

# Package: convey (via r-universe)

October 17, 2024

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

**Version** 1.0.1

**Date** 2024-10-16

**URL** <https://www.convey-r.org/>

**BugReports** <https://github.com/ajdamico/convey/issues>

**Description** Variance estimation on indicators of income concentration and poverty using complex sample survey designs. Wrapper around the 'survey' package.

**Depends** R (>= 4.0.0)

**Imports** survey (>= 4.2-1), methods

**License** GPL-3

**Suggests** testthat, knitr, rmarkdown, vardpoor, laeken, DBI, RSQLite

**RoxygenNote** 7.3.2

**NeedsCompilation** no

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

**Maintainer** Anthony Damico <ajdamico@gmail.com>

**Repository** CRAN

**Date/Publication** 2024-10-16 14:20:03 UTC

## Contents

|                       |    |
|-----------------------|----|
| contrastinf . . . . . | 2  |
| convey_prep . . . . . | 4  |
| densfun . . . . .     | 5  |
| h_fun . . . . .       | 6  |
| icdf . . . . .        | 7  |
| svyarpr . . . . .     | 8  |
| svyarpt . . . . .     | 10 |
| svyatk . . . . .      | 12 |

|                       |    |
|-----------------------|----|
| svyfgt . . . . .      | 16 |
| svyfgtdec . . . . .   | 20 |
| svygei . . . . .      | 23 |
| svygeidec . . . . .   | 27 |
| svygini . . . . .     | 30 |
| svygp . . . . .       | 33 |
| svyqalpha . . . . .   | 35 |
| svyisq . . . . .      | 37 |
| svyjdiv . . . . .     | 40 |
| svyjdivdec . . . . .  | 43 |
| svylorenz . . . . .   | 46 |
| svypoormed . . . . .  | 49 |
| svyqsr . . . . .      | 52 |
| svyrich . . . . .     | 55 |
| svyrmir . . . . .     | 58 |
| svyrmpg . . . . .     | 61 |
| svywatts . . . . .    | 64 |
| svywattsdec . . . . . | 68 |
| svyzenga . . . . .    | 71 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>75</b> |
|--------------|-----------|

---

|             |  |
|-------------|--|
| contrastinf | <i>Generalized linearization of a smooth function of survey statistics</i> |
|-------------|--|

---

## 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, Guilherme Jacob, 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 <https://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, linearized = TRUE)
S20_val <- coef (S20); attributes (S20_val) <- NULL
S20_lin <- attr(S20 , "linearized" )
S80 <- svyisq(~ eqincome, design = des_eusilc, .80, linearized = TRUE)
S80_val <- coef (S80); attributes (S80_val) <- NULL
S80_lin <- attr(S80 , "linearized" )
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

sets the population of reference for poverty threshold estimation (needed for convey functions that use a global poverty threshold) within the design. this function generally should 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, Anthony Damico, and Guilherme Jacob

**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 <code>survey.design</code> from the <code>survey</code> library.  |
| x       | the point where the derivative is calculated   |
| h       | value of the bandwidth based on the whole sample   |
| FUN     | if <code>F</code> estimates the derivative of the cdf function; if <code>big_s</code> 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 | <i>Computes the bandwidth needed to compute the derivative of the cdf function</i> |
|-------|--|

---

**Description**

Using the whole sample, computes the bandwidth 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, Guilherme Jacob, 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 <code>survey.design</code> or class <code>svyrep.design</code> 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 <https://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 <https://www150.statcan.gc.ca/n1/en/catalogue/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 )

```

---

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 <code>survey.design</code> or class <code>svyrep.design</code> from the <code>survey</code> library. |
| ...       | arguments passed on to <code>'svyarpr'</code>   |
| quantiles | income quantile, usually <code>.50</code> (median)  |
| percent   | fraction of the quantile, usually <code>.60</code>  |
| 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 <https://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 <https://www150.statcan.gc.ca/n1/en/catalogue/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 <code>survey.design</code> or class <code>svyrep.design</code> from the survey library. |
| ...       | arguments passed on to <code>'survey::oldsvyquantile'</code>   |
| quantiles | income quantile quantiles, usually <code>.50</code> (median)   |
| percent   | fraction of the quantile, usually <code>.60</code>   |
| 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 <https://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 <https://www150.statcan.gc.ca/n1/en/catalogue/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, an inequality measure

**Usage**

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

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

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

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

**Arguments**

|                   |  |
|-------------------|--|
| formula           | a formula specifying the income variable   |
| design            | a design object of class <code>survey.design</code> or class <code>svyrep.design</code> from the survey library.                         |
| ...               | future expansion   |
| epsilon           | a parameter that determines the sensitivity towards inequality in the bottom of the distribution. Defaults to <code>epsilon = 1</code> . |
| na.rm             | Should cases with missing values be dropped?   |
| deff              | Return the design effect (see <code>survey::svymean</code> )   |
| linearized        | Should a matrix of linearized variables be returned  |
| influence         | Should a matrix of (weighted) influence functions be returned? (for compatibility with <a href="#">svyby</a> )                           |
| return.replicates | Return the replicate estimates?  |

**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](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_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 )

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)

```

---

svyfgt

*FGT measure of poverty*


---

## Description

Estimate the FGT measure.

## 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,
  deff = FALSE,
  linearized = FALSE,
  influence = 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,
  deff = FALSE,
  linearized = FALSE,
  return.replicates = FALSE,
  ...
)

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

```

### Arguments

|                   |  |
|-------------------|--|
| formula           | a formula specifying the income variable   |
| design            | a design object of class <code>survey.design</code> or class <code>svyrep.design</code> from the survey library.   |
| ...               | passed to <code>svyarpr</code> and <code>svyarpt</code>  |
| g                 | If <code>g=0</code> estimates the headcount ratio; If <code>g=1</code> estimates the average normalised poverty gap, and if <code>g=2</code> estimates the average squared normalised poverty gap        |
| type_thresh       | type of poverty threshold. If "abs" the threshold is fixed and given the value of <code>abs_thresh</code> ; if "relq" it is given by percent times the quantile; if "relm" it is percent times the mean. |
| abs_thresh        | poverty threshold value if <code>type_thresh</code> 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   |
| deff              | Return the design effect (see <code>survey::svymean</code> )   |
| linearized        | Should a matrix of linearized variables be returned?   |
| influence         | Should a matrix of (weighted) influence functions be returned? (for compatibility with <a href="#">svyby</a> ). Not implemented yet for linearized designs.  |
| return.replicates | Return the replicate estimates?  |

## Details

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

The FGT poverty measures have three special cases. When  $g = 0$ , the FGT measure is the headcount poverty rate, assigning the same "poverty-weight" to all persons below the poverty line. When  $g = 1$ , it becomes the poverty gap ratio, a measure which accounts for the intensity of income shortfall among the poor. When  $g = 2$ , it becomes the squared poverty gap ratio, a measure that also accounts for inequality of poverty intensity across the poor. The  $g$  is a poverty sensitivity parameter, adding more weight to people with greater income shortfalls as it increases.

## 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, Anthony Damico, and Guilherme Jacob

## References

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

Y.G. Berger and C. J. Skinner (2003), Variance estimation for a low income proportion. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, Vol. 52, No. 4, pp. 457-468. DOI [doi:10.1111/14679876.00417](https://doi.org/10.1111/14679876.00417)

Buhong Zheng (2001). Statistical inference for poverty measures with relative poverty lines. *Journal of Econometrics*, Vol. 101, pp. 337-356.

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 <https://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 <https://www150.statcan.gc.ca/n1/en/catalogue/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*


---

### 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,
  return.replicates = FALSE,
  ...
)

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

```

### Arguments

|                   |  |
|-------------------|--|
| formula           | a formula specifying the income variable   |
| design            | a design object of class <code>survey.design</code> or class <code>svyrep.design</code> 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 $g \geq 2$ only,   |
| type_thresh       | type of poverty threshold. If "abs" the threshold is fixed and given the value of <code>abs_thresh</code> ; if "relq" it is given by <code>percent</code> times the quantile; if "relm" it is <code>percent</code> times the mean. |
| abs_thresh        | poverty threshold value if <code>type_thresh</code> 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   |
| return.replicates | Return the replicate estimates?  |

### 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.

**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. <doi: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 <https://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 <https://www150.statcan.gc.ca/n1/en/catalogue/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
svyfgtdec(~eqincome, des_eusilc, g=2, type_thresh= "relq" , thresh = TRUE)
# poverty threshold equal to 0.6 times the mean
svyfgtdec(~eqincome, des_eusilc, g=2, type_thresh= "reln" , thresh = TRUE)

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

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

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

```

### Arguments

|                   |   |
|-------------------|---|
| formula           | a formula specifying the income variable  |
| design            | a design object of class <code>survey.design</code> or class <code>svyrep.design</code> from the survey library.                      |
| ...               | future expansion  |
| epsilon           | a parameter that determines the sensitivity towards inequality in the top of the distribution. Defaults to <code>epsilon = 1</code> . |
| na.rm             | Should cases with missing values be dropped?  |
| deff              | Return the design effect (see <code>survey::svymean</code> )  |
| linearized        | Should a matrix of linearized variables be returned   |
| influence         | Should a matrix of (weighted) influence functions be returned? (for compatibility with <a href="#">svyby</a> )                        |
| return.replicates | Return the replicate estimates?   |

### 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](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,
  deff = FALSE,
  linearized = FALSE,
  influence = FALSE,
  ...
)

## S3 method for class 'svyrep.design'
svygeidec(
  formula,
  subgroup,
  design,
  epsilon = 1,
  na.rm = FALSE,
  deff = FALSE,
  linearized = FALSE,
  return.replicates = 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 <code>survey.design</code> or class <code>svyrep.design</code> from the survey library.                      |
| ...               | future expansion  |
| epsilon           | a parameter that determines the sensitivity towards inequality in the top of the distribution. Defaults to <code>epsilon = 1</code> . |
| na.rm             | Should cases with missing values be dropped? Observations containing missing values in income or group variables will be dropped.     |
| deff              | Return the design effect (see <code>survey::svymean</code> )  |
| linearized        | Should a matrix of linearized variables be returned   |
| influence         | Should a matrix of (weighted) influence functions be returned? (for compatibility with <code>svyby</code> )                           |
| return.replicates | Return the replicate estimates?   |

**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. DOI [doi:10.2307/1913511](https://doi.org/10.2307/1913511).

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](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, an inequality measure

### Usage

```

svygini(formula, design, ...)

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

```

```

    linearized = FALSE,
    influence = FALSE,
    ...
)

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

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

```

### Arguments

|                   |  |
|-------------------|--|
| formula           | a formula specifying the income variable   |
| design            | a design object of class <code>survey.design</code> or class <code>svyrep.design</code> from the survey library. |
| ...               | future expansion   |
| na.rm             | Should cases with missing values be dropped?   |
| deff              | Return the design effect (see <code>survey::svymean</code> )   |
| linearized        | Should a matrix of linearized variables be returned  |
| influence         | Should a matrix of (weighted) influence functions be returned? (for compatibility with <a href="#">svyby</a> )   |
| return.replicates | Return the replicate estimates?  |

### 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, Guilherme Jacob, 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 <https://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 <https://www150.statcan.gc.ca/n1/en/catalogue/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)

```

svygpq

*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**

```

svygpq(formula, design, ...)

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

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

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

```

**Arguments**

|         |  |
|---------|--|
| formula | a formula specifying the gross hourly earnings variable  |
| design  | a design object of class <code>survey.design</code> or class <code>svyrep.design</code> 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 <https://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 <https://www150.statcan.gc.ca/n1/en/catalogue/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
svygp(~earningshour, des_ses, ~sex)
# replicate-weighted design
des_ses_rep <- as.svrepdesign( des_ses , type = "bootstrap" )
des_ses_rep <- convey_prep(des_ses_rep)

svygp(~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 )

svyggp(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 <code>survey.design</code> or class <code>svyrep.design</code> from the survey library. |
| ...     | arguments passed on to <code>'survey::oldsvyquantile'</code>   |
| 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 <https://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 <https://www150.statcan.gc.ca/n1/en/catalogue/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,
  upper = FALSE,
  na.rm = FALSE,
  deff = FALSE,
  linearized = FALSE,
  influence = FALSE,
  ...
)

```

```
## S3 method for class 'svyrep.design'
svyisq(
  formula,
  design,
  alpha,
  quantile = FALSE,
  upper = FALSE,
  na.rm = FALSE,
  deff = FALSE,
  linearized = FALSE,
  return.replicates = FALSE,
  ...
)

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

### Arguments

|                   |  |
|-------------------|--|
| formula           | a formula specifying the income variable   |
| design            | a design object of class <code>survey.design</code> or class <code>svyrep.design</code> from the survey library. |
| ...               | arguments passed on to <code>'survey::oldsvyquantile'</code>   |
| alpha             | the order of the quantile  |
| quantile          | return the upper bound of the lower tail   |
| upper             | return the total in the total in the upper tail. Defaults to FALSE.  |
| na.rm             | Should cases with missing values be dropped?   |
| deff              | Return the design effect (see <code>survey::svymean</code> )   |
| linearized        | Should a matrix of linearized variables be returned  |
| influence         | Should a matrix of (weighted) influence functions be returned? (for compatibility with <a href="#">svyby</a> )   |
| return.replicates | Return the replicate estimates?  |

### 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, Guilherme Jacob, 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 <https://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 <https://www150.statcan.gc.ca/n1/en/catalogue/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*


---

## 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,
  deff = FALSE,
  linearized = FALSE,
  influence = FALSE,
  ...
)

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

```



```

    ...
)

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

### Arguments

|                   |  |
|-------------------|--|
| formula           | a formula specifying the income variable   |
| design            | a design object of class <code>survey.design</code> or class <code>svyrep.design</code> from the survey library. |
| ...               | future expansion   |
| na.rm             | Should cases with missing values be dropped?   |
| deff              | Return the design effect (see <code>survey::svymean</code> )   |
| linearized        | Should a matrix of linearized variables be returned  |
| influence         | Should a matrix of (weighted) influence functions be returned? (for compatibility with <a href="#">svyby</a> )   |
| return.replicates | Return the replicate estimates?  |

### 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

Nicholas Rohde (2016). J-divergence measurements of economic inequality. *J. R. Statist. Soc. A*, v. 179, Part 3 (2016), pp. 847-870. DOI [doi:10.1111/rssa.12153](https://doi.org/10.1111/rssa.12153).

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](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)

svyjddiv( ~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)

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

## Not run:

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

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

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )

```

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

---

```
svyjdivdec
```

```
J-Divergence Decomposition
```

---

## 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,  
  deff = FALSE,  
  linearized = FALSE,  
  influence = FALSE,  
  ...  
)
```

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

```
svyjdivdec(  
  formula,  
  subgroup,  
  design,  
  na.rm = FALSE,  
  deff = FALSE,  
  linearized = FALSE,  
  return.replicates = 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 <code>survey.design</code> or class <code>svyrep.design</code> 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. |
| deff              | Return the design effect (see <code>survey::svymean</code> )  |
| linearized        | Should a matrix of linearized variables be returned   |
| influence         | Should a matrix of (weighted) influence functions be returned? (for compatibility with <code>svyby</code> )                       |
| return.replicates | Return the replicate estimates?   |

### 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 by population subgroups. *Econometrica*, v. 52, n. 6, 1984, pp. 1369-1385. DOI [doi:10.2307/1913511](https://doi.org/10.2307/1913511).

Nicholas Rohde (2016). J-divergence measurements of economic inequality. *J. R. Statist. Soc. A*, v. 179, Part 3 (2016), pp. 847-870. DOI [doi:10.1111/rssa.12153](https://doi.org/10.1111/rssa.12153).

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](https://www.diw.de/documents/publikationen/73/diw_01.c.40394.de/dp345.pdf).

### See Also

[svyjdidec](#)

### 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,  
  deff = FALSE,  
  linearized = FALSE,  
  influence = 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,  
  deff = FALSE,
```

```

    linearized = FALSE,
    return.replicates = FALSE,
    ...
)

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

```

### Arguments

|                   |  |
|-------------------|--|
| formula           | a formula specifying the income variable   |
| design            | a design object of class <code>survey.design</code> or class <code>svyrep.design</code> 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 specifies the confidence level for the graph.  |
| na.rm             | Should cases with missing values be dropped? Defaults to FALSE.  |
| deff              | Return the design effect (see <code>survey::svymean</code> )   |
| linearized        | Should a matrix of linearized variables be returned  |
| influence         | Should a matrix of (weighted) influence functions be returned? (for compatibility with <a href="#">svyby</a> )   |
| return.replicates | Return the replicate estimates?  |

### 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 approaches the observation-based curve.

### Value

Object of class `"survey::oldsvyquantile"`, 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/ca21efb41fee47d293bb5bf7be7fb3/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**

[oldsvyquantile](#)

**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 )

# highlighting 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 <code>survey.design</code> or class <code>svyrep.design</code> from the survey library. |
| ...       | arguments passed on to <code>'survey::oldsvyquantile'</code>   |
| quantiles | income quantile, usually <code>.5</code> (median)  |
| percent   | fraction of the quantile, usually <code>.60</code>   |
| 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 <https://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 <https://www150.statcan.gc.ca/n1/en/catalogue/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,
  deff = FALSE,
  linearized = FALSE,
  influence = 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,
  deff = FALSE,
  linearized = FALSE,
  return.replicates = FALSE,
  ...
)
```

```
)

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

### Arguments

|                   |  |
|-------------------|--|
| formula           | a formula specifying the income variable   |
| design            | a design object of class <code>survey.design</code> or class <code>svyrep.design</code> 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  |
| deff              | Return the design effect (see <code>survey::svymean</code> )   |
| linearized        | Should a matrix of linearized variables be returned  |
| influence         | Should a matrix of (weighted) influence functions be returned? (for compatibility with <code>svyby</code> )      |
| return.replicates | Return the replicate estimates?  |

### 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 <https://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 <https://www150.statcan.gc.ca/n1/en/catalogue/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)
```

---

svyrich

*Richness measures*

---

## 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,  
  percent = 1.5,  
  quantiles = 0.5,  
  thresh = FALSE,  
  na.rm = FALSE,  
  deff = FALSE,  
  linearized = FALSE,  
  ...  
)
```

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

```
svyrich(  
  formula,  
  design,  
  type_measure,  
  g,  
  type_thresh = "abs",  
  abs_thresh = NULL,  
  percent = 1.5,  
  quantiles = 0.5,  
  thresh = FALSE,  
  na.rm = FALSE,
```

```

    deff = FALSE,
    linearized = FALSE,
    return.replicates = FALSE,
    ...
)

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

```

### Arguments

|                   |   |
|-------------------|---|
| formula           | a formula specifying the income variable  |
| design            | a design object of class <code>survey.design</code> or class <code>svyrep.design</code> from the survey library.  |
| ...               | passed to <code>svyarpt</code>  |
| 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 <code>abs_thresh</code> ; if "relq" it is given by percent times the quantile; if "relm" it is percent times the mean. |
| abs_thresh        | richness threshold value if <code>type_thresh</code> is "abs"   |
| percent           | the multiple of the quantile or mean used in the richness threshold definition. Defaults to <code>percent = 1.5</code> ; i.e., 1.5 times the quantile or mean.  |
| quantiles         | the quantile used used in the richness threshold definition. Defaults to <code>quantiles = .5</code> , the median.  |
| thresh            | return the richness threshold value   |
| na.rm             | Should cases with missing values be dropped?  |
| deff              | Return the design effect (see <code>survey::svymean</code> )  |
| linearized        | Should a matrix of linearized variables be returned   |
| return.replicates | Return the replicate estimates?   |

### 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

Michał Brzezinski (2014). Statistical Inference for Richness Measures. *Applied Economics*, Vol. 46, No. 14, pp. 1599-1608, DOI [doi:10.1080/00036846.2014.880106](https://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 <https://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 Chakravarty richness measure
# higher g= parameters tend toward headcount ratio, richness threshold fixed
svyrich(~eqincome, des_eusilc, type_measure = "Cha" , g=3, abs_thresh=30000)
# g=1 parameter computes the richness gap index, richness threshold fixed
svyrich(~eqincome, des_eusilc, type_measure = "Cha" , g=1, abs_thresh=30000)
# higher g= parameters tend toward headcount ratio, richness threshold equal to the median
svyrich(~eqincome, des_eusilc, type_measure = "Cha" , g=3, type_thresh= "relq" )
# g=1 parameter computes the richness gap index, richness threshold equal to the median
svyrich(~eqincome, des_eusilc, type_measure = "Cha" , g=1, type_thresh= "relq" )
# higher g= parameters tend toward headcount ratio, richness threshold equal to the mean
svyrich(~eqincome, des_eusilc, type_measure = "Cha" , g=3, type_thresh= "relm" )
# g=1 parameter computes the richness gap index, richness threshold equal to the mean
svyrich(~eqincome, des_eusilc, type_measure = "Cha" , g=1, type_thresh= "relm" )

# using svrep.design:
# higher g= parameters tend toward headcount ratio, richness threshold fixed
svyrich(~eqincome, des_eusilc_rep, type_measure = "Cha" , g=3, abs_thresh=30000 )
# g=1 parameter computes the richness gap index, richness threshold fixed
svyrich(~eqincome, des_eusilc_rep, type_measure = "Cha" , g=1, abs_thresh=30000 )
# higher g= parameters tend toward headcount ratio, richness threshold equal to the median
svyrich(~eqincome, des_eusilc_rep, type_measure = "Cha" , g=3, type_thresh= "relq" )
# g=1 parameter computes the richness gap index, richness threshold equal to the median
```

```

svyrich(~eqincome, des_eusilc_rep, type_measure = "Cha" , g=1, type_thresh= "relq" )
# higher g= parameters tend toward headcount ratio, richness threshold equal to the mean
svyrich(~eqincome, des_eusilc_rep, type_measure = "Cha" , g=3, type_thresh= "relm" )
# g=1 parameter computes the richness gap index, richness threshold equal to the mean
svyrich(~eqincome, des_eusilc_rep, type_measure = "Cha" , 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 )

# higher g= parameters tend toward headcount ratio, richness threshold fixed
svyrich(~eqincome, dbd_eusilc, type_measure = "Cha" , g=3, abs_thresh=30000 )
# g=1 parameter computes the richness gap index, richness threshold fixed
svyrich(~eqincome, dbd_eusilc, type_measure = "Cha" , g=1, abs_thresh=30000 )
# higher g= parameters tend toward headcount ratio, richness threshold equal to the median
svyrich(~eqincome, dbd_eusilc, type_measure = "Cha" , g=3, type_thresh= "relq" )
# g=1 parameter computes the richness gap index, richness threshold equal to the median
svyrich(~eqincome, dbd_eusilc, type_measure = "Cha" , g=1, type_thresh= "relq" )
# higher g= parameters tend toward headcount ratio, richness threshold equal to the mean
svyrich(~eqincome, dbd_eusilc, type_measure = "Cha" , g=3, type_thresh= "relm" )
# g=1 parameter computes the richness gap index, richness threshold equal to the mean
svyrich(~eqincome, dbd_eusilc, type_measure = "Cha" , g=1, type_thresh= "relm" )

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )

## End(Not run)

```

**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 <code>survey.design</code> or class <code>svyrep.design</code> from the survey library. |
| ...       | arguments passed on to <code>'survey::oldsvyquantile'</code>   |
| 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 <https://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 <https://www150.statcan.gc.ca/n1/en/catalogue/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 <code>survey.design</code> or class <code>svyrep.design</code> 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 <https://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 <https://www150.statcan.gc.ca/n1/en/catalogue/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)

```

---

svywatts

*Watts measure of poverty*


---

### Description

Estimate the Watts measure for the cases:  $\alpha=0$  headcount ratio and  $\alpha=1$  poverty gap index.

### 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,
  thresh = FALSE,
  na.rm = FALSE,
  deff = FALSE,
  linearized = FALSE,

```



```

    influence = FALSE,
    ...
)

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

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

```

### Arguments

|                   |  |
|-------------------|--|
| formula           | a formula specifying the income variable   |
| design            | a design object of class <code>survey.design</code> or class <code>svyrep.design</code> from the survey library.   |
| ...               | passed to <code>svyarpr</code> and <code>svyarpt</code>  |
| type_thresh       | type of poverty threshold. If "abs" the threshold is fixed and given the value of <code>abs_thresh</code> ; if "relq" it is given by percent times the quantile; if "relm" it is percent times the mean. |
| abs_thresh        | poverty threshold value if <code>type_thresh</code> 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   |
| thresh            | return the poverty threshold value   |
| na.rm             | Should cases with missing values be dropped?   |
| deff              | Return the design effect (see <code>survey::svymean</code> )   |
| linearized        | Should a matrix of linearized variables be returned  |
| influence         | Should a matrix of (weighted) influence functions be returned? (for compatibility with <a href="#">svyby</a> ). Not implemented yet for linearized designs.  |
| return.replicates | Return the replicate estimates?  |

## Details

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

For the `svywatts` and `svywattsdec` functions, zeroes and negative numbers in the analysis domain cause an error because of the logarithm function in the definition of this poverty measure. However, zeroes and negative values in the full survey design that are outside of the domain of analysis are valid to calculate the poverty threshold because zeroes and negatives are not a problem for computing quantiles (used when `type_thresh = "relq"`) or means (used when `type_thresh = "relm"`). Missing values are treated differently. NA values anywhere in the full survey design (not only the subset, or the domain of analysis) will cause these quantiles and means to return NA results. To ignore NA values throughout, set `na.rm = TRUE`.

## 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

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>.

Buhong Zheng (2001). Statistical inference for poverty measures with relative poverty lines. *Journal of Econometrics*, Vol. 101, pp. 337-356.

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, DOI [doi:10.1080/02664763.2010.515674](https://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, DOI [doi:10.2307/1911028](https://doi.org/10.2307/1911028).

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 <https://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 <https://www150.statcan.gc.ca/n1/en/catalogue/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 )

# filter positive incomes
des_eusilc <- subset( des_eusilc , eqincome > 0 )
des_eusilc_rep <- subset( des_eusilc_rep , eqincome > 0 )

# poverty threshold fixed
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= "reln" , thresh = TRUE)
# using svrep.design:
# poverty threshold fixed
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= "reln" , 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 )

```

```

# filter positive incomes
dbd_eusilc <- subset( dbd_eusilc , eqincome > 0 )

# poverty threshold fixed
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= "reln" , thresh = TRUE)

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )

## End(Not run)

```

---

svywattsdec

*Watts poverty index decomposition*


---

## 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,
    return.replicates = FALSE,
    ...
)

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

```

### Arguments

|                   |  |
|-------------------|--|
| formula           | a formula specifying the income variable   |
| design            | a design object of class <code>survey.design</code> or class <code>svyrep.design</code> 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 <code>abs_thresh</code> ; if "relq" it is given by percent times the quantile; if "reln" it is percent times the mean. |
| abs_thresh        | poverty threshold value if <code>type_thresh</code> 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   |
| return.replicates | Return the replicate estimates?  |

### Details

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

For the `svywatts` and `svywattsdec` functions, zeroes and negative numbers in the analysis domain cause an error because of the logarithm function in the definition of this poverty measure. However, zeroes and negative values in the full survey design that are outside of the domain of analysis are valid to calculate the poverty threshold because zeroes and negatives are not a problem for computing quantiles (used when `type_thresh = "relq"`) or means (used when `type_thresh = "reln"`). Missing values are treated differently. NA values anywhere in the full survey design (not only the subset, or the domain of analysis) will cause these quantiles and means to return NA results. To ignore NA values throughout, set `na.rm = TRUE`.

### 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.

**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, DOI [doi:10.1080/07350015.1989.10509760](https://doi.org/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 <https://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 <https://www150.statcan.gc.ca/n1/en/catalogue/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 )

# filter positive incomes
des_eusilc <- subset( des_eusilc , eqincome > 0 )
des_eusilc_rep <- subset( des_eusilc_rep , eqincome > 0 )

# 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 )
dbd_eusilc <- subset( dbd_eusilc , eqincome > 0 )

# 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*


---

### 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,
  deff = FALSE,
  linearized = FALSE,
  influence = FALSE,
  ...
)

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

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

**Arguments**

|                   |  |
|-------------------|--|
| formula           | a formula specifying the income variable   |
| design            | a design object of class <code>survey.design</code> or class <code>svyrep.design</code> from the survey library. |
| ...               | future expansion   |
| na.rm             | Should cases with missing values be dropped?   |
| deff              | Return the design effect (see <code>survey::svymean</code> )   |
| linearized        | Should a matrix of linearized variables be returned  |
| influence         | Should a matrix of (weighted) influence functions be returned? (for compatibility with <a href="#">svyby</a> )   |
| return.replicates | Return the replicate estimates?  |

**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, Guilherme Jacob, and Anthony Damico

**References**

- Lucio Barabesi, Giancarlo Diana and Pier Francesco Perri (2016). Linearization of inequality indices in the design-based framework. *Statistics*, 50(5), 1161-1172. DOI [doi:10.1080/02331888.2015.1135924](https://doi.org/10.1080/02331888.2015.1135924).
- Matti Langel and Yves Tille (2012). Inference by linearization for Zenga's new inequality index: a comparison with the Gini index. *Metrika*, 75, 1093-1110. DOI [doi:10.1007/s0018401103691](https://doi.org/10.1007/s0018401103691).
- 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)

svyzenga( ~eqincome , design = des_eusilc )

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

svyzenga( ~eqincome , design = des_eusilc_rep )

## Not run:

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

svyzenga( ~ eqincome , design = dbd_eusilc )

dbRemoveTable( conn , 'eusilc' )

dbDisconnect( conn , shutdown = TRUE )

## End(Not run)
```

# Index

## \* survey

- convey\_prep, 4
  - densfun, 5
  - h\_fun, 6
  - icdf, 7
  - svyarpr, 8
  - svyarpt, 10
  - svyatk, 12
  - svyfgt, 16
  - svyfgtdec, 20
  - svygei, 23
  - svygeidec, 27
  - svygini, 30
  - svygp, 33
  - svyiqalpha, 35
  - svyisq, 37
  - svyjdiv, 40
  - svyjdivdec, 43
  - svylorenz, 46
  - svypoormed, 49
  - svyqsr, 52
  - svyrich, 55
  - svyrmir, 58
  - svyrmpg, 61
  - svywatts, 64
  - svywattsdec, 68
  - svyzenga, 71
  - svyatk, 12, 25
  - svyby, 13, 17, 24, 28, 31, 38, 41, 44, 47, 53, 65, 72
  - svyfgt, 16, 22, 57, 70
  - svyfgtdec, 20
  - svygei, 14, 23, 28, 41
  - svygeidec, 27
  - svygini, 30, 73
  - svygp, 33
  - svyiqalpha, 35
  - svyisq, 37
  - svyjdiv, 40, 44
  - svyjdivdec, 43
  - svylorenz, 46
  - svypoormed, 49
  - svyqsr, 52
  - svyrich, 55
  - svyrmir, 58
  - svyrmpg, 61
  - svywatts, 64, 70
  - svywattsdec, 68
  - svyzenga, 71
- contrastinf, 2
- convey\_prep, 4
- densfun, 5
- h\_fun, 6
- icdf, 7
- oldsvyquantile, 48
- svyarpr, 7, 8, 11, 32, 36, 39
- svyarpt, 9, 10, 18, 34, 50, 54, 60, 63, 66