# Package ‘convey’

May 23, 2020

**Title** Income Concentration Analysis with Complex Survey Samples

**Version** 0.2.2

**URL** `https://guilhermejacob.github.io/context/`

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

**Description** Variance estimation on indicators of income concentration and
poverty using complex sample 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, laeken, DBI,
RODBC, RSQLite, srvyr

**RoxygenNote** 7.1.0

**NeedsCompilation** no

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

**Maintainer** Djalma Pessoa <`pessoad@gmail.com`>

**Repository** CRAN

**Date/Publication** 2020-05-22 22:40:11 UTC

## 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(laeken)
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(laeken)
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(laeken)
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(laeken)
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 )
```

---

svyafc *Alkire-Foster multidimensional poverty class*

---

## Description

Estimate indices from the Alkire-Foster class, a class of poverty measures.

## Usage

```
svyafc(formula, design, ...)

## S3 method for class 'survey.design'
svyafc(formula, design, k, g, cutoffs, dimw = NULL, na.rm = FALSE, ...)

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

```
svyafc(formula, design, k, g, cutoffs, dimw = NULL, na.rm = FALSE, ...)

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

## Arguments

| | |
|---|---|
| `formula` | a formula specifying the variables. Variables can be numeric or ordered factors. |
| `design` | a design object of class `survey.design` or class `svyrep.design` from the `survey` library. |
| `...` | future expansion |
| `k` | a scalar defining the multidimensional cutoff. |
| `g` | a scalar defining the exponent of the indicator. |
| `cutoffs` | a list defining each variable's deprivation limit. |
| `dimw` | a vector defining the weight of each dimension in the multidimensional deprivation sum. |
| `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

Sabina Alkire and James Foster (2011). Counting and multidimensional poverty measurement. Journal of Public Economics, v. 95, n. 7-8, August 2011, pp. 476-487, ISSN 0047-2727. URL `http://dx.doi.org/10.1016/j.jpubeco.2010.11.006`.

Alkire et al. (2015). Multidimensional Poverty Measurement and Analysis. Oxford University Press, 2015.

Daniele Pacifico and Felix Poege (2016). MPI: Stata module to compute the Alkire-Foster multidimensional poverty measures and their decomposition by deprivation indicators and population sub-groups. URL `http://EconPapers.repec.org/RePEc:boc:bocode:s458120`.

## See Also

`svyfgt`

## Examples

```
library(survey)
library(laeken)
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)
des_eusilc <- update(des_eusilc, pb220a = ordered( pb220a ) )

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

# cutoffs
cos <- list( 10000, 5000 )

# variables without missing values
svyafc( ~ eqincome + hy050n , design = des_eusilc , k = .5 , g = 0, cutoffs = cos )
svyafc( ~ eqincome + hy050n , design = des_eusilc_rep , k = .5 , g = 0, cutoffs = cos )

# subsetting:
sub_des_eusilc <- subset( des_eusilc, db040 == "Styria")
sub_des_eusilc_rep <- subset( des_eusilc_rep, db040 == "Styria")

svyafc( ~ eqincome + hy050n , design = sub_des_eusilc , k = .5, g = 0, cutoffs = cos )
svyafc( ~ eqincome + hy050n , design = sub_des_eusilc_rep , k = .5, g = 0, cutoffs = cos )

## Not run:

# including factor variable with missings
cos <- list( 10000, 5000, "EU" )
svyafc(~eqincome+hy050n+pb220a, des_eusilc, k = .5, g = 0, cutoffs = cos , na.rm = FALSE )
svyafc(~eqincome+hy050n+pb220a, des_eusilc, k = .5, g = 0, cutoffs = cos , na.rm = TRUE )
svyafc(~eqincome+hy050n+pb220a, des_eusilc_rep, k = .5, g = 0, cutoffs = cos , na.rm = FALSE )
svyafc(~eqincome+hy050n+pb220a, des_eusilc_rep, k = .5, g = 0, cutoffs = cos , na.rm = TRUE )

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

dbd_eusilc <- convey_prep( dbd_eusilc )
dbd_eusilc <- update( dbd_eusilc, pb220a = ordered( pb220a ) )

# cutoffs
cos <- list( 10000 , 5000 )

# variables without missing values
svyafc(~eqincome+hy050n, design = dbd_eusilc, k = .5, g = 0, cutoffs = cos )

# subsetting:
sub_dbd_eusilc <- subset( dbd_eusilc, db040 == "Styria")
svyafc(~eqincome+hy050n, design = sub_dbd_eusilc, k = .5, g = 0, cutoffs = cos )

# cutoffs
cos <- list( 10000, 5000, "EU" )

# including factor variable with missings
svyafc(~eqincome+hy050n+pb220a, dbd_eusilc, k = .5, g = 0, cutoffs = cos , na.rm = FALSE )
svyafc(~eqincome+hy050n+pb220a, dbd_eusilc, k = .5, g = 0, cutoffs = cos , na.rm = TRUE )

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## End(Not run)
```

## svyafcdec *Alkire-Foster multidimensional poverty decompostition*

### Description

Decomposition of indices from the Alkire-Foster class

### Usage

```
svyafcdec(formula, subgroup, design, ...)

## S3 method for class 'survey.design'
svyafcdec(
  formula,
  subgroup = ~1,
  design,
  g,
  cutoffs,
  k,
  dimw = NULL,
  na.rm = FALSE,
  ...
)

## S3 method for class 'svyrep.design'
svyafcdec(
  formula,
  subgroup = ~1,
  design,
  g,
  cutoffs,
  k,
  dimw = NULL,
  na.rm = FALSE,
  ...
)

## S3 method for class 'DBIsvydesign'
svyafcdec(formula, subgroup = ~1, design, ...)
```

## Arguments

| | |
|---|---|
| `formula` | a formula specifying the variables. Variables can be numeric or ordered factors. |
| `subgroup` | a formula defining the group variable for decomposition. |
| `design` | a design object of class `survey.design` or class `svyrep.design` from the `survey` library. |
| `...` | future expansion |
| `g` | a scalar defining the exponent of the indicator. |
| `cutoffs` | a list defining each variable's deprivation limit. |
| `k` | a scalar defining the multidimensional cutoff. |
| `dimw` | a vector defining the weight of each dimension in the multidimensional deprivation sum. |
| `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

Sabina Alkire and James Foster (2011). Counting and multidimensional poverty measurement. Journal of Public Economics, v. 95, n. 7-8, August 2011, pp. 476-487, ISSN 0047-2727. URL http://dx.doi.org/10.1016/j.jpubeco.2010.11.006.

Alkire et al. (2015). Multidimensional Poverty Measurement and Analysis. Oxford University Press, 2015.

Daniele Pacifico and Felix Poege (2016). MPI: Stata module to compute the Alkire-Foster multidimensional poverty measures and their decomposition groups deprivation indicators and population sub-groups. URL http://EconPapers.repec.org/RePEc:boc:bocode:s458120.

## See Also

svyfgt

## Examples

```
library(survey)
library(laeken)
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)
des_eusilc <- update(des_eusilc, pb220a = ordered( pb220a ) )

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

# cutoffs
cos <- list( 10000, 5000 )


# variables without missing values
svyafcdec( ~ eqincome + hy050n ,  ~1 , des_eusilc , k = .5 , g = 0, cutoffs = cos )
svyafcdec( ~ eqincome + hy050n ,  ~1 , des_eusilc_rep , k = .5 , g = 0, cutoffs = cos )

svyafcdec( ~ eqincome + hy050n ,  ~rb090 , des_eusilc , k = .5 , g = 0, cutoffs = cos )
svyafcdec( ~ eqincome + hy050n ,  ~rb090 , des_eusilc_rep , k = .5 , g = 0, cutoffs = cos )

# subsetting:
sub_des_eusilc <- subset( des_eusilc, db040 == "Styria")
sub_des_eusilc_rep <- subset( des_eusilc_rep, db040 == "Styria")

svyafcdec( ~ eqincome + hy050n ,  ~1 , sub_des_eusilc , k = .5 , g = 0, cutoffs = cos )
svyafcdec( ~ eqincome + hy050n ,  ~1 , sub_des_eusilc_rep , k = .5 , g = 0, cutoffs = cos )

svyafcdec( ~ eqincome + hy050n ,  ~rb090 , sub_des_eusilc ,
k = .5 , g = 0, cutoffs = cos )
svyafcdec( ~ eqincome + hy050n ,  ~rb090 , sub_des_eusilc_rep ,
k = .5 , g = 0, cutoffs = cos )
```

```
## Not run:

# including factor variable with missings
cos <- list( 10000, 5000, "EU" )
svyafcdec(~eqincome+hy050n+pb220a,  ~1 , des_eusilc,
k = .5, g = 0, cutoffs = cos , na.rm = FALSE )
svyafcdec(~eqincome+hy050n+pb220a,  ~1 , des_eusilc,
k = .5, g = 0, cutoffs = cos , na.rm = TRUE )
svyafcdec(~eqincome+hy050n+pb220a,  ~1 , des_eusilc_rep,
k = .5, g = 0, cutoffs = cos , na.rm = FALSE )
svyafcdec(~eqincome+hy050n+pb220a,  ~1 , des_eusilc_rep,
k = .5, g = 0, cutoffs = cos , na.rm = TRUE )

svyafcdec(~eqincome+hy050n+pb220a,  ~rb090 , des_eusilc,
k = .5, g = 0, cutoffs = cos , na.rm = FALSE )
svyafcdec(~eqincome+hy050n+pb220a,  ~rb090 , des_eusilc,
k = .5, g = 0, cutoffs = cos , na.rm = TRUE )
svyafcdec(~eqincome+hy050n+pb220a,  ~rb090 , des_eusilc_rep,
k = .5, g = 0, cutoffs = cos , na.rm = FALSE )
svyafcdec(~eqincome+hy050n+pb220a,  ~rb090 , des_eusilc_rep,
k = .5, g = 0, cutoffs = cos , na.rm = TRUE )

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

dbd_eusilc <- convey_prep( dbd_eusilc )
dbd_eusilc <- update( dbd_eusilc, pb220a = ordered( pb220a ) )

# cutoffs
cos <- list( 10000 , 5000 )

# variables without missing values
svyafcdec( ~eqincome+hy050n ,  ~1 , des_eusilc ,
k = .5 , g = 0, cutoffs = cos )
svyafcdec( ~eqincome+hy050n ,  ~rb090 , des_eusilc ,
k = .5 , g = 0, cutoffs = cos )

# subsetting:
```

```
sub_des_eusilc <- subset( des_eusilc, db040 == "Styria")

svyafcdec( ~ eqincome + hy050n ,  ~1 , sub_des_eusilc ,
k = .5 , g = 0, cutoffs = cos )

svyafcdec( ~ eqincome + hy050n ,  ~rb090 , sub_des_eusilc ,
k = .5 , g = 0, cutoffs = cos )

# including factor variable with missings
cos <- list( 10000, 5000, "EU" )

svyafcdec(~eqincome+hy050n+pb220a,  ~1 , dbd_eusilc,
k = .5, g = 0, cutoffs = cos , na.rm = FALSE )
svyafcdec(~eqincome+hy050n+pb220a,  ~1 , dbd_eusilc,
k = .5, g = 0, cutoffs = cos , na.rm = TRUE )

svyafcdec(~eqincome+hy050n+pb220a,  ~rb090 , dbd_eusilc,
k = .5, g = 0, cutoffs = cos , na.rm = FALSE )
svyafcdec(~eqincome+hy050n+pb220a,  ~rb090 , dbd_eusilc,
k = .5, g = 0, cutoffs = cos , na.rm = TRUE )

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## End(Not run)
```

---

`svyamato` *Amato index (EXPERIMENTAL)*

---

## Description

Estimate the Amato index, a measure of inequality.

## Usage

```
svyamato(formula, design, ...)

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

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

## S3 method for class 'DBIsvydesign'
svyamato(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 |
| `standardized` | If `standardized = TRUE`, returns the standardized Amato index, i.e., a linear tranformation of the amato index. |
| `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.

The Amato index is the length of the Lorenz curve.

## 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.

## Note

This function is experimental and is subject to change in later versions.

## Author(s)

Guilherme Jacob, Djalma Pessoa and Anthony Damico

## References

Lucio Barabesi, Giancarlo Diana and Pier Francesco Perri (2016). Linearization of inequality indexes in the design-based framework. Statistics. URL `http://www.tandfonline.com/doi/pdf/10.1080/02331888.2015.1135924`.

Barry C. Arnold (2012). On the Amato inequality index. Statistics & Probability Letters, v. 82, n. 8, August 2012, pp. 1504-1506, ISSN 0167-7152. URL `http://dx.doi.org/10.1016/j.spl.2012.04.020`.

## See Also

`svygini`

## Examples

```
library(survey)
library(laeken)
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)


# variable without missing values
svyamato(~eqincome, des_eusilc)
svyamato(~eqincome, des_eusilc_rep)

# subsetting:
svyamato(~eqincome, subset( des_eusilc, db040 == "Styria"))
svyamato(~eqincome, subset( des_eusilc_rep, db040 == "Styria"))

## Not run:

# variable with with missings
svyamato(~py010n, des_eusilc )
svyamato(~py010n, des_eusilc_rep )

svyamato(~py010n, des_eusilc, na.rm = TRUE )
svyamato(~py010n, des_eusilc_rep, na.rm = TRUE )

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

dbd_eusilc <- convey_prep( dbd_eusilc )


# variable without missing values
svyamato(~eqincome, dbd_eusilc)

# subsetting:
svyamato(~eqincome, subset( dbd_eusilc, db040 == "Styria"))

# variable with with missings
svyamato(~py010n, dbd_eusilc )
svyamato(~py010n, dbd_eusilc, na.rm = TRUE )


dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## End(Not run)
```

---

## 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, quantiles = 0.5, percent = 0.6, na.rm = FALSE, ...)

## S3 method for class 'svyrep.design'
svyarpr(formula, design, quantiles = 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. |
| `...` | arguments passed on to ‘svyarpt‘ |
| `quantiles` | income quantile, 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

`svyarpt`

## Examples

```
library(survey)
library(laeken)
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 )

## Not run:

# 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 )

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

dbd_eusilc <- convey_prep( dbd_eusilc )

svyarpr( ~ eqincome , design = dbd_eusilc )

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## 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, quantiles = 0.5, percent = 0.6, na.rm = FALSE, ...)

## S3 method for class 'svyrep.design'
svyarpt(formula, design, quantiles = 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. |
| ... | arguments passed on to ‘survey::svyquantile‘ |
| quantiles | income quantile quantiles, 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(laeken)
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 )

## Not run:

# 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 )


# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

dbd_eusilc <- convey_prep( dbd_eusilc )

svyarpt( ~ eqincome , design = dbd_eusilc )

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## 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(laeken)
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 )

## Not run:

# 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 )


# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

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' )

dbDisconnect( conn , shutdown = TRUE )


## End(Not run)
```

## svybcc    *Bourguignon-Chakravarty multidimensional poverty class (EXPERIMENTAL)*

## Description

Estimate indices from the Bourguignon-Chakravarty class, a class of poverty measures.

## Usage

```
svybcc(formula, design, ...)

## S3 method for class 'survey.design'
svybcc(
  formula,
  design,
  theta = 0.5,
  alpha = 0.5,
  cutoffs,
  dimw = NULL,
  na.rm = FALSE,
  ...
)

## S3 method for class 'svyrep.design'
svybcc(
  formula,
  design,
  theta = 0.5,
  alpha = 0.5,
  cutoffs,
  dimw = NULL,
  na.rm = FALSE,
  ...
)

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

## Arguments

| | |
|---|---|
| `formula` | a formula specifying the variables. Variables can be numeric or ordered factors. |
| `design` | a design object of class `survey.design` or class `svyrep.design` from the `survey` library. |
| `...` | future expansion |
| `theta` | a scalar defining the elasticity of substitution between the normalized gaps of the attributes. |
| `alpha` | a scalar that can be interpreted as the society's aversion to poverty. |
| `cutoffs` | a list defining each variable's deprivation limit. |
| `dimw` | a vector defining the weight of each dimension in the multidimensional deprivation sum. |
| `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.

## Note

This function is experimental and is subject to changes in later versions.

## Author(s)

Guilherme Jacob, Djalma Pessoa and Anthony Damico

## References

Francois Bourguignon and Satya R. Chakravarty (2003). The measurement of multidimensional poverty. Journal of Economic Inequality, v. 1, n. 1, April 2003, pp. 1-25. URL http://dx.doi.org/10.1023/A:1023913831342.

Maria Casilda Lasso de la Vega, Ana Urrutia and Henar Diez (2009). The Bourguignon and Chakravarty multidimensional poverty family: a characterization. Working Papers 109, ECINEQ, Society for the Study of Economic Inequality.

## See Also

svyafc

## Examples

```
library(survey)
library(laeken)
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)
des_eusilc <- update(des_eusilc, pb220a = ordered( pb220a ) )

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

# cutoffs
cos <- list( 10000, 5000 )

# variables without missing values
svybcc( ~ eqincome + hy050n , design = des_eusilc , cutoffs = cos )
svybcc( ~ eqincome + hy050n , design = des_eusilc_rep , cutoffs = cos )

# subsetting:
sub_des_eusilc <- subset( des_eusilc, db040 == "Styria")
sub_des_eusilc_rep <- subset( des_eusilc_rep, db040 == "Styria")

svybcc( ~ eqincome + hy050n , design = sub_des_eusilc , cutoffs = cos )
svybcc( ~ eqincome + hy050n , design = sub_des_eusilc_rep , cutoffs = cos )

## Not run:

# including factor variable with missings
cos <- list( 10000, 5000, "EU" )
svybcc(~eqincome+hy050n+pb220a, des_eusilc, cutoffs = cos, na.rm = FALSE )
svybcc(~eqincome+hy050n+pb220a, des_eusilc, cutoffs = cos, na.rm = TRUE )
svybcc(~eqincome+hy050n+pb220a, des_eusilc_rep, cutoffs = cos, na.rm = FALSE )
svybcc(~eqincome+hy050n+pb220a, des_eusilc_rep, cutoffs = cos, na.rm = TRUE )

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

dbd_eusilc <- convey_prep( dbd_eusilc )
dbd_eusilc <- update( dbd_eusilc, pb220a = ordered( pb220a ) )

# cutoffs
cos <- list( 10000 , 5000 )

# variables without missing values
svybcc(~eqincome+hy050n, design = dbd_eusilc, cutoffs = cos )

# subsetting:
sub_dbd_eusilc <- subset( dbd_eusilc, db040 == "Styria")
svybcc(~eqincome+hy050n, design = sub_dbd_eusilc, cutoffs = cos )

# cutoffs
cos <- list( 10000, 5000, "EU" )

# including factor variable with missings
svybcc(~eqincome+hy050n+pb220a, dbd_eusilc, cutoffs = cos, na.rm = FALSE )
svybcc(~eqincome+hy050n+pb220a, dbd_eusilc, cutoffs = cos, na.rm = TRUE )

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## End(Not run)
```

---

svybmi *Bourguignon (1999) multidimensional inequality indices (EXPERIMENTAL)*

---

## Description

Estimate indices from the Bourguignon (1999) class, a class of multidimensional inequality measures.

## Usage

```
svybmi(formula, design, ...)

## S3 method for class 'survey.design'
svybmi(
  formula,
  design,
  alpha = 0.5,
  beta = -2,
  dimw = NULL,
  na.rm = FALSE,
  ...
)

## S3 method for class 'svyrep.design'
svybmi(
  formula,
  design,
  alpha = 0.5,
  beta = -2,
  dimw = NULL,
  na.rm = FALSE,
  ...
)

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

## Arguments

| | |
|---|---|
| `formula` | a formula specifying the variables. Variables can be numeric or ordered factors. |
| `design` | a design object of class `survey.design` or class `svyrep.design` from the `survey` library. |
| `...` | future expansion |
| `alpha` | a scalar defining the exponent of the indicator. |
| `beta` | a scalar defining the exponent of the indicator. |
| `dimw` | a vector defining the weight of each dimension in the multidimensional deprivation sum. |
| `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.

## Note

This function is experimental and is subject to change in later versions.

## Author(s)

Guilherme Jacob, Djalma Pessoa and Anthony Damico

## References

Francois Bourguignon (1999). Comment to 'Multidimensioned Approaches to Welfare Analysis' by Maasoumi, E. In: Handbook of income inequality measurement., ed. J. Silber, Boston, Dordrecht and London: Kluwer Academic, p. 477-484.

Maria Ana Lugo (2007). Comparing multidimensional indices of inequality: Methods and application. In: Research on Economic Inequality. Emerald, p. 213-236.

## See Also

`svyfgt`

## Examples

```
library(survey)
library(laeken)
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)
des_eusilc <- update(des_eusilc, pb220a = ordered( pb220a ) )

# replicate-weighted design
des_eusilc_rep <- as.svrepdesign( des_eusilc , type = "bootstrap", replicates = 50 )
des_eusilc_rep <- convey_prep(des_eusilc_rep)

# linearized
svybmi(~eqincome+hy050n, design = des_eusilc, alpha = .5, beta = .5, na.rm = FALSE )
svybmi(~eqincome+hy050n, design = des_eusilc, alpha = .5, beta = 0, na.rm = FALSE )
svybmi(~eqincome+hy050n, design = des_eusilc, alpha = .5, beta = -.5, na.rm = FALSE )
svybmi(~eqincome+hy050n, design = des_eusilc, alpha = .5, beta = -1, na.rm = FALSE )
svybmi(~eqincome+hy050n, design = des_eusilc, alpha = .5, beta = -2, na.rm = FALSE )

# replicate
svybmi(~eqincome+hy050n, design = des_eusilc_rep, alpha = .5, beta = .5, na.rm = FALSE )
svybmi(~eqincome+hy050n, design = des_eusilc_rep, alpha = .5, beta = 0, na.rm = FALSE )
svybmi(~eqincome+hy050n, design = des_eusilc_rep, alpha = .5, beta = -.5, na.rm = FALSE )
svybmi(~eqincome+hy050n, design = des_eusilc_rep, alpha = .5, beta = -1, na.rm = FALSE )
svybmi(~eqincome+hy050n, design = des_eusilc_rep, alpha = .5, beta = -2, na.rm = FALSE )

## Not run:

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

dbd_eusilc <- convey_prep( dbd_eusilc )

# linearized
svybmi(~eqincome+hy050n, design = dbd_eusilc, alpha = .5, beta = .5, na.rm = FALSE )
svybmi(~eqincome+hy050n, design = dbd_eusilc, alpha = .5, beta = 0, na.rm = FALSE )
svybmi(~eqincome+hy050n, design = dbd_eusilc, alpha = .5, beta = -.5, na.rm = FALSE )
svybmi(~eqincome+hy050n, design = dbd_eusilc, alpha = .5, beta = -1, na.rm = FALSE )
svybmi(~eqincome+hy050n, design = dbd_eusilc, alpha = .5, beta = -2, na.rm = FALSE )

# subsetting:
sub_dbd_eusilc <- subset( dbd_eusilc, db040 == "Styria")
svybmi(~eqincome+hy050n, design = sub_dbd_eusilc, alpha = .5, beta = .5, na.rm = FALSE )
```

```
dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## End(Not run)
```

---

svychu *CHU class of poverty measures (EXPERIMENTAL)*

---

## Description

Estimate the Clark, Hemming and Ulph (1981) class of poverty measures

## Usage

```
svychu(formula, design, ...)

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

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

```
## S3 method for class 'DBIsvydesign'
svychu(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` | A parameter where (1 - g) defines the inequality aversion among the poor. If g = 0, the CHU class becomes a monotonic transform of the Watts poverty measure. |
| `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 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 |
| `quantiles` | the quantile 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.

## Note

This function is experimental and is subject to change in later versions.

## Author(s)

Guilherme Jacob, Djalma Pessoa and Anthony Damico

## References

Vijay Verma and Gianni Betti (2011). Taylor linearization sampling errors and design effects for poverty measures and other complex statistics. *Journal Of Applied Statistics*, Vol.38, No.8, pp. 1549-1576, URL `http://dx.doi.org/10.1080/02664763.2010.515674`.

Anthony B. Atkinson (1987). On the measurement of poverty. *Econometrica*, Vol.55, No.4, (Jul., 1987), pp. 749-764, URL `http://www.jstor.org/stable/1911028`.

Stephen Clark, Richard Hemming and David Ulph (1981). On Indices for the Measurement of Poverty. *The Economic Journal*, Vol.91, No.362, (Jun., 1981), pp. 515-526, URL http://www.jstor.org/stable/2232600.

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

svywatts

## Examples

```
library(survey)
library(laeken)
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 )

# absolute poverty threshold
svychu(~eqincome, des_eusilc, g=1,  abs_thresh=10000)
# poverty threshold equal to arpt
svychu(~eqincome, des_eusilc, g=1, type_thresh= "relq" , thresh = TRUE)
# poverty threshold equal to 0.6 times the mean
svychu(~eqincome, des_eusilc, g=1, type_thresh= "relm" , thresh = TRUE)

#  using svrep.design:
# absolute poverty threshold
svychu(~eqincome, des_eusilc_rep, g=1,  abs_thresh=10000)
# poverty threshold equal to arpt
svychu(~eqincome, des_eusilc_rep, g=1, type_thresh= "relq" , thresh = TRUE)
# poverty threshold equal to 0.6 times the mean
svychu(~eqincome, des_eusilc_rep, g=1, type_thresh= "relm" , thresh = TRUE)

## Not run:

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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


dbd_eusilc <- convey_prep( dbd_eusilc )

# absolute poverty threshold
svychu(~eqincome, dbd_eusilc, g=1,  abs_thresh=10000)
# poverty threshold equal to arpt
svychu(~eqincome, dbd_eusilc, g=1, type_thresh= "relq" , thresh = TRUE)
# poverty threshold equal to 0.6 times the mean
svychu(~eqincome, dbd_eusilc, g=1, type_thresh= "relm" , thresh = TRUE)

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## 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,
  quantiles = 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,
  quantiles = 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 g=0 estimates the headcount ratio; If g=1 estimates the average normalised poverty gap, and if g=2 estimates the average squared normalised poverty gap |
| `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 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 |
| `quantiles` | the quantile 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

James Foster, Joel Greer and Erik Thorbecke (1984). A class of decomposable poverty measures. *Econometrica*, Vol.52, No.3, pp. 761-766.

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(laeken)
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)

## Not run:

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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


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' )

dbDisconnect( conn , shutdown = TRUE )


## End(Not run)
```

svyfgtdec *FGT indices decomposition (EXPERIMENTAL)*

## Description

Estimate the Foster et al. (1984) poverty class and its components

## Usage

```
svyfgtdec(formula, design, ...)

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

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

## S3 method for class 'DBIsvydesign'
svyfgtdec(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. Currently not used. |
| `g` | If g=2 estimates the average squared normalised poverty gap. This function is defined for g >= 2 only, |
| `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 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 |
| `quantiles` | the quantile 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 "`cvydstat`", with estimates for the FGT(g), FGT(0), FGT(1), income gap ratio and GEI(income gaps; epsilon = g) with a "`var`" attribute giving the variance-covariance matrix. A "`statistic`" attribute giving the name of the statistic.

## Note

This function is experimental and is subject to change in later versions.

## Author(s)

Guilherme Jacob, Djalma Pessoa and Anthony Damico

## References

Oihana Aristondo, Cassilda Lasso De La vega and Ana Urrutia (2010). A new multiplicative decomposition for the Foster-Greer-Thorbecke poverty indices. *Bulletin of Economic Research*, Vol.62, No.3, pp. 259-267. University of Wisconsin. URL `http://dx.doi.org/10.1111/j.1467-8586.2009.00320.x`.

James Foster, Joel Greer and Erik Thorbecke (1984). A class of decomposable poverty measures. *Econometrica*, Vol.52, No.3, pp. 761-766.

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

svyfgt,svyfgt,svyfgt

## Examples

```
library(survey)
library(laeken)
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 )

# absolute poverty threshold
svyfgtdec(~eqincome, des_eusilc, g=2, abs_thresh=10000)
# poverty threshold equal to arpt
svywattsdec(~eqincome, des_eusilc, g=2, type_thresh= "relq" , thresh = TRUE)
# poverty threshold equal to 0.6 times the mean
svywattsdec(~eqincome, des_eusilc, g=2, type_thresh= "relm" , thresh = TRUE)

# using svrep.design:
# absolute poverty threshold
svyfgtdec(~eqincome, des_eusilc_rep, g=2, abs_thresh=10000)
# poverty threshold equal to arpt
svywattsdec(~eqincome, des_eusilc_rep, g=2, type_thresh= "relq" , thresh = TRUE)
# poverty threshold equal to 0.6 times the mean
svywattsdec(~eqincome, des_eusilc_rep, g=2, type_thresh= "relm" , thresh = TRUE)

## Not run:

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

```
dbd_eusilc <- convey_prep( dbd_eusilc )


# absolute poverty threshold
svyfgtdec(~eqincome, dbd_eusilc, g=2, abs_thresh=10000)
# poverty threshold equal to arpt
svywattsdec(~eqincome, dbd_eusilc, g=2, type_thresh= "relq" , thresh = TRUE)
# poverty threshold equal to 0.6 times the mean
svywattsdec(~eqincome, dbd_eusilc, g=2, type_thresh= "relm" , thresh = TRUE)

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## 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.

This measure 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(laeken)
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 , subset(des_eusilc, eqincome > 0), epsilon = .5 )
svygei( ~eqincome , subset(des_eusilc, eqincome > 0), epsilon = 1 )
svygei( ~eqincome , subset(des_eusilc, eqincome > 0), epsilon = 2 )

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

## Not run:

# 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 , subset(des_eusilc, py010n > 0 | is.na(py010n) ), epsilon = .5 )
svygei( ~py010n , subset(des_eusilc, py010n > 0 | is.na(py010n) ), 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 , subset(des_eusilc, py010n > 0 | is.na(py010n) ), epsilon = 2 )
svygei( ~py010n , subset(des_eusilc, py010n > 0 | is.na(py010n) ), 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 , subset(des_eusilc_rep, py010n > 0 | is.na(py010n) ), epsilon = .5 )
svygei( ~py010n , subset(des_eusilc_rep, py010n > 0 | is.na(py010n) ), 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 , subset(des_eusilc_rep, py010n > 0 | is.na(py010n) ), epsilon = 2 )
svygei( ~py010n , subset(des_eusilc_rep, py010n > 0 | is.na(py010n) ), epsilon = 2, na.rm = TRUE )

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

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' )

dbDisconnect( conn , shutdown = TRUE )


## End(Not run)
```

## svygeidec *Generalized entropy index Decomposition*

### Description

Estimates the group decomposition of the generalized entropy index

### Usage

```
svygeidec(formula, subgroup, design, ...)

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

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

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

### Arguments

| | |
|---|---|
| `formula` | a formula specifying the income variable |
| `subgroup` | a formula specifying the group 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? Observations containing missing values in income or group variables will 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.

This measure only allows for strictly positive variables.

## Value

Object of class "cvydstat", which are vectors with a "var" attribute giving the variance-covariance matrix and a "statistic" attribute giving the name of the statistic.

## Author(s)

Guilherme Jacob, Djalma Pessoa and Anthony Damico

## References

Anthony F. Shorrocks (1984). Inequality decomposition groups population subgroups. *Econometrica*, v. 52, n. 6, 1984, pp. 1369-1385. URL http://www.jstor.org/stable/1913511.

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(laeken)
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
svygeidec( ~eqincome , ~rb090 , subset( des_eusilc, eqincome > 0 ) , epsilon = 0 )
svygeidec( ~eqincome , ~rb090 , subset( des_eusilc, eqincome > 0 ) , epsilon = .5 )
svygeidec( ~eqincome , ~rb090 , subset( des_eusilc, eqincome > 0 ) , epsilon = 1 )
svygeidec( ~eqincome , ~rb090 , subset( des_eusilc, eqincome > 0 ) , epsilon = 2 )

# replicate-weighted design
svygeidec( ~eqincome , ~rb090 , subset( des_eusilc_rep, eqincome > 0 ) , epsilon = 0 )
svygeidec( ~eqincome , ~rb090 , subset( des_eusilc_rep, eqincome > 0 ) , epsilon = .5 )
svygeidec( ~eqincome , ~rb090 , subset( des_eusilc_rep, eqincome > 0 ) , epsilon = 1 )
svygeidec( ~eqincome , ~rb090 , subset( des_eusilc_rep, eqincome > 0 ) , epsilon = 2 )

## Not run:

# linearized design using a variable with missings
sub_des_eusilc <- subset(des_eusilc, py010n > 0 | is.na(py010n) )
svygeidec( ~py010n , ~rb090 , sub_des_eusilc , epsilon = 0 )
svygeidec( ~py010n , ~rb090 , sub_des_eusilc , epsilon = 0, na.rm = TRUE )
svygeidec( ~py010n , ~rb090 , sub_des_eusilc , epsilon = 1 )
svygeidec( ~py010n , ~rb090 , sub_des_eusilc , epsilon = 1, na.rm = TRUE )

# replicate-weighted design using a variable with missings
sub_des_eusilc_rep <- subset(des_eusilc_rep, py010n > 0 | is.na(py010n) )
svygeidec( ~py010n , ~rb090 , sub_des_eusilc_rep , epsilon = 0 )
svygeidec( ~py010n , ~rb090 , sub_des_eusilc_rep , epsilon = 0, na.rm = TRUE )
svygeidec( ~py010n , ~rb090 , sub_des_eusilc_rep , epsilon = 1 )
svygeidec( ~py010n , ~rb090 , sub_des_eusilc_rep , epsilon = 1, na.rm = TRUE )

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

dbd_eusilc <- convey_prep( dbd_eusilc )

# database-backed linearized design
svygeidec( ~eqincome , ~rb090 , subset(dbd_eusilc, eqincome > 0) , epsilon = 0 )
svygeidec( ~eqincome , ~rb090 , subset(dbd_eusilc, eqincome > 0) , epsilon = .5 )
svygeidec( ~eqincome , ~rb090 , subset(dbd_eusilc, eqincome > 0) , epsilon = 1 )
svygeidec( ~eqincome , ~rb090 , subset(dbd_eusilc, eqincome > 0) , epsilon = 2 )

# database-backed linearized design using a variable with missings
sub_dbd_eusilc <- subset(dbd_eusilc, py010n > 0 | is.na(py010n) )
svygeidec( ~py010n , ~rb090 , sub_dbd_eusilc , epsilon = 0 )
svygeidec( ~py010n , ~rb090 , sub_dbd_eusilc , epsilon = 0, na.rm = TRUE )
svygeidec( ~py010n , ~rb090 , sub_dbd_eusilc , epsilon = .5 )
svygeidec( ~py010n , ~rb090 , sub_dbd_eusilc , epsilon = .5, na.rm = TRUE )
svygeidec( ~py010n , ~rb090 , sub_dbd_eusilc , epsilon = 1 )
svygeidec( ~py010n , ~rb090 , sub_dbd_eusilc , epsilon = 1, na.rm = TRUE )
svygeidec( ~py010n , ~rb090 , sub_dbd_eusilc , epsilon = 2 )
svygeidec( ~py010n , ~rb090 , sub_dbd_eusilc , epsilon = 2, na.rm = TRUE )
```

```
dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## 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(laeken)
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 )

## Not run:

# 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 )

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

dbd_eusilc <-
```

```
svydesign(
ids = ~rb030 ,
strata = ~db040 ,
weights = ~rb050 ,
data="eusilc",
dbname=dbfile,
dbtype="SQLite"
)

dbd_eusilc <- convey_prep( dbd_eusilc )

svygini( ~ eqincome , design = dbd_eusilc )

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## 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(laeken)
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)

## Not run:

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'ses' , ses )

dbd_ses <- svydesign(id=~1, weights=~weights, data="ses", dbname=dbfile, dbtype="SQLite")
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. |
| `...` | arguments passed on to ‘survey::svyquantile‘ |
| `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(laeken)
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 )

## Not run:

# 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 )

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

```
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

dbd_eusilc <- convey_prep( dbd_eusilc )

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

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## 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. |
| `...` | arguments passed on to ‘survey::svyquantile‘ |
| `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(laeken)
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 )
```

```
## Not run:

# 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 )

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

dbd_eusilc <- convey_prep( dbd_eusilc )

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

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## End(Not run)
```

## svyjdiv *J-divergence measure (EXPERIMENTAL)*

### Description

Estimate the j-divergence measure, an entropy-based measure of inequality

### Usage

```
svyjdiv(formula, design, ...)

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

```
svyjdiv(formula, design, na.rm = FALSE, ...)

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

## S3 method for class 'DBIsvydesign'
svyjdiv(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.

This measure 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.

## Note

This function is experimental and is subject to change in later versions.

## Author(s)

Guilherme Jacob

## References

Nicholas Rohde (2016). J-divergence measurements of economic inequality. J. R. Statist. Soc. A, v. 179, Part 3 (2016), pp. 847-870. URL `http://onlinelibrary.wiley.com/doi/10.1111/rssa.12153/abstract`.

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(laeken)
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)

svyjdiv( ~eqincome , design = subset( des_eusilc , eqincome > 0 ) )

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

svyjdiv( ~eqincome , design = subset( des_eusilc_rep , eqincome > 0 ) )

## Not run:

# linearized design using a variable with missings
svyjdiv( ~py010n , design = subset( des_eusilc , py010n > 0 | is.na( py010n ) ) )
svyjdiv( ~py010n , design = subset( des_eusilc , py010n > 0 | is.na( py010n ) ), na.rm = TRUE )
# replicate-weighted design using a variable with missings
svyjdiv( ~py010n , design = subset( des_eusilc_rep , py010n > 0 | is.na( py010n ) ) )
svyjdiv( ~py010n , design = subset( des_eusilc_rep , py010n > 0 | is.na( py010n ) ) , na.rm = TRUE )

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

dbd_eusilc <- convey_prep( dbd_eusilc )

svyjdiv( ~eqincome , design = subset( dbd_eusilc , eqincome > 0 ) )

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )
```

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

---

`svyjdivdec` *J-Divergence Decomposition (EXPERIMENTAL)*

---

## Description

Estimates the group decomposition of the generalized entropy index

## Usage

```
svyjdivdec(formula, subgroup, design, ...)

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

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

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

## Arguments

| | |
|---|---|
| `formula` | a formula specifying the income variable |
| `subgroup` | a formula specifying the group 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? Observations containing missing values in income or group variables will 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.

This measure only allows for strictly positive variables.

## Value

Object of class "`cvydstat`", which are vectors with a "`var`" attribute giving the variance-covariance matrix and a "`statistic`" attribute giving the name of the statistic.

## Note

This function is experimental and is subject to change in later versions.

## Author(s)

Guilherme Jacob, Djalma Pessoa and Anthony Damico

## References

Anthony F. Shorrocks (1984). Inequality decomposition by population subgroups. *Econometrica*, v. 52, n. 6, 1984, pp. 1369-1385. URL http://www.jstor.org/stable/1913511.

Nicholas Rohde (2016). J-divergence measurements of economic inequality. J. R. Statist. Soc. A, v. 179, Part 3 (2016), pp. 847-870. URL http://onlinelibrary.wiley.com/doi/10.1111/rssa.12153/abstract.

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

svyjdiv

## Examples

```
library(survey)
library(laeken)
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
svyjdivdec( ~eqincome , ~rb090 , subset(des_eusilc, eqincome > 0) )

# replicate-weighted design
svyjdivdec( ~eqincome , ~rb090 , subset(des_eusilc_rep, eqincome > 0) )

## Not run:

# linearized design using a variable with missings
sub_des_eusilc <- subset(des_eusilc, py010n > 0 | is.na(py010n) )
svyjdivdec( ~py010n , ~rb090 , sub_des_eusilc )
svyjdivdec( ~py010n , ~rb090 , sub_des_eusilc , na.rm = TRUE )

# replicate-weighted design using a variable with missings
sub_des_eusilc_rep <- subset(des_eusilc_rep, py010n > 0 | is.na(py010n) )
svyjdivdec( ~py010n , ~rb090 , sub_des_eusilc_rep )
svyjdivdec( ~py010n , ~rb090 , sub_des_eusilc_rep , na.rm = TRUE )
```

```
# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

dbd_eusilc <- convey_prep( dbd_eusilc )

# database-backed linearized design
svyjdivdec( ~eqincome , ~rb090 , subset(dbd_eusilc, eqincome > 0) )

# database-backed linearized design using a variable with missings
sub_dbd_eusilc <- subset(dbd_eusilc, py010n > 0 | is.na(py010n) )
svyjdivdec( ~py010n , ~rb090 , sub_dbd_eusilc )
svyjdivdec( ~py010n , ~rb090 , sub_dbd_eusilc , na.rm = TRUE )

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## 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 quantile-based curve approacches 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 `https://www.scb.se/contentassets/ca21efb41fee47d293bbee5bf7be7fb3/variance-estimation-for-measures-of-income-inequality-and-polarization---the-estimating-equations-pdf`.

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

```
library(survey)
library(laeken)
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 )

## Not run:

# 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 )



# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

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' )

dbDisconnect( conn , shutdown = TRUE )


## 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, quantiles = 0.5, percent = 0.6, na.rm = FALSE, ...)

## S3 method for class 'svyrep.design'
svypoormed(formula, design, quantiles = 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. |
| `...` | arguments passed on to ‘survey::svyquantile‘ |
| `quantiles` | income quantile, usually .5 (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

svyarpt

## Examples

```
library(survey)
library(laeken)
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 )

## Not run:

# 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 )

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

dbd_eusilc <- convey_prep( dbd_eusilc )

svypoormed( ~ eqincome , design = dbd_eusilc )

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## 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,
  alpha1 = 0.2,
  alpha2 = (1 - alpha1),
  na.rm = FALSE,
  upper_quant = FALSE,
  lower_quant = FALSE,
  upper_tot = FALSE,
  lower_tot = FALSE,
  ...
)

## S3 method for class 'svyrep.design'
svyqsr(
  formula,
  design,
  alpha1 = 0.2,
  alpha2 = (1 - alpha1),
  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 |
| `alpha1` | order of the lower quintile |
| `alpha2` | order of the upper quintile |
| `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

```
library(survey)
library(laeken)
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 )

## Not run:

# linearized design using a variable with missings
svyqsr( ~ db090 , design = des_eusilc )
svyqsr( ~ db090 , design = des_eusilc , na.rm = TRUE )
# replicate-weighted design using a variable with missings
svyqsr( ~ db090 , design = des_eusilc_rep )
svyqsr( ~ db090 , design = des_eusilc_rep , na.rm = TRUE )

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

```
dbd_eusilc <- convey_prep( dbd_eusilc )

svyqsr( ~ eqincome , design = dbd_eusilc )

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## End(Not run)
```

---

## svyrenyi *Renyi divergence measure (EXPERIMENTAL)*

---

### 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.

## Note

This function is experimental and is subject to change in later versions.

## 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(laeken)
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 )

## Not run:

# 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 )

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

```
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

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' )

dbDisconnect( conn , shutdown = TRUE )


## End(Not run)
```

## svyrich *Richness measures (EXPERIMENTAL)*

### Description

Estimate Peichl, Schaefer and Scheicher (2010) richness measures.

### Usage

```
svyrich(formula, design, ...)

## S3 method for class 'survey.design'
svyrich(
  formula,
  design,
  type_measure,
  g,
  type_thresh = "abs",
  abs_thresh = NULL,
  times = 1,
  quantiles = 0.5,
  na.rm = FALSE,
  thresh = FALSE,
  ...
)

## S3 method for class 'svyrep.design'
svyrich(
  formula,
  design,
  type_measure,
  g,
  type_thresh = "abs",
  abs_thresh = NULL,
  times = 1,
  quantiles = 0.5,
  na.rm = FALSE,
  thresh = FALSE,
  ...
)

## S3 method for class 'DBIsvydesign'
svyrich(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 `svyarpt` |
| `type_measure` | A string "Cha", "FGTT1" or "FGTT2" defining the richness measure. |
| `g` | Richness preference parameter. |
| `type_thresh` | type of richness threshold. If "abs" the threshold is fixed and given the value of abs_thresh; if "relq" it is given by `times` times the quantile; if "relm" it is `times` times the mean. |
| `abs_thresh` | richness threshold value if type_thresh is "abs" |
| `times` | the multiple of the quantile or mean used in the richness threshold definition |
| `quantiles` | the quantile used used in the richness threshold definition |
| `na.rm` | Should cases with missing values be dropped? |
| `thresh` | return the richness 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.

## Note

This function is experimental and is subject to change in later versions.

## Author(s)

Guilherme Jacob, Djalma Pessoa and Anthony Damico

## References

Michal Brzezinski (2014). Statistical Inference for Richness Measures. *Applied Economics*, Vol. 46, No. 14, pp. 1599-1608, URL http://dx.doi.org/10.1080/00036846.2014.880106.

Andreas Peichl, Thilo Schaefer, and Christoph Scheicher (2010). Measuring richness and poverty: A micro data application to Europe and Germany. *Review of Income and Wealth*, Vol. 56, No.3, pp. 597-619.

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

svyfgt

## Examples

```
library(survey)
library(laeken)
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 )

# concave FGT-like richness measure
# headcount ratio, richness threshold fixed
svyrich(~eqincome, des_eusilc, type_measure = "FGTT1" , g=0,  abs_thresh=30000)
# richness gap index, richness threshold fixed
svyrich(~eqincome, des_eusilc, type_measure = "FGTT1" , g=1,  abs_thresh=30000)
# headcount ratio, richness threshold equal to the median
svyrich(~eqincome, des_eusilc, type_measure = "FGTT1" , g=0, type_thresh= "relq" )
# richness gap index, richness threshold equal to the median
```

```
svyrich(~eqincome, des_eusilc, type_measure = "FGTT1" , g=1, type_thresh= "relq" )
# headcount ratio, richness threshold equal to the mean
svyrich(~eqincome, des_eusilc, type_measure = "FGTT1" , g=0, type_thresh= "relm" )
# richness gap index, richness threshold equal to the mean
svyrich(~eqincome, des_eusilc, type_measure = "FGTT1" , g=1, type_thresh= "relm" )

#  using svrep.design:
# headcount ratio, richness threshold fixed
svyrich(~eqincome, des_eusilc_rep, type_measure = "FGTT1" , g=0, abs_thresh=30000 )
# richness gap index, richness threshold fixed
svyrich(~eqincome, des_eusilc_rep, type_measure = "FGTT1" , g=1, abs_thresh=30000 )
# headcount ratio, richness threshold equal to the median
svyrich(~eqincome, des_eusilc_rep, type_measure = "FGTT1" , g=0, type_thresh= "relq" )
# richness gap index, richness threshold equal to the median
svyrich(~eqincome, des_eusilc_rep, type_measure = "FGTT1" , g=1, type_thresh= "relq" )
# headcount ratio, richness threshold equal to the mean
svyrich(~eqincome, des_eusilc_rep, type_measure = "FGTT1" , g=0, type_thresh= "relm" )
# richness gap index, richness threshold equal to the mean
svyrich(~eqincome, des_eusilc_rep, type_measure = "FGTT1" , g=1, type_thresh= "relm" )

## Not run:

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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


dbd_eusilc <- convey_prep( dbd_eusilc )

# headcount ratio, richness threshold fixed
svyrich(~eqincome, dbd_eusilc, type_measure = "FGTT1" , g=0, abs_thresh=30000 )
# richness gap index, richness threshold fixed
svyrich(~eqincome, dbd_eusilc, type_measure = "FGTT1" , g=1, abs_thresh=30000 )
# headcount ratio, richness threshold equal to the median
svyrich(~eqincome, dbd_eusilc, type_measure = "FGTT1" , g=0, type_thresh= "relq" )
# richness gap index, richness threshold equal to the median
svyrich(~eqincome, dbd_eusilc, type_measure = "FGTT1" , g=1, type_thresh= "relq" )
# headcount ratio, richness threshold equal to the mean
svyrich(~eqincome, dbd_eusilc, type_measure = "FGTT1" , g=0, type_thresh= "relm" )
# richness gap index, richness threshold equal to the mean
```

```
svyrich(~eqincome, dbd_eusilc, type_measure = "FGTT1" , g=1, type_thresh= "relm" )

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## 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,
  quantiles = 0.5,
  na.rm = FALSE,
  med_old = FALSE,
  med_young = FALSE,
  ...
)

## S3 method for class 'svyrep.design'
svyrmir(
  formula,
  design,
  age,
  agelim = 65,
  quantiles = 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. |
| `...` | arguments passed on to ‘survey::svyquantile‘ |
| `age` | formula defining the variable age |
| `agelim` | the age cutpoint, the default is 65 |
| `quantiles` | income quantile, usually .5 (median) |
| `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(laeken)
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 )

## Not run:

# 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 )

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

dbd_eusilc <- convey_prep( dbd_eusilc )

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

```
dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## 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,
  quantiles = 0.5,
  percent = 0.6,
  na.rm = FALSE,
  thresh = FALSE,
  poor_median = FALSE,
  ...
)

## S3 method for class 'svyrep.design'
svyrmpg(
  formula,
  design,
  quantiles = 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 |
| quantiles | income quantile, usually .5 (median) |
| 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

```
library(survey)
library(laeken)
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 )

## Not run:

# 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 )

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

dbd_eusilc <- convey_prep( dbd_eusilc )

svyrmpg( ~ eqincome , design = dbd_eusilc )

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## End(Not run)
```

## svysen *Sen (1976) poverty index (EXPERIMENTAL)*

## Description

Estimate the Sen (1976) poverty measure.

## Usage

```
svysen(formula, design, ...)

## S3 method for class 'survey.design'
svysen(
  formula,
  design,
  abs_thresh = NULL,
  na.rm = FALSE,
  components = FALSE,
  ...
)

## S3 method for class 'svyrep.design'
svysen(
  formula,
  design,
  abs_thresh = NULL,
  na.rm = FALSE,
  components = FALSE,
  ...
)

## S3 method for class 'DBIsvydesign'
svysen(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 |
| `abs_thresh` | poverty threshold value |
| `na.rm` | Should cases with missing values be dropped? |
| `components` | Keep estimates of FGT(0), FGT(1), Gini index of poor incomes. |

## 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.

## Note

This function is experimental and is subject to change in later versions.

## Author(s)

Guilherme Jacob, Djalma Pessoa and Anthony Damico

## References

Amartya K. Sen (1976). Poverty: An Ordinal Approach to Measurement. *Econometrica*, v. 44, n. 3, pp. 219-231. URL http://www.jstor.org/stable/1912718.

## See Also

svysst, svyfgt, svygini.

## Examples

```
library(survey)
library(laeken)
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 )

# using linearized design
svysen( ~eqincome, des_eusilc, abs_thresh=10000 )

# using replicate design:
svysen( ~eqincome, des_eusilc_rep, abs_thresh = 10000 )


## Not run:

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )
```

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


dbd_eusilc <- convey_prep( dbd_eusilc )

# linearized SE:
svysen(~eqincome, dbd_eusilc, abs_thresh=10000)

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## End(Not run)
```

## svysst *Sen-Shorrocks-Thon poverty index (EXPERIMENTAL)*

### Description

Estimate the Sen-Shorrocks-Thon poverty measure.

### Usage

```
svysst(formula, design, ...)

## S3 method for class 'survey.design'
svysst(
  formula,
  design,
  abs_thresh = NULL,
  na.rm = FALSE,
  components = FALSE,
  ...
)

## S3 method for class 'svyrep.design'
svysst(
  formula,
  design,
  abs_thresh = NULL,
  na.rm = FALSE,
  components = FALSE,
  ...
)

## S3 method for class 'DBIsvydesign'
svysst(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 |
| `abs_thresh` | poverty threshold value |
| `na.rm` | Should cases with missing values be dropped? |
| `components` | Keep estimates of FGT(0), FGT(1), Gini index of poverty gap ratios. |

## 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.

## Note

This function is experimental and is subject to change in later versions.

## Author(s)

Guilherme Jacob, Djalma Pessoa and Anthony Damico

## References

Anthony F. Shorrocks (1995). Revisiting the Sen Poverty Index. *Econometrica*, v. 63, n. 5, pp. 1225-230. URL `http://www.jstor.org/stable/2171728`.

Dominique Thon (1979). On measuring poverty. *Review of Income and Wealth*, v. 25, n. 4, pp. 429-439. URL `http://dx.doi.org/10.1111/j.1475-4991.1979.tb00117.x`.

Amartya K. Sen (1976). Poverty: An Ordinal Approach to Measurement. *Econometrica*, v. 44, n. 3, pp. 219-231. URL `http://www.jstor.org/stable/1912718`.

## See Also

svysen, svyfgt, svygini.

## Examples

```
library(survey)
library(laeken)
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 )

# using linearized design
svysst( ~eqincome, des_eusilc, abs_thresh=10000 )

# using replicate design:
svysst( ~eqincome, des_eusilc_rep, abs_thresh = 10000 )


## Not run:

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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


dbd_eusilc <- convey_prep( dbd_eusilc )

# linearized SE:
svysst(~eqincome, dbd_eusilc, abs_thresh=10000)

dbRemoveTable( conn , 'eusilc' )
```

```
dbDisconnect( conn , shutdown = TRUE )


## End(Not run)
```

---

svywatts *Watts poverty index (EXPERIMENTAL)*

---

## Description

Estimate the Watts (1968) poverty measure

## Usage

```
svywatts(formula, design, ...)

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

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

## S3 method for class 'DBIsvydesign'
svywatts(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` |
| `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 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 |
| `quantiles` | the quantile 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.

## Note

This function is experimental and is subject to change in later versions.

## Author(s)

Guilherme Jacob, Djalma Pessoa and Anthony Damico

## References

Harold W. Watts (1968). An economic definition of poverty. *Institute For Research on Poverty Discussion Papers*, n.5. University of Wisconsin. URL `https://www.irp.wisc.edu/publications/dps/pdfs/dp568.pdf`.

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(laeken)
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 )

# absolute poverty threshold
svywatts(~eqincome, des_eusilc, abs_thresh=10000)
# poverty threshold equal to arpt
svywatts(~eqincome, des_eusilc, type_thresh= "relq" , thresh = TRUE)
# poverty threshold equal to 0.6 times the mean
svywatts(~eqincome, des_eusilc, type_thresh= "relm" , thresh = TRUE)

#  using svrep.design:
# absolute poverty threshold
svywatts(~eqincome, des_eusilc_rep, abs_thresh=10000)
# poverty threshold equal to arpt
svywatts(~eqincome, des_eusilc_rep, type_thresh= "relq" , thresh = TRUE)
# poverty threshold equal to 0.6 times the mean
svywatts(~eqincome, des_eusilc_rep, type_thresh= "relm" , thresh = TRUE)

## Not run:

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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


dbd_eusilc <- convey_prep( dbd_eusilc )

# absolute poverty threshold
```

```
svywatts(~eqincome, dbd_eusilc, abs_thresh=10000)
# poverty threshold equal to arpt
svywatts(~eqincome, dbd_eusilc, type_thresh= "relq" , thresh = TRUE)
# poverty threshold equal to 0.6 times the mean
svywatts(~eqincome, dbd_eusilc, type_thresh= "relm" , thresh = TRUE)

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## End(Not run)
```

---

## svywattsdec *Watts poverty index decomposition (EXPERIMENTAL)*

---

### Description

Estimate the Watts (1968) poverty measure and its components

### Usage

```
svywattsdec(formula, design, ...)

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

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

## S3 method for class 'DBIsvydesign'
svywattsdec(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. Currently not used. |
| `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 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 |
| `quantiles` | the quantile 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 "`cvydstat`", with estimates for the Watts index, FGT(0), Watts Poverty Gap Ratio, and Theil(poor incomes) with a "`var`" attribute giving the variance-covariance matrix. A "`statistic`" attribute giving the name of the statistic.

## Note

This function is experimental and is subject to change in later versions.

## Author(s)

Guilherme Jacob, Djalma Pessoa and Anthony Damico

## References

McKinley L. Blackburn (1989). Poverty measurement: an index related to a Theil measure of inequality. *Journal of Business & Economic Statistics*, Vol.7, No.4, pp. 475-481, URL `http://amstat.tandfonline.com/doi/abs/10.1080/07350015.1989.10509760`.

Satya R. Chakravarty, Joseph Deutsch and Jacques Silber (2008). On the Watts multidimensional poverty index and its decomposition. *World Development*, Vol.36, No.6, pp.1067-1077.

Harold W. Watts (1968). An economic definition of poverty. *Institute For Research on Poverty Discussion Papers*, n.5. University of Wisconsin. URL https://www.irp.wisc.edu/publications/dps/pdfs/dp568.pdf.

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

svywatts,svyfgt,svyfgt

## Examples

```
library(survey)
library(laeken)
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 )

# absolute poverty threshold
svywattsdec(~eqincome, des_eusilc, abs_thresh=10000)
# poverty threshold equal to arpt
svywattsdec(~eqincome, des_eusilc, type_thresh= "relq" , thresh = TRUE)
# poverty threshold equal to 0.6 times the mean
svywattsdec(~eqincome, des_eusilc, type_thresh= "relm" , thresh = TRUE)

#  using svrep.design:
# absolute poverty threshold
svywattsdec(~eqincome, des_eusilc_rep, abs_thresh=10000)
# poverty threshold equal to arpt
svywattsdec(~eqincome, des_eusilc_rep, type_thresh= "relq" , thresh = TRUE)
# poverty threshold equal to 0.6 times the mean
svywattsdec(~eqincome, des_eusilc_rep, type_thresh= "relm" , thresh = TRUE)

## Not run:

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
```

```
dbWriteTable( conn , 'eusilc' , eusilc )

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


dbd_eusilc <- convey_prep( dbd_eusilc )

# absolute poverty threshold
svywattsdec(~eqincome, dbd_eusilc, abs_thresh=10000)
# poverty threshold equal to arpt
svywattsdec(~eqincome, dbd_eusilc, type_thresh= "relq" , thresh = TRUE)
# poverty threshold equal to 0.6 times the mean
svywattsdec(~eqincome, dbd_eusilc, type_thresh= "relm" , thresh = TRUE)

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## End(Not run)
```

---

`svyzenga` *Zenga index (EXPERIMENTAL)*

---

## 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.

## Note

This function is experimental and is subject to changes in later versions.

## Author(s)

Guilherme Jacob, Djalma Pessoa and Anthony Damico

## References

Lucio Barabesi, Giancarlo Diana and Pier Francesco Perri (2016). Linearization of inequality indexes in the design-based framework. Statistics. URL `http://www.tandfonline.com/doi/pdf/10.1080/02331888.2015.1135924`.

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

```
library(survey)
library(laeken)
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)


# variable without missing values
svyzenga(~eqincome, des_eusilc)
svyzenga(~eqincome, des_eusilc_rep)

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

## Not run:

# variable with with missings
svyzenga(~py010n, des_eusilc )
svyzenga(~py010n, des_eusilc_rep )

svyzenga(~py010n, des_eusilc, na.rm = TRUE )
svyzenga(~py010n, des_eusilc_rep, na.rm = TRUE )

# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )

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

dbd_eusilc <- convey_prep( dbd_eusilc )


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

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

# variable with with missings
svyzenga(~py010n, dbd_eusilc )

svyzenga(~py010n, dbd_eusilc, na.rm = TRUE )


dbRemoveTable( conn , 'eusilc' )
```

```
dbDisconnect( conn , shutdown = TRUE )


## End(Not run)
```

---

`svyzengacurve` *Zenga inequality curve (EXPERIMENTAL)*

---

## Description

Estimate the Zenga curve, an inequality graph

## Usage

```
svyzengacurve(formula, design, ...)

## S3 method for class 'survey.design'
svyzengacurve(
  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'
svyzengacurve(
  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'
svyzengacurve(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 Zenga curve be estimated as well? Defaults to `FALSE`. |
| `plot` | Should the Zenga 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

WARNING: this is an experimental version. Use with caution.

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 based on the Lorenz curve formula actually used to calculate the Gini index. On the other hand, the quantile-based curve is used to estimate the complement of the ratio between the means, SEs and confidence intervals.

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

## Value

Object of class "`svyquantile`", which are vectors with a "`quantiles`" attribute giving the complement of the ratio between the lower and upper means, and a "`SE`" attribute giving the standard errors of the estimates.

## Note

This function is experimental and is subject to changes in later versions.

## Author(s)

Guilherme Jacob, Djalma Pessoa and Anthony Damico

## References

Marcella Polisicchio and Francesco Porro (2011). A Comparison Between Lorenz L(P) Curve and Zenga I(P) Curve. Statistica Applicata, v. 21, n. 3-4, 289-301.

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

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

svyquantile

## Examples

```
library(survey)
library(laeken)
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 )
svyzengacurve( ~eqincome , des_eusilc,  alpha = .01 )

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

svyzengacurve( ~eqincome , des_eusilc_rep,  alpha = .01 )

## Not run:

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



# database-backed design
library(RSQLite)
library(DBI)
dbfile <- tempfile()
conn <- dbConnect( RSQLite::SQLite() , dbfile )
dbWriteTable( conn , 'eusilc' , eusilc )
```

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

dbd_eusilc <- convey_prep( dbd_eusilc )

svyzengacurve( ~eqincome , dbd_eusilc, alpha = .01 )

# highlighting the difference between the quantile-based curve and the empirical version:
svyzengacurve( ~eqincome , dbd_eusilc, seq(0,1,.1), empirical = TRUE, curve.col = "green" )
svyzengacurve( ~eqincome , dbd_eusilc, seq(0,1,.1), alpha = .01, add = TRUE )
legend( "bottomleft", 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
svyzengacurve( ~eqincome , dbd_eusilc, seq(0,1,.01), empirical = TRUE, curve.col = "green" )
svyzengacurve( ~eqincome , dbd_eusilc, seq(0,1,.01), alpha = .01, add = TRUE )
legend( "bottomleft", c("Quantile-based", "Empirical"), lwd = c(1,1), col = c("red", "green"))

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )


## End(Not run)
```

# Index