Package 'convey'

June 24, 2016

Title Income Concentration Analysis with Complex Survey Samples

Version 0.1.0

URL https://github.com/djalmapessoa/convey

BugReports https://github.com/djalmapessoa/convey/issues

Description Variance estimation on indicators of income concentration and poverty using linearized or replication-based survey designs. Wrapper around the survey package.

Depends R (>= 3.2.1)

Imports survey, methods

License GPL-3

LazyData true

Suggests testthat, knitr, rmarkdown, IC2, vardpoor, DBI, MonetDBLite, RODBC

VignetteBuilder knitr

RoxygenNote 5.0.1

NeedsCompilation no

Author Djalma Pessoa [aut, cre], Anthony Damico [aut], Guilherme Jacob [aut]

Maintainer Djalma Pessoa <pessoad@gmail.com>

Repository CRAN

Date/Publication 2016-06-24 09:53:38

R topics documented:

`contrastinf` *Generalized linearization of a smooth function of survey statistics*

Description

Generalized linearization of a smooth function of survey statistics

Usage

```
contrastinf(exprlist, infunlist)
```

Arguments

`exprlist`	a call
`infunlist`	a list of lists, each having two components: value - the estimate value and lin - the linearized variable

Details

The call must use function that `deriv` knows how to differentiate. It allows to compute the linearized variable of a complex indicator from the linearized variables of simpler component variables, avoiding the formal derivatives calculations.

Value

a list with two components: values - the estimate value and lin - the linearized variable

Author(s)

Djalma Pessoa and Anthony Damico

References

Guillaume Osier (2009). Variance estimation for complex indicators of poverty and inequality. *Journal of the European Survey Research Association*, Vol.3, No.3, pp. 167-195, ISSN 1864-3361, URL http://ojs.ub.uni-konstanz.de/srm/article/view/369.

See Also

svyqsr

Examples

```
library(survey)
library(vardpoor)
data(eusilc) ; names( eusilc ) <- tolower( names( eusilc ) )

# linearized design
des_eusilc <- svydesign( ids = ~rb030 , strata = ~db040 ,  weights = ~rb050 , data = eusilc )
des_eusilc <- convey_prep(des_eusilc)

w <- weights(des_eusilc)

# ratio linearization
T1 = list(value = sum(w*eusilc$eqincome) , lin = eusilc$eqincome )
T2 = list(value = sum(w), lin = rep (1, nrow(eusilc)) )
list_all <- list( T1 = T1, T2 = T2)
lin_R = contrastinf (quote(T1/T2), list_all)

# estimate of the variable eqincome mean
lin_R$value
# se estimate of the variable eqincome mean
SE(svytotal(lin_R$lin, des_eusilc))
# to check, use
svymean (~eqincome, des_eusilc)

# quintile share ratio (qsr) linearization
S20 <- svyisq(~ eqincome, design = des_eusilc, .20)
S20_val <- coef (S20); attributes (S20_val) <- NULL
S20_lin <- attr(S20 , "lin" )
S80 <- svyisq(~ eqincome, design = des_eusilc, .80)
S80_val <- coef (S80); attributes (S80_val) <- NULL
S80_lin <- attr(S80 , "lin" )
SU <- list (value = S80_val, lin = S80_lin )
SI <- list (value = S20_val, lin = S20_lin )
TOT <- list(value = sum( w * eusilc$eqincome) , lin = eusilc$eqincome )
list_all <- list (TOT = TOT, SI = SI, SU = SU )
lin_QSR <- contrastinf( quote((TOT-SU)/SI), list_all)

# estimate of the qsr
lin_QSR$value
# se estimate of the qsr:
SE(svytotal(lin_QSR$lin, des_eusilc))
# to check, use
```

```
svyqsr(~eqincome, des_eusilc )
# proportion of income below the quantile .20
list_all <- list (TOT = TOT, SI = SI )
lin_Lor <- contrastinf( quote(SI/TOT), list_all)
# estimate of the proportion of income below the quantile .20
lin_Lor$value
# se estimate
SE(svytotal(lin_Lor$lin,des_eusilc))
```

---

convey_prep *prepare svydesign and svyrep.design objects for the convey package*

Description

stores the full survey design (needed for convey functions that use a global poverty threshold) within the design. this function must be run immediately after the full design object creation with `svydesign` or `svrepdesign`

Usage

```
convey_prep(design)
```

Arguments

`design`	a survey design object of the library survey.

Details

functions in the convey package that use a global poverty threshold require the complete (pre-subsetted) design in order to calculate variances correctly. this function stores the full design object as a separate attribute so that functions from the `survey` package such as `subset` and `svyby` do not disrupt the calculation of error terms.

Value

the same survey object with a `full_design` attribute as the storage space for the unsubsetted survey design

Author(s)

Djalma Pessoa and Anthony Damico

Examples

```
library(survey)
library(vardpoor)
data(eusilc) ; names( eusilc ) <- tolower( names( eusilc ) )

# linearized design: convey_prep must be run as soon as the linearized design has been created
des_eusilc <- svydesign( ids = ~rb030 , strata = ~db040 ,  weights = ~rb050 , data = eusilc )
des_eusilc <- convey_prep( des_eusilc )
# now this linearized design object is ready for analysis!

# # # CORRECT usage example # # #
des_eusilc <- svydesign( ids = ~rb030 , strata = ~db040 ,  weights = ~rb050 , data = eusilc )
des_eusilc <- convey_prep( des_eusilc )
sub_eusilc <- subset( des_eusilc , age > 20 )
# since convey_prep() was run immediately after creating the design
# this will calculate the variance accurately
SE( svyarpt( ~ eqincome , sub_eusilc ) )
# # # end of CORRECT usage example # # #

# # # INCORRECT usage example # # #
des_eusilc <- svydesign( ids = ~rb030 , strata = ~db040 ,  weights = ~rb050 , data = eusilc )
sub_eusilc <- subset( des_eusilc , age > 20 )
sub_eusilc <- convey_prep( sub_eusilc )
# since convey_prep() was not run immediately after creating the design
# this will make the variance wrong
SE( svyarpt( ~ eqincome , sub_eusilc ) )
# # # end of INCORRECT usage example # # #
```

---

densfun *Estimate the derivative of the cdf function using kernel estimator*

Description

computes the derivative of a function in a point using kernel estimation

Usage

```
densfun(formula, design, x, h = NULL, FUN = "F", na.rm = FALSE, ...)
```

Arguments

formula	a formula specifying the income variable
design	a design object of class survey.design from the survey library.
x	the point where the derivative is calculated
h	value of the bandwidth based on the whole sample

FUN	if F estimates the derivative of the cdf function; if big_s estimates the derivative of total in the tails of the distribution
na.rm	Should cases with missing values be dropped?
...	future expansion

Value

the value of the derivative at x

Author(s)

Djalma Pessoa and Anthony Damico

Examples

```
library(vardpoor)
data(eusilc) ; names( eusilc ) <- tolower( names( eusilc ) )
library(survey)
des_eusilc <- svydesign(ids = ~rb030, strata =~db040,  weights = ~rb050, data = eusilc)
des_eusilc <- convey_prep( des_eusilc )
densfun (~eqincome, design=des_eusilc, 10000, FUN="F" )
# linearized design using a variable with missings
densfun ( ~ py010n , design = des_eusilc, 10000, FUN="F" )
densfun ( ~ py010n , design = des_eusilc , 10000,FUN="F", na.rm = TRUE )
```

---

h_fun *Computes the bandwidth needed to compute the derivative of the cdf function*

Description

Using the whole sample, computes the bandwith used to get the linearized variable

Usage

```
h_fun(incvar, w)
```

Arguments

incvar	income variable used in the estimation of the indicators
w	vector of design weights

Value

value of the bandwidth

Author(s)

Djalma Pessoa and Anthony Damico

icdf *Linearization of the cumulative distribution function (cdf) of a variable*

Description

Computes the linearized variable of the cdf function in a point.

Usage

```
icdf(formula, design, x, na.rm = FALSE, ...)
```

Arguments

formula	a formula specifying the income variable
design	a design object of class survey.design or class svyrep.design from the survey library.
x	the point where the cdf is calculated
na.rm	Should cases with missing values be dropped?
...	future expansion

Value

Object of class "cvystat", which are vectors with a "var" attribute giving the variance and a "statistic" attribute giving the name of the statistic.

Author(s)

Djalma Pessoa and Anthony Damico

References

Guillaume Osier (2009). Variance estimation for complex indicators of poverty and inequality. *Journal of the European Survey Research Association*, Vol.3, No.3, pp. 167-195, ISSN 1864-3361, URL http://ojs.ub.uni-konstanz.de/srm/article/view/369. Jean-Claude Deville (1999). Variance estimation for complex statistics and estimators: linearization and residual techniques. Survey Methodology, 25, 193-203, URL http://www5.statcan.gc.ca/bsolc/olc-cel/olc-cel?lang=eng&catno=12-001-X19990024882.

See Also

svyarpr

Examples

```
library(vardpoor)
data(eusilc) ; names( eusilc ) <- tolower( names( eusilc ) )
library(survey)
des_eusilc <- svydesign(ids = ~rb030, strata =~db040,  weights = ~rb050, data = eusilc)
des_eusilc <- convey_prep( des_eusilc )
icdf(~eqincome, design=des_eusilc, 10000 )
# linearized design using a variable with missings
icdf( ~ py010n , design = des_eusilc, 10000 )
icdf( ~ py010n , design = des_eusilc , 10000, na.rm = TRUE )
```

---

svyarpr *At-risk-of-poverty rate*

Description

Estimate the proportion of persons with income below the at-risk-of-poverty threshold.

Usage

```
svyarpr(formula, design, ...)

## S3 method for class 'survey.design'
svyarpr(formula, design, order = 0.5, percent = 0.6,
  na.rm = FALSE, ...)

## S3 method for class 'svyrep.design'
svyarpr(formula, design, order = 0.5, percent = 0.6,
  na.rm = FALSE, ...)

## S3 method for class 'DBIsvydesign'
svyarpr(formula, design, ...)
```

Arguments

formula	a formula specifying the income variable
design	a design object of class survey.design or class svyrep.design from the survey library.
...	future expansion
order	income quantile order, usually .5
percent	fraction of the quantile, usually .60
na.rm	Should cases with missing values be dropped?

Details

you must run the convey_prep function on your survey design object immediately after creating it with the svydesign or svrepdesign function.

Value

Object of class "cvystat", which are vectors with a "var" attribute giving the variance and a "statistic" attribute giving the name of the statistic.

Author(s)

Djalma Pessoa and Anthony Damico

References

Guillaume Osier (2009). Variance estimation for complex indicators of poverty and inequality. *Journal of the European Survey Research Association*, Vol.3, No.3, pp. 167-195, ISSN 1864-3361, URL http://ojs.ub.uni-konstanz.de/srm/article/view/369.

Jean-Claude Deville (1999). Variance estimation for complex statistics and estimators: linearization and residual techniques. Survey Methodology, 25, 193-203, URL http://www5.statcan.gc.ca/bsolc/olc-cel/olc-cel?lang=eng&catno=12-001-X19990024882.

See Also

svyarpt

Examples

```
## Not run:
library(survey)
library(vardpoor)
data(eusilc) ; names( eusilc ) <- tolower( names( eusilc ) )

# linearized design
des_eusilc <- svydesign( ids = ~rb030 , strata = ~db040 ,  weights = ~rb050 , data = eusilc )
des_eusilc <- convey_prep( des_eusilc )

svyarpr( ~eqincome , design = des_eusilc )

# replicate-weighted design
des_eusilc_rep <- as.svrepdesign( des_eusilc , type = "bootstrap" )
des_eusilc_rep <- convey_prep( des_eusilc_rep )

svyarpr( ~eqincome , design = des_eusilc_rep )

# linearized design using a variable with missings
svyarpr( ~ py010n , design = des_eusilc )
svyarpr( ~ py010n , design = des_eusilc , na.rm = TRUE )
# replicate-weighted design using a variable with missings
svyarpr( ~ py010n , design = des_eusilc_rep )
svyarpr( ~ py010n , design = des_eusilc_rep , na.rm = TRUE )

# library(MonetDBLite) is only available on 64-bit machines,
# so do not run this block of code in 32-bit R

# database-backed design
```

```
library(MonetDBLite)
library(DBI)
dbfolder <- tempdir()
conn <- dbConnect( MonetDBLite::MonetDBLite() , dbfolder )
dbWriteTable( conn , 'eusilc' , eusilc )

dbd_eusilc <-
svydesign(
ids = ~rb030 ,
strata = ~db040 ,
weights = ~rb050 ,
data="eusilc",
dbname=dbfolder,
dbtype="MonetDBLite"
)

dbd_eusilc <- convey_prep( dbd_eusilc )

svyarpr( ~ eqincome , design = dbd_eusilc )

dbRemoveTable( conn , 'eusilc' )


## End(Not run)
```

---

svyarpt *At-risk-of-poverty threshold*

Description

The standard definition is to use 60% of the median income.

Usage

```
svyarpt(formula, design, ...)

## S3 method for class 'survey.design'
svyarpt(formula, design, order = 0.5, percent = 0.6,
  na.rm = FALSE, ...)

## S3 method for class 'svyrep.design'
svyarpt(formula, design, order = 0.5, percent = 0.6,
  na.rm = FALSE, ...)

## S3 method for class 'DBIsvydesign'
svyarpt(formula, design, ...)
```

Arguments

formula	a formula specifying the income variable
design	a design object of class survey.design or class svyrep.design from the survey library.
...	future expansion
order	income quantile order, usually .50 (median)
percent	fraction of the quantile, usually .60
na.rm	Should cases with missing values be dropped?

Details

you must run the convey_prep function on your survey design object immediately after creating it with the svydesign or svrepdesign function.

Value

Object of class "cvystat", which are vectors with a "var" attribute giving the variance and a "statistic" attribute giving the name of the statistic.

Author(s)

Djalma Pessoa and Anthony Damico

References

Guillaume Osier (2009). Variance estimation for complex indicators of poverty and inequality. *Journal of the European Survey Research Association*, Vol.3, No.3, pp. 167-195, ISSN 1864-3361, URL http://ojs.ub.uni-konstanz.de/srm/article/view/369.

Jean-Claude Deville (1999). Variance estimation for complex statistics and estimators: linearization and residual techniques. Survey Methodology, 25, 193-203, URL http://www5.statcan.gc.ca/bsolc/olc-cel/olc-cel?lang=eng&catno=12-001-X19990024882.

See Also

svyarpr

Examples

```
library(survey)
library(vardpoor)
data(eusilc) ; names( eusilc ) <- tolower( names( eusilc ) )

# linearized design

des_eusilc <- svydesign( ids = ~rb030 , strata = ~db040 ,  weights = ~rb050 , data = eusilc )
des_eusilc <- convey_prep( des_eusilc )
svyarpt( ~eqincome , design = des_eusilc )
```

```
# replicate-weighted design
des_eusilc_rep <- as.svrepdesign( des_eusilc , type = "bootstrap" )
des_eusilc_rep <- convey_prep( des_eusilc_rep )
svyarpt( ~eqincome , design = des_eusilc_rep )

# linearized design using a variable with missings
svyarpt( ~ py010n , design = des_eusilc )
svyarpt( ~ py010n , design = des_eusilc , na.rm = TRUE )
# replicate-weighted design using a variable with missings
svyarpt( ~ py010n , design = des_eusilc_rep )
svyarpt( ~ py010n , design = des_eusilc_rep , na.rm = TRUE )


# library(MonetDBLite) is only available on 64-bit machines,
# so do not run this block of code in 32-bit R
## Not run:

# database-backed design
library(MonetDBLite)
library(DBI)
dbfolder <- tempdir()
conn <- dbConnect( MonetDBLite::MonetDBLite() , dbfolder )
dbWriteTable( conn , 'eusilc' , eusilc )

dbd_eusilc <-
svydesign(
ids = ~rb030 ,
strata = ~db040 ,
weights = ~rb050 ,
data="eusilc",
dbname=dbfolder,
dbtype="MonetDBLite"
)

dbd_eusilc <- convey_prep( dbd_eusilc )

svyarpt( ~ eqincome , design = dbd_eusilc )

dbRemoveTable( conn , 'eusilc' )


## End(Not run)
```

---

svyatk *Atkinson index*

Description

Estimate the Atkinson index, a measure of inequality

Usage

```
svyatk(formula, design, ...)

## S3 method for class 'survey.design'
svyatk(formula, design, epsilon = 1, na.rm = FALSE,
  ...)

## S3 method for class 'svyrep.design'
svyatk(formula, design, epsilon = 1, na.rm = FALSE,
  ...)

## S3 method for class 'DBIsvydesign'
svyatk(formula, design, ...)
```

Arguments

formula	a formula specifying the income variable
design	a design object of class survey.design or class svyrep.design from the survey library.
...	future expansion
epsilon	a parameter that determines the sensivity towards inequality in the bottom of the distribution. Defaults to epsilon = 1.
na.rm	Should cases with missing values be dropped?

Details

you must run the convey_prep function on your survey design object immediately after creating it with the svydesign or svrepdesign function.

Value

Object of class "cvystat", which are vectors with a "var" attribute giving the variance and a "statistic" attribute giving the name of the statistic.

Author(s)

Guilherme Jacob, Djalma Pessoa and Anthony Damico

References

Matti Langel (2012). Measuring inequality in finite population sampling. PhD thesis: Universite de Neuchatel, URL https://doc.rero.ch/record/29204/files/00002252.pdf.

Martin Biewen and Stephen Jenkins (2002). Estimation of Generalized Entropy and Atkinson Inequality Indices from Complex Survey Data. *DIW Discussion Papers*, No.345, URL https://www.diw.de/documents/publikationen/73/diw_01.c.40394.de/dp345.pdf.

See Also

svygei

Examples

```
library(survey)
library(vardpoor)
data(eusilc) ; names( eusilc ) <- tolower( names( eusilc ) )

# linearized design
des_eusilc <- svydesign( ids = ~rb030 , strata = ~db040 ,  weights = ~rb050 , data = eusilc )
des_eusilc <- convey_prep(des_eusilc)

# replicate-weighted design
des_eusilc_rep <- as.svrepdesign( des_eusilc , type = "bootstrap" )
des_eusilc_rep <- convey_prep(des_eusilc_rep)


# subset all designs to positive income and non-missing records only
des_eusilc_pos_inc <- subset( des_eusilc , eqincome > 0 )
des_eusilc_rep_pos_inc <- subset( des_eusilc_rep , eqincome > 0 )


# linearized design
svyatk( ~eqincome , des_eusilc_pos_inc, epsilon = .5 )
svyatk( ~eqincome , des_eusilc_pos_inc )
svyatk( ~eqincome , des_eusilc_pos_inc, epsilon = 2 )

# replicate-weighted design
svyatk( ~eqincome , des_eusilc_rep_pos_inc, epsilon = .5 )
svyatk( ~eqincome , des_eusilc_rep_pos_inc )
svyatk( ~eqincome , des_eusilc_rep_pos_inc, epsilon = 2 )


# subsetting
svyatk( ~eqincome , subset(des_eusilc_pos_inc, db040 == "Styria"), epsilon = .5 )
svyatk( ~eqincome , subset(des_eusilc_pos_inc, db040 == "Styria") )
svyatk( ~eqincome , subset(des_eusilc_pos_inc, db040 == "Styria"), epsilon = 2 )

svyatk( ~eqincome , subset(des_eusilc_rep_pos_inc, db040 == "Styria"), epsilon = .5 )
svyatk( ~eqincome , subset(des_eusilc_rep_pos_inc, db040 == "Styria") )
svyatk( ~eqincome , subset(des_eusilc_rep_pos_inc, db040 == "Styria"), epsilon = 2 )


# linearized design using a variable with missings (but subsetted to remove negatives)
svyatk( ~py010n , subset(des_eusilc, py010n > 0 | is.na(py010n)), epsilon = .5 )
svyatk( ~py010n , subset(des_eusilc, py010n > 0 | is.na(py010n)), epsilon = .5 , na.rm=TRUE )

# replicate-weighted design using a variable with missings (but subsetted to remove negatives)
svyatk( ~py010n , subset(des_eusilc_rep, py010n > 0 | is.na(py010n)), epsilon = .5 )
svyatk( ~py010n , subset(des_eusilc_rep, py010n > 0 | is.na(py010n)), epsilon = .5 , na.rm=TRUE )
```

```
# library(MonetDBLite) is only available on 64-bit machines,
# so do not run this block of code in 32-bit R
## Not run:

# database-backed design
library(MonetDBLite)
library(DBI)
dbfolder <- tempdir()
conn <- dbConnect( MonetDBLite::MonetDBLite() , dbfolder )
dbWriteTable( conn , 'eusilc' , eusilc )

dbd_eusilc <-
svydesign(
ids = ~rb030 ,
strata = ~db040 ,
weights = ~rb050 ,
data="eusilc",
dbname=dbfolder,
dbtype="MonetDBLite"
)

dbd_eusilc <- convey_prep( dbd_eusilc )


# subset all designs to positive income and non-missing records only
dbd_eusilc_pos_inc <- subset( dbd_eusilc , eqincome > 0 )


# database-backed linearized design
svyatk( ~eqincome , dbd_eusilc_pos_inc, epsilon = .5 )
svyatk( ~eqincome , dbd_eusilc_pos_inc )
svyatk( ~eqincome , dbd_eusilc_pos_inc, epsilon = 2 )

svyatk( ~eqincome , subset(dbd_eusilc_pos_inc, db040 == "Styria"), epsilon = .5 )
svyatk( ~eqincome , subset(dbd_eusilc_pos_inc, db040 == "Styria") )
svyatk( ~eqincome , subset(dbd_eusilc_pos_inc, db040 == "Styria"), epsilon = 2 )

# database-backed linearized design using a variable with missings
# (but subsetted to remove negatives)
svyatk( ~py010n , subset(dbd_eusilc, py010n > 0 | is.na(py010n)), epsilon = .5 )
svyatk( ~py010n , subset(dbd_eusilc, py010n > 0 | is.na(py010n)), epsilon = .5 , na.rm=TRUE )


dbRemoveTable( conn , 'eusilc' )


## End(Not run)
```

svyfgt *FGT measure of poverty*

Description

Estimate the FGT measure for the cases: `alpha=0` headcount ratio and `alpha=1` poverty gap index.

Usage

```
svyfgt(formula, design, ...)

## S3 method for class 'survey.design'
svyfgt(formula, design, g, type_thresh = "abs",
  abs_thresh = NULL, percent = 0.6, order = 0.5, na.rm = FALSE,
  thresh = FALSE, ...)

## S3 method for class 'svyrep.design'
svyfgt(formula, design, g, type_thresh = "abs",
  abs_thresh = NULL, percent = 0.6, order = 0.5, na.rm = FALSE,
  thresh = FALSE, ...)

## S3 method for class 'DBIsvydesign'
svyfgt(formula, design, ...)
```

Arguments

`formula`	a formula specifying the income variable
`design`	a design object of class `survey.design` or class `svyrep.design` from the `survey` library.
`...`	passed to `svyarpr` and `svyarpt`
`g`	If 0 estimates the headcount ratio and if 1 the poverty gap index
`type_thresh`	type of poverty threshold. If "abs" the threshold is fixed and given the value of abs_thresh; if "relq" it is given by percent times the order quantile; if "relm" it is percent times the mean.
`abs_thresh`	poverty threshold value if type_thresh is "abs"
`percent`	the multiple of the the quantile or mean used in the poverty threshold definition
`order`	the quantile order used used in the poverty threshold definition
`na.rm`	Should cases with missing values be dropped?
`thresh`	return the poverty threshold value

Details

you must run the `convey_prep` function on your survey design object immediately after creating it with the `svydesign` or `svrepdesign` function.

Value

Object of class "cvystat", which are vectors with a "var" attribute giving the variance and a "statistic" attribute giving the name of the statistic.

Author(s)

Djalma Pessoa and Anthony Damico

References

Guillaume Osier (2009). Variance estimation for complex indicators of poverty and inequality. *Journal of the European Survey Research Association*, Vol.3, No.3, pp. 167-195, ISSN 1864-3361, URL http://ojs.ub.uni-konstanz.de/srm/article/view/369.

Jean-Claude Deville (1999). Variance estimation for complex statistics and estimators: linearization and residual techniques. Survey Methodology, 25, 193-203, URL http://www5.statcan.gc.ca/bsolc/olc-cel/olc-cel?lang=eng&catno=12-001-X19990024882.

See Also

svyarpt

Examples

```
library(survey)
library(vardpoor)
data(eusilc) ; names( eusilc ) <- tolower( names( eusilc ) )

# linearized design

des_eusilc <- svydesign( ids = ~rb030 , strata = ~db040 ,  weights = ~rb050 , data = eusilc )
des_eusilc <- convey_prep( des_eusilc )

# replicate-weighted design
des_eusilc_rep <- as.svrepdesign( des_eusilc , type = "bootstrap" )
des_eusilc_rep <- convey_prep( des_eusilc_rep )

# headcount ratio, poverty threshold fixed
svyfgt(~eqincome, des_eusilc, g=0,  abs_thresh=10000)
# poverty gap index, poverty threshold fixed
svyfgt(~eqincome, des_eusilc, g=1,  abs_thresh=10000)
# headcount ratio, poverty threshold equal to arpt
svyfgt(~eqincome, des_eusilc, g=0, type_thresh= "relq" , thresh = TRUE)
# poverty gap index, poverty threshold equal to arpt
svyfgt(~eqincome, des_eusilc, g=1, type_thresh= "relq", thresh = TRUE)
# headcount ratio, poverty threshold equal to .6 times the mean
svyfgt(~eqincome, des_eusilc, g=0, type_thresh= "relm", thresh = TRUE)
# poverty gap index, poverty threshold equal to 0.6 times the mean
svyfgt(~eqincome, des_eusilc, g=1, type_thresh= "relm" , thresh = TRUE)

#  using svrep.design:
# headcount ratio, poverty threshold fixed
```

```
svyfgt(~eqincome, des_eusilc_rep, g=0,  abs_thresh=10000)
# poverty gap index, poverty threshold fixed
svyfgt(~eqincome, des_eusilc, g=1,  abs_thresh=10000)
# headcount ratio, poverty threshold equal to arpt
svyfgt(~eqincome, des_eusilc_rep, g=0, type_thresh= "relq" , thresh = TRUE)
# poverty gap index, poverty threshold equal to arpt
svyfgt(~eqincome, des_eusilc, g=1, type_thresh= "relq", thresh = TRUE)
# headcount ratio, poverty threshold equal to .6 times the mean
svyfgt(~eqincome, des_eusilc_rep, g=0, type_thresh= "relm" , thresh = TRUE)
# poverty gap index, poverty threshold equal to 0.6 times the mean
svyfgt(~eqincome, des_eusilc_rep, g=1, type_thresh= "relm", thresh = TRUE)


# library(MonetDBLite) is only available on 64-bit machines,
# so do not run this block of code in 32-bit R
## Not run:

# database-backed design
library(MonetDBLite)
library(DBI)
dbfolder <- tempdir()
conn <- dbConnect( MonetDBLite::MonetDBLite() , dbfolder )
dbWriteTable( conn , 'eusilc' , eusilc )

dbd_eusilc <-
svydesign(
ids = ~rb030 ,
strata = ~db040 ,
weights = ~rb050 ,
data="eusilc",
dbname=dbfolder,
dbtype="MonetDBLite"
)


dbd_eusilc <- convey_prep( dbd_eusilc )

# headcount ratio, poverty threshold fixed
svyfgt(~eqincome, dbd_eusilc, g=0, abs_thresh=10000)
# poverty gap index, poverty threshold fixed
svyfgt(~eqincome, dbd_eusilc, g=1, abs_thresh=10000)
# headcount ratio, poverty threshold equal to arpt
svyfgt(~eqincome, dbd_eusilc, g=0, type_thresh= "relq", thresh = TRUE)
# poverty gap index, poverty threshold equal to arpt
svyfgt(~eqincome, dbd_eusilc, g=1, type_thresh= "relq")
# headcount ratio, poverty threshold equal to .6 times the mean
svyfgt(~eqincome, dbd_eusilc, g=0, type_thresh= "relm")
# poverty gap index, poverty threshold equal to 0.6 times the mean
svyfgt(~eqincome, dbd_eusilc, g=1, type_thresh= "relm")

dbRemoveTable( conn , 'eusilc' )
```

```
## End(Not run)
```

---

svygei *Generalized entropy index*

Description

Estimate the generalized entropy index, a measure of inequality

Usage

```
svygei(formula, design, ...)

## S3 method for class 'survey.design'
svygei(formula, design, epsilon = 1, na.rm = FALSE,
  ...)

## S3 method for class 'svyrep.design'
svygei(formula, design, epsilon = 1, na.rm = FALSE,
  ...)

## S3 method for class 'DBIsvydesign'
svygei(formula, design, ...)
```

Arguments

formula	a formula specifying the income variable
design	a design object of class survey.design or class svyrep.design from the survey library.
...	future expansion
epsilon	a parameter that determines the sensivity towards inequality in the top of the distribution. Defaults to epsilon = 1.
na.rm	Should cases with missing values be dropped?

Details

you must run the convey_prep function on your survey design object immediately after creating it with the svydesign or svrepdesign function.

If epsilon == 0 or epsilon == 1, the logarithm in the function only allows for strictly positive variables.

Value

Object of class "cvystat", which are vectors with a "var" attribute giving the variance and a "statistic" attribute giving the name of the statistic.

Author(s)

Guilherme Jacob, Djalma Pessoa and Anthony Damico

References

Matti Langel (2012). Measuring inequality in finite population sampling. PhD thesis: Universite de Neuchatel, URL https://doc.rero.ch/record/29204/files/00002252.pdf.

Martin Biewen and Stephen Jenkins (2002). Estimation of Generalized Entropy and Atkinson Inequality Indices from Complex Survey Data. *DIW Discussion Papers*, No.345, URL https://www.diw.de/documents/publikationen/73/diw_01.c.40394.de/dp345.pdf.

See Also

svyatk

Examples

```
library(survey)
library(vardpoor)
data(eusilc) ; names( eusilc ) <- tolower( names( eusilc ) )

# linearized design
des_eusilc <- svydesign( ids = ~rb030 , strata = ~db040 ,  weights = ~rb050 , data = eusilc )
des_eusilc <- convey_prep(des_eusilc)

# replicate-weighted design
des_eusilc_rep <- as.svrepdesign( des_eusilc , type = "bootstrap" )
des_eusilc_rep <- convey_prep(des_eusilc_rep)

# linearized design
svygei( ~eqincome , subset(des_eusilc, eqincome > 0), epsilon = 0 )
svygei( ~eqincome , des_eusilc, epsilon = .5 )
svygei( ~eqincome , subset(des_eusilc, eqincome > 0), epsilon = 1 )
svygei( ~eqincome , des_eusilc, epsilon = 2 )

# replicate-weighted design
svygei( ~eqincome , subset(des_eusilc_rep, eqincome > 0), epsilon = 0 )
svygei( ~eqincome , des_eusilc_rep, epsilon = .5 )
svygei( ~eqincome , subset(des_eusilc_rep, eqincome > 0), epsilon = 1 )
svygei( ~eqincome , des_eusilc_rep, epsilon = 2 )

# linearized design using a variable with missings
svygei( ~py010n , subset(des_eusilc, py010n > 0 | is.na(py010n)), epsilon = 0 )
svygei( ~py010n , subset(des_eusilc, py010n > 0 | is.na(py010n)), epsilon = 0, na.rm = TRUE )
svygei( ~py010n , des_eusilc, epsilon = .5 )
svygei( ~py010n , des_eusilc, epsilon = .5, na.rm = TRUE )
svygei( ~py010n , subset(des_eusilc, py010n > 0 | is.na(py010n)), epsilon = 1 )
svygei( ~py010n , subset(des_eusilc, py010n > 0 | is.na(py010n)), epsilon = 1, na.rm = TRUE )
svygei( ~py010n , des_eusilc, epsilon = 2 )
svygei( ~py010n , des_eusilc, epsilon = 2, na.rm = TRUE )
```

```
# replicate-weighted design using a variable with missings
svygei( ~py010n , subset(des_eusilc_rep, py010n > 0 | is.na(py010n)), epsilon = 0 )
svygei( ~py010n , subset(des_eusilc_rep, py010n > 0 | is.na(py010n)), epsilon = 0, na.rm = TRUE )
svygei( ~py010n , des_eusilc_rep, epsilon = .5 )
svygei( ~py010n , des_eusilc_rep, epsilon = .5, na.rm = TRUE )
svygei( ~py010n , subset(des_eusilc_rep, py010n > 0 | is.na(py010n)), epsilon = 1 )
svygei( ~py010n , subset(des_eusilc_rep, py010n > 0 | is.na(py010n)), epsilon = 1, na.rm = TRUE )
svygei( ~py010n , des_eusilc_rep, epsilon = 2 )
svygei( ~py010n , des_eusilc_rep, epsilon = 2, na.rm = TRUE )


# library(MonetDBLite) is only available on 64-bit machines,
# so do not run this block of code in 32-bit R
## Not run:

# database-backed design
library(MonetDBLite)
library(DBI)
dbfolder <- tempdir()
conn <- dbConnect( MonetDBLite::MonetDBLite() , dbfolder )
dbWriteTable( conn , 'eusilc' , eusilc )

dbd_eusilc <-
svydesign(
ids = ~rb030 ,
strata = ~db040 ,
weights = ~rb050 ,
data="eusilc",
dbname=dbfolder,
dbtype="MonetDBLite"
)

dbd_eusilc <- convey_prep( dbd_eusilc )

# database-backed linearized design
svygei( ~eqincome , subset(dbd_eusilc, eqincome > 0), epsilon = 0 )
svygei( ~eqincome , dbd_eusilc, epsilon = .5 )
svygei( ~eqincome , subset(dbd_eusilc, eqincome > 0), epsilon = 1 )
svygei( ~eqincome , dbd_eusilc, epsilon = 2 )

# database-backed linearized design using a variable with missings
svygei( ~py010n , subset(dbd_eusilc, py010n > 0 | is.na(py010n)), epsilon = 0 )
svygei( ~py010n , subset(dbd_eusilc, py010n > 0 | is.na(py010n)), epsilon = 0, na.rm = TRUE )
svygei( ~py010n , dbd_eusilc, epsilon = .5 )
svygei( ~py010n , dbd_eusilc, epsilon = .5, na.rm = TRUE )
svygei( ~py010n , subset(dbd_eusilc, py010n > 0 | is.na(py010n)), epsilon = 1 )
svygei( ~py010n , subset(dbd_eusilc, py010n > 0 | is.na(py010n)), epsilon = 1, na.rm = TRUE )
svygei( ~py010n , dbd_eusilc, epsilon = 2 )
svygei( ~py010n , dbd_eusilc, epsilon = 2, na.rm = TRUE )

dbRemoveTable( conn , 'eusilc' )
```

```
## End(Not run)
```

---

svygini *Gini coefficient*

Description

Estimate the Gini coefficient, a measure of inequalty

Usage

```
svygini(formula, design, ...)

## S3 method for class 'survey.design'
svygini(formula, design, na.rm = FALSE, ...)

## S3 method for class 'svyrep.design'
svygini(formula, design, na.rm = FALSE, ...)

## S3 method for class 'DBIsvydesign'
svygini(formula, design, ...)
```

Arguments

formula	a formula specifying the income variable
design	a design object of class survey.design or class svyrep.design from the survey library.
...	future expansion
na.rm	Should cases with missing values be dropped?

Details

you must run the convey_prep function on your survey design object immediately after creating it with the svydesign or svrepdesign function.

Value

Object of class "cvystat", which are vectors with a "var" attribute giving the variance and a "statistic" attribute giving the name of the statistic.

Author(s)

Djalma Pessoa and Anthony Damico

References

Guillaume Osier (2009). Variance estimation for complex indicators of poverty and inequality. *Journal of the European Survey Research Association*, Vol.3, No.3, pp. 167-195, ISSN 1864-3361, URL http://ojs.ub.uni-konstanz.de/srm/article/view/369.

Jean-Claude Deville (1999). Variance estimation for complex statistics and estimators: linearization and residual techniques. Survey Methodology, 25, 193-203, URL http://www5.statcan.gc.ca/bsolc/olc-cel/olc-cel?lang=eng&catno=12-001-X19990024882.

See Also

svyarpr

Examples

```
library(survey)
library(vardpoor)
data(eusilc) ; names( eusilc ) <- tolower( names( eusilc ) )

# linearized design
des_eusilc <- svydesign( ids = ~rb030 , strata = ~db040 ,  weights = ~rb050 , data = eusilc )
des_eusilc <- convey_prep(des_eusilc)

svygini( ~eqincome , design = des_eusilc )

# replicate-weighted design
des_eusilc_rep <- as.svrepdesign( des_eusilc , type = "bootstrap" )
des_eusilc_rep <- convey_prep(des_eusilc_rep)

svygini( ~eqincome , design = des_eusilc_rep )

# linearized design using a variable with missings
svygini( ~ py010n , design = des_eusilc )
svygini( ~ py010n , design = des_eusilc , na.rm = TRUE )
# replicate-weighted design using a variable with missings
svygini( ~ py010n , design = des_eusilc_rep )
svygini( ~ py010n , design = des_eusilc_rep , na.rm = TRUE )


# library(MonetDBLite) is only available on 64-bit machines,
# so do not run this block of code in 32-bit R
## Not run:

# database-backed design
library(MonetDBLite)
library(DBI)
dbfolder <- tempdir()
conn <- dbConnect( MonetDBLite::MonetDBLite() , dbfolder )
dbWriteTable( conn , 'eusilc' , eusilc )

dbd_eusilc <-
svydesign(
```

```
ids = ~rb030 ,
strata = ~db040 ,
weights = ~rb050 ,
data="eusilc",
dbname=dbfolder,
dbtype="MonetDBLite"
)

dbd_eusilc <- convey_prep( dbd_eusilc )

svygini( ~ eqincome , design = dbd_eusilc )

dbRemoveTable( conn , 'eusilc' )


## End(Not run)
```

---

svygpg *Linearization of the gender pay (wage) gap*

Description

Estimate the difference between the average gross hourly earnings of men and women expressed as a percentage of the average gross hourly earnings of men.

Usage

```
svygpg(formula, design, ...)

## S3 method for class 'survey.design'
svygpg(formula, design, sex, na.rm = FALSE, ...)

## S3 method for class 'svyrep.design'
svygpg(formula, design, sex, na.rm = FALSE, ...)

## S3 method for class 'DBIsvydesign'
svygpg(formula, design, sex, ...)
```

Arguments

formula	a formula specifying the gross hourly earnings variable
design	a design object of class survey.design or class svyrep.design from the survey library.
...	future expansion
sex	formula with a factor with labels 'male' and 'female'
na.rm	Should cases with missing values be dropped?

Details

you must run the convey_prep function on your survey design object immediately after creating it with the svydesign or svrepdesign function.

Value

Object of class "cvystat", which are vectors with a "var" attribute giving the variance and a "statistic" attribute giving the name of the statistic.

Author(s)

Djalma Pessoa and Anthony Damico

References

Guillaume Osier (2009). Variance estimation for complex indicators of poverty and inequality. *Journal of the European Survey Research Association*, Vol.3, No.3, pp. 167-195, ISSN 1864-3361, URL http://ojs.ub.uni-konstanz.de/srm/article/view/369.

Jean-Claude Deville (1999). Variance estimation for complex statistics and estimators: linearization and residual techniques. Survey Methodology, 25, 193-203, URL http://www5.statcan.gc.ca/bsolc/olc-cel/olc-cel?lang=eng&catno=12-001-X19990024882.

See Also

svyarpt

Examples

```
library(vardpoor)
library(survey)
data(ses)
names( ses ) <- gsub( "size" , "size_" , tolower( names( ses ) ) )
des_ses <- svydesign(id=~1, weights=~weights, data=ses)
des_ses <- convey_prep(des_ses)

# linearized design
svygpg(~earningshour, des_ses, ~sex)
# replicate-weighted design
des_ses_rep <-  as.svrepdesign( des_ses , type = "bootstrap" )
des_ses_rep <- convey_prep(des_ses_rep)

svygpg(~earningshour, des_ses_rep, ~sex)


# library(MonetDBLite) is only available on 64-bit machines,
# so do not run this block of code in 32-bit R
## Not run:

# database-backed design
library(MonetDBLite)
library(DBI)
```

```
dbfolder <- tempdir()
conn <- dbConnect( MonetDBLite::MonetDBLite() , dbfolder )
dbWriteTable( conn , 'ses' , ses )

dbd_ses <- svydesign(id=~1, weights=~weights, data="ses", dbname=dbfolder, dbtype="MonetDBLite")
dbd_ses <- convey_prep( dbd_ses )

svygpg(formula=~earningshour, design=dbd_ses, sex= ~sex)

dbRemoveTable( conn , 'ses' )


## End(Not run)
```

---

svyiqalpha *Linearization of a variable quantile*

Description

Computes the linearized variable of a quantile of variable.

Usage

```
svyiqalpha(formula, design, ...)

## S3 method for class 'survey.design'
svyiqalpha(formula, design, alpha, na.rm = FALSE, ...)

## S3 method for class 'svyrep.design'
svyiqalpha(formula, design, alpha, na.rm = FALSE, ...)

## S3 method for class 'DBIsvydesign'
svyiqalpha(formula, design, ...)
```

Arguments

formula	a formula specifying the income variable
design	a design object of class survey.design or class svyrep.design from the survey library.
...	future expansion
alpha	the order of the quantile
na.rm	Should cases with missing values be dropped?

Details

you must run the convey_prep function on your survey design object immediately after creating it with the svydesign or svrepdesign function.

Value

Object of class "cvystat", which are vectors with a "var" attribute giving the variance and a "statistic" attribute giving the name of the statistic.

Author(s)

Djalma Pessoa and Anthony Damico

References

Guillaume Osier (2009). Variance estimation for complex indicators of poverty and inequality. *Journal of the European Survey Research Association*, Vol.3, No.3, pp. 167-195, ISSN 1864-3361, URL http://ojs.ub.uni-konstanz.de/srm/article/view/369.

Jean-Claude Deville (1999). Variance estimation for complex statistics and estimators: linearization and residual techniques. Survey Methodology, 25, 193-203, URL http://www5.statcan.gc.ca/bsolc/olc-cel/olc-cel?lang=eng&catno=12-001-X19990024882.

See Also

svyarpr

Examples

```
library(vardpoor)
data(eusilc) ; names( eusilc ) <- tolower( names( eusilc ) )
library(survey)
# linearized design
des_eusilc <- svydesign( ids = ~rb030 , strata = ~db040 ,  weights = ~rb050 , data = eusilc )
des_eusilc <- convey_prep(des_eusilc)

svyiqalpha( ~eqincome , design = des_eusilc, .50 )

# replicate-weighted design
des_eusilc_rep <- as.svrepdesign( des_eusilc , type = "bootstrap" )
des_eusilc_rep <- convey_prep(des_eusilc_rep)

svyiqalpha( ~eqincome , design = des_eusilc_rep, .50 )

# linearized design using a variable with missings
svyiqalpha( ~ py010n , design = des_eusilc, .50 )
svyiqalpha( ~ py010n , design = des_eusilc , .50, na.rm = TRUE )
# replicate-weighted design using a variable with missings
svyiqalpha( ~ py010n , design = des_eusilc_rep, .50 )
svyiqalpha( ~ py010n , design = des_eusilc_rep ,.50, na.rm = TRUE )


# library(MonetDBLite) is only available on 64-bit machines,
# so do not run this block of code in 32-bit R
## Not run:
```

```
# database-backed design
library(MonetDBLite)
library(DBI)
dbfolder <- tempdir()
conn <- dbConnect( MonetDBLite::MonetDBLite() , dbfolder )
dbWriteTable( conn , 'eusilc' , eusilc )

dbd_eusilc <-
svydesign(
ids = ~rb030 ,
strata = ~db040 ,
weights = ~rb050 ,
data="eusilc",
dbname=dbfolder,
dbtype="MonetDBLite"
)

dbd_eusilc <- convey_prep( dbd_eusilc )

svyiqalpha( ~ eqincome , design = dbd_eusilc, .50 )

dbRemoveTable( conn , 'eusilc' )


## End(Not run)
```

---

svyisq *Linearization of the total below a quantile*

Description

Computes the linearized variable of the total in the lower tail of the distribution of a variable.

Usage

```
svyisq(formula, design, ...)

## S3 method for class 'survey.design'
svyisq(formula, design, alpha, quantile = FALSE,
  na.rm = FALSE, ...)

## S3 method for class 'svyrep.design'
svyisq(formula, design, alpha, quantile = FALSE,
  na.rm = FALSE, ...)

## S3 method for class 'DBIsvydesign'
svyisq(formula, design, ...)
```

Arguments

formula	a formula specifying the income variable
design	a design object of class survey.design or class svyrep.design from the survey library.
...	future expansion
alpha	the order of the quantile
quantile	return the upper bound of the lower tail
na.rm	Should cases with missing values be dropped?

Details

you must run the convey_prep function on your survey design object immediately after creating it with the svydesign or svrepdesign function.

Value

Object of class "cvystat", which are vectors with a "var" attribute giving the variance and a "statistic" attribute giving the name of the statistic.

Author(s)

Djalma Pessoa and Anthony Damico

References

Guillaume Osier (2009). Variance estimation for complex indicators of poverty and inequality. *Journal of the European Survey Research Association*, Vol.3, No.3, pp. 167-195, ISSN 1864-3361, URL http://ojs.ub.uni-konstanz.de/srm/article/view/369.

Jean-Claude Deville (1999). Variance estimation for complex statistics and estimators: linearization and residual techniques. Survey Methodology, 25, 193-203, URL http://www5.statcan.gc.ca/bsolc/olc-cel/olc-cel?lang=eng&catno=12-001-X19990024882.

See Also

svyarpr

Examples

```
library(vardpoor)
data(eusilc) ; names( eusilc ) <- tolower( names( eusilc ) )
library(survey)
des_eusilc <- svydesign(ids = ~rb030, strata =~db040,  weights = ~rb050, data = eusilc)
des_eusilc <- convey_prep(des_eusilc)
svyisq(~eqincome, design=des_eusilc,.20 , quantile = TRUE)

# replicate-weighted design
des_eusilc_rep <- as.svrepdesign( des_eusilc , type = "bootstrap" )
des_eusilc_rep <- convey_prep(des_eusilc_rep)
```

```
svyisq( ~eqincome , design = des_eusilc_rep, .20 , quantile = TRUE )
# linearized design using a variable with missings
svyisq( ~ py010n , design = des_eusilc, .20 )
svyisq( ~ py010n , design = des_eusilc , .20, na.rm = TRUE )
# replicate-weighted design using a variable with missings
svyisq( ~ py010n , design = des_eusilc_rep, .20 )
svyisq( ~ py010n , design = des_eusilc_rep , .20,  na.rm = TRUE )


# library(MonetDBLite) is only available on 64-bit machines,
# so do not run this block of code in 32-bit R
## Not run:

# database-backed design
library(MonetDBLite)
library(DBI)
dbfolder <- tempdir()
conn <- dbConnect( MonetDBLite::MonetDBLite() , dbfolder )
dbWriteTable( conn , 'eusilc' , eusilc )

dbd_eusilc <-
svydesign(
ids = ~rb030 ,
strata = ~db040 ,
weights = ~rb050 ,
data="eusilc",
dbname=dbfolder,
dbtype="MonetDBLite"
)

dbd_eusilc <- convey_prep( dbd_eusilc )

svyisq( ~ eqincome , design = dbd_eusilc, .20 )

dbRemoveTable( conn , 'eusilc' )


## End(Not run)
```

---

svylorenz *Lorenz curve*

Description

Estimate the Lorenz curve, an inequality graph

Usage

```
svylorenz(formula, design, ...)

## S3 method for class 'survey.design'
svylorenz(formula, design, quantiles = seq(0, 1, 0.1),
  empirical = FALSE, plot = TRUE, add = FALSE, curve.col = "red",
  ci = TRUE, alpha = 0.05, na.rm = FALSE, ...)

## S3 method for class 'svyrep.design'
svylorenz(formula, design, quantiles = seq(0, 1, 0.1),
  empirical = FALSE, plot = TRUE, add = FALSE, curve.col = "red",
  ci = TRUE, alpha = 0.05, na.rm = FALSE, ...)

## S3 method for class 'DBIsvydesign'
svylorenz(formula, design, ...)
```

Arguments

formula	a formula specifying the income variable
design	a design object of class survey.design or class svyrep.design from the survey library.
...	additional arguments passed to plot methods
quantiles	a sequence of probabilities that defines the quantiles sum to be calculated
empirical	Should an empirical Lorenz curve be estimated as well? Defaults to FALSE.
plot	Should the Lorenz curve be plotted? Defaults to TRUE.
add	Should a new curve be plotted on the current graph?
curve.col	a string defining the color of the curve.
ci	Should the confidence interval be plotted? Defaults to TRUE.
alpha	a number that especifies de confidence level for the graph.
na.rm	Should cases with missing values be dropped? Defaults to FALSE.

Details

you must run the convey_prep function on your survey design object immediately after creating it with the svydesign or svrepdesign function.

Notice that the 'empirical' curve is observation-based and is the one actually used to calculate the Gini index. On the other hand, the quantile-based curve is used to estimate the shares, SEs and confidence intervals.

This way, as the number of quantiles of the quantile-based function increases, the qwuantile-based curve approaccches the observation-based curve.

Value

Object of class "svyquantile", which are vectors with a "quantiles" attribute giving the proportion of income below that quantile, and a "SE" attribute giving the standard errors of the estimates.

Author(s)

Guilherme Jacob, Djalma Pessoa and Anthony Damico

References

Milorad Kovacevic and David Binder (1997). Variance Estimation for Measures of Income Inequality and Polarization - The Estimating Equations Approach. *Journal of Official Statistics*, Vol.13, No.1, 1997. pp. 41 58. URL http://www.jos.nu/Articles/abstract.asp?article=13141.

Shlomo Yitzhaki and Robert Lerman (1989). Improving the accuracy of estimates of Gini coefficients. *Journal of Econometrics*, Vol.42(1), pp. 43-47, September.

Matti Langel (2012). *Measuring inequality in finite population sampling*. PhD thesis. URL http://doc.rero.ch/record/29204.

See Also

svyquantile

Examples

```
## Not run:
library(survey)
library(vardpoor)
data(eusilc) ; names( eusilc ) <- tolower( names( eusilc ) )

# linearized design
des_eusilc <- svydesign( ids = ~rb030 , strata = ~db040 ,  weights = ~rb050 , data = eusilc )
des_eusilc <- convey_prep( des_eusilc )
svylorenz( ~eqincome , des_eusilc, seq(0,1,.05), alpha = .01 )

# replicate-weighted design
des_eusilc_rep <- as.svrepdesign( des_eusilc , type = "bootstrap" )
des_eusilc_rep <- convey_prep( des_eusilc_rep )

svylorenz( ~eqincome , des_eusilc_rep, seq(0,1,.05), alpha = .01 )


# linearized design using a variable with missings
svylorenz( ~py010n , des_eusilc, seq(0,1,.05), alpha = .01 )
svylorenz( ~py010n , des_eusilc, seq(0,1,.05), alpha = .01, na.rm = TRUE )
# demonstration of `curve.col=` and `add=` parameters
svylorenz( ~eqincome , des_eusilc, seq(0,1,.05), alpha = .05 , add = TRUE , curve.col = 'green' )
# replicate-weighted design using a variable with missings
svylorenz( ~py010n , des_eusilc_rep, seq(0,1,.05), alpha = .01 )
svylorenz( ~py010n , des_eusilc_rep, seq(0,1,.05), alpha = .01, na.rm = TRUE )


# library(MonetDBLite) is only available on 64-bit machines,
# so do not run this block of code in 32-bit R

# database-backed design
library(MonetDBLite)
```

```
library(DBI)
dbfolder <- tempdir()
conn <- dbConnect( MonetDBLite::MonetDBLite() , dbfolder )
dbWriteTable( conn , 'eusilc' , eusilc )

dbd_eusilc <-
svydesign(
ids = ~rb030 ,
strata = ~db040 ,
weights = ~rb050 ,
data="eusilc",
dbname=dbfolder,
dbtype="MonetDBLite"
)

dbd_eusilc <- convey_prep( dbd_eusilc )

svylorenz( ~eqincome , dbd_eusilc, seq(0,1,.05), alpha = .01 )

# highlithing the difference between the quantile-based curve and the empirical version:
svylorenz( ~eqincome , dbd_eusilc, seq(0,1,.5), empirical = TRUE, ci = FALSE, curve.col = "green" )
svylorenz( ~eqincome , dbd_eusilc, seq(0,1,.5), alpha = .01, add = TRUE )
legend( "topleft", c("Quantile-based", "Empirical"), lwd = c(1,1), col = c("red", "green"))
# as the number of quantiles increases, the difference between the curves gets smaller
svylorenz( ~eqincome , dbd_eusilc, seq(0,1,.01), empirical = TRUE, ci = FALSE, curve.col = "green" )
svylorenz( ~eqincome , dbd_eusilc, seq(0,1,.01), alpha = .01, add = TRUE )
legend( "topleft", c("Quantile-based", "Empirical"), lwd = c(1,1), col = c("red", "green"))

dbRemoveTable( conn , 'eusilc' )


## End(Not run)
```

---

svypoormed *Relative median poverty gap*

Description

Estimate the median of incomes less than the at-risk-of-poverty threshold (arpt).

Usage

```
svypoormed(formula, design, ...)

## S3 method for class 'survey.design'
svypoormed(formula, design, order = 0.5,
  percent = 0.6, na.rm = FALSE, ...)

## S3 method for class 'svyrep.design'
```

```
svypoormed(formula, design, order = 0.5,
  percent = 0.6, na.rm = FALSE, ...)

## S3 method for class 'DBIsvydesign'
svypoormed(formula, design, ...)
```

Arguments

formula	a formula specifying the income variable
design	a design object of class survey.design or class svyrep.design from the survey library.
...	future expansion
order	income quantile order, usually .5
percent	fraction of the quantile, usually .60
na.rm	Should cases with missing values be dropped?

Details

you must run the convey_prep function on your survey design object immediately after creating it with the svydesign or svrepdesign function.

Value

Object of class "cvystat", which are vectors with a "var" attribute giving the variance and a "statistic" attribute giving the name of the statistic.

Author(s)

Djalma Pessoa and Anthony Damico

References

Guillaume Osier (2009). Variance estimation for complex indicators of poverty and inequality. *Journal of the European Survey Research Association*, Vol.3, No.3, pp. 167-195, ISSN 1864-3361, URL http://ojs.ub.uni-konstanz.de/srm/article/view/369.

Jean-Claude Deville (1999). Variance estimation for complex statistics and estimators: linearization and residual techniques. Survey Methodology, 25, 193-203, URL http://www5.statcan.gc.ca/bsolc/olc-cel/olc-cel?lang=eng&catno=12-001-X19990024882.

See Also

svyarpt

Examples

```
## Not run:
library(survey)
library(vardpoor)
data(eusilc) ; names( eusilc ) <- tolower( names( eusilc ) )

# linearized design
des_eusilc <- svydesign( ids = ~rb030 , strata = ~db040 ,  weights = ~rb050 , data = eusilc )
des_eusilc <- convey_prep( des_eusilc )

svypoormed( ~eqincome , design = des_eusilc )

# replicate-weighted design
des_eusilc_rep <- as.svrepdesign( des_eusilc , type = "bootstrap" )
des_eusilc_rep <- convey_prep( des_eusilc_rep )

svypoormed( ~eqincome , design = des_eusilc_rep )

# linearized design using a variable with missings
svypoormed( ~ py010n , design = des_eusilc )
svypoormed( ~ py010n , design = des_eusilc , na.rm = TRUE )
# replicate-weighted design using a variable with missings
svypoormed( ~ py010n , design = des_eusilc_rep )
svypoormed( ~ py010n , design = des_eusilc_rep , na.rm = TRUE )

# library(MonetDBLite) is only available on 64-bit machines,
# so do not run this block of code in 32-bit R

# database-backed design
library(MonetDBLite)
library(DBI)
dbfolder <- tempdir()
conn <- dbConnect( MonetDBLite::MonetDBLite() , dbfolder )
dbWriteTable( conn , 'eusilc' , eusilc )

dbd_eusilc <-
svydesign(
ids = ~rb030 ,
strata = ~db040 ,
weights = ~rb050 ,
data="eusilc",
dbname=dbfolder,
dbtype="MonetDBLite"
)

dbd_eusilc <- convey_prep( dbd_eusilc )

svypoormed( ~ eqincome , design = dbd_eusilc )

dbRemoveTable( conn , 'eusilc' )
```

```
## End(Not run)
```

---

svyqsr *Quintile Share Ratio*

Description

Estimate ratio of the total income received by the highest earners to the total income received by lowest earners, defaulting to 20

Usage

```
svyqsr(formula, design, ...)

## S3 method for class 'survey.design'
svyqsr(formula, design, alpha = 0.2, na.rm = FALSE,
  upper_quant = FALSE, lower_quant = FALSE, upper_tot = FALSE,
  lower_tot = FALSE, ...)

## S3 method for class 'svyrep.design'
svyqsr(formula, design, alpha = 0.2, na.rm = FALSE,
  upper_quant = FALSE, lower_quant = FALSE, upper_tot = FALSE,
  lower_tot = FALSE, ...)

## S3 method for class 'DBIsvydesign'
svyqsr(formula, design, ...)
```

Arguments

formula	a formula specifying the income variable
design	a design object of class survey.design or class svyrep.design from the survey library.
...	future expansion
alpha	order of the quintile ratio
na.rm	Should cases with missing values be dropped?
upper_quant	return the lower bound of highest earners
lower_quant	return the upper bound of lowest earners
upper_tot	return the highest earners total
lower_tot	return the lowest earners total

Details

you must run the convey_prep function on your survey design object immediately after creating it with the svydesign or svrepdesign function.

Value

Object of class "cvystat", which are vectors with a "var" attribute giving the variance and a "statistic" attribute giving the name of the statistic.

Author(s)

Djalma Pessoa and Anthony Damico

References

Guillaume Osier (2009). Variance estimation for complex indicators of poverty and inequality. *Journal of the European Survey Research Association*, Vol.3, No.3, pp. 167-195, ISSN 1864-3361, URL http://ojs.ub.uni-konstanz.de/srm/article/view/369.

Jean-Claude Deville (1999). Variance estimation for complex statistics and estimators: linearization and residual techniques. Survey Methodology, 25, 193-203, URL http://www5.statcan.gc.ca/bsolc/olc-cel/olc-cel?lang=eng&catno=12-001-X19990024882.

See Also

svyarpt

Examples

```
## Not run:
library(survey)
library(vardpoor)
data(eusilc) ; names( eusilc ) <- tolower( names( eusilc ) )

# linearized design
des_eusilc <- svydesign( ids = ~rb030 , strata = ~db040 ,  weights = ~rb050 , data = eusilc )
des_eusilc <- convey_prep( des_eusilc )

svyqsr( ~eqincome , design = des_eusilc, upper_tot = TRUE, lower_tot = TRUE )

# replicate-weighted design
des_eusilc_rep <- as.svrepdesign( des_eusilc , type = "bootstrap" )
des_eusilc_rep <- convey_prep( des_eusilc_rep )

svyqsr( ~eqincome , design = des_eusilc_rep, upper_tot = TRUE, lower_tot = TRUE )

# linearized design using a variable with missings
svyqsr( ~ py010n , design = des_eusilc )
svyqsr( ~ py010n , design = des_eusilc , na.rm = TRUE )
# replicate-weighted design using a variable with missings
svyqsr( ~ py010n , design = des_eusilc_rep )
svyqsr( ~ py010n , design = des_eusilc_rep , na.rm = TRUE )

# library(MonetDBLite) is only available on 64-bit machines,
# so do not run this block of code in 32-bit R

# database-backed design
```

```
library(MonetDBLite)
library(DBI)
dbfolder <- tempdir()
conn <- dbConnect( MonetDBLite::MonetDBLite() , dbfolder )
dbWriteTable( conn , 'eusilc' , eusilc )

dbd_eusilc <-
svydesign(
ids = ~rb030 ,
strata = ~db040 ,
weights = ~rb050 ,
data="eusilc",
dbname=dbfolder,
dbtype="MonetDBLite"
)

dbd_eusilc <- convey_prep( dbd_eusilc )

svyqsr( ~ eqincome , design = dbd_eusilc )

dbRemoveTable( conn , 'eusilc' )


## End(Not run)
```

---

svyrenyi *Renyi divergence measure*

Description

Estimate the Renyi divergence measure, a measure of inequality

Usage

```
svyrenyi(formula, design, ...)

## S3 method for class 'survey.design'
svyrenyi(formula, design, epsilon = 1,
  na.rm = FALSE, ...)

## S3 method for class 'svyrep.design'
svyrenyi(formula, design, epsilon = 1,
  na.rm = FALSE, ...)

## S3 method for class 'DBIsvydesign'
svyrenyi(formula, design, ...)
```

Arguments

formula	a formula specifying the income variable
design	a design object of class survey.design or class svyrep.design from the survey library.
...	future expansion
epsilon	a parameter that determines the sensivity towards inequality on the top of the distribution. Defaults to epsilon = 1.
na.rm	Should cases with missing values be dropped?

Details

you must run the convey_prep function on your survey design object immediately after creating it with the svydesign or svrepdesign function.

If epsilon == 1, the result matches svygei with epsilon == 1. As in the generalized entropy index, when epsilon == 1, the logarithm in the function only allows for strictly positive variables.

Value

Object of class "cvystat", which are vectors with a "var" attribute giving the variance and a "statistic" attribute giving the name of the statistic.

Author(s)

Guilherme Jacob, Djalma Pessoa and Anthony Damico

References

Matti Langel (2012). Measuring inequality in finite population sampling. PhD thesis: Universite de Neuchatel, URL https://doc.rero.ch/record/29204/files/00002252.pdf.

See Also

svygei

Examples

```
library(survey)
library(vardpoor)
data(eusilc) ; names( eusilc ) <- tolower( names( eusilc ) )

# linearized design
des_eusilc <- svydesign( ids = ~rb030 , strata = ~db040 ,  weights = ~rb050 , data = eusilc )
des_eusilc <- convey_prep(des_eusilc)

svyrenyi( ~eqincome , design = des_eusilc, epsilon = .5 )

# replicate-weighted design
des_eusilc_rep <- as.svrepdesign( des_eusilc , type = "bootstrap" )
des_eusilc_rep <- convey_prep(des_eusilc_rep)
```

```
svyrenyi( ~eqincome , design = des_eusilc_rep, epsilon = .5 )

# linearized design using a variable with missings
svyrenyi( ~py010n , design = des_eusilc, epsilon = .5 )
svyrenyi( ~py010n , design = des_eusilc, epsilon = .5, na.rm = TRUE )
# replicate-weighted design using a variable with missings
svyrenyi( ~py010n , design = des_eusilc_rep, epsilon = .5 )
svyrenyi( ~py010n , design = des_eusilc_rep, epsilon = .5, na.rm = TRUE )


# library(MonetDBLite) is only available on 64-bit machines,
# so do not run this block of code in 32-bit R
## Not run:

# database-backed design
library(MonetDBLite)
library(DBI)
dbfolder <- tempdir()
conn <- dbConnect( MonetDBLite::MonetDBLite() , dbfolder )
dbWriteTable( conn , 'eusilc' , eusilc )

dbd_eusilc <-
svydesign(
ids = ~rb030 ,
strata = ~db040 ,
weights = ~rb050 ,
data="eusilc",
dbname=dbfolder,
dbtype="MonetDBLite"
)

dbd_eusilc <- convey_prep( dbd_eusilc )

svyrenyi( ~eqincome , design = dbd_eusilc, epsilon = .5 )

# Testing if Renyi and GEI match when epsilon == 1:
svyrenyi( ~eqincome , design = subset(dbd_eusilc, eqincome > 0 ), epsilon = 1 )
svygei( ~eqincome , design = subset(dbd_eusilc, eqincome > 0 ), epsilon = 1 )

dbRemoveTable( conn , 'eusilc' )


## End(Not run)
```

---

svyrmir *Relative median income ratio*

Description

Estimate the ratio between the median income of people with age above 65 and the median income of people with age below 65.

Usage

```
svyrmir(formula, design, ...)

## S3 method for class 'survey.design'
svyrmir(formula, design, age, agelim = 65,
  order = 0.5, na.rm = FALSE, med_old = FALSE, med_young = FALSE, ...)

## S3 method for class 'svyrep.design'
svyrmir(formula, design, age, agelim = 65,
  order = 0.5, na.rm = FALSE, med_old = FALSE, med_young = FALSE, ...)

## S3 method for class 'DBIsvydesign'
svyrmir(formula, design, age, ...)
```

Arguments

formula	a formula specifying the income variable
design	a design object of class survey.design or class svyrep.design from the survey library.
...	future expansion
age	formula defining the variable age
agelim	the age cutpoint, the default is 65
order	income quantile order, usually .5
na.rm	Should cases with missing values be dropped?
med_old	return the median income of people older than agelim
med_young	return the median income of people younger than agelim

Details

you must run the convey_prep function on your survey design object immediately after creating it with the svydesign or svrepdesign function.

Value

Object of class "cvystat", which are vectors with a "var" attribute giving the variance and a "statistic" attribute giving the name of the statistic.

Author(s)

Djalma Pessoa and Anthony Damico

References

Guillaume Osier (2009). Variance estimation for complex indicators of poverty and inequality. *Journal of the European Survey Research Association*, Vol.3, No.3, pp. 167-195, ISSN 1864-3361, URL http://ojs.ub.uni-konstanz.de/srm/article/view/369.

Jean-Claude Deville (1999). Variance estimation for complex statistics and estimators: linearization and residual techniques. Survey Methodology, 25, 193-203, URL http://www5.statcan.gc.ca/bsolc/olc-cel/olc-cel?lang=eng&catno=12-001-X19990024882.

See Also

svyarpt

Examples

```
library(survey)
library(vardpoor)
data(eusilc) ; names( eusilc ) <- tolower( names( eusilc ) )

# missing completely at random, missingness rate = .20
ind_miss <- rbinom(nrow(eusilc), 1, .20 )
eusilc$eqincome_miss <- eusilc$eqincome
is.na(eusilc$eqincome_miss)<- ind_miss==1

# linearized design
des_eusilc <- svydesign( ids = ~rb030 , strata = ~db040 ,  weights = ~rb050 , data = eusilc )
des_eusilc <- convey_prep(des_eusilc)

svyrmir( ~eqincome , design = des_eusilc , age = ~age, med_old = TRUE )

# replicate-weighted design
des_eusilc_rep <- as.svrepdesign( des_eusilc , type = "bootstrap" )
des_eusilc_rep <- convey_prep(des_eusilc_rep)

svyrmir( ~eqincome , design = des_eusilc_rep, age= ~age, med_old = TRUE )

# linearized design using a variable with missings
svyrmir( ~ eqincome_miss , design = des_eusilc,age= ~age)
svyrmir( ~ eqincome_miss , design = des_eusilc , age= ~age, na.rm = TRUE )
# replicate-weighted design using a variable with missings
svyrmir( ~ eqincome_miss , design = des_eusilc_rep,age= ~age )
svyrmir( ~ eqincome_miss , design = des_eusilc_rep ,age= ~age, na.rm = TRUE )


# library(MonetDBLite) is only available on 64-bit machines,
# so do not run this block of code in 32-bit R
## Not run:

# database-backed design
library(MonetDBLite)
library(DBI)
dbfolder <- tempdir()
```

```
conn <- dbConnect( MonetDBLite::MonetDBLite() , dbfolder )
dbWriteTable( conn , 'eusilc' , eusilc )

dbd_eusilc <-
svydesign(
ids = ~rb030 ,
strata = ~db040 ,
weights = ~rb050 ,
data="eusilc",
dbname=dbfolder,
dbtype="MonetDBLite"
)

dbd_eusilc <- convey_prep( dbd_eusilc )

svyrmir( ~eqincome , design = dbd_eusilc , age = ~age )

dbRemoveTable( conn , 'eusilc' )


## End(Not run)
```

---

svyrmpg *Relative median poverty gap*

Description

Estimate the difference between the at-risk-of-poverty threshold (`arpt`) and the median of incomes less than the `arpt` relative to the `arpt`.

Usage

```
svyrmpg(formula, design, ...)

## S3 method for class 'survey.design'
svyrmpg(formula, design, order = 0.5, percent = 0.6,
  na.rm = FALSE, thresh = FALSE, poor_median = FALSE, ...)

## S3 method for class 'svyrep.design'
svyrmpg(formula, design, order = 0.5, percent = 0.6,
  na.rm = FALSE, thresh = FALSE, poor_median = FALSE, ...)

## S3 method for class 'DBIsvydesign'
svyrmpg(formula, design, ...)
```

Arguments

formula	a formula specifying the income variable
design	a design object of class survey.design or class svyrep.design from the survey library.
...	future expansion
order	income quantile order, usually .5
percent	fraction of the quantile, usually .60
na.rm	Should cases with missing values be dropped?
thresh	return the poverty poverty threshold
poor_median	return the median income of poor people

Details

you must run the convey_prep function on your survey design object immediately after creating it with the svydesign or svrepdesign function.

Value

Object of class "cvystat", which are vectors with a "var" attribute giving the variance and a "statistic" attribute giving the name of the statistic.

Author(s)

Djalma Pessoa and Anthony Damico

References

Guillaume Osier (2009). Variance estimation for complex indicators of poverty and inequality. *Journal of the European Survey Research Association*, Vol.3, No.3, pp. 167-195, ISSN 1864-3361, URL http://ojs.ub.uni-konstanz.de/srm/article/view/369.

Jean-Claude Deville (1999). Variance estimation for complex statistics and estimators: linearization and residual techniques. Survey Methodology, 25, 193-203, URL http://www5.statcan.gc.ca/bsolc/olc-cel/olc-cel?lang=eng&catno=12-001-X19990024882.

See Also

svyarpt

Examples

```
## Not run:
library(survey)
library(vardpoor)
data(eusilc) ; names( eusilc ) <- tolower( names( eusilc ) )

# linearized design
des_eusilc <- svydesign( ids = ~rb030 , strata = ~db040 ,  weights = ~rb050 , data = eusilc )
```

```
des_eusilc <- convey_prep( des_eusilc )

svyrmpg( ~eqincome , design = des_eusilc, thresh = TRUE )

# replicate-weighted design
des_eusilc_rep <- as.svrepdesign( des_eusilc , type = "bootstrap" )
des_eusilc_rep <- convey_prep( des_eusilc_rep )

svyrmpg( ~eqincome , design = des_eusilc_rep, thresh = TRUE )


# linearized design using a variable with missings
svyrmpg( ~ py010n , design = des_eusilc )
svyrmpg( ~ py010n , design = des_eusilc , na.rm = TRUE )
# replicate-weighted design using a variable with missings
svyrmpg( ~ py010n , design = des_eusilc_rep )
svyrmpg( ~ py010n , design = des_eusilc_rep , na.rm = TRUE )

# library(MonetDBLite) is only available on 64-bit machines,
# so do not run this block of code in 32-bit R

# database-backed design
library(MonetDBLite)
library(DBI)
dbfolder <- tempdir()
conn <- dbConnect( MonetDBLite::MonetDBLite() , dbfolder )
dbWriteTable( conn , 'eusilc' , eusilc )

dbd_eusilc <-
svydesign(
ids = ~rb030 ,
strata = ~db040 ,
weights = ~rb050 ,
data="eusilc",
dbname=dbfolder,
dbtype="MonetDBLite"
)

dbd_eusilc <- convey_prep( dbd_eusilc )

svyrmpg( ~ eqincome , design = dbd_eusilc )

dbRemoveTable( conn , 'eusilc' )


## End(Not run)
```

---

svyzenga *Zenga index*

Description

Estimate the Zenga index, a measure of inequality

Usage

```
svyzenga(formula, design, ...)

## S3 method for class 'survey.design'
svyzenga(formula, design, na.rm = FALSE, ...)

## S3 method for class 'svyrep.design'
svyzenga(formula, design, na.rm = FALSE, ...)

## S3 method for class 'DBIsvydesign'
svyzenga(formula, design, ...)
```

Arguments

formula	a formula specifying the income variable.
design	a design object of class survey.design or class svyrep.design from the survey library.
...	future expansion
na.rm	Should cases with missing values be dropped?

Details

you must run the convey_prep function on your survey design object immediately after creating it with the svydesign or svrepdesign function.

Value

Object of class "cvystat", which are vectors with a "var" attribute giving the variance and a "statistic" attribute giving the name of the statistic.

Author(s)

Guilherme Jacob, Djalma Pessoa and Anthony Damico

References

Matti Langel (2012). Measuring inequality in finite population sampling. PhD thesis: Universite de Neuchatel, URL https://doc.rero.ch/record/29204/files/00002252.pdf.

See Also

svygini

Examples

```
## Not run:
library(survey)
library(vardpoor)
data(eusilc) ; names( eusilc ) <- tolower( names( eusilc ) )

# linearized design
des_eusilc <- svydesign( ids = ~rb030 , strata = ~db040 ,  weights = ~rb050 , data = eusilc )
des_eusilc <- convey_prep(des_eusilc)

# replicate-weighted design
des_eusilc_rep <- as.svrepdesign( des_eusilc , type = "bootstrap" )
des_eusilc_rep <- convey_prep(des_eusilc_rep)

# subset all designs to positive income and non-missing records only
des_eusilc_pos_inc <- subset( des_eusilc , eqincome > 0 )
des_eusilc_rep_pos_inc <- subset( des_eusilc_rep , eqincome > 0 )


# variable without missing values
svyzenga(~eqincome, des_eusilc_pos_inc)
svyzenga(~eqincome, des_eusilc_rep_pos_inc)

# subsetting:
svyzenga(~eqincome, subset( des_eusilc_pos_inc, db040 == "Styria"))
svyzenga(~eqincome, subset( des_eusilc_rep_pos_inc, db040 == "Styria"))

# variable with with missings (but subsetted to remove negatives)
# svyzenga(~py010n, subset( des_eusilc, py010n > 0 | is.na(py010n)) )
# svyzenga(~py010n, subset( des_eusilc_rep, py010n > 0 | is.na(py010n)) )

# svyzenga(~py010n, subset( des_eusilc, py010n > 0 | is.na(py010n)), na.rm = TRUE)
# svyzenga(~py010n, subset( des_eusilc_rep, py010n > 0 | is.na(py010n)), na.rm = TRUE)


# library(MonetDBLite) is only available on 64-bit machines,
# so do not run this block of code in 32-bit R

# database-backed design
library(MonetDBLite)
library(DBI)
dbfolder <- tempdir()
conn <- dbConnect( MonetDBLite::MonetDBLite() , dbfolder )
dbWriteTable( conn , 'eusilc' , eusilc )

dbd_eusilc <-
svydesign(
ids = ~rb030 ,
strata = ~db040 ,
weights = ~rb050 ,
data="eusilc",
dbname=dbfolder,
```

```
dbtype="MonetDBLite"
)

dbd_eusilc <- convey_prep( dbd_eusilc )


# subset all designs to positive income and non-missing records only
dbd_eusilc_pos_inc <- subset( dbd_eusilc , eqincome > 0 )

# variable without missing values
svyzenga(~eqincome, dbd_eusilc_pos_inc)

# subsetting:
svyzenga(~eqincome, subset( dbd_eusilc_pos_inc, db040 == "Styria"))

# variable with with missings (but subsetted to remove negatives)
# svyzenga(~py010n, subset( dbd_eusilc, py010n > 0 | is.na(py010n)) )

# svyzenga(~py010n, subset( dbd_eusilc, py010n > 0 | is.na(py010n)), na.rm = TRUE)

dbRemoveTable( conn , 'eusilc' )


## End(Not run)
```

Index