from the original values based on the synthesised ranks that are transformed from their synthesised Normal deviates.

Usage

```
syn.normrank(y, x, xp, smoothing = "", proper = FALSE, ...)
```

Arguments

y	an original data vector of length n.
x	a matrix (n x p) of original covariates.
xp	a matrix (k x p) of synthesised covariates.
smoothing	smoothing method. See syn.smooth.
proper	a logical value specifying whether proper synthesis should be conducted. See details.
...	additional parameters.

Details

First generates synthetic values of Normal deviates of ranks of the values in y using the spread around the fitted linear regression line of Normal deviates of ranks given x. Then synthetic Normal deviates of ranks are transformed back to get synthetic ranks which are used to assign values from y. For proper synthesis first the regression coefficients are drawn from normal distribution with mean and variance from the fitted model. A smoothing methods can be applied by setting smoothing parameter (see syn.smooth). It is recommended as a tool to decrease the disclosure risk.

Value

A list with two components:

res	a vector of length k with synthetic values of y.
fit	a data frame with regression coefficients and error estimates.

See Also

syn, syn.norm, syn.lognorm, syn.smooth

syn.passive *Passive synthesis*

Description

Derives a new variable according to a specified function of synthesised data.

Usage

```
syn.passive(data, func)
```

Arguments

data	a data frame with synthesised data.
func	a formula specifying transformations on data. It is specified as a string starting with ~.

Details

Any function of the synthesised data can be specified. Note that several operators such as +, -, * and ^ have different meanings in formula syntax. Use the identity function I() if they should be interpreted as arithmetic operators, e.g. "~I(age^2)". Function syn() checks whether the passive assignment is correct in the original data and fails with a warning if this is not true. The variables synthesised passively can be used to predict later variables in the synthesis except when they are numeric variables with missing data. A warning is produced in this last case.

Value

A list with two components:

res	a vector of length k including the result of applying the formula.
fit	a name of the method used for synthesis ("passive").

Author(s)

Gillian Raab, 2021 based on Stef van Buuren, Karin Groothuis-Oudshoorn, 2000

References

Van Buuren, S. and Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. doi:10.18637/jss.v045.i03

See Also

syn

Examples

```
### the examples shows how inconsistencies in the SD2011 data are picked up
### by syn.passive()
ods <- SD2011[, c("height", "weight", "bmi", "age", "agegr")]
ods$hsq <- ods$height^2
ods$sex <- SD2011$sex
meth <- c("cart", "cart", "~I(weight / height^2 * 10000)",
          "cart", "~I(cut(age, c(15, 24, 34, 44, 59, 64, 120)))",
          "~I(height^2)", "logreg")

## Not run:
### fails for bmi
s1 <- syn(ods, method = meth, seed = 6756, models = TRUE)

### fails for agegr
ods$bmi <- ods$weight / ods$height^2 * 10000
s2 <- syn(ods, method = meth, seed = 6756, models = TRUE)

### fails because of wrong order
ods$agegr <- cut(ods$age, c(15, 24, 34, 44, 59, 64, 120))
s3 <- syn(ods, method = meth, visit.sequence = 7:1,
          seed = 6756, models = TRUE)

## End(Not run)

### runs without errors
ods$bmi   <- ods$weight / ods$height^2 * 10000
ods$agegr <- cut(ods$age, c(15, 24, 34, 44, 59, 64, 120))
s4 <- syn(ods, method = meth, seed = 6756, models = TRUE)
### bmi and hsq do not predict sex because of missing values
s4$models$sex

### hsq with no missing values used to predict sex
ods2 <- ods[!is.na(ods$height),]
s5 <- syn(ods2, method = meth, seed = 6756, models = TRUE)
s5$models$sex

### agegr with missing values used to predict sex because not numeric
ods3 <- ods
ods3$age[1:4] <- NA
ods3$agegr <- cut(ods3$age, c(15, 24, 34, 44, 59, 64, 120))
s6 <- syn(ods3, method = meth, seed = 6756, models = TRUE)
s6$models$sex
```

---

syn.pmm *Synthesis by predictive mean matching*

Description

Generates univariate synthetic data using predictive mean matching.

Usage

```
syn.pmm(y, x, xp, smoothing = "", proper = FALSE, ...)
```

Arguments

y	an original data vector of length n.
x	a matrix (n x p) of original covariates.
xp	a matrix (k x p) of synthesised covariates.
proper	a logical value specifying whether proper synthesis should be conducted. See details.
smoothing	smoothing method. See documentation for syn.smooth.
...	additional parameters.

Details

Synthesis of y by predictive mean matching. The procedure is as follows:

1. Fit a linear regression to the original data.
2. Compute predicted values y.hat and ysyn.hat for the original x and synthesised xp covariates respectively.
3. For each predicted value ysyn.hat find donor observations with the closest predicted values y.hat (ties are broken by random selection), randomly sample one of them and take its observed value y as the synthetic value.

The Bayesian version (for proper synthesis) includes additional step before computing predicted values:

- Draw coefficients from normal distribution with mean and variance estimated in step 1 and use them to calculate predicted values for the synthesised covariates.

Value

A list with two components:

res	a vector of length k with synthetic values of y.
fit	a data frame with regression coefficients and error estimates.

See Also

syn, syn.smooth

syn.polr *Synthesis by ordered polytomous regression*

Description

Generates a synthetic categorical variable using ordered polytomous regression (without or with bootstrap).

Usage

```
syn.polr(y, x, xp, proper = FALSE, maxit = 1000, trace = FALSE,
  MaxNWts = 10000, ...)
```

Arguments

y	an original data vector of length n.
x	a matrix (n x p) of original covariates.
xp	a matrix (k x p) of synthesised covariates.
proper	for proper synthesis (proper = TRUE) a model is fitted to a bootstrapped sample of the original data.
maxit	the maximum number of iterations for nnet.
trace	switch for tracing optimization for nnet.
MaxNWts	the maximum allowable number of weights for nnet.
...	additional parameters passed to optim or nnet.

Details

Generates synthetic ordered categorical variables by the proportional odds logistic regression (polr) model. The function repeatedly applies logistic regression on the successive splits. The model is also known as the cumulative link model.

The algorithm of syn.polr uses the function polr from the **MASS** package.

In order to avoid bias due to perfect prediction, the data are augmented by the method of White, Daniel and Royston (2010).

In case the call to polr fails, usually because the data are very sparse, multinom function is used instead.

Value

A list with two components:

res	a vector of length k with synthetic values of y.
fit	a summary of the model fitted to the observed data and used to produce synthetic values.

References

White, I.R., Daniel, R. and Royston, P. (2010). Avoiding bias due to perfect prediction in multiple imputation of incomplete categorical variables. *Computational Statistics and Data Analysis*, **54**, 2267–2275.

See Also

syn,syn.polyreg multinom, polr

---

syn.polyreg *Synthesis by unordered polytomous regression*

Description

Generates a synthetic categorical variable using unordered polytomous regression (without or with bootstrap).

Usage

```
syn.polyreg(y, x, xp, proper = FALSE, maxit = 1000, trace = FALSE,
            MaxNWts = 10000, ...)
```

Arguments

y	an original data vector of length n.
x	a matrix (n x p) of original covariates.
xp	a matrix (k x p) of synthesised covariates.
proper	for proper synthesis (proper = TRUE) a multinomial model is fitted to a bootstrapped sample of the original data.
maxit	the maximum number of iterations for nnet.
trace	switch for tracing optimization for nnet.
MaxNWts	the maximum allowable number of weights for nnet.
...	additional parameters passed to nnet.

Details

Generates synthetic categorical variables by the polytomous regression model. The method consists of the following steps:

1. Fit categorical response as a multinomial model.
2. Compute predicted categories.
3. Add appropriate noise to predictions.

The algorithm of syn.polyreg uses the function multinom from the **nnet** package. Any numerical variables are scaled to cover the range (0,1) before fitting. Warnings are printed if the algorithm fails to converge in maxit iterations and also if the synthesised data has only one category. The latter may occur if the variable being synthesised is sparse so that the algorithm fails to iterate.

In order to avoid bias due to perfect prediction, the data are augmented by the method of White, Daniel and Royston (2010).

NOTE that when the function is called by setting elements of method in syn() to "polyreg", the parameters maxit, trace and MaxNWts can be supplied to syn() as e.g. polyreg.maxit.

Value

A list with two components:

res	a vector of length k with synthetic values of y.
fit	a summary of the model fitted to the observed data and used to produce synthetic values.

References

White, I.R., Daniel, R. and Royston, P. (2010). Avoiding bias due to perfect prediction in multiple imputation of incomplete categorical variables. *Computational Statistics and Data Analysis*, **54**, 2267–2275.

See Also

syn, syn.polr, multinom, polr

---

syn.ranger *Synthesis with a fast implementation of random forests*

Description

Generates univariate synthetic data using a fast implementation of random forests. It uses ranger function from the **ranger** package.

Usage

```
syn.ranger(y, x, xp, smoothing = "", proper = FALSE, ...)
```

Arguments

y	an original data vector of length n.
x	a matrix (n x p) of original covariates.
xp	a matrix (k x p) of synthesised covariates.
smoothing	smoothing method for numeric variable. See syn.smooth.
proper	for proper synthesis (proper = TRUE) a model is fitted to a bootstrapped sample of the original data.
...	additional parameters passed to ranger.

Details

...

Value

A list with two components:

`res`	a vector of length `k` with synthetic values of `y`.
`fit`	the model fitted to the observed data that was used to produce synthetic values.

References

...

See Also

`syn`, `syn.rf`, `syn.bag`, `syn.cart`, `ranger`, `syn.smooth`

`syn.rf` *Synthesis with random forest*

Description

Generates univariate synthetic data using Breiman's random forest algorithm classification and regression. It uses `randomForest` function from the **randomForest** package.

Usage

```
syn.rf(y, x, xp, smoothing = "", proper = FALSE, ntree = 10, ...)
```

Arguments

`y`	an original data vector of length `n`.
`x`	a matrix (`n` x `p`) of original covariates.
`xp`	a matrix (`k` x `p`) of synthesised covariates.
`smoothing`	smoothing method for numeric variable. See `syn.smooth`.
`proper`	for proper synthesis (`proper = TRUE`) a model is fitted to a bootstrapped sample of the original data.
`ntree`	number of trees to grow.
`...`	additional parameters passed to `randomForest`.

Details

...

Value

A list with two components:

res	a vector of length k with synthetic values of y.
fit	the fitted model which is an object of class randomForest.

References

...

See Also

syn, syn.rf, syn.bag, syn.cart, randomForest, syn.smooth

---

syn.sample *Synthesis by simple random sampling*

Description

Generates a random sample from the observed data.

Usage

```
syn.sample(y, xp, smoothing = "", cont.na = NA, proper = FALSE, ...)
```

Arguments

y	an original data vector of length n.
xp	a target length k of a synthetic data vector.
smoothing	smoothing method for numeric variable. See documentation for syn.smooth.
cont.na	a vector of codes for missing values for continuous variables that should be excluded from smoothing.
proper	if proper = TRUE values are sampled from a bootstrapped sample of the original data.
...	additional parameters passed to sample.

Details

A simple random sample with replacement is taken from the observed values in y and used as synthetic values. A Guassian kernel smoothing can be applied to continuous variables by setting smoothing parameter to "density". It is recommended as a tool to decrease the disclosure risk.

Value

A list with two components:

res	a vector of length k with synthetic values of y.
fit	a name of the method used for synthesis ("sample").

See Also

syn, syn.smooth

---

syn.satcat *Synthesis from a saturated model based on all combinations of the predictor variables.*

Description

Synthesises one variable (y) from all possible combinations of its predictors (x). A bootstrap sample is created from the original values of y within each unique combinations of of xp (the synthesisied values of the grouping variable). Note that only combinations of predictor variable levels that appear in the original data can be in the synthetic data. The related method (syn.catall) overcomes this by adding a small prior probability to all zero cells in the cross tabulation from the original data that are not structural zeros. But it has the limitation of requiring a complete cross tabulation of all the variables.

Usage

```
syn.satcat(y, x, xp, proper = FALSE, ...)
```

Arguments

y	an original data vector of length n for the satcat variable.
x	a matrix (n x p) with the original predictor variables for y.
xp	a matrix (k x p) with synthetic values of x.
proper	if proper = TRUE x and y are replaced with a bootstrap sample before synthesis, thus effectively sampling from the posterior distribution of the model, given the data.
...	additional parameters.

Details

It is intended that the variables in x are categorical (factor) variables. If y is also a categorical variable syn.satcat will give the same results as fitting a saturated polychotomous regression model but will usually be much faster. syn.satcat will fail with an error message if previous syntheses have generated a combination of variables in xp that was not present in x. Use of the syn.catall method for grouped variables can overcome this.

Value

A list with two components:

res	a data frame of dimension k x p containing the synthesised data.
fit	the cross-tabulation of the original predictor variables.

Examples

```
ods <- SD2011[, c("region", "sex", "agegr", "placesize")]

s1 <- syn(ods, method =  "satcat", seed = 7856)
s2 <- syn(ods, method = c("sample", "cart", "satcat", "cart"), seed = 7856)

## Not run:
### mostly fails because previous synthesis has produced
### combinations not found in the original data
s3 <- syn(ods, method = c("sample", "cart", "cart", "satcat"), seed = 7856)
## End(Not run)
```

---

syn.smooth *syn.smooth*

Description

Implements three different smoothing methods for numeric data.

Usage

```
syn.smooth(ysyn, yobs = NULL, smoothing = "spline", window = 5, ...)
```

Arguments

ysyn	non-missing synthetic data to be smoothed.
yobs	original data used by all methods to determine number of decimal places and by method "density" to identify top-coding.
smoothing	a character vector that can take values "spline", "density", or "rmean".
window	width of window for running mean.
...	additional parameters.

Details

Smooths numeric variables by three methods. Default is "spline" that uses a smoothing spline, others are "density" that uses a Gaussian kernel density estimator with bandwidth selected using the Sheather-Jones 'solve-the-equation' method (see bw.SJ) and "rmean" that smooths with a running mean of width "window" (see runningmean).

Value

A vector of smoothed values of ysyn.

See Also

syn, syn.sample, syn.normrank, syn.pmm, syn.ctree, syn.cart, syn.bag, syn.rf, syn.ranger, syn.nested

syn.survctree *Synthesis of survival time by classification and regression trees (CART)*

Description

Generates synthetic event indicator and time to event data using classification and regression trees (without or with bootstrap).

Usage

```
syn.survctree(y, yevent, x, xp, proper = FALSE, minbucket = 5, ...)
```

Arguments

y	a vector of length n with original time data.
yevent	a vector of length n with original event indicator data.
x	a matrix (n x p) of original covariates.
xp	a matrix (k x p) of synthesised covariates.
proper	for proper synthesis (proper = TRUE) a CART model is fitted to a bootstrapped sample of the original data.
minbucket	the minimum number of observations in any terminal node. See ctree_control for details.
...	additional parameters passed to ctree.

Details

The procedure for synthesis by a CART model is as follows:

1. Fit a tree-structured survival model by binary recursive partitioning (the terminal nodes include Kaplan-Meier estimates of the survival time).
2. For each xp find the terminal node.
3. Randomly draw a donor from the members of the node and take the observed value of yevent and y from that draw as the synthetic values.

The function is used in syn() to generate survival times by setting elements of method in syn() to "survctree". Additional parameters related to ctree function, e.g. minbucket can be supplied to syn() as survctree.minbucket.

Where the survival variable is censored this information must be supplied to syn() as a named list (event) that gives the name of the variable for each event indicator. Event variables can be a numeric variable with values 1/0 (1 = event), TRUE/FALSE (TRUE = event) or a factor with 2 levels (level 2 = event). The event variable(s) will be synthesised along with the survival time(s).

Value

A list with the following components:

`syn.time`	a vector of length k with synthetic time values.
`syn.event`	a vector of length k with synthetic event indicator values.
`fit`	the fitted model which is an item of class `ctree.object`.

See Also

syn, syn.ctree

Examples

```
### This example uses the data set 'mgus2' from the survival package.
### It has a follow-up time variable 'futime' and an event indicator 'death'.
library(survival)

### first exclude the 'id' variable and run a dummy synthesis to get
### a method vector
ods <- mgus2[-1]
s0 <- syn(ods)

### create new method vector including 'survctree' for 'futime' and create
### an event list for it; the names of the list element must correspond to
### the name of the follow-up variable for which the event indicator
### need to be specified.
meth <- s0$method
meth[names(meth) == "futime"] <- "survctree"
evlist <- list(futime = "death")

s1 <- syn(ods, method = meth, event = evlist)

### evaluate outputs
## compare selected variables
compare(s1, ods, vars = c("futime", "death", "sex", "creat"))

## compare original and synthetic follow up time by an event indicator
multi.compare(s1, ods, var = "futime", by = "death")

## compare survival curves for original and synthetic data
par(mfrow = c(2,1))
plot(survfit(Surv(futime, death) ~ sex, data = ods),
     col = 1:2, xlim = c(0,450), main = "Original data")
legend("topright", levels(ods$sex), col = 1:2, lwd = 1, bty = "n")
plot(survfit(Surv(futime, death) ~ sex, data = s1$syn),
     col = 1:2, xlim = c(0,450), main = "Synthetic data")
```

synorig.compare *check synthetic and original if not produced by synthpop.*

Description

Check, and attempt to adjust, synthetic datasets NOT created by `syn()` if not compatible with the original. The output is a list with 4 components, the first two giving adjusted versions of the input synthetic data and original and the third `needsfix` an indicator of whether the functions listed below are likely to run correctly on the adjusted data, the fourth `unchanged` indicates whether any changes have been made to the original or the synthetic data.

Variables that are in both the synthetic and the original data are checked to 1) convert any character variables to R factors 2) check that data types match 3) check differences in whether variables have missing values 4) check if the levels of factors agree and if not use the combination of both sets.

`needsfix` becomes TRUE if 1) some variables are in the synthetic but not in the original 2) variables have different classes (after characters converted to factor) 3) there are missing values in the synthetic data but not in the original 4) some levels of a factor are in the synthetic data but not in the original. Some warning messages are printed if level differences are not just due to missing values.

The function can be run to compare original and synthetic or it can be called by setting compare.synorig = TRUE when data are not supplied as an object of class synds created by synthpop in the following functions : `utility.tab()` `utility.tables()` `compare()` `disclosure()` `disclosure.summary()`. In this case the function attempts to correct differences and continue or, if this is impossible, will prompt the user as to which variables need changing.

Usage

```
synorig.compare(syn,orig, print.flag = TRUE)
```

Arguments

`syn`	A data set containing the synthesised data, or a list of such data sets. When `syn` is a list only the first member of the list is used and the syn component of the result is the adjusted version of this first member.
`orig`	The original data set.
`print.flag`	If TRUE prints non-essential summary messages.

Details

Error messages explain briefly what adjustments have been made to the data sets, what could not be fixed and what might need to be checked. Both `orig` and `syn` are made into simple data frames for comparison (e.g. if tibbles or matrices)

Value

A list with 3 components

`syn`	adjusted version of `syn` or of the first element of the list `syn` if it is a list.

orig	adjusted version of orig.
needsfix	TRUE/FALSE as to whether the outputs need to be fixed before utility and disclosure functions could be used on them.
unchanged	TRUE/FALSE indicating if the outputs of the function are unchanged from the outputs.

References

to add

See Also

utility.gen utility.tab utility.tables compare.synds disclosure.synds

Examples

```
library(synthpop)
orig <- SD2011[1:2000,]
pretendsyn <- SD2011[2001:5000, 1:5]
orig[,1] <- as.character(orig[,1])
codebook.syn(orig[,1:5])
newdata <- synorig.compare(pretendsyn, orig)
codebook.syn(newdata$orig[,1:5])
```

---

utility.gen *Distributional comparison of synthesised and observed data*

Description

Distributional comparison of synthesised data set with the original (observed) data set using propensity scores.

This function can be also used with synthetic data NOT created by syn(), but then additional parameters not.synthesised and cont.na might need to be provided.

Usage

```
## S3 method for class 'synds'
utility.gen(object, data,
            method = "cart", maxorder = 1, k.syn = FALSE, tree.method = "rpart",
        max.params = 400, print.stats = c("pMSE", "S_pMSE"), resamp.method = NULL,
           nperms = 50, cp = 1e-3, minbucket = 5, mincriterion = 0, vars = NULL,
            aggregate = FALSE, maxit = 200, ngroups = NULL, print.flag = TRUE,
            print.every = 10, digits = 6, print.zscores = FALSE, zthresh = 1.6,
            print.ind.results = FALSE, print.variable.importance = FALSE, ...)

## S3 method for class 'data.frame'
utility.gen(object, data, not.synthesised = NULL, cont.na = NULL,
```

```
            method = "cart", maxorder = 1, k.syn = FALSE, tree.method = "rpart",
        max.params = 400, print.stats = c("pMSE", "S_pMSE"), resamp.method = NULL,
           nperms = 50, cp = 1e-3, minbucket = 5, mincriterion = 0, vars = NULL,
            aggregate = FALSE, maxit = 200, ngroups = NULL, print.flag = TRUE,
            print.every = 10, digits = 6, print.zscores = FALSE, zthresh = 1.6,
            print.ind.results = FALSE, print.variable.importance = FALSE, ...)

## S3 method for class 'list'
utility.gen(object, data, not.synthesised = NULL, cont.na = NULL,
            method = "cart", maxorder = 1, k.syn = FALSE, tree.method = "rpart",
        max.params = 400, print.stats = c("pMSE", "S_pMSE"), resamp.method = NULL,
           nperms = 50, cp = 1e-3, minbucket = 5, mincriterion = 0, vars = NULL,
            aggregate = FALSE, maxit = 200, ngroups = NULL, print.flag = TRUE,
            print.every = 10, digits = 6, print.zscores = FALSE, zthresh = 1.6,
            print.ind.results = FALSE, print.variable.importance = FALSE, ...)


## S3 method for class 'utility.gen'
print(x, digits = NULL, zthresh = NULL,
               print.zscores = NULL, print.stats = NULL,
               print.ind.results = NULL, print.variable.importance = NULL, ...)
```

Arguments

`object` it can be an object of class `synds`, which stands for 'synthesised data set'. It is typically created by function `syn()` and it includes `object$m` synthesised data set(s) as `object$syn`. This a single data set when `object$m = 1` or a list of length `object$m` when `object$m > 1`. Alternatively, when data are synthesised not using `syn()`, it can be a data frame with a synthetic data set or a list of data frames with synthetic data sets, all created from the same original data with the same variables and the same method.

`data` the original (observed) data set.

`not.synthesised` a vector of variable names for any variables that has been left unchanged in the synthetic data. Not required if oject is of class `synds`

`cont.na` a named list of codes for missing values for continuous variables if different from the `R` missing data code `NA`. The names of the list elements must correspond to the variables names for which the missing data codes need to be specified. Not required if oject is of class `synds`

`method` a single string specifying the method for modeling the propensity scores. Method can be selected from `"logit"` and `"cart"`.

`maxorder` maximum order of interactions to be considered in `"logit"` method. For model without interactions `0` should be provided.

`k.syn` a logical indicator as to whether the sample size itself has been synthesised.

`tree.method` implementation of `"cart"` method that is used when `method = "cart"`. It can be `"rpart"` or `"ctree"`.

max.params	the maximum number of parameters for a "logit" model which alerts the user to possible fitting failure.
print.stats	statistics to be printed must be a selection from "pMSE", "SPECKS", "PO50", "S_pMSE", "S_SPECKS", "S_PO50". If print.stats = "all", all of the measures mentioned above will be printed.
resamp.method	method used for resampling estimates of standardized measures can be "perm", "pairs" or "none". Defaults to "pairs" if print.stats includes "S_SPECKS" or "S_PO50" or synthesis is incomplete else defaults to "perm" if method is "cart" or to NULL, no resampling needed, if method is "logit". "none" can be used to get results without standardized measures e.g. in simulations.
nperms	number of permutations for the permutation test to obtain the null distribution of the utility measure when resamp.method = "perm".
cp	complexity parameter for classification with tree.method "rpart". Small values grow bigger trees.
minbucket	minimum number of observations allowed in a leaf for classification when method = "cart".
mincriterion	criterion between 0 and 1 to use to control tree.method = "ctree" when the tree will not be allowed to split further. A value of 0.95 would be equivalent to a 5% significance test. Here we set it to 0 to effectively disable this test and grow large trees.
vars	variables to be included in the utility comparison. It can be a character vector of names of variables or an integer vector of their column indices. If none are specified all the variables in the synthesised data will be included.
aggregate	logical flag as to whether the data should be aggregated by collapsing identical rows before computation. This can lead to much faster computation when all the variables are categorical. Only works for method = "logit".
maxit	maximum iterations to use when method = "logit". If the model does not converge in this number a warning will suggest increasing it.
ngroups	target number of groups for categorisation of each numeric variable: final number may differ if there are many repeated values. If NULL (default) variables are not categorised into groups.
print.flag	TRUE/FALSE to indicate if any messages should be printed during calculations. Change to FALSE for simulations.
print.every	controls the printing of progress of resampling when resamp.method is not NULL. When print.every = 0 no progress is reported, otherwise the resample number is printed every print.every.
...	additional parameters passed to glm, rpart, or ctree.
x	an object of class utility.gen.
digits	number of digits to print in the default output values.
zthresh	threshold value to use to suppress the printing of z-scores under +/- this value for method = "logit". If set to NA all z-scores are printed.
print.zscores	logical value as to whether z-scores for coefficients of the logit model should be printed.

`print.ind.results`
: logical value as to whether utility score results from individual syntheses should be printed.

`print.variable.importance`
: logical value as to whether the variable importance measure should be printed when `tree.method = "rpart"`.

Details

This function follows the method for evaluating the utility of masked data as given in Snoke et al. (2018) and originally proposed by Woo et al. (2009). The original and synthetic data are combined into one dataset and propensity scores, as detailed in Rosenbaum and Rubin (1983), are calculated to estimate the probability of membership in the synthetic data set. The utility measure is based on the mean squared difference between these probabilities and the probability expected if the data did not distinguish the synthetic data from the original.

If `k.syn = FALSE` the expected probability is just the proportion of synthetic data in the combined data set, `0.5` when the original and synthetic data have the same number of records. Setting `k.syn = TRUE` indicates that the numbers of observations in the synthetic data was synthesised and not fixed by the synthesiser. In this case the expected probability will be `0.5` in all cases and the model to discriminate between observed and synthetic will include an intercept term. This will usually only apply when the standalone version of this function `utility.gen.sa()` is used.

Propensity scores can be modeled by logistic regression `method = "logit"` or by two different implementations of classification and regression trees as method `"cart"`. For logistic regression the predictors are all variables in the data and their interactions up to order `maxorder`. The default of `1` gives all main effects and first order interactions. For logistic regression the null distribution of the propensity score is derived and is used to calculate ratios and standardised values.

For `method = "cart"` the expectation and variance of the null distribution is calculated from a permutation test. Our recent work indicates that this method can sometimes give misleading results.

If missing values exist, indicator variables are added and included in the model as recommended by Rosenbaum and Rubin (1984). For categorical variables, `NA` is treated as a new category.

Value

An object of class `utility.gen` which is a list including the utility measures their expected null values for each synthetic set with the following components:

`call`	the call that produced the result.
`m`	number of synthetic data sets in object.
`method`	method used to fit propensity score.
`tree.method`	cart function used to fit propensity score when `method = "cart"`.
`resamp.method`	type of resampling used to get `pMSEExp` and `pval`.
`maxorder`	see above.
`vars`	see above.
`nfix`	see above.
`aggregate`	see above.

maxit	see above.
ngroups	see above.
df	degrees of freedom for the chi-squared test for logit models derived from the number of non-aliased coefficients in the logistic model, minus 1 for k.syn = FALSE.
mincriterion	see above.
nperms	see above.
incomplete	TRUE/FALSE indicator if any of the variables being compared are not synthesised.
pMSE	propensity score mean square error from the utility model or a vector of these values if object$m > 1.
S_pMSE	ratio(s) of pMSE to its Null expectation.
PO50	percentage over 50% of each synthetic data set where the model used correctly predicts whether real or synthetic.
S_PO50	ratio(s) of PO50 to its Null expectation.
SPECKS	Kolmogorov-Smirnov statistic to compare the propensity scores for the original and synthetic records.
S_SPECKS	ratio(s) of SPECKS to its Null expectation.
print.stats	see above.
fit	the fitted model for the propensity score or a list of fitted models of length m if m > 0.
nosplits	for resampling methods and cart models, a list of the number of times from the total each resampled cart model failed to select any splits to classify the indicator. Indicates that this method is not working correctly and results should not be used but a logit model selected instead.
digits	see above.
print.ind.results	see above.
print.zscores	see above.
zthresh	see above.
print.variable.importance	see above.

References

Woo, M-J., Reiter, J.P., Oganian, A. and Karr, A.F. (2009). Global measures of data utility for microdata masked for disclosure limitation. *Journal of Privacy and Confidentiality*, **1**(1), 111-124.

Rosenbaum, P.R. and Rubin, D.B. (1984). Reducing bias in observational studies using subclassification on the propensity score. *Journal of the American Statistical Association*, **79**(387), 516-524.

Snoke, J., Raab, G.M., Nowok, B., Dibben, C. and Slavkovic, A. (2018). General and specific utility measures for synthetic data. *Journal of the Royal Statistical Society: Series A*, **181**, Part 3, 663-688.

See Also

utility.tab

Examples

```
## Not run:
  ods <- SD2011[1:1000, c("age", "bmi", "depress", "alcabuse", "nofriend")]
  s1 <- syn(ods, m = 5, method = "parametric",
            cont.na = list(nofriend = -8))

  ### synthetic data provided as a 'synds' object
  u1 <- utility.gen(s1, ods)
  print(u1, print.zscores = TRUE, zthresh = 1, digits = 6)
  u2 <- utility.gen(s1, ods, ngroups = 3, print.flag = FALSE)
  print(u2, print.zscores = TRUE)
  u3 <- utility.gen(s1, ods, method = "cart", nperms = 20)
  print(u3, print.variable.importance = TRUE)

  ### synthetic data provided as 'list'
  utility.gen(s1$syn, ods, cont.na = list(nofriend = -8))

## End(Not run)
```

---

utility.tab *Tabular utility*

Description

Produces tables from observed and synthesised data and calculates utility measures to compare them with their expectation if the synthesising model is correct.

It can be also used with synthetic data NOT created by syn(), but then an additional parameter cont.na might need to be provided.

Usage

```
## S3 method for class 'synds'
utility.tab(object, data, vars = NULL, ngroups = 5,
            useNA = TRUE, max.table = 1e6,
            print.tables = length(vars) < 4,
            print.stats = c("pMSE", "S_pMSE", "df"),
            print.zdiff = FALSE, print.flag = TRUE,
            digits = 4, k.syn = FALSE,  ...)

## S3 method for class 'data.frame'
utility.tab(object, data, vars = NULL, cont.na = NULL,
            ngroups = 5, useNA = TRUE, max.table = 1e6,
            print.tables = length(vars) < 4,
            print.stats = c("pMSE", "S_pMSE", "df"),
```

```
            print.zdiff = FALSE, print.flag = TRUE,
            digits = 4, k.syn = FALSE,
            compare.synorig  = TRUE, ...)

## S3 method for class 'list'
utility.tab(object, data, vars = NULL, cont.na = NULL,
            ngroups = 5, useNA = TRUE, max.table = 1e6,
            print.tables = length(vars) < 4,
            print.stats = c("pMSE", "S_pMSE", "df"),
            print.zdiff = FALSE, print.flag = TRUE,
            digits = 4, k.syn = FALSE,
            compare.synorig = TRUE, ...)


## S3 method for class 'utility.tab'
print(x, print.tables = NULL,
      print.zdiff = NULL, print.stats = NULL,
      digits = NULL, ...)
```

Arguments

`object`	an object of class `synds`, which stands for 'synthesised data set'. It is typically created by function `syn()` or `syn.strata()` and it includes `object$m` number of synthesised data set(s), as well as `object$syn` the synthesised data set, if `m = 1`, or a list of `m` such data sets. Alternatively, when data are synthesised not using `syn()`, it can be a data frame with a synthetic data set or a list of data frames with synthetic data sets, all created from the same original data with the same variables and the same method.
`data`	the original (observed) data set.
`vars`	a single string or a vector of strings with the names of variables to be used to form the table.
`cont.na`	a named list of codes for missing values for continuous variables if different from the `R` missing data code `NA`. The names of the list elements must correspond to the variables names for which the missing data codes need to be specified.
`max.table`	a maximum table size. You could try increasing the default value, but memory problems are likely.
`ngroups`	if numerical (non-factor) variables are included they will be classified into this number of groups to form tables. Classification is performed using `classIntervals()` function for `n = ngroups`. By default, `style = "quantile"` to get appropriate groups for skewed data. Problems for variables with a small number of unique values are handled by selecting only unique values of breaks. Arguments of `classIntervals()` may be, however, specified in the call to `utility.tab()`.
`useNA`	determines if NA values are to be included in tables.
`print.tables`	a logical value that determines if tables of observed and synthesised data are to be printed. By default tables are printed if they have up to three dimensions.
`print.stats`	a single string or a vector of strings that determines which utility measures to print. Must be a selection from: `"VW"`, `"FT"`,`"JSD"`, `"SPECKS"`, `"WMabsDD"`,

"U", "G", "pMSE", "PO50", "MabsDD", "dBhatt", "S_VW", "S_FT", "S_JSD", "S_WMabsDD", "S_G", "S_pMSE", "df", dfG. If print.stats = "all", all of these will be printed. For more information see the details section below.

print.zdiff — a logical value that determines if tables of Z scores for differences between observed and expected are to be printed.

print.flag — a logical value that determines if messages are to be printed during computation.

digits — an integer indicating the number of decimal places for printing statistics, tab.zdiff and mean results for m > 1.

k.syn — a logical indicator as to whether the sample size itself has been synthesised. The default value is FALSE, which will apply to synthetic data created by synthpop.

compare.synorig — a logical value to determine if the functions synorig.compare() should be used to check that data sets can be compared. Used when the synthetic data are supplied as a data.frame or a list when default set to TRUE.

... — additional parameters; can be passed to classIntervals() function.

x — an object of class utility.tab.

Details

Forms tables of observed and synthesised values for the variables specified in vars. Several utility measures are calculated from the cells of the tables, as described below. Details of all of these measures can be found in Raab et al. (2021). If the synthesising model is correct the measures VW, FT, G and JSD should have chi-square distributions with df degrees of freedom for large samples. Standardised versions of each measure are available (e.g. S_VW for VW, where S_VW = VW/df) that will have an expected value of 1 if the synthesising model is correct. Four other measures are calculated by considering the table as a prediction model. The propensity score mean-squared error pMSE, and from a comparison of propensity scores for the synthetic and original data the Kolmogorov-Smirnov statistic SPECKS and the Wilcoxon rank-sum statistic U and also the percentage of the observations correctly predicted in the combined tables over 50%(PO50) where the majority of observations in each grouping are in agreement with category (real or synthetic) of the observation. The first of these pMSE is identical except for a constant to VW. No expected values are computed for the last three of these measures, but they can be obtained by replication from utility.gen(). Three further measures are calculated from the tables. The mean absolute difference in distributions: firstly MabsDD, the avarage absolute difference in the proportions of original and synthetic data from all the cells in the table. Secondly a weighted version of this measure WMabsDD where the weights are proportional to the inverse of the variance of the absolute differences so that this measure can be standardised by its expected value, df. Finally the Bhattacharyya distances BhattD derived from the overlap of the histograms of the original and synthetic data sets.

Value

An object of class utility.tab which is a list with the following components:

m — number of synthetic data sets in object, i.e. object$m.

VW — a vector with object$m values for the Voas Williamson utility measure.; linearly related to pMSE.

FT — a vector with object$m values for the Freeman-Tukey utility measure.

JSD	a vector with object$m values for the Jensen-Shannaon divergence for comparing the tables.
SPECKS	a vector with object$m values for the Kolmogorov-Smirnov statistic for comparing the propensity scores for the original and synthetic data.
WMabsDD	a vector with object$m values of the weighted mean absolute difference in distributions for original and synthetic data.
U	a vector with object$m values of the Wilcoxon statistic comparing the propensity scores for the original and synthetic data.
G	a vector with object$m values for the adjusted likelihood ratio utility measure.
pMSE	a vector with object$m values of the propensity score mean-squared error; linearly related to VW.
PO50	a vector with object$m values of the percentage over 50% of observations correctly predicted from the propensity scores linearly related to SPECKS and MabsDD.
MabsDD	a vector with object$m values of the mean absolute difference in distributions for original and synthetic data linearly related to SPECKS and PO50.
dBhatt	a vector with object$m values of the Bhattacharyya distances between the synthetic and original data, linearly related to the square root of FT.
S_VW	VW/df.
S_FT	FT/df.
S_JSD	JSD/df.
S_WMabsDD	WMabsDD/df.
S_G	G/df.
S_pMSE	standardised measure from pMSE, identical to S_VW.
df	a vector of degrees of freedom for the chi-square tests which equal to the number of cells in the tables with any observed or synthesised counts minus one when k.syn == FALSE or equal to the the number of cells when k.syn == TRUE.
dfG	degrees of freedom used in standardising G.
nempty	a vector of length object$m with number of cells not contributing to the statistics.
tab.obs	a table from the observed data.
tab.syn	a table or a list of m tables from the synthetic data.
tab.zdiff	a table or a list of m tables of Z statistics for differences between observed and synthesised cells of the tables. Large absolute values indicate a large contribution to lack-of-fit.
digits	an integer indicating the number of decimal places for printing statistics, tab.zdiff and mean results for m > 1.
print.tables	a logical value that determines if tables of observed and synthesised are to be printed.
print.stats	a single string or a vector of strings with utility measures to be printed out.
print.zdiff	a logical value that determines if tables of Z scores for differences between observed and expected are to be printed.
n	number of observation in the original dataset.
k.syn	a logical indicator as to whether the sample size itself has been synthesised.

References

Nowok, B., Raab, G.M and Dibben, C. (2016). synthpop: Bespoke creation of synthetic data in R. *Journal of Statistical Software*, **74**(11), 1-26. doi:10.18637/jss.v074.i11.

Raab, G.M., Nowok, B. and Dibben, C. (2021). Assessing, visualizing and improving the utility of synthetic data. Available from https://arxiv.org/abs/2109.12717.

Read, T.R.C. and Cressie, N.A.C. (1988) *Goodness–of–Fit Statistics for Discrete Multivariate Data*, Springer–Verlag, New York.

Voas, D. and Williamson, P. (2001) Evaluating goodness-of-fit measures for synthetic microdata. *Geographical and Environmental Modelling*, **5**(2), 177-200.

See Also

utility.gen

Examples

```
ods <- SD2011[1:1000, c("sex", "age", "marital", "nofriend")]

s1 <- syn(ods, m = 10, cont.na = list(nofriend = -8))
utility.tab(s1, ods, vars = c("marital", "sex"), print.stats = "all")

s2 <- syn(ods, m = 1, cont.na = list(nofriend = -8))
u2 <- utility.tab(s2, ods, vars = c("marital", "age", "sex"), ngroups = 3)
print(u2, print.tables = TRUE, print.zdiff = TRUE)

### synthetic data provided as 'data.frame'
utility.tab(s2$syn, ods, vars = c("marital", "nofriend"), ngroups = 3,
            print.tables = TRUE, cont.na = list(nofriend = -8), digits = 4)
```

---

utility.tables *Tables and plots of utility measures*

Description

Calculates and plots tables of utility measures. The calculations of utility measures are done by the function utility.tab. Options are all one-way tables, all two-way tables or three-way tables for a specified third variable along with pairs of all other variables.

This function can be also used with synthetic data NOT created by syn(), but then an additional parameters not.synthesised and cont.na might need to be provided.

Usage

```
## S3 method for class 'synds'
utility.tables(object, data,
               tables = "twoway", maxtables = 5e4,
               vars = NULL, third.var = NULL,
```

```
                useNA = TRUE, ngroups = 5,
                tab.stats = c("pMSE", "S_pMSE", "df"),
                plot.stat = "S_pMSE", plot = TRUE, max.table = 1e07,
                print.tabs = FALSE, digits.tabs = 4,
                max.scale = NULL, min.scale = 0, plot.title = NULL,
                nworst = 5, ntabstoprint = 0, k.syn = FALSE,
                low = "grey92", high = "#E41A1C",
                n.breaks = NULL, breaks = NULL, print.flag = TRUE, ...)

## S3 method for class 'data.frame'
utility.tables(object, data,
                cont.na = NULL, not.synthesised = NULL,
                tables = "twoway", maxtables = 5e4,
                vars = NULL, third.var = NULL,
                useNA = TRUE, ngroups = 5,
                tab.stats = c("pMSE", "S_pMSE", "df"),
                plot.stat = "S_pMSE", plot = TRUE, max.table = 1e07,
                print.tabs = FALSE, digits.tabs = 4,
                max.scale = NULL, min.scale = 0, plot.title = NULL,
                nworst = 5, ntabstoprint = 0, k.syn = FALSE,
                low = "grey92", high = "#E41A1C",
                n.breaks = NULL, breaks = NULL,
                compare.synorig = TRUE, print.flag = TRUE,...)

## S3 method for class 'list'
utility.tables(object, data,
                cont.na = NULL, not.synthesised = NULL,
                tables = "twoway", maxtables = 5e4,
                vars = NULL, third.var = NULL,
                useNA = TRUE, ngroups = 5,
                tab.stats = c("pMSE", "S_pMSE", "df"),
                plot.stat = "S_pMSE", plot = TRUE,  max.table = 1e07,
                print.tabs = FALSE, digits.tabs = 4,
                max.scale = NULL, min.scale = 0, plot.title = NULL,
                nworst = 5, ntabstoprint = 0, k.syn = FALSE,
                low = "grey92", high = "#E41A1C",
                n.breaks = NULL, breaks = NULL,
                compare.synorig = TRUE, print.flag = TRUE,...)

## S3 method for class 'utility.tables'
print(x, print.tabs = NULL, digits.tabs = NULL,
      plot = NULL, plot.title = NULL, max.scale = NULL, min.scale = NULL,
      nworst = NULL, ntabstoprint = NULL, ...)
```

Arguments

`object` an object of class synds, which stands for 'synthesised data set'. It is typically created by function `syn()` and it includes `object$m` synthesised data set(s) as `object$syn`. This a single data set when `object$m = 1` or a list of length

object$m when object$m > 1. Alternatively, when data are synthesised not using syn(), it can be a data frame with a synthetic data set or a list of data frames with synthetic data sets, all created from the same original data with the same variables and the same method.

data — the original (observed) data set.

cont.na — a named list of codes for missing values for continuous variables if different from the R missing data code NA. The names of the list elements must correspond to the variables names for which the missing data codes need to be specified.

not.synthesised — a vector of variable names for any variables that has been left unchanged in the synthetic data.

tables — defines the type of tables to produce. Options are "oneway", "twoway" (default) or "threeway". If set to "oneway" or "twoway" all possible tables from vars are produced. For "threeway", third.var may be specified and all three-way tables between this variable and other pairs of variables are produced. If a third variable is not specified the function chooses the variable with the largest median utility measure for all three-way tables it contributes to.

maxtables — maximum number of tables that will be produced. If number of tables is larger, then utility is only measured for a sample of size maxtables. You cannot produce plots of twoway or three way tables from sampled tables

.

vars — a vector of strings with the names of variables to be used to form the table, or a vector of variable numbers in the original data. Defaults to all variables in both original and synthetic data.

third.var — when tables is "threeway" a variable to make the third variable with all other pairs

useNA — determines if NA values are to be included in tables. Only applies for method "tab".

ngroups — if numerical (non-factor) variables included with method = "tab" will be classified into this number of groups to form tables. Classification is performed using classIntervals() function for n = ngroups. By default, style = "quantile", to get appropriate groups for skewed data. Problems for variables with a small number of unique values are handled by selecting only unique values of breaks. Arguments of classIntervals() may be, however, specified in the call to utility.tables().

tab.stats — statistics to include in the table of results. Must be a selection from: "VW", "FT","JSD", "SPECKS", "WMabsDD", "U", "G", "pMSE", "PO50", "MabsDD", "dBhatt", "S_VW", "S_FT", "S_JSD", "S_WMabsDD", "S_G", "S_pMSE", "df", dfG. If tab.stats = "all", all of these will be included. See utility.tab for explanations of measures.

plot.stat — statistics to plot. Choice is "VW", "FT", "JSD", "SPECKS", "WMabsDD", "U", "G", "pMSE", "PO50", "MabsDD", "dBhatt", "S_VW", "S_FT", "S_JSD", "S_WMabsDD", "S_G", "S_pMSE". See utility.tab for explanations of measures.

plot — determines if plot will be produced when the result is printed.

max.table	Value of maximum number of cells allowed in a table by the function utility.tab
print.tabs	logical value that determines if table of results is to be printed.
digits.tabs	number of digits to print for table, except for p-values that are always printed to 4 places.
max.scale	a numeric value for the maximum value used in calculating the shading of the plots. If it is NULL then the maximum value will be replaced by the maximum value in the data.
min.scale	a numeric value for the minimum value used in calculating the shading of the plots. If it is NULL then the minimum value will be replaced by zero.
plot.title	title for the plot.
nworst	a number of variable combinations with worst utility scores to be printed.
ntabstoprint	a number of tables to print for observed and synthetic data with the worst utility.
k.syn	a logical indicator as to whether the sample size itself has been synthesised.
low	colour for low end of the gradient.
high	colour for high end of the gradient.
n.breaks	a number of break points to create if breaks are not given directly.
breaks	breaks for a two colour binned gradient.
compare.synorig	a logical value to determine if the functions synorig.compare() should be used to check that data sets can be compared. Used when the synthetic data are supplied as a data.frame or a list when default set to TRUE.
print.flag	Allows printing of message as metrics are calculated for each element of the table. Default is TRUE.
...	additional parameters
x	an object of class utility.tables.

Details

Calculates tables of observed and synthesised values for the variables specified in vars with the function utility.tab and produces tables and plots of one-way, two-way or three-way utility measures formed from vars. Several options for utility measures can be selected for printing or plotting. Details are in help file for utility.tab.

The tables and variables with the worst utility scores are identified. Visualisations of the matrices of utility scores are plotted. For threeway tables a third variable can be defined to select all tables involving that variable for plotting. If it is not specified the variable with tables giving the worst utility is selected as the third variable.

Value

An object of class utility.tab which is a list with the following components:

tabs	a table with all the selected measures for all combinations of variables defined by tables, third.var, and vars.
plot.stat	measure used in mat and toplot.

tables	see above.
third.var	see above.
utility.plot	plot of the selected utility measure.
var.scores	an average of utility scores for all combinations with other variables.
plot	see above.
print.tabs	see above.
digits.tabs	see above.
plot.title	see above.
max.scale	see above.
min.scale	see above.
ntabstoprint	see above.
nworst	see above.
worstn	variable combinations with nworst worst utility scores.
worsttabs	observed and synthetic cross-tabulations for worstn.

References

Read, T.R.C. and Cressie, N.A.C. (1988) *Goodness–of–Fit Statistics for Discrete Multivariate Data*, Springer–Verlag, New York.

Voas, D. and Williamson, P. (2001) Evaluating goodness-of-fit measures for synthetic microdata. *Geographical and Environmental Modelling*, **5**(2), 177-200.

See Also

utility.tab

Examples

```
ods <- SD2011[1:1000, c("sex", "age", "edu", "marital", "region", "income")]
s1 <- syn(ods)

### synthetic data provided as a 'synds' object
(t1 <- utility.tables(s1, ods, tab.stats = "all", print.tabs = TRUE))
### synthetic data provided as a 'data.frame' object
(t1 <- utility.tables(s1$syn, ods, tab.stats = "all", print.tabs = TRUE))

t2 <- utility.tables(s1, ods, tables = "twoway")
print(t2, max.scale = 3)

(t3 <- utility.tables(s1, ods, tab.stats = "all", tables = "threeway",
                      third.var = "sex", print.tabs = TRUE))

(t4 <- utility.tables(s1, ods, tab.stats = "all", tables = "threeway",
                      third.var = "sex", useNA = FALSE, print.tabs = TRUE))

(t5 <- utility.tables(s1, ods,  tab.stats = "all",
                      print.tabs = TRUE))
```

`write.syn` *Exporting synthetic data sets to external files*

Description

Exports synthetic data set(s) from synthesised data set (`synds`) object to external files of selected format. Currently supported file formats include: SPSS, Stata, SAS, csv, tab, rda, RData and txt. For SPSS, Stata and SAS it uses functions from the `foreign` package with some adjustments where necessary. Information about the synthesis is written into a separate text file.

NOTE: Currently numeric codes and labels can be preserved correctly only for SPSS files imported into R using `read.obs` function.

Usage

```
write.syn(object, filename,
filetype = c("csv", "tab", "txt",
                    "SPSS", "Stata", "SAS", "rda", "RData"),
convert.factors = "numeric", data.labels = NULL, save.complete = TRUE,
extended.info = TRUE, ...)
```

Arguments

`object`	an object of class `synds`, which stands for 'synthesised data set'. It is typically created by function `syn` and it includes `object$m` synthesised data set(s).
`filename`	the name of the file (excluding extension) which the synthetic data are to be written into. For multiple synthetic data sets it will be used as a prefix folowed respectively by `_1`, `_...`, `_m`.
`filetype`	a desired format of the output files.
`convert.factors`	a single string indicating how to handle factors in Stata output files. The default value is set to `"numeric"` in order to preserve the numeric codes from the original data. See `write.dta` for other possible values.
`data.labels`	a list with variable labels and value labels.
`save.complete`	a logical value indicating whether a complete 'synthesised data set' (`synds`) object should be saved into a file (`synobject_filename.RData`).
`extended.info`	a logical value indicating whether extended information should be saved into an information file.
`...`	additional parameters passed to write functions.

Value

File(s) with synthesised data set(s) and a text file with information about synthesis are produced. Optionally a complete synthesised data set object is saved into `synobject_filename.RData` file.

See Also

read.obs

Index

∗ **datagen**
syn, 37
syn.bag, 44
syn.catall, 45
syn.ctree, syn.cart, 47
syn.ipf, 49
syn.lognorm, syn.sqrtnorm, syn.cubertnorm, 51
syn.logreg, 52
syn.nested, 53
syn.norm, 54
syn.normrank, 55
syn.passive, 56
syn.pmm, 57
syn.polr, 59
syn.polyreg, 60
syn.ranger, 61
syn.rf, 62
syn.sample, 63
syn.satcat, 64
syn.survctree, 66
∗ **datasets**
SD2011, 30
∗ **manip**
sdc, 32
∗ **multivariate**
glm.synds, lm.synds, 16
multinom.synds, 24
polr.synds, 26
∗ **package**
synthpop-package, 3
∗ **regression**
syn, 37
∗ **smoothing**
syn.smooth, 65
∗ **tree**
syn, 37

bw.SJ, *65*

codebook.syn, 4
compare, 5
compare.data.frame *(*compare.synds*)*, 9
compare.fit.synds, *3*, *5*, 6, *17*, *19*, *25*, *27*, *34*, *35*
compare.list *(*compare.synds*)*, 9
compare.synds, *3*, *5*, 9, *19*, *43*, *69*
ctree, *48*, *66*, *71*
ctree_control, *48*, *66*

disclosure, 11, *20*, *22–24*
disclosure.synds, *69*

family, *16*
format, *36*
formula, *16*, *24*, *27*

ggplot, *7*, *19*
glm, *16*, *17*, *53*, *71*
glm.synds, *3*, *6*, *25*, *27*, *33*, *34*
glm.synds *(*glm.synds, lm.synds*)*, 16
glm.synds, lm.synds, 16

Ipfp, *49*, *50*

lm, *16*, *17*
lm.synds, *3*, *6*, *33*, *34*
lm.synds *(*glm.synds, lm.synds*)*, 16

mergelevels.syn, 17
multi.compare, *11*, 19
multi.disclosure, *14*, *15*, 20
multinom, *24*, *25*, *59–61*
multinom.synds, *17*, 24, *27*, *33*, *34*

nnet, *59*, *60*
numtocat.syn, 25

optim, *59*

polr, *26*, *27*, *59–61*

polr.synds, *17*, *25*, 26, *33, 34*
print, *35*, *37*
print.compare.fit.synds (compare.fit.synds), 6
print.compare.synds (compare.synds), 9
print.disclosure (disclosure), 11
print.fit.synds, *24, 25*, *27*
print.fit.synds (glm.synds, lm.synds), 16
print.multi.disclosure (multi.disclosure), 20
print.repuniq.synds (replicated.uniques), 29
print.summary.fit.synds (summary.fit.synds), 33
print.summary.synds (summary.synds), 36
print.synds (syn), 37
print.utility.gen (utility.gen), 69
print.utility.tab (utility.tab), 74
print.utility.tables (utility.tables), 78

randomForest, *44, 45*, *62, 63*
ranger, *61, 62*
read.obs, 28, *83, 84*
replicated.uniques, 29, *32, 33*
rpart, *48*, *71*
rpart.control, *48*
runningmean, *65*

SD2011, 30
sdc, *18*, *29, 30*, 32
smooth.spline, *33*
summary, *16*, *24*, *27*, *35–37*, *42*
summary.fit.synds, *8*, *16, 17*, *24, 25*, *27*, 33
summary.synds, 36, *42, 43*
syn, *3*, *15, 16*, *24*, *27*, *36*, 37, *45*, *48*, *52, 53*, *55, 56*, *58*, *60–65*, *67*, *83*
syn.bag, *41*, 44, *62, 63*, *65*
syn.cart, *38*, *41*, *45*, *62, 63*, *65*
syn.cart (syn.ctree, syn.cart), 47
syn.catall, *41*, 45
syn.ctree, *65*, *67*
syn.ctree (syn.ctree, syn.cart), 47
syn.ctree, syn.cart, 47
syn.cubertnorm (syn.lognorm, syn.sqrtnorm, syn.cubertnorm), 51
syn.ipf, *41*, 49
syn.lognorm, *41*, *55*
syn.lognorm (syn.lognorm, syn.sqrtnorm, syn.cubertnorm), 51
syn.lognorm, syn.sqrtnorm, syn.cubertnorm, 51
syn.logreg, *41*, 52
syn.nested, *41*, 53, *65*
syn.norm, *41*, *52*, 54, *55*
syn.normrank, *41*, *52*, *55*, 55, *65*
syn.passive, *38*, *41*, 56
syn.pmm, *41*, 57, *65*
syn.polr, *41*, 59, *61*
syn.polyreg, *41*, *60*, 60
syn.ranger, *41*, 61, *65*
syn.rf, *41*, *45*, *62*, 62, *63*, *65*
syn.sample, *41*, 63, *65*
syn.satcat, *41*, 64
syn.smooth, *39*, *44, 45*, *47, 48*, *53*, *55*, *58*, *61–64*, 65
syn.sqrtnorm (syn.lognorm, syn.sqrtnorm, syn.cubertnorm), 51
syn.survctree, *41*, *48*, 66
synorig.compare, 68
synthpop (synthpop-package), 3
synthpop-package, 3

utility.gen, *69*, 69, *78*
utility.tab, *10*, *69*, *74*, 74, *78*, *80–82*
utility.tables, *69*, 78

write.dta, *83*
write.syn, *28*, 83