Package ‘EdSurvey’

April 13, 2020

Version 2.5.0

Date 2020-04-09

Title Analysis of NCES Education Survey and Assessment Data

Author Paul Bailey [aut, cre],
Ahmad Emad [aut],
Huade Huo [aut],
Michael Lee [aut],
Yuqi Liao [aut],
Alex Lishinski [aut],
Trang Nguyen [aut],
Qingshu Xie [aut],
Jiao Yu [aut],
Ting Zhang [aut],
Jeppe Bundsgaard [ctb],
Ren C'deBaca [ctb]

Maintainer Paul Bailey <pbailey@air.org>

Depends R (>= 3.5.0), car, lfactors (>= 1.0.3)

Imports data.table (>= 1.11.4), Formula, glm2, haven, LaF, lme4, MASS,
Matrix, methods, NAEPprimer, quantreg, readxl, tibble, wCorr,
WeMix (>= 3.1.3), xtable

URL https://www.air.org/project/nces-data-r-project-edsurvey

Description
Read in and analyze functions for education survey and assessment data from the National Center for Education Statistics (NCES) <https://nces.ed.gov/>, including National Assessment of Educational Progress (NAEP) data <https://nces.ed.gov/nationsreportcard/> and data from the International Assessment Database: Organisation for Economic Co-operation and Development (OECD) <https://www.oecd.org/>, including Programme for International Student Assessment (PISA), Teaching and Learning International Survey (TALIS), Programme for the International Assessment of Adult Competencies (PIAAC), and International Association for the Evaluation of Educational Achievement (IEA) <https://www.iea.nl/>, including Trends in International Mathematics and Science Study (TIMSS), TIMSS Advanced, Progress in International Reading Literacy Study (PIRLS), International Civic and Citizenship Study (ICCS), International Computer and Information Literacy Study (ICILS), and Civic Education Study (CivEd).

License GPL-2

VignetteBuilder knitr

Suggests dplyr, knitr, testthat, withr, RColorBrewer

LazyData true

ByteCompile true

RoxygenNote 7.0.2

Note This publication was prepared for NCES under Contract No. ED-IES-12-D-0002 with the American Institutes for Research. Mention of trade names, commercial products, or organizations does not imply endorsement by the U.S. Government.

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2020-04-13 19:50:03 UTC

R topics documented:

EdSurvey-package . . . 4
achievementLevels . . . 4
as.data.frame . . . 8
cbind . . . 9
contourPlot . . . 10
cor.sdf . . . 11
dim.edsurvey.data.frame . . . 14
DoFCorrection . . . 15
downloadCivEDICCS . . . 16
downloadECLS_K . . . 17
downloadELS . . . 18
downloadHSLS . . . 19
downloadICILS . . . 20
downloadPIAAC . . . 21
downloadPIRLS . . . 22
downloadPISA . . . 23
downloadTALIS . . . 24
downloadTIMSS . . . 25
downloadTIMSSAdv . . . 26
download_ePIRLS . . . 27
edsurvey.data.frame . . . 28
edsurvey.data.frame.list . . . 32
edsurveyTable . . . 35
edsurveyTable2pdf . . . 38
gap . . . 40
getData . . . 47
getPlausibleValue . . . 50

getWeightJkReplicates 51
glm.sdf 52
hasPlausibleValue 56
isWeight 57
levelsSDF 58
lm.sdf 59
merge 63
mixed.sdf 64
mvrlm.sdf 67
oddsRatio 71
percentile 72
print.achievementLevels 75
print.edsurvey.data.frame 76
print.gap 76
readBB_2001 77
readBB_2003 78
readBB_2012 80
readBPS_1994 81
readBPS_2001 82
readBPS_2009 84
readBPS_2014 85
readBTLS 86
readCivEDICCS 87
readECLS_K1998 89
readECLS_K2011 91
readELS 92
readHSB_Senior 94
readHSB_Sophomore 95
readHSLS 97
readICILS 98
readNAEP 100
readNLS72 102
readPIAAC 103
readPIRLS 104
readPISA 106
readTALIS 108
readTIMSS 109
readTIMSSAdv 111
read_ePIRLS 113
rebindAttributes 115
recode.sdf 116
rename.sdf 117
rq.sdf 118
SD 121
searchSDF 123
showCodebook 124
showCutPoints 125
showPlausibleValues 126

showWeights 127
subset 127
summary2 130
updatePlausibleValue 132
varEstToCov 133
waldTest 134

Index **137**

`EdSurvey-package` *Analysis of NCES Education Survey and Assessment Data*

Description

The `EdSurvey` package uses appropriate methods for analyzing NCES datasets with a small memory footprint. Existing system control files, included with the data, are used to read in and format the data for further processing.

Details

To get started using `EdSurvey`, see the vignettes for tutorials and the statistical methodologies. Use `vignette("introduction",package="EdSurvey")` to see the vignettes.

The package provides functions called `readNAEP`, `readCivEDICCS`, `readICILS`, `readPIAAC`, `readPIRLS`, `read_ePIRLS`, `readPISA`, `readTALIS`, `readTIMSS`, `readTIMSSAdv`, and `readECLS_K2011` to read in NCES datasets. The functions `achievementLevels`, `cor.sdf`, `edsurveyTable`, `summary2`, `lm.sdf`, `logit.sdf`, `mixed.sdf`, `rq.sdf`, `percentile`, and `gap` can then be used to analyze data. For advanced users, `getData` extracts the data of interest as a data frame for further processing.

`achievementLevels` *Achievement Levels*

Description

Returns achievement levels using weights and variance estimates appropriate for the `edsurvey.data.frame`.

Usage

```
achievementLevels(
  achievementVars = NULL,
  aggregateBy = NULL,
  data,
  cutpoints = NULL,
  returnDiscrete = TRUE,
  returnCumulative = FALSE,
  weightVar = NULL,
  jrrIMax = 1,
  omittedLevels = TRUE,
  defaultConditions = TRUE,
  recode = NULL,
  returnNumberOfPSU = FALSE,
  returnVarEstInputs = FALSE
)
```

Arguments

`achievementVars`	character vector indicating variables to be included in the achievement levels table, potentially with a subject scale or subscale. When the subject scale or subscale is omitted, the default subject scale or subscale is used. You can find the default composite scale and all subscales using the function `showPlausibleValues`.
`aggregateBy`	character vector specifying variables by which to aggregate achievement levels. The percentage column sums up to 100 for all levels of all variables specified here. When set to the default of `NULL`, the percentage column sums up to 100 for all levels of all variables specified in `achievementVars`.
`data`	an `edsurvey.data.frame`
`cutpoints`	numeric vector indicating cutpoints. Set to standard NAEP cutpoints for Basic, Proficient, and Advanced by default.
`returnDiscrete`	logical indicating if discrete achievement levels should be returned. Defaults to `TRUE`.
`returnCumulative`	logical indicating if cumulative achievement levels should be returned. Defaults to `FALSE`. The first and last categories are the same as defined for discrete levels.
`weightVar`	character string indicating the weight variable to use. Only the name of the weight variable needs to be included here, and any replicate weights will be automatically included. When this argument is `NULL`, the function uses the default. Use `showWeights` to find the default.
`jrrIMax`	a numeric value. When using the jackknife variance estimation method, the default estimation option, `jrrIMax=1`, uses the sampling variance from the first plausible value as the component for sampling variance estimation. The V_{jrr} term (see *Statistical Methods Used in EdSurvey* for the definition of V_{jrr}) can be estimated with any number of plausible values, and values larger than the number of plausible values on the survey (including `Inf`) will result in all plausible values being used. Higher values of `jrrIMax` lead to longer computing times and more accurate variance estimates.
`omittedLevels`	a logical value. When set to the default value (`TRUE`), it drops those levels in all factor variables that are specified in `achievementVars` and `aggregateBy`. Use `print` on an `edsurvey.data.frame` to see the omitted levels.
`defaultConditions`	a logical value. When set to the default value of `TRUE`, uses the default conditions stored in an `edsurvey.data.frame` to subset the data. Use `print` on an `edsurvey.data.frame` to see the default conditions.
`recode`	a list of lists to recode variables. Defaults to `NULL`. Can be set as `recode = list(var1= list(from=c("a", "b", "c"), to ="d"))`. See Examples.

`returnNumberOfPSU`	a logical value set to `TRUE` to return the number of primary sampling units (PSUs)
`returnVarEstInputs`	a logical value set to `TRUE` to return the inputs to the jackknife and imputation variance estimates, which allows for the computation of covariances between estimates.

Details

The `achievementLevels` function applies appropriate weights and the variance estimation method for each `edsurvey.data.frame`, with several arguments for customizing the aggregation and output of the analysis results. Namely, by using these optional arguments, users can choose to generate the percentage of students performing at each achievement level (discrete), generate the percentage of students performing at or above each achievement level (cumulative), calculate the percentage distribution of students by achievement level (discrete or cumulative) and selected characteristics (specified in `aggregateBy`), and compute the percentage distribution of students by selected characteristics within a specific achievement level.

Calculation of percentages: The details of the methods are shown in the vignette titled Statistical Methods Used in EdSurvey in “Estimation of Weighted Percentages When Plausible Values Are Present” and are used to calculate all cumulative and discrete probabilities.

When the requested achievement levels are discrete (`returnDiscrete = TRUE`), the percentage $\mathcal{A}$ is the percentage of students (within the categories specified in `aggregateBy`) whose scores lie in the range $[cutPoints_i, cutPoints_{i+1}), i = 0, 1, ..., n$. `cutPoints` is the score thresholds provided by the user with $cutPoints_0$ taken to be 0. `cutPoints` are set to NAEP standard cutpoints for achievement levels by default. To aggregate by a specific variable, for example, `dsex`, specify `dsex` in `aggregateBy` and all other variables in `achievementVars`. To aggregate by subscale, specify the name of the subscale (e.g., `num_oper`) in `aggregateBy` and all other variables in `achievementVars`.

When the requested achievement levels are cumulative (`returnCumulative = TRUE`), the percentage $\mathcal{A}$ is the percentage of students (within the categories specified in `aggregateBy`) whose scores lie in the range $[cutPoints_i, \infty)$, $i = 1, 2, ..., n-1$. The first and last categories are the same as defined for discrete levels.

Calculation of standard error of percentages: The method used to calculate the standard error of the percentages is described in the vignette titled Statistical Methods Used in EdSurvey in the sections “Estimation of the Standard Error of Weighted Percentages When Plausible Values Are Present, Using the Jackknife Method” and “Estimation of the Standard Error of Weighted Percentages When Plausible Values Are Not Present, Using the Taylor Series Method.” For “Estimation of the Standard Error of Weighted Percentages When Plausible Values Are Present, Using the Jackknife Method,” the value of `jrrIMax` sets the value of m^*.

Value

A `list` containing up to two data frames, one discrete achievement levels (when `returnDiscrete` is `TRUE`) and one for cumulative achievement levels (when `returnCumulative` is `TRUE`). The `data.frame` contains the following columns:

`Level`	one row for each level of the specified achievement cutpoints

`Variables in achievementVars`	one column for each variable in `achievementVars` and one row for each level of each variable in `achievementVars`
`Percent`	the percentage of students at or above each achievement level aggregated as specified by `aggregateBy`
`StandardError`	the standard error of the percentage, accounting for the survey sampling methodology. See the vignette titled Statistical Methods Used in EdSurvey.
`N`	the number of observations in the incoming data (the number of rows when `omittedLevels` and `defaultConditions` are set to `FALSE`)
`wtdN`	the weighted number of observations in the data
`nPSU`	the number of PSUs at or above each achievement level aggregated as specified by `aggregateBy`. Only returned with `returnNumberOfPSU=TRUE`.

Author(s)

Huade Huo, Ahmad Emad, and Trang Nguyen

References

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys*. New York, NY: Wiley.

Examples

```
## Not run:
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package="NAEPprimer"))

# discrete achievement levels
achievementLevels(achievementVars=c("composite"), aggregateBy=NULL, data=sdf)

# discrete achievement levels with a different subscale
achievementLevels(achievementVars=c("num_oper"), aggregateBy=NULL, data=sdf)

# cumulative achievement levels
achievementLevels(achievementVars=c("composite"), aggregateBy=NULL, data=sdf,
                  returnCumulative=TRUE)

# cumulative achievement levels with a different subscale
achievementLevels(achievementVars=c("num_oper"), aggregateBy=NULL, data=sdf,
                  returnCumulative=TRUE)

# achievement levels as independent variables, by sex aggregated by composite
achievementLevels(achievementVars=c("composite", "dsex"), aggregateBy="composite",
                  data=sdf, returnCumulative=TRUE)

# achievement levels as independent variables, by sex aggregated by sex
achievementLevels(achievementVars=c("composite", "dsex"), aggregateBy="dsex",
                  data=sdf, returnCumulative=TRUE)

# achievement levels as independent variables, by race aggregated by race
```

```
achievementLevels(achievementVars=c("composite", "sdracem"),
                  aggregateBy="sdracem", data=sdf, returnCumulative=TRUE)

# use customized cutpoints
achievementLevels(achievementVars=c("composite"), aggregateBy=NULL, data=sdf,
                  cutpoints = c("Customized Basic" = 200,
                                "Customized Proficient" = 300,
                                "Customized Advanced" = 400))

# use recode to change values for specified variables:
achievementLevels(achievementVars=c("composite", "dsex", "b017451"),
                  aggregateBy = "dsex", sdf,
                  recode=list(b017451=list(from=c("Never or hardly ever",
                                                  "Once every few weeks",
                                                  "About once a week"),
                                           to="Infrequently"),
                              b017451=list(from=c("2 or 3 times a week",
                                                  "Every day"),
                                           to="Frequently")))


## End(Not run)
```

`as.data.frame` *Coerce to a Data Frame*

Description

Function to coerce a `light.edsurvey.data.frame` to a `data.frame`.

Usage

```
## S3 method for class 'light.edsurvey.data.frame'
as.data.frame(x, ...)
```

Arguments

`x`	a `light.edsurvey.data.frame`
`...`	other arguments to be passed to `as.data.frame`

Value

a `data.frame`

Author(s)

Trang Nguyen

cbind — *Combine R Objects by Rows or Columns*

Description

Implements `cbind` and `rbind` for `light.edsurvey.data.frame` class. It takes a sequence of `vector`, `matrix`, `data.frame`, or `light.edsurvey.data.frame` arguments and combines by columns or rows, respectively.

Usage

```
cbind(..., deparse.level = 1)

rbind(..., deparse.level = 1)
```

Arguments

`...`	one or more objects of class `vector`, `data.frame`, `matrix`, or `light.edsurvey.data.frame`
`deparse.level`	integer determining under which circumstances column and row names are built from the actual arguments. See `cbind`.

Details

Because `cbind` and `rbind` are standard generic functions that do not use method dispatch, we set this function as generic, which means it overwrites `base::cbind` and `base::rbind` on loading. If none of the specified elements are of class `light.edsurvey.data.frame`, the function will revert to the standard `base` method. However, to be safe, you might want to explicitly use `base::cbind` when needed after loading the package.

The returned object will contain attributes only from the first `light.edsurvey.data.frame` object in the call to `cbind.light.edsurvey.data.frame`.

Value

a matrix-like object like `matrix` or `data.frame`. Returns a `light.edsurvey.data.frame` if there is at least one `light.edsurvey.data.frame` in the list of arguments.

Author(s)

Trang Nguyen, Michael Lee, and Paul Bailey

See Also

`cbind`

`contourPlot` *Overlaid Scatter and Contour Plots*

Description

Diagnostic plots for regressions can become too dense to interpret. This function helps by adding a contour plot over the points to allow the density of points to be seen, even when an area is entirely covered in points.

Usage

```
contourPlot(
  x,
  y,
  m = 30L,
  xrange,
  yrange,
  xkernel,
  ykernel,
  nlevels = 9L,
  colors = heat.colors(nlevels),
  ...
)
```

Arguments

`x`	numeric vector of the `x` data to be plotted
`y`	numeric vector of the `y` data to be plotted
`m`	integer value of the number of `x` and `y` grid points
`xrange`	numeric vector of length two indicating `x`-range of plot; defaults to range(x)
`yrange`	numeric vector of length two indicating `y`-range of plot; defaults to range(y)
`xkernel`	numeric indicating the standard deviation of Normal `x` kernel to use in generating contour plot
`ykernel`	numeric indicating the standard deviation of Normal `y` kernel to use in generating contour plot
`nlevels`	integer with the number of levels of the contour plot
`colors`	colors to use, specified as in `par`. Defaults to the `heat.colors` with `nlevels`. When specified, `colors` overrides `nlevels`.
`...`	additional arguments to be passed to a plot call that generates the scatter plot and the contour plot

Author(s)

Yuqi Liao and Paul Bailey

Examples

```
## Not run:
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package = "NAEPprimer"))
lm1 <- lm.sdf(composite ~ pared * dsex + sdracem, sdf)
# plot the results
contourPlot(x=lm1$fitted.values,
          y=lm1$residuals[,1], # use only the first plausible value
          m=30,
          xlab="fitted values",
          ylab="residuals",
          main="Figure 1")
# add a line indicating where the residual is zero
abline(0,0)

## End(Not run)
```

`cor.sdf` *Bivariate Correlation*

Description

Computes the correlation of two variables on an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`. The correlation accounts for plausible values and the survey design.

Usage

```
cor.sdf(
  x,
  y,
  data,
  method = c("Pearson", "Spearman", "Polychoric", "Polyserial"),
  weightVar = "default",
  reorder = NULL,
  omittedLevels = TRUE,
  defaultConditions = TRUE,
  recode = NULL,
  condenseLevels = TRUE
)
```

Arguments

`x`	a character variable name from the `data` to be correlated with `y`
`y`	a character variable name from the `data` to be correlated with `x`
`data`	an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`
`method`	a character string indicating which correlation coefficient (or covariance) is to be computed. One of `Pearson` (default), `Spearman`, `Polychoric`, or `Polyserial`.

`weightVar`	character indicating the weight variable to use. See Details section in `lm.sdf`.
`reorder`	a list of variables to reorder. Defaults to `NULL` (no variables are reordered). Can be set as `reorder = list(var1 = c("a","b","c"), var2 = c("4","3","2","1"))`. See Examples.
`omittedLevels`	a logical value. When set to the default value of `TRUE`, drops those levels of all factor variables that are specified in an `edsurvey.data.frame`. Use `print` on an `edsurvey.data.frame` to see the omitted levels.
`defaultConditions`	a logical value. When set to the default value of `TRUE`, uses the default conditions stored in an `edsurvey.data.frame` to subset the data. Use `print` on an `edsurvey.data.frame` to see the default conditions.
`recode`	a list of lists to recode variables. Defaults to `NULL`. Can be set as `recode = list(var1 = list(from = c("a","b","c"), to = "d"))`. See Examples.
`condenseLevels`	a logical value. When set to the default value of `TRUE` and either `x` or `y` is a categorical variable, the function will drop all unused levels and rank the levels of the variable before calculating the correlation. When set to `FALSE`, the numeric levels of the variable remain the same as in the codebook. See Examples.

Details

The `getData` arguments and `recode.sdf` may be useful. (See Examples.) The correlation methods are calculated as described in the documentation for the `wCorr` package—see `browseVignettes(package="wCorr")`.

Value

An `edsurvey.cor` that has print and summary methods.

The class includes the following elements:

`correlation`	numeric estimated correlation coefficient
`Zse`	standard error of the correlation (`Vimp` + `Vjrr`). In the case of Pearson, this is calculated in the linear atanh space and is not a standard error in the usual sense.
`correlates`	a vector of length two showing the columns for which the correlation coefficient was calculated
`variables`	`correlates` that are discrete
`order`	a list that shows the order of each variable
`method`	the type of correlation estimated
`Vjrr`	the jackknife component of the variance estimate. For Pearson, in the atanh space.
`Vimp`	the imputation component of the variance estimate. For Pearson, in the atanh space.
`weight`	the weight variable used
`npv`	the number of plausible values used
`njk`	the number of the jackknife replicates used

Author(s)

Paul Bailey; relies heavily on the `wCorr` package, written by Ahmad Emad and Paul Bailey

See Also

`cor` and `weightedCorr`

Examples

```
## Not run:
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package = "NAEPprimer"))

# for two categorical variables any of the following work
c1_pears <- cor.sdf(x="b017451", y="b003501", data=sdf, method="Pearson",
                    weightVar="origwt")
c1_spear <- cor.sdf(x="b017451", y="b003501", data=sdf, method="Spearman",
                    weightVar="origwt")
c1_polyc <- cor.sdf(x="b017451", y="b003501", data=sdf, method="Polychoric",
                    weightVar="origwt")

c1_pears
c1_spear
c1_polyc

# for categorical variables, users can either keep the original numeric levels of the variables
# or condense the levels (default)
# the following call condenses the levels of the variable 'c046501'
cor.sdf(x="c046501", y="c044006", data=sdf)

# the following call keeps the original levels of the variable 'c046501'
cor.sdf(x="c046501", y="c044006", data=sdf, condenseLevels = FALSE)

# these take awhile to calculate for large datasets, so limit to a subset
sdf_dnf <- subset(sdf, b003601 == 1)

# for a categorical variable and a scale score any of the following work
c2_pears <- cor.sdf(x="composite", y="b017451", data=sdf_dnf, method="Pearson",
                    weightVar="origwt")
c2_spear <- cor.sdf(x="composite", y="b017451", data=sdf_dnf, method="Spearman",
                    weightVar="origwt")
c2_polys <- cor.sdf(x="composite", y="b017451", data=sdf_dnf, method="Polyserial",
                    weightVar="origwt")

c2_pears
c2_spear
c2_polys

# recode two variables
cor.sdf(x="c046501", y="c044006", data=sdf, method="Spearman", weightVar="origwt",
        recode=list(c046501=list(from="0%",to="None"),
                    c046501=list(from=c("1-5%", "6-10%", "11-25%", "26-50%",
                                        "51-75%", "76-90%", "Over 90%"),
                                 to="Between 0% and 100%"),
                    c044006=list(from=c("1-5%", "6-10%", "11-25%", "26-50%",
                                        "51-75%", "76-90%", "Over 90%"),
                                 to="Between 0% and 100%")))

# reorder two variables
cor.sdf(x="b017451", y="sdracem", data=sdf, method="Spearman", weightVar="origwt",
        reorder=list(sdracem=c("White", "Hispanic", "Black", "Asian/Pacific Island",
                               "Amer Ind/Alaska Natv", "Other"),
                     b017451=c("Every day", "2 or 3 times a week", "About once a week",
                               "Once every few weeks", "Never or hardly ever")))

# recode two variables and reorder
cor.sdf(x="pared", y="b013801", data=subset(sdf, !pared %in% "I Don\'t Know"),
        method="Spearman", weightVar = "origwt",
        recode=list(pared=list(from="Some ed after H.S.", to="Graduated H.S."),
                    pared=list(from="Graduated college", to="Graduated H.S."),
                    b013801=list(from="0-10", to="Less than 100"),
                    b013801=list(from="11-25", to="Less than 100"),
                    b013801=list(from="26-100", to="Less than 100")),
        reorder=list(b013801=c("Less than 100", ">100")))

## End(Not run)
```

dim.edsurvey.data.frame

Dimensions of an edsurvey.data.frame or an edsurvey.data.frame.list

Description

Returns the dimensions of an `edsurvey.data.frame` or an `edsurvey.data.frame.list`.

Usage

```
## S3 method for class 'edsurvey.data.frame'
dim(x)
```

Arguments

x	an `edsurvey.data.frame` or an `edsurvey.data.frame.list`

Value

For an `edsurvey.data.frame`, returns a numeric vector of length two, with the first element being the number of rows and the second element being the number of columns.

For an `edsurvey.data.frame.list`, returns a list of length two, where the first element is named `nrow` and is a numeric vector containing the number of rows for each element of the `edsurvey.data.frame.list`. The second element is named `ncol` and is the number of columns for each element. This is done so that the `nrow` and `ncol` functions return meaningful results, even if nonstandard.

Author(s)

Paul Bailey

`DoFCorrection` *Degrees of Freedom*

Description

Calculates the degrees of freedom for a statistic (or of a contrast between two statistics) based on the jackknife and imputation variance estimates.

Usage

```
DoFCorrection(
  varEstA,
  varEstB = varEstA,
  varA,
  varB = varA,
  method = c("WS", "JR")
)
```

Arguments

`varEstA`	the `varEstInput` object returned from certain functions, such as `lm.sdf` when `returnVarEstInputs=TRUE`). The variable `varA` must be on this dataset. See Examples.
`varEstB`	similar to the `varEstA` argument. If left blank, both are assumed to come from `varEstA`. When set, the degrees of freedom are for a contrast between `varA` and `varB`, and the `varB` values are taken from `varEstB`.
`varA`	a character that names the statistic in the `varEstA` argument for which the degrees of freedom calculation is required.
`varB`	a character that names the statistic in the `varEstB` argument for which a covariance is required. When `varB` is specified, returns the degrees of freedom for the contrast between `varA` and `varB`.
`method`	a character that is either `WS` for the Welch-Satterthwaite formula or `JR` for the Johnson-Rust correction to the Welch-Satterthwaite formula

Details

This calculation happens under the notion that statistics have little variance within strata, and some strata will contribute fewer than a full degree of freedom.

The functions are not vectorized, so both `varA` and `varB` must contain exactly one variable name.

The method used to compute the degrees of freedom is in the vignette titled *Statistical Methods Used in EdSurvey* section “Estimation of Degrees of Freedom.”

Value

numeric; the estimated degrees of freedom

Author(s)

Paul Bailey

References

Johnson, E. G., & Rust, K. F. (1992). Population inferences and variance estimation for NAEP data. *Journal of Educational Statistics, 17,* 175–190.

Examples

```
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package="NAEPprimer"))
lm1 <- lm.sdf(composite ~ dsex + b017451, sdf, returnVarEstInputs=TRUE)
summary(lm1)
# this output agrees with summary of lm1 coefficient for dsex
DoFCorrection(lm1$varEstInputs,
              varA="dsexFemale",
              method="JR")
# second example, a covariance term requires more work
# first, estimate the covariance between two regression coefficients
# note that the variable names are parallel to what they are called in lm1 output
covFEveryDay <- varEstToCov(lm1$varEstInputs,
                            varA="dsexFemale",
                            varB="b017451Every day",
                      jkSumMultiplier=EdSurvey:::getAttributes(sdf, "jkSumMultiplier"))
# second, find the difference and the SE of the difference
se <- lm1$coefmat["dsexFemale","se"] + lm1$coefmat["b017451Every day","se"] +
      -2*covFEveryDay
# third, calculate the t-statistic
tv <- (coef(lm1)["dsexFemale"] - coef(lm1)["b017451Every day"])/se
# fourth, calculate the p-value, which requires the estimated degrees of freedom
dofFEveryDay <- DoFCorrection(lm1$varEstInputs,
                              varA="dsexFemale",
                              varB="b017451Every day",
                              method="JR")
# finally, the p-value
2*(1-pt(abs(tv), df=dofFEveryDay))
```

`downloadCivEDICCS` *Instructions for Downloading and Unzipping CivED or ICCS Files*

Description

Provides instructions to download CivED or ICCS data to be processed in `readCivEDICCS`.

Usage

```
downloadCivEDICCS(years = c(1999, 2009))
```

Arguments

`years`	an integer vector indicating the study year. Valid years are 1999 and 2009.

Author(s)

Tom Fink

See Also

`readCivEDICCS`

Examples

```
## Not run:
# view instructions to manually download study data
downloadCivEDICCS()

## End(Not run)
```

`downloadECLS_K` *Download and Unzip ECLS_K Files*

Description

Uses an Internet connection to download ECLS_K data. Data come from nces.ed.gov zip files. This function works for 1998 and 2011 data.

Usage

```
downloadECLS_K(root, years = c(1998, 2011), cache = FALSE, verbose = TRUE)
```

Arguments

`root`	a character string indicating the directory where the ECLS_K data should be stored. Files are placed in a subdirectory named ECLS_K/[year].
`years`	an integer vector of the assessment years to download. Valid years are 1998 and 2011.
`cache`	a logical value set to process and cache the text (.txt) version of files. This takes a very long time but saves time for future uses of the data. Default value is `FALSE`.
`verbose`	a logical value to either print or suppress status message output. The default value is `TRUE`.

Author(s)

Tom Fink

See Also

`readECLS_K1998` and `readECLS_K2011`

Examples

```
## Not run:
# root argument will vary by operating system conventions
downloadECLS_K(years=c(1998, 2011), root = "C:/")

# cache=TRUE will download then process the datafiles
downloadECLS_K(years=c(1998, 2011), root = "C:/", cache = TRUE)

# set verbose=FALSE for silent output
# if year not specified, download all years
downloadECLS_K(root="C:/", verbose = FALSE)

## End(Not run)
```

`downloadELS` *Download and Unzip ELS Files*

Description

Uses an Internet connection to download ELS data. Data come from nces.ed.gov zip files. This function works for 2002 data.

Usage

```
downloadELS(root, years = c(2002), cache = FALSE, verbose = TRUE)
```

Arguments

`root`	a character string indicating the directory where the ELS data should be stored. Files are placed in a subdirectory named ELS/[year].
`years`	an integer vector of the assessment years to download. Valid year is 2002 only.
`cache`	a logical value set to process and cache the text (.txt) version of files. This takes a very long time but saves time for future uses of the data. Default value is `FALSE`.
`verbose`	a logical value to either print or suppress status message output. The default value is `TRUE`.

Author(s)

Tom Fink

See Also

`readELS`

Examples

```
## Not run:
# root argument will vary by operating system conventions
downloadELS(years=2002, root = "C:/")

# cache=TRUE will download then process the datafiles
downloadELS(years=2002, root = "C:/", cache = TRUE)

# set verbose=FALSE for silent output
# if year not specified, download all years
downloadELS(root="C:/", verbose = FALSE)

## End(Not run)
```

`downloadHSLS` *Download and Unzip HSLS Files*

Description

Uses an Internet connection to download HSLS data. Data come from nces.ed.gov zip files. This function works for 2009 data.

Usage

```
downloadHSLS(root, years = c(2009), cache = FALSE, verbose = TRUE)
```

Arguments

`root`	a character string indicating the directory where the HSLS data should be stored. Files are placed in a subdirectory named HSLS/[year].
`years`	an integer vector of the assessment years to download. Valid year is 2009 only.
`cache`	a logical value set to process and cache the text (.txt) version of files. This takes a very long time but saves time for future uses of the data. Default value is `FALSE`.
`verbose`	a logical value to either print or suppress status message output. The default value is `TRUE`.

Author(s)

Tom Fink

See Also

`readHSLS`

Examples

```
## Not run:
# root argument will vary by operating system conventions
downloadHSLS(root = "C:/", years=2009)

# set verbose=FALSE for silent output
# if year not specified, download all years
downloadHSLS(root="C:/", verbose = FALSE)

## End(Not run)
```

---

downloadICILS *Instructions for Downloading and Unzipping ICILS Files*

Description

Provides instructions to download ICILS data to be processed in `readICILS`.

Usage

```
downloadICILS(years = c(2013))
```

Arguments

`years`	an integer vector indicating the study year. Valid year is 2013 only.

Author(s)

Tom Fink

See Also

`readICILS`

Examples

```
## Not run:
# view instructions to manually download study data
downloadICILS()

## End(Not run)
```

`downloadPIAAC` *Download and Unzip PIAAC Files*

Description

Uses an Internet connection to download PIAAC data to a computer. Data come from the OECD website.

Usage

```
downloadPIAAC(root, cycle = 1, cache = FALSE, verbose = TRUE)
```

Arguments

`root`	a character string indicating the directory where the PIAAC data should be stored. Files are placed in a folder named PIAAC/cycle [cycle number].
`cycle`	a numeric value indicating the assessment cycle to download. Valid cycle is 1 only.
`cache`	a logical value set to process and cache the text (.txt) version of files. This takes a very long time but saves time for future uses of the data. Default value is `FALSE`.
`verbose`	a logical value to either print or suppress status message output. The default value is `TRUE`.

Author(s)

Paul Bailey and Trang Nguyen

Examples

```
## Not run:
# download all available data for PIAAC round 1 to "C:/PIAAC/Round 1" folder
# root argument will vary by operating system conventions
downloadPIAAC(root="C:/")

## End(Not run)
```

downloadPIRLS *Download and Unzip PIRLS Files*

Description

Uses an Internet connection to download PIRLS data. Data come from timssandpirls.bc.edu zip files. This function works for 2001, 2006, 2011, and 2016 data.

Usage

```
downloadPIRLS(
  root,
  years = c(2001, 2006, 2011, 2016),
  cache = FALSE,
  verbose = TRUE
)
```

Arguments

`root`	a character string indicating the directory where the PIRLS data should be stored. Files are placed in a subdirectory named PIRLS/[year].
`years`	an integer vector of the assessment years to download. Valid years are 2001, 2006, 2011, and 2016.
`cache`	a logical value set to process and cache the text (.txt) version of files. This takes a very long time but saves time for future uses of the data. Default value is `FALSE`.
`verbose`	a logical value to either print or suppress status message output. The default value is `TRUE`.

Author(s)

Tom Fink

See Also

`readPIRLS`

Examples

```
## Not run:
# root argument will vary by operating system conventions
downloadPIRLS(year=c(2006, 2011), root = "C:/")

# cache=TRUE will download then process the datafiles
downloadPIRLS(year=2011, root = "C:/", cache = TRUE)

# set verbose=FALSE for silent output
# if year not specified, download all years
downloadPIRLS(root="C:/", verbose = FALSE)

## End(Not run)
```

`downloadPISA` *Download and Unzip PISA Files*

Description

Uses an Internet connection to download PISA data to a computer. Data come from the OECD website.

Usage

```
downloadPISA(
  root,
  years = c(2000, 2003, 2006, 2009, 2012, 2015),
  database = c("INT", "CBA", "FIN"),
  cache = FALSE,
  verbose = TRUE
)
```

Arguments

`root`	a character string indicating the directory where the PISA data should be stored. Files are placed in a folder named PISA/[year].
`years`	an integer vector of the assessment years to download. Valid years are 2000, 2003, 2006, 2009, 2012, and 2015.
`database`	a character vector to indicate which database to download from. For 2012, three databases are available (`INT` = International, `CBA` = Computer-Based Assessment, and `FIN` = Financial Literacy). For other years, only `INT` is available (for example, if PISA 2015 financial literacy is to be downloaded, the database argument should be set to `INT`). Defaults to `INT`.
`cache`	a logical value set to process and cache the text (.txt) version of files. This takes a very long time but saves time for future uses of the data. Default value is `FALSE`.
`verbose`	a logical value to either print or suppress status message output. The default value is `TRUE`.

Details

The function uses `download.file` to download files from provided URLs. Some machines might require a different user agent in HTPP(S) requests. If the downloading gives an error or behaves unexpectedly (e.g., a zip file cannot be unzipped or a data file is significantly smaller than expected), users can toggle `HTTPUserAgent` options to find one that works for their machines. One common alternative option is

`options(HTTPUserAgent="Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0")`

Author(s)

Yuqi Liao, Paul Bailey, and Trang Nguyen

See Also

`readPISA`, `download.file`, `options`

Examples

```
## Not run:
# download PISA 2012 data (for all three databases)
downloadPISA(years = 2012, database = c("INT","CBA","FIN"), root="C:/")

# download PISA 2009, 2012, and 2015 data (International Database only)
# to C:/PISA/2009, C:/PISA/2012, and C:/PISA/2015 folders, respectively
downloadPISA(years = c(2009,2012,2015), root="C:/")

## End(Not run)
```

`downloadTALIS` *Instructions for Downloading TALIS Files*

Description

Provides instructions to download TALIS data to be processed in `readTALIS`.

Usage

```
downloadTALIS(years)
```

Arguments

`years`	a numeric value indicating the assessment year. Available years are 2008 and 2013.

Author(s)

Trang Nguyen

See Also

`readTALIS`

Examples

```
## Not run:
# print out downloading instructions for TALIS 2008 database
downloadTALIS(2008)

## End(Not run)
```

`downloadTIMSS` *Download and Unzip TIMSS Files*

Description

Uses an Internet connection to download TIMSS data. Data come from timssandpirls.bc.edu zip files. This function works for 2003, 2007, 2011, and 2015 data.

Usage

```
downloadTIMSS(
  root,
  years = c(2003, 2007, 2011, 2015),
  cache = FALSE,
  verbose = TRUE
)
```

Arguments

`root`	a character string indicating the directory where the TIMSS data should be stored. Files are placed in a subdirectory named TIMSS/[year].
`years`	an integer vector of the assessment years to download. Valid years are 2003, 2007, 2011, and 2015.
`cache`	a logical value set to process and cache the text (.txt) version of files. This takes a very long time but saves time for future uses of the data. Default value is `FALSE`.
`verbose`	a logical value to either print or suppress status message output. The default value is `TRUE`.

Author(s)

Tom Fink

See Also

`readTIMSS`

Examples

```
## Not run:
# root argument will vary by operating system conventions
downloadTIMSS(year=c(2015, 2011), root = "C:/")

# cache=TRUE will download then process the datafiles
downloadTIMSS(year=2015, root = "C:/", cache = TRUE)

# set verbose=FALSE for silent output
# if year not specified, download all years
downloadTIMSS(root="C:/", verbose = FALSE)

## End(Not run)
```

---

downloadTIMSSAdv *Download and Unzip TIMSS Advanced Files*

Description

Uses an Internet connection to download TIMSS Advanced data. Data come from timssandpirls.bc.edu zip files. This function works for 1995, 2008, and 2015 data.

Usage

```
downloadTIMSSAdv(
  root,
  years = c(1995, 2008, 2015),
  cache = FALSE,
  verbose = TRUE
)
```

Arguments

`root`	a character string indicating the directory where the TIMSS Advanced data should be stored. Files are placed in a subdirectory named TIMSSAdv/[year].
`years`	an integer vector of the assessment years to download. Valid years are 1995, 2008, and 2015.
`cache`	a logical value set to process and cache the text (.txt) version of files. This takes a very long time but saves time for future uses of the data. Default value is `FALSE`.
`verbose`	a logical value to either print or suppress status message output. The default value is `TRUE`.

Author(s)

Tom Fink

See Also

`readTIMSSAdv`

Examples

```
## Not run:
# root argument will vary by operating system conventions
downloadTIMSSAdv(year=c(2008, 2015), root = "C:/")

# cache=TRUE will download then process the datafiles
downloadTIMSSAdv(year=2015, root = "C:/", cache = TRUE)

# set verbose=FALSE for silent output
# if year not specified, download all years
downloadTIMSSAdv(root="C:/", verbose = FALSE)

## End(Not run)
```

`download_ePIRLS` *Download and Unzip ePIRLS Files*

Description

Uses an Internet connection to download ePIRLS data. Data come from timssandpirls.bc.edu zip files. This function works for 2016 data.

Usage

```
download_ePIRLS(root, years = c(2016), cache = FALSE, verbose = TRUE)
```

Arguments

`root`	a character string indicating the directory where the ePIRLS data should be stored. Files are placed in a subdirectory named ePIRLS/[year].
`years`	an integer vector of the assessment years to download. Valid year is 2016 only.
`cache`	a logical value set to process and cache the text (.txt) version of files. This takes a very long time but saves time for future uses of the data. Default value is `FALSE`.
`verbose`	a logical value to either print or suppress status message output. The default value is `TRUE`.

Author(s)

Tom Fink

See Also

`read_ePIRLS`

Examples

```
## Not run:
# root argument will vary by operating system conventions
download_ePIRLS(years=2016, root = "C:/")

# cache=TRUE will download then process the datafiles
download_ePIRLS(years=2016, root = "C:/", cache = TRUE)

# set verbose=FALSE for silent output
# if year not specified, download all years
download_ePIRLS(root="C:/", verbose = FALSE)

## End(Not run)
```

`edsurvey.data.frame` *EdSurvey Class Constructors and Helpers*

Description

Two new classes in `EdSurvey` are described in this section: the `edsurvey.data.frame` and `light.edsurvey.data.frame`. The `edsurvey.data.frame` class stores metadata about survey data, and data are stored on the disk (via the `LaF` package), allowing gigabytes of data to be used easily on a machine otherwise inappropriate for manipulating large datasets. The `light.edsurvey.data.frame` is typically generated by the `getData` function and stores the data in a `data.frame`. Both classes use attributes to manage metadata and allow for correct statistics to be used in calculating results; the `getAttributes` acts as an accessor for these attributes, whereas `setAttributes` acts as a mutator for the attributes. As a convenience, `edsurvey.data.frame` implements the `$` function to extract a variable.

Usage

```
edsurvey.data.frame(
  userConditions,
  defaultConditions,
  dataList = list(),
  weights,
  pvvars,
  subject,
  year,
  assessmentCode,
  dataType,
  gradeLevel,
  achievementLevels,
  omittedLevels,
  survey,
  country,
  psuVar,
  stratumVar,
  jkSumMultiplier,
  recodes = NULL,
  validateFactorLabels = FALSE,
  forceLower = TRUE,
  reqDecimalConversion = TRUE
)

## S3 method for class 'edsurvey.data.frame'
x$i

## S3 replacement method for class 'edsurvey.data.frame'
x$name <- value

getAttributes(data, attribute = NULL)

setAttributes(data, attribute, value)

getPSUVar(data, weightVar = NULL)

getStratumVar(data, weightVar = NULL)
```

Arguments

`userConditions`	a list of user conditions that includes subsetting or recoding conditions
`defaultConditions`	a list of default conditions that often are set for each survey
`dataList`	a list of `dataListItem` objects to model the data structure of the survey
`weights`	a list that stores information regarding weight variables. See Details.
`pvvars`	a list that stores information regarding plausible values. See Details.
`subject`	a character that indicates the subject domain of the given data
`year`	a character or numeric that indicates the year of the given data
`assessmentCode`	a character that indicates the code of the assessment. Can be `National` or `International`.
`dataType`	a character that indicates the unit level of the main data. Examples include `Student`, `teacher`, `school`, `Adult Data`.
`gradeLevel`	a character that indicates the grade level of the given data
`achievementLevels`	a list of achievement-level categories and cutpoints
`omittedLevels`	a list of default omitted levels for the given data
`survey`	a character that indicates the name of the survey
`country`	a character that indicates the country of the given data
`psuVar`	a character that indicates the PSU sampling unit variable. Ignored when weights have `psuVar` defined.
`stratumVar`	a character that indicates the stratum variable. Ignored when weights have `stratumVar` defined.

`jkSumMultiplier`	a numeric value of the jackknife coefficient (used in calculating the jackknife replication estimation)
`recodes`	a list of variable recodes of the given data
`validateFactorLabels`	a Boolean that indicates whether the `getData` function needs to validate factor variables
`forceLower`	a Boolean; when set to `TRUE`, will automatically lowercase variable names
`reqDecimalConversion`	a Boolean; when set to `TRUE`, a `getData` call will multiply the raw file value by a decimal multiplier
`x`	an `edsurvey.data.frame`
`i`	a character, the column name to extract
`name`	a character vector of the column to edit
`value`	outside of the assignment context, new value of the given `attribute`
`data`	an `edsurvey.data.frame`
`attribute`	a character, name of an attribute to get or set
`weightVar`	a character indicating the full sample weights

Details

The `weight` list has an element named after each weight variable name that is a list with elements `jkbase` and `jksuffixes`. The `jkbase` variable is a single character indicating the jackknife replicate weight base name, whereas `jksuffixes` is a vector with one element for each jackknife replicate weight. When the two are pasted together, they should form the complete set of the jackknife replicate weights. The `weights` argument also can have an attribute that is the default weight. If the primary sampling unit and stratum variables change by weight, they also can be defined on the weight list as `psuVar` and `stratumVar`. When this option is used, it overrides the `psuVar` and `stratumVar` on the `edsurvey.data.frame`, which can be left blank. A weight must define only one of `psuVar` and `stratumVar`.

The `pvvars` list has an element for each subject or subscale score that has plausible values. Each element is a list with a `varnames` element that indicates the column names of the plausible values and an `achievementLevel` argument that is a named vector of the achievement-level cutpoints.

Value

An object of class `edsurvey.data.frame` with the following elements:

Elements that store data connections and data codebooks

`dataList`	a `list` object containing the surveys `dataListItem` objects

Elements that store sample design and default subsetting information of the given survey data

`userConditions`	a list containing all user conditions, set using the `subset.edsurvey.data.frame` method
`defaultConditions`	the default subsample conditions

`weights`	a list containing the weights. See Details.
`stratumVar`	a character that indicates the default strata identification variable name in the data. Often used in Taylor series estimation.
`psuVar`	a character that indicates the default PSU (sampling unit) identification variable name in the data. Often used in Taylor series estimation.
`pvvars`	a list containing the plausible values. See Details.
`achievementLevels`	default achievement cutoff scores and names. See Details.
`omittedLevels`	the levels of the factor variables that will be omitted from the `edsurvey.data.frame`

Elements that store descriptive information of the survey

`survey`	the type of survey data
`subject`	the subject of the data
`year`	the year of assessment
`assessmentCode`	the assessment code
`dataType`	the type of data (e.g., `student` or `school`)
`gradeLevel`	the grade of the dataset contained in the `edsurvey.data.frame`

EdSurvey Classes

`edsurvey.data.frame` is an object that stores connection to data on the disk along with important survey sample design information.

`edsurvey.data.frame.list` is a list of `edsurvey.data.frame` objects. It often is used in trend or cross-regional analysis in the `gap` function. See `edsurvey.data.frame.list` for more information on how to create an `edsurvey.data.frame.list`. Users also can refer to the vignette titled *Using EdSurvey for Trend Analysis* for examples.

Besides `edsurvey.data.frame` class, the `EdSurvey` package also implements the `light.edsurvey.data.frame` class, which can be used by both `EdSurvey` and non-`EdSurvey` functions. More particularly, `light.edsurvey.data.frame` is a `data.frame` that has basic survey and sample design information (i.e., plausible values and weights), which will be used for variance estimation in analytical functions. Because it also is a base R `data.frame`, users can apply base R functions for data manipulation. See the vignette titled *Using the* `getData` *Function in EdSurvey* for more examples.

Many functions will remove attributes from a data frame, such as a `light.edsurvey.data.frame`, and the `rebindAttributes` function can add them back.

Users can get a `light.edsurvey.data.frame` object by using the `getData` method with `addAttributes=TRUE`.

Basic Methods for EdSurvey Classes

Extracting a column from an `edsurvey.data.frame`

Users can extract a column from an `edsurvey.data.frame` object using `$` or `[]` like a normal data frame.

Extracting and updating attributes of an object of class `edsurvey.data.frame` *or* `light.edsurvey.data.frame`

Users can use the `getAttributes` method to extract any attribute of an `edsurvey.data.frame` or a `light.edsurvey.data.frame`. A `light.edsurvey.data.frame` will not have attributes related to data connection because data have already been read in memory.

If users want to update an attribute (i.e., `omittedLevels`), they can use the `setAttributes` method.

Author(s)

Tom Fink, Trang Nguyen, and Paul Bailey

See Also

`rebindAttributes`

Examples

```
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package="NAEPprimer"))

# run a base R function on a column of edsurvey.data.frame
table(sdf$dsex)
# assignment
table(sdf$b013801)
sdf$books <- ifelse(sdf$b013801 %in% c("0-10", "11-25"), "0-25 books", "26+ books")
table(sdf$books, sdf$b013801)

# extract default omitted levels of NAEP primer data
getAttributes(sdf, "omittedLevels") #[1] "Multiple" NA         "Omitted"

# update default omitted levels of NAEP primer data
sdf <- setAttributes(sdf, "omittedLevels", c("Multiple", "Omitted", NA, "(Missing)"))
getAttributes(sdf, "omittedLevels") #[1] "Multiple"  "Omitted"   NA          "(Missing)"
```

`edsurvey.data.frame.list`

EdSurvey Dataset Vectorization

Description

The `edsurvey.data.frame.list` function creates an `edsurvey.data.frame.list` object from a series of `edsurvey.data.frame` objects. `append.edsurvey.data.frame.list` creates an `edsurvey.data.frame.list` from two `edsurvey.data.frame` or `edsurvey.data.frame.list` objects.

An `edsurvey.data.frame.list` is useful for looking at data, for example, across time or graphically, and reduces repetition in function calls. The user may specify a variable that varies across the `edsurvey.data.frame` objects that is then included in further output.

Usage

```
edsurvey.data.frame.list(datalist, cov = NULL, labels = NULL)

append.edsurvey.data.frame.list(sdfA, sdfB, labelsA = NULL, labelsB = NULL)
```

Arguments

`datalist`	a list of `edsurvey.data.frames` to be combined
`cov`	a character vector that indicates what varies across the `edsurvey.data.frame` objects. Guessed if not supplied. For example, if several `edsurvey.data.frames` for several different countries are supplied, then `cov` would be set to the country.
`labels`	a character vector that specifies labels. Must be the same length as `datalist`. Not needed if `cov` exists or can be guessed. See Examples.
`sdfA`	an `edsurvey.data.frame` or an `edsurvey.data.frame.list` to be combined
`sdfB`	an `edsurvey.data.frame` or an `edsurvey.data.frame.list` to be combined
`labelsA`	a character vector that specifies `labels` for `sdfA` when creating the new `edsurvey.data.frame.list`. `labelsA` would be ignored if `sdfA` is an `edsurvey.data.frame.list` with labels supplied.
`labelsB`	a character vector that specifies `labels` for `sdfB` when creating the new `edsurvey.data.frame.list`. `labelsB` would be ignored if `sdfB` is an `edsurvey.data.frame.list` with labels supplied.

Details

The `edsurvey.data.frame.list` can be used in place of an `edsurvey.data.frame` in function calls, and results are returned for each of the component `edsurvey.data.frames`, with the organization of the results varying by the particular method.

An `edsurvey.data.frame.list` can be created from several `edsurvey.data.frame` objects that are related; for example, all are NAEP mathematics assessments but have one or more differences (e.g., they are all from different years). Another example could be data from multiple countries for an international assessment.

When `cov` and `labels` are both missing, `edsurvey.data.frame.list` attempts to guess what variables may be varying and uses those. When there are no varying covariates, generic labels are automatically generated.

Value

`edsurvey.data.frame.list` returns an `edsurvey.data.frame.list` with elements

`datalist`	a list of `edsurvey.data.frame` objects
`covs`	a character vector of key variables that vary within the `edsurvey.data.frame.list`. When labels are included, they will be included in `covs`. In the unusual circumstance that `sdfA` or `sdfB` is an `edsurvey.data.frame.list` has `covs`, and labels are not supplied, the `covs` are simply pasted together with colons between them.

`append.edsurvey.data.frame.list` returns an `edsurvey.data.frame.list` with elements

`datalist`	a list of `edsurvey.data.frame` objects
`covs`	a character vector of key variables that vary within the `edsurvey.data.frame.list`. When labels are included, they will be included in `covs`.

Author(s)

Paul Bailey, Huade Huo

Examples

```
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package="NAEPprimer"))

# NOTE: the following code would not normally have to be run but is used here
# to generate demo data.
# Specifically, make subsets of sdf by the scrpsu variable,
# "Scrambled PSU and school code"
sdfA <- subset(sdf, scrpsu %in% c(5,45,56))
sdfB <- subset(sdf, scrpsu %in% c(75,76,78))
sdfC <- subset(sdf, scrpsu %in% 100:200)
sdfD <- subset(sdf, scrpsu %in% 201:300)

# construct an edsurvey.data.frame.list from these four data sets
sdfl <- edsurvey.data.frame.list(list(sdfA, sdfB, sdfC, sdfD),
                                 labels=c("A locations",
                                          "B locations",
                                          "C locations",
                                          "D locations"))


# this shows how these datasets will be described
sdfl$covs
## Not run:
# get the gaps between Male and Female for each data set
gap1 <- gap("composite", sdfl, dsex=="Male", dsex=="Female")
gap1

## End(Not run)

# make combine sdfA and sdfB
sdfl1a <- edsurvey.data.frame.list(list(sdfA, sdfB),
                                   labels=c("A locations",
                                            "B locations"))

# combine sdfC and sdfD
sdfl1b <- edsurvey.data.frame.list(list(sdfC, sdfD),
                                   labels=c("C locations",
                                            "D locations"))

# append to make sdf3 the same as sdfl
sdfl3 <- append.edsurvey.data.frame.list(sdfl1a, sdfl1b)
identical(sdfl, sdfl3) #TRUE

# append to make sdf4 the same as sdfl
sdfl4 <- append.edsurvey.data.frame.list(
  append.edsurvey.data.frame.list(sdfl1a, sdfC, labelsB = "C locations"),
  sdfD,
  labelsB = "D locations")
identical(sdfl, sdfl4) #TRUE
```

`edsurveyTable` *EdSurvey Tables With Conditional Means*

Description

Returns a summary table (as a `data.frame`) that shows the number of students, the percentage of students, and the mean value of the outcome (or left-hand side) variable by the predictor (or right-hand side) variable(s).

Usage

```
edsurveyTable(
  formula,
  data,
  weightVar = NULL,
  jrrIMax = 1,
  pctAggregationLevel = NULL,
  returnMeans = TRUE,
  returnSepct = TRUE,
  varMethod = c("jackknife", "Taylor"),
  drop = FALSE,
  omittedLevels = TRUE,
  defaultConditions = TRUE,
  recode = NULL,
  returnVarEstInputs = FALSE
)
```

Arguments

`formula`	object of class `formula`, potentially with a subject scale or subscale on the left-hand side and variables to tabulate on the right-hand side. When the left-hand side of the formula is omitted and `returnMeans` is `TRUE`, then the default subject scale or subscale is used. You can find the default composite scale and all subscales using the function `showPlausibleValues`. Note that the order of the right-hand side variables affects the output.
`data`	object of class `edsurvey.data.frame`. See `readNAEP` for how to generate an `edsurvey.data.frame`.
`weightVar`	character string indicating the weight variable to use. Note that only the name of the weight variable needs to be included here, and any replicate weights will be automatically included. When this argument is `NULL`, the function uses the default. Use `showWeights` to find the default.

`jrrIMax`	a numeric value; when using the jackknife variance estimation method, the default estimation option, `jrrIMax=1`, uses the sampling variance from the first plausible value as the component for sampling variance estimation. The V_{jrr} term (see the Details section of `lm.sdf` to see the definition of V_{jrr}) can be estimated with any number of plausible values, and values larger than the number of plausible values on the survey (including `Inf`) will result in all of the plausible values being used. Higher values of `jrrIMax` lead to longer computing times and more accurate variance estimates.
`pctAggregationLevel`	the percentage variable sums up to 100 for the first `pctAggregationLevel` columns. So, when set to `0`, the `PCT` column adds up to 1 across the entire sample. When set to `1`, the `PCT` column adds up to 1 within each level of the first variable on the right-hand side of the formula; when set to `2`, then the percentage adds up to 100 within the interaction of the first and second variable, and so on. Default is `NULL`, which will result in the lowest feasible aggregation level. See Examples section.
`returnMeans`	a logical value; set to `TRUE` (the default) to get the `MEAN` and `SE(MEAN)` columns in the returned table described in the Value section.
`returnSepct`	set to `TRUE` (the default) to get the `SEPCT` column in the returned table described in the Value section.
`varMethod`	a character set to `jackknife` or `Taylor` that indicates the variance estimation method to be used.
`drop`	a logical value. When set to the default value of `FALSE`, when a single column is returned, it is still represented as a `data.frame` and is not converted to a vector.
`omittedLevels`	a logical value. When set to the default value of `TRUE`, drops those levels of all factor variables that are specified in an `edsurvey.data.frame`. Use `print` on an `edsurvey.data.frame` to see the omitted levels.
`defaultConditions`	a logical value. When set to the default value of `TRUE`, uses the default conditions stored in an `edsurvey.data.frame` to subset the data. Use `print` on an `edsurvey.data.frame` to see the default conditions.
`recode`	a list of lists to recode variables. Defaults to `NULL`. Can be set as `recode = list(var1 = list(from = c("a","b","c"), to = "c"))`.
`returnVarEstInputs`	a logical value set to `TRUE` to return the inputs to the jackknife and imputation variance estimates, which allows for the computation of covariances between estimates.

Details

This method can be used to generate a simple one-way, two-way, or *n*-way table with unweighted and weighted *n* values and percentages. It also can calculate the average of the subject scale or subscale for students at each level of the cross-tabulation table.

A detailed description of all statistics is given in the vignette titled *Statistical Methods Used in EdSurvey*.

Value

A table with the following columns:

`RHS levels`	one column for each right-hand side variable. Each row regards students who are at the levels shown in that row.
`N`	count of the number of students in the survey in the `RHS levels`
`WTD_N`	the weighted *N* count of students in the survey in `RHS levels`
`PCT`	the percentage of students at the aggregation level specified by `pctAggregationLevel` (see Arguments). See the vignette titled *Statistical Methods Used in EdSurvey* in the section "Estimation of Weighted Percentages" and its first subsection "Estimation of Weighted Percentages When Plausible Values Are Not Present."
`SE(PCT)`	the standard error of the percentage, accounting for the survey sampling methodology. When `varMethod` is the `jackknife`, the calculation of this column is described in the vignette titled *Statistical Methods Used in EdSurvey* in the section "Estimation of the Standard Error of Weighted Percentages When Plausible Values Are Not Present, Using the Jackknife Method." When `varMethod` is set to `Taylor`, the calculation of this column is described in "Estimation of the Standard Error of Weighted Percentages When Plausible Values Are Not Present, Using the Taylor Series Method."
`MEAN`	the mean assessment score for units in the `RHS levels`, calculated according to the vignette titled *Statistical Methods Used in EdSurvey* in the section "Estimation of Weighted Means When Plausible Values Are Present."
`SE(MEAN)`	the standard error of the `MEAN` column (the mean assessment score for units in the `RHS levels`), calculated according to the vignette titled *Statistical Methods Used in EdSurvey* in the sections "Estimation of Standard Errors of Weighted Means When Plausible Values Are Present, Using the Jackknife Method" or "Estimation of Standard Errors of Weighted Means When Plausible Values Are Present, Using the Taylor Series Method," depending on the value of `varMethod`.

When `returnVarEstInputs` is `TRUE`, two additional elements are returned. These are `meanVarEstInputs` and `pctVarEstInputs` and regard the `MEAN` and `PCT` columns, respectively. These two objects can be used for calculating covariances with `varEstToCov`.

Author(s)

Paul Bailey and Ahmad Emad

References

Binder, D. A. (1983). On the variances of asymptotically normal estimators from complex surveys. *International Statistical Review*, *51*(3), 279–292.

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys*. New York, NY: Wiley.

Examples

```
## Not run:
# read in the example data (generated, not real student data)
```

```
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package = "NAEPprimer"))

# create a table that shows only the breakdown of dsex
edsurveyTable(composite ~ dsex, data=sdf, returnMeans=FALSE, returnSepct=FALSE)

# create a table with composite scores by dsex
edsurveyTable(composite ~ dsex, data=sdf)

# add a second variable
edsurveyTable(composite ~ dsex + b017451, data=sdf)

# add a second variable, do not omit any levels
edsurveyTable(composite ~ dsex + b017451 + b003501, data=sdf, omittedLevels=FALSE)

# add a second variable, do not omit any levels, change aggregation level
edsurveyTable(composite ~ dsex + b017451 + b003501, data=sdf, omittedLevels=FALSE,
            pctAggregationLevel=0)

edsurveyTable(composite ~ dsex + b017451 + b003501, data=sdf, omittedLevels=FALSE,
            pctAggregationLevel=1)

edsurveyTable(composite ~ dsex + b017451 + b003501, data=sdf, omittedLevels=FALSE,
            pctAggregationLevel=2)

# variance estimation using the Taylor series
edsurveyTable(composite ~ dsex + b017451 + b003501, data=sdf, varMethod="Taylor")

## End(Not run)
```

edsurveyTable2pdf *PDF File From an edsurveyTable*

Description

Produces the LaTeX code and compiles to a PDF file from the `edsurveyTable` results.

Usage

```
edsurveyTable2pdf(
  data,
  formula,
  caption = NULL,
  filename = "",
  toCSV = "",
  returnMeans = TRUE,
  estDigits = 2,
  seDigits = 3
)
```

Arguments

`data`	the result of a call to `edsurveyTable`
`formula`	a formula of the form `LHS ~ RHS` to cast the `edsurveyTable` results from long format to wide format. This formula takes the form `LHS ~ RHS` (e.g., `var1 + var2 ~ var3`). The order of the entries in the formula is essential.
`caption`	character vector of length one or two containing the table's caption or title. If the length is two, the second item is the "short caption" used when LaTeX generates a `List of Tables`. Set to `NULL` to suppress the caption. Default value is `NULL`.
`filename`	a character string containing filenames and paths. By default (`filename = ""`), table will be saved in the working directory (`getwd()`). Use `filename = "CONSOLE"` to print LaTeX code in R console without generating a PDF file.
`toCSV`	a character string containing filenames and paths of .csv table output. `""` indicates no .csv output. `toCSV` is independent to `filename`, so both a csv file and PDF file would be generated if both `filename` and `toCSV` were specified.
`returnMeans`	a logical value set to `TRUE` (the default) to generate a PDF with the `MEAN` and `SE(MEAN)`. It is set to `FALSE` to generate a PDF with the `PCT` and `SE(PCT)`. See Value in `edsurveyTable`.
`estDigits`	an integer indicating the number of decimal places to be used for estimates. Negative values are allowed. See Details.
`seDigits`	an integer indicating the number of decimal places to be used for standard errors. Negative values are allowed.

Details

Rounding to a negative number of digits means rounding to a power of 10, so, for example, `estDigits = -2` rounds estimates to the nearest hundred.

Note

For more details, see the vignette titled *Producing* `LaTeX` *Tables From* `edsurveyTable` *Results With* `edsurveyTable2pdf`.

Author(s)

Huade Huo

Examples

```
## Not run:
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package="NAEPprimer"))

# create a table with composite scores by dsex and b017451
est1 <- edsurveyTable(composite ~ dsex + b017451, sdf)

# create a table with csv output
edsurveyTable2pdf(data = est1,
                  formula = b017451~dsex,
                  toCSV = "C:/example table.csv",
                  filename = "C:/example table.pdf",
                  returnMeans = FALSE)

# create a pdf file using the default subject scale or subscale
# and keep two digits for estimates and three digits for SE after decimal point
edsurveyTable2pdf(est1, b017451~dsex,
                  returnMeans = TRUE, estDigits = 2, seDigits = 3)

# create a pdf file using the percentage of students at the
# aggregation level specified by \code{pctAggregationLevel}
# output will be saved as "C:/example table.pdf"
edsurveyTable2pdf(est1,
                  b017451~dsex,
                  "C:/example table.pdf",
                  returnMeans = FALSE)

## End(Not run)
```

gap *Gap Analysis*

Description

Compares the average levels of a variable between two groups that potentially share members.

Usage

```
gap(
  variable,
  data,
  groupA = "default",
  groupB = "default",
  percentiles = NULL,
  achievementLevel = NULL,
  achievementDiscrete = FALSE,
  stDev = FALSE,
  targetLevel = NULL,
  weightVar = NULL,
  jrrIMax = 1,
  varMethod = c("jackknife"),
  omittedLevels = TRUE,
  defaultConditions = TRUE,
  recode = NULL,
  referenceDataIndex = 1,
  returnVarEstInputs = FALSE,
  returnSimpleDoF = FALSE,
  returnSimpleN = FALSE,
  returnNumberOfPSU = FALSE,
  noCov = FALSE,
  pctMethod = c("unbiased", "simple"),
  includeLinkingError = FALSE
)
```

Arguments

`variable`	a character indicating the variable to be compared, potentially with a subject scale or subscale
`data`	an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`
`groupA`	an expression or character expression that defines a condition for the subset. This subset will be compared to `groupB`. If not specified, it will define a whole sample as in `data`.
`groupB`	an expression or character expression that defines a condition for the subset. This subset will be compared to `groupA`. If not specified, it will define a whole sample as in `data`. If set to `NULL`, estimates for the second group will be dropped.
`percentiles`	a numeric vector. The `gap` function calculates the mean when this argument is omitted or set to `NULL`. Otherwise, the gap at the percentile given is calculated.
`achievementLevel`	the achievement level(s) at which percentages should be calculated
`achievementDiscrete`	a logical indicating if the achievement level specified in the `achievementLevel` argument should be interpreted as discrete so that just the percentage in that particular achievement level will be included. Defaults to `FALSE` so that the percentage at or above that achievement level will be included in the percentage.
`stDev`	a logical, set to `TRUE` to calculate the gap in standard deviations.
`targetLevel`	a character string. When specified, calculates the gap in the percentage of students at `targetLevel` in the `variable` argument. This is useful for comparing the gap in the percentage of students at a survey response level.
`weightVar`	a character indicating the weight variable to use. See Details.
`jrrIMax`	a numeric value; when using the jackknife variance estimation method, the default estimation option, `jrrIMax=1`, uses the sampling variance from the first plausible value as the component for sampling variance estimation. The `Vjrr` term, or sampeling variance term, can be estimated with any number of plausible values, and values larger than the number of plausible values on the survey (including `Inf`) will result in all plausible values being used. Higher values of `jrrIMax` lead to longer computing times and more accurate variance estimates.
`varMethod`	deprecated parameter, `gap` always uses the jackknife variance estimation
`omittedLevels`	a logical value. When set to the default value of `TRUE`, drops those levels of all factor variables. Use `print` on an `edsurvey.data.frame` to see the omitted levels.
`defaultConditions`	a logical value. When set to the default value of `TRUE`, uses the default conditions stored in `edsurvey.data.frame` to subset the data. Use `print` on an `edsurvey.data.frame` to see the default conditions.

`recode`	a list of lists to recode variables. Defaults to `NULL`. Can be set as `recode = list(var1 = list(from = c("a", "b", "c"), to = "d"))`.
`referenceDataIndex`	a numeric used only when the `data` argument is an `edsurvey.data.frame.list`, indicating which dataset is the reference dataset that other datasets are compared with. Defaults to 1.
`returnVarEstInputs`	a logical value; set to `TRUE` to return the inputs to the jackknife and imputation variance estimates which allows for the computation of covariances between estimates.
`returnSimpleDoF`	a logical value set to `TRUE` to return the degrees of freedom for some statistics (see Value section) that do not have a *t*-test; useful primarily for further computation
`returnSimpleN`	a logical value set to `TRUE` to add the count (*n*-size) of observations included in groups A and B in the percentage object
`returnNumberOfPSU`	a logical value set to `TRUE` to return the number of PSUs used in the calculation
`noCov`	set the covariances to zero in result
`pctMethod`	a character that is one of `unbiased` or `simple`. See the help for `percentile` for more information.
`includeLinkingError`	a logical value set to `TRUE` to include the linking error in variance estimation. Standard errors (e.g., `diffAAse`, `diffBBse`, and `diffABABse`) and *p*-values (e.g., `diffAApValue`, `diffBBpValue`, and `diffABABpValue`) would be adjusted for comparisons between digitally based assessments (DBA) and paper-based assessments (PBA) data. This option is supported only for NAEP data.

Details

This function calculates the gap between `groupA` and `groupB` (which may be omitted to indicate the full sample). The gap is calculated for one of four statistics:

the gap in means The mean score gap (in the score variable) identified in the `variable` argument. This is the default. The means and their standard errors are calculated using the methods described in the `lm.sdf` function documentation.

the gap in percentiles The gap between respondents at the percentiles specified in the `percentiles` argument. This is returned when the `percentiles` argument is defined. The mean and standard error are computed as described in the `percentile` function documentation.

the gap in achievement levels The gap in the percentage of students at (when `achievementDiscrete` is `TRUE`) or at or above (when `achievementDiscrete` is `FALSE`) a particular achievement level. This is used when the `achievementLevel` argument is defined. The mean and standard error are calculated as described in the `achievementLevels` function documentation.

the gap in a survey response The gap in the percentage of respondents responding at `targetLevel` to `variable`. This is used when `targetLevel` is defined. The mean and standard deviation are calculated as described in the `edsurveyTable` function documentation.

Value

The return type depends on if the class of the `data` argument is an `edsurvey.data.frame` or an `edsurvey.data.frame.list`. Both include the call (called `call`), a list called `labels`, an object named `percentage` that shows the percentage in `groupA` and `groupB`, and an object that shows the gap called `results`.

The labels include the following elements:

`definition`	the definitions of the groups
`nFullData`	the *n*-size for the full dataset (before applying the definition)
`nUsed`	the *n*-size for the data after the group is subsetted and other restrictions (such as omitted values) are applied
`nPSU`	the number of PSUs used in calculation–only returned when `returnNumberOfPSU = TRUE`

The percentages are computed according to the vignette titled *Statistical Methods Used in EdSurvey* in the section "Estimation of Weighted Percentages When Plausible Values Are Not Present." The standard errors are calculated according to "Estimation of the Standard Error of Weighted Percentages When Plausible Values Are Not Present, Using the Jackknife Method." Standard errors of differences are calculated as the square root of the typical variance formula

$$Var(A-B) = Var(A) + Var(B) - 2Cov(A,B)$$

where the covariance term is calculated as described in the vignette titled *Statistical Methods Used in EdSurvey* in the section "Estimation of Covariances." These degrees of freedom are available only with the jackknife variance estimation. The degrees of freedom used for hypothesis testing are always set to the number of jackknife replicates in the data.

the data argument is an edsurvey.data.frame When the `data` argument is an `edsurvey.data.frame`, `gap` returns an S3 object of class `gap`.

The `percentage` object is a numeric vector with the following elements:

`pctA`	the percentage of respondents in `groupA` compared with the whole sample in `data`
`pctAse`	the standard error on the percentage of respondents in `groupA`
`dofA`	degrees of freedom appropriate for a *t*-test involving `pctA`. This value is returned only if `returnSimpleDoF=TRUE`.
`pctB`	the percentage of respondents in `groupB`.
`pctBse`	the standard error on the percentage of respondents in `groupB`
`dofB`	degrees of freedom appropriate for a *t*-test involving `pctA`. This value is returned only if `returnSimpleDoF=TRUE`.
`diffAB`	the value of `pctA` minus `pctB`
`covAB`	the covariance of `pctA` and `pctB`; used in calculating `diffABse`.
`diffABse`	the standard error of `pctA` minus `pctB`
`diffABpValue`	the *p*-value associated with the *t*-test used for the hypothesis test that `diffAB` is zero

`dofAB`	degrees of freedom used in calculating `diffABpValue`

The `results` object is a numeric data frame with the following elements:

`estimateA`	the mean estimate of `groupA` (or the percentage estimate if `achievementLevel` or `targetLevel` is specified)
`estimateAse`	the standard error of `estimateA`
`dofA`	degrees of freedom appropriate for a *t*-test involving `meanA`. This value is returned only if `returnSimpleDoF=TRUE`.
`estimateB`	the mean estimate of `groupB` (or the percentage estimate if `achievementLevel` or `targetLevel` is specified)
`estimateBse`	the standard error of `estimateB`
`dofB`	degrees of freedom appropriate for a *t*-test involving `meanB`. This value is returned only if `returnSimpleDoF=TRUE`.
`diffAB`	the value of `estimateA` minus `estimateB`
`covAB`	the covariance of `estimateA` and `estimateB`. Used in calculating `diffABse`.
`diffABse`	the standard error of `diffAB`
`diffABpValue`	the *p*-value associated with the *t*-test used for the hypothesis test that `diffAB` is zero.
`dofAB`	degrees of freedom used for the *t*-test on `diffAB`

If the gap was in achievement levels or percentiles and more than one percentile or achievement level is requested, then an additional column labeled `percentiles` or `achievementLevel` is included in the `results` object.

When `results` has a single row and when `returnVarEstInputs` is `TRUE`, the additional elements `varEstInputs` and `pctVarEstInputs` also are returned. These can be used for calculating covariances with `varEstToCov`.

the data argument is an edsurvey.data.frame.list When the `data` argument is an `edsurvey.data.frame.list`, `gap` returns an S3 object of class `gapList`.

The `results` object in the `edsurveyResultList` is a `data.frame`. Each row regards a particular dataset from the `edsurvey.data.frame`, and a reference dataset is dictated by the `referenceDataIndex` argument.

The `percentage` object is a `data.frame` with the following elements:

`covs`	a data frame with a column for each column in the `covs`. See previous section for more details.
`...`	all elements in the `percentage` object in the previous section
`diffAA`	the difference in `pctA` between the reference data and this dataset. Set to `NA` for the reference dataset.
`covAA`	the covariance of `pctA` in the reference data and `pctA` on this row. Used in calculating `diffAAse`.
`diffAAse`	the standard error for `diffAA`
`diffAApValue`	the *p*-value associated with the *t*-test used for the hypothesis test that `diffAA` is zero

`diffBB`	the difference in `pctB` between the reference data and this dataset. Set to `NA` for the reference dataset.
`covBB`	the covariance of `pctB` in the reference data and `pctB` on this row. Used in calculating `diffAAse`.
`diffBBse`	the standard error for `diffBB`
`diffBBpValue`	the *p*-value associated with the *t*-test used for the hypothesis test that `diffBB` is zero
`diffABAB`	the value of `diffAB` in the reference dataset minus the value of `diffAB` in this dataset. Set to `NA` for the reference dataset.
`covABAB`	the covariance of `diffAB` in the reference data and `diffAB` on this row. Used in calculating `diffABABse`.
`diffABABse`	the standard error for `diffABAB`
`diffABABpValue`	the *p*-value associated with the *t*-test used for the hypothesis test that `diffABAB` is zero

The `results` object is a `data.frame` with the following elements:

`...`	all elements in the `results` object in the previous section
`diffAA`	the value of `groupA` in the reference dataset minus the value in this dataset. Set to `NA` for the reference dataset.
`covAA`	the covariance of `meanA` in the reference data and `meanA` on this row. Used in calculating `diffAAse`.
`diffAAse`	the standard error for `diffAA`
`diffAApValue`	the *p*-value associated with the *t*-test used for the hypothesis test that `diffAA` is zero
`diffBB`	the value of `groupB` in the reference dataset minus the value in this dataset. Set to `NA` for the reference dataset.
`covBB`	the covariance of `meanB` in the reference data and `meanB` on this row. Used in calculating `diffBBse`.
`diffBBse`	the standard error for `diffBB`
`diffBBpValue`	the *p*-value associated with the *t*-test used for the hypothesis test that `diffBB` is zero
`diffABAB`	the value of `diffAB` in the reference dataset minus the value of `diffAB` in this dataset. Set to `NA` for the reference dataset.
`covABAB`	the covariance of `diffAB` in the reference data and `diffAB` on this row. Used in calculating `diffABABse`.
`diffABABse`	the standard error for `diffABAB`
`diffABABpValue`	the *p*-value associated with the *t*-test used for the hypothesis test that `diffABAB` is zero
`sameSurvey`	a logical value indicating if this line uses the same survey as the reference line. Set to `NA` for the reference line.

Author(s)

Paul Bailey, Trang Nguyen, and Huade Huo

Examples

```
## Not run:
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package = "NAEPprimer"))

# find the mean score gap in the primer data between males and females
gap("composite", sdf, dsex=="Male", dsex=="Female")

# find the score gap of the quartiles in the primer data between males and females
gap("composite", sdf, dsex=="Male", dsex=="Female", percentile=50)
gap("composite", sdf, dsex=="Male", dsex=="Female", percentile=c(25, 50, 75))

# find the percent proficient (or higher) gap in the primer data between males and females
gap("composite", sdf, dsex=="Male", dsex=="Female",
    achievementLevel=c("Basic", "Proficient", "Advanced"))

# find the discrete achievement level gap--this is harder to interpret
gap("composite", sdf, dsex=="Male", dsex=="Female",
    achievementLevel="Proficient", achievementDiscrete=TRUE)

# find the percent talk about studies at home (b017451) never or hardly
# ever gap in the primer data between males and females
gap("b017451", sdf, dsex=="Male", dsex=="Female",
    targetLevel="Never or hardly ever")

# example showing how to compare multiple levels
gap("b017451",sdf, dsex=="Male", dsex=="Female", targetLevel="Infrequently",
    recode=list(b017451=list(from=c("Never or hardly ever",
                                    "Once every few weeks",
                                    "About once a week"),
                             to=c("Infrequently"))))

# make subsets of sdf by scrpsu, "Scrambled PSU and school code"
sdfA <- subset(sdf, scrpsu %in% c(5,45,56))
sdfB <- subset(sdf, scrpsu %in% c(75,76,78))
sdfC <- subset(sdf, scrpsu %in% 100:200)
sdfD <- subset(sdf, scrpsu %in% 201:300)

sdfl <- edsurvey.data.frame.list(list(sdfA, sdfB, sdfC, sdfD),
                                 labels=c("A locations", "B locations",
                                          "C locations", "D locations"))

gap("composite", sdfl, dsex=="Male", dsex=="Female", percentile=c(50))

# example showing using linking error with gap
# load Grade 4 math data
g4math2015 <- readNAEP(".../M46NT1AT.dat")
g4math2017 <- readNAEP(".../M48NT1AT.dat")
```

```
g4math2019 <- readNAEP(".../M50NT1AT.dat")

# make an edsurvey.data.frame.list from math grade 4 2015, 2017, and 2019 data
g4math <- edsurvey.data.frame.list(list(g4math2019, g4math2017, g4math2015),
                                   labels = c("2019", "2017", "2015"))

# gap analysis with linking error in variance estimation across surveys
gap("composite", g4math, dsex == "Male", dsex == "Female", includeLinkingError=TRUE)
gap("composite", g4math, dsex == "Male", dsex == "Female", percentiles = c(10, 25),
    includeLinkingError=TRUE)
gap("composite", g4math, dsex == "Male", dsex == "Female",
    achievementDiscrete = TRUE, achievementLevel=c("Basic", "Proficient", "Advanced"),
    includeLinkingError=TRUE)


## End(Not run)
```

`getData` *Read Data to a Data Frame*

Description

Reads in selected columns to a `data.frame` or a `light.edsurvey.data.frame`. On an `edsurvey.data.frame`, the data are stored on disk.

Usage

```
getData(
  data,
  varnames = NULL,
  drop = FALSE,
  dropUnusedLevels = TRUE,
  omittedLevels = TRUE,
  defaultConditions = TRUE,
  formula = NULL,
  recode = NULL,
  includeNaLabel = FALSE,
  addAttributes = FALSE,
  returnJKreplicates = TRUE
)
```

Arguments

`data`	an `edsurvey.data.frame` or a `light.edsurvey.data.frame`
`varnames`	a character vector of variable names that will be returned. When both `varnames` and a `formula` are specified, variables associated with both are returned. Set to `NULL` by default.

`drop`	a logical value. When set to the default value of `FALSE`, when a single column is returned, it is still represented as a `data.frame` and is not converted to a vector.
`dropUnusedLevels`	a logical value. When set to the default value of `TRUE`, drops unused levels of all factor variables.
`omittedLevels`	a logical value. When set to the default value of `TRUE`, drops those levels of all factor variables that are specified in an `edsurvey.data.frame`. Use `print` on an `edsurvey.data.frame` to see the omitted levels. The omitted levels also can be adjusted with `setAttributes`; see Examples.
`defaultConditions`	a logical value. When set to the default value of `TRUE`, uses the default conditions stored in an `edsurvey.data.frame` to subset the data. Use `print` on an `edsurvey.data.frame` to see the default conditions.
`formula`	a `formula`. When included, `getData` returns data associated with all variables of the `formula`. When both `varnames` and a formula are specified, the variables associated with both are returned. Set to `NULL` by default.
`recode`	a list of lists to recode variables. Defaults to `NULL`. Can be set as `recode = list(var1 = list(from = c("a","b","c"),to = "d"))`. See Examples.
`includeNaLabel`	a logical value to indicate if `NA` (missing) values are returned as literal `NA` values or as factor levels coded as `NA`
`addAttributes`	a logical value set to `TRUE` to get a `data.frame` that can be used in calls to other functions that usually would take an `edsurvey.data.frame`. This `data.frame` also is called a `light.edsurvey.data.frame`. See Description section in `edsurvey.data.frame` for more information on `light.edsurvey.data.frame`.
`returnJKreplicates`	a logical value indicating if JK replicate weights should be returned. Defaults to `TRUE`.

Details

By default, an `edsurvey.data.frame` does not have data read into memory until `getData` is called and returns a data frame. This structure allows `EdSurvey` to have a minimal memory footprint. To keep the footprint small, you need to limit `varnames` to just the necessary variables.

There are two methods of attaching survey attributes to a `data.frame` to make it usable by the functions in the `EdSurvey` package (e.g., `lm.sdf`): (a) setting the `addAttributes` argument to `TRUE` at in the call to `getData` or (b) by appending the attributes to the data frame with `rebindAttributes`.

When `getData` is called, it returns a data frame. Setting the `addAttributes` argument to `TRUE` adds the survey attributes and changes the resultant `data.frame` to a `light.edsurvey.data.frame`.

Alternatively, a `data.frame` can be coerced into a `light.edsurvey.data.frame` using `rebindAttributes`. See Examples in the `rebindAttributes` documentation.

If both `formula` and `varnames` are populated, the variables on both will be included.

See the vignette titled *Using the `getData` Function in EdSurvey* for long-form documentation on this function.

Value

When `addAttributes` is `FALSE`, `getData` returns a `data.frame` containing data associated with the requested variables. When `addAttributes` is `TRUE`, `getData` returns a `light.edsurvey.data.frame`.

Author(s)

Tom Fink, Paul Bailey, and Ahmad Emad

See Also

`rebindAttributes`, `subset.edsurvey.data.frame`

Examples

```
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package = "NAEPprimer"))

# get two variables, without weights
df <- getData(data=sdf, varnames=c("dsex", "b017451"))
table(df)

# example of using recode
df2 <- getData(data=sdf, varnames=c("dsex", "t088301"),
               recode=list(t088301=list(from=c("Yes, available","Yes, I have access"),
                                         to=c("Yes")),
                           t088301=list(from=c("No, have no access"),
                                         to=c("No"))))
table(df2)

# when readNAEP is called on a data file, it appends a default
# condition to the edsurvey.data.frame. You can see these conditions
# by printing the sdf
sdf

# As per the default condition specified, getData restricts the data to only
# Reporting Sample. This behavior can be changed as follows:
df2 <- getData(data=sdf, varnames=c("dsex", "b017451"), defaultConditions = FALSE)
table(df2)

# similarly, the default behavior of omitting certain levels specified
# in the edsurvey.data.frame can be changed as follows:
df2 <- getData(data=sdf, varnames=c("dsex", "b017451"), omittedLevels = FALSE)
table(df2)

# omittedLevels can also be edited with setAttributes()
# here, the omitted level "Multiple" is removed from the list
sdfIncludeMultiple <- setAttributes(sdf, "omittedLevels", c(NA, "Omitted"))
# check that it was set
getAttributes(sdfIncludeMultiple, "omittedLevels")
# notice that omittedLevels is TRUE, removing NA and "Omitted" still
dfIncludeMultiple <- getData(data=sdfIncludeMultiple, varnames=c("dsex", "b017451"))
table(dfIncludeMultiple)
```

```
# the variable "c052601" is from the school-level data file; merging is handled automatically.
# returns a light.edsurvey.data.frame using addAttributes=TRUE argument
gddat <- getData(data=sdf,
                 varnames=c("composite", "dsex", "b017451","c052601"),
                 addAttributes = TRUE)
class(gddat)
# look at the first few lines
head(gddat)

# get a selection of variables, recode using ifelse, and reappend attributes
# with rebindAttributes so that it can be used with EdSurvey analysis functions
df0 <- getData(sdf, c("composite", "dsex", "b017451", "origwt"))
df0$sex <- ifelse(df0$dsex=="Male", "boy", "girl")
df0 <- rebindAttributes(df0, sdf)

## Not run:
# getting all the data can use up all the memory and is generally a bad idea
df0 <- getData(sdf, varnames=colnames(sdf),
               omittedLevels=FALSE, defaultConditions=FALSE)

## End(Not run)
```

`getPlausibleValue` *Get Plausible Value Variables*

Description

Gets the set of variables on an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list` associated with the given subject or subscale.

Usage

```
getPlausibleValue(var, data)
```

Arguments

`var`	a character vector naming the subject scale or subscale
`data`	an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`

Details

This function will return a set of plausible value names for variables that `hasPlausibleValue` returns as true.

Value

a character vector of the set of variable names for the plausible values

Author(s)

Michael Lee and Paul Bailey

See Also

showPlausibleValues, updatePlausibleValue

Examples

```
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package="NAEPprimer"))

getPlausibleValue(var="composite", data=sdf)
```

---

getWeightJkReplicates *Retrieve the Jackknife Replicate Weights*

Description

Returns the jackknife replicate weights on an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list` associated with a weight variable.

Usage

```
getWeightJkReplicates(var, data)
```

Arguments

`var`	character indicating the name of the weight variable for which the jackknife replicate weights are desired
`data`	an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`

Value

a character vector of the jackknife replicate weights

Author(s)

Michael Lee and Paul Bailey

Examples

```
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package="NAEPprimer"))

getWeightJkReplicates(var="origwt", data=sdf)
```

glm.sdf *EdSurvey Generalized Linear Models*

Description

Fits a logit or probit that uses weights and variance estimates appropriate for the `edsurvey.data.frame`, the `light.edsurvey.data.frame`, or the `edsurvey.data.frame.list`.

Usage

```
glm.sdf(formula, family = binomial(link = "logit"), data,
  weightVar = NULL, relevels = list(),
  varMethod=c("jackknife", "Taylor"), jrrIMax = 1,
  omittedLevels = TRUE, defaultConditions = TRUE, recode = NULL,
  returnNumberOfPSU=FALSE, returnVarEstInputs = FALSE)

logit.sdf(
  formula,
  data,
  weightVar = NULL,
  relevels = list(),
  varMethod = c("jackknife", "Taylor"),
  jrrIMax = 1,
  omittedLevels = TRUE,
  defaultConditions = TRUE,
  recode = NULL,
  returnNumberOfPSU = FALSE,
  returnVarEstInputs = FALSE
)

probit.sdf(
  formula,
  data,
  weightVar = NULL,
  relevels = list(),
  varMethod = c("jackknife", "Taylor"),
  jrrIMax = 1,
  omittedLevels = TRUE,
  defaultConditions = TRUE,
  recode = NULL,
  returnVarEstInputs = FALSE
)
```

Arguments

`formula`	a `formula` for the linear model. See `glm`. For logit and probit, we recommend using the `I()` function to define the level used for success. (See Examples.)

`family`	the `glm.sdf` function currently fits only the binomial outcome models, such as logit and probit, although other link functions are available for binomial models. See the `link` argument in the help for `family`.
`data`	an `edsurvey.data.frame`
`weightVar`	character indicating the weight variable to use (see Details). The `weightVar` must be one of the weights for the `edsurvey.data.frame`. If `NULL`, uses the default for the `edsurvey.data.frame`.
`relevels`	a list; used to change the contrasts from the default treatment contrasts to the treatment contrasts with a chosen omitted group. The name of each element should be the variable name, and the value should be the group to be omitted.
`varMethod`	a character set to "jackknife" or "Taylor" that indicates the variance estimation method to be used. See Details.
`jrrIMax`	the `Vjrr` sampling variance term (see *Statistical Methods Used in EdSurvey*) can be estimated with any positive number of plausible values and is estimated on the lower of the number of available plausible values and `jrrIMax`. When `jrrIMax` is set to `Inf`, all plausible values will be used. Higher values of `jrrIMax` lead to longer computing times and more accurate variance estimates.
`omittedLevels`	a logical value. When set to the default value of `TRUE`, drops those levels of all factor variables that are specified in `edsurvey.data.frame`. Use `print` on an `edsurvey.data.frame` to see the omitted levels.
`defaultConditions`	a logical value. When set to the default value of `TRUE`, uses the default conditions stored in an `edsurvey.data.frame` to subset the data. Use `print` on an `edsurvey.data.frame` to see the default conditions.
`recode`	a list of lists to recode variables. Defaults to `NULL`. Can be set as `recode= list(var1= list(from= c("a", "b", "c"), to="d"))`.
`returnNumberOfPSU`	a logical value set to `TRUE` to return the number of primary sampling units (PSUs)
`returnVarEstInputs`	a logical value set to `TRUE` to return the inputs to the jackknife and imputation variance estimates, which allow for the computation of covariances between estimates.

Details

This function implements an estimator that correctly handles left-hand side variables that are logical, allows for survey sampling weights, and estimates variances using the jackknife replication or Taylor series. The vignette titled *Statistical Methods Used in EdSurvey* describes estimation of the reported statistics and how it depends on `varMethod`.

The coefficients are estimated using the sample weights according to the section "Estimation of Weighted Means When Plausible Values Are Not Present" or the section "Estimation of Weighted Means When Plausible Values Are Present," depending on if there are assessment variables or variables with plausible values in them.

How the standard errors of the coefficients are estimated depends on the presence of plausible values (assessment variables), But once it is obtained, the *t* statistic is given by

$$t = \frac{\hat{\beta}}{\sqrt{\mathrm{var}(\hat{\beta})}}$$

where $\hat{\beta}$ is the estimated coefficient and $\mathrm{var}(\hat{\beta})$ is its variance of that estimate.

`logit.sdf` and `probit.sdf` are included for convenience only; they give the same results as a call to `glm.sdf` with the binomial family and the link function named in the function call (logit or probit). By default, `glm` fits a logistic regression when `family` is not set, so the two are expected to give the same results in that case. Other types of generalized linear models are not supported.

Variance estimation of coefficients: All variance estimation methods are shown in the vignette titled *Statistical Methods Used in EdSurvey*. When the predicted value does not have plausible values and `varMethod` is set to `jackknife`, the variance of the coefficients is estimated according to the section "Estimation of Standard Errors of Weighted Means When Plausible Values Are Not Present, Using the Jackknife Method."

When plausible values are present and `varMethod` is set to `jackknife`, the variance of the coefficients is estimated according to the section "Estimation of Standard Errors of Weighted Means When Plausible Values Are Present, Using the Jackknife Method."

When the predicted value does not have plausible values and `varMethod` is set to `Taylor`, the variance of the coefficients is estimated according to the section "Estimation of Standard Errors of Weighted Means When Plausible Values Are Not Present, Using the Taylor Series Method."

When plausible values are present and `varMethod` is set to `Taylor`, the variance of the coefficients is estimated according to the section "Estimation of Standard Errors of Weighted Means When Plausible Values Are Present, Using the Taylor Series Method."

Value

An `edsurveyGlm` with the following elements:

`call`	the function call
`formula`	the formula used to fit the model
`coef`	the estimates of the coefficients
`se`	the standard error estimates of the coefficients
`Vimp`	the estimated variance caused by uncertainty in the scores (plausible value variables)
`Vjrr`	the estimated variance from sampling
`M`	the number of plausible values
`nPSU`	the number of PSUs used in the calculation
`varm`	the variance estimates under the various plausible values
`coefm`	the values of the coefficients under the various plausible values
`coefmat`	the coefficient matrix (typically produced by the summary of a model)
`weight`	the name of the weight variable

`npv`	the number of plausible values
`njk`	the number of the jackknife replicates used
`varMethod`	always `jackknife`
`varEstInputs`	when `returnVarEstInputs` is `TRUE`, this element is returned. These are used for calculating covariances with `varEstToCov`.

Testing

Of the common hypothesis tests for joint parameter testing, only the Wald test is widely used with plausible values and sample weights. As such, it replaces, if imperfectly, the Akaike Information Criteria (AIC), the likelihood ratio test, chi-squared, and analysis of variance (ANOVA, including F-tests). See `waldTest` or the vignette titled *Methods and Overview of Using EdSurvey for Running Wald Tests*.

Author(s)

Paul Bailey

See Also

`glm`

Examples

```
## Not run:
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package = "NAEPprimer"))

# by default uses the jackknife variance method using replicate weights
table(sdf$b013801)
logit1 <- logit.sdf(I(b013801 %in% c("26-100", ">100")) ~ dsex + b017451, data=sdf)
# use summary to get detailed results
summary(logit1)

# Taylor series variance estimation
logit1t <- logit.sdf(I(b013801 %in% c("26-100", ">100")) ~ dsex + b017451, data=sdf,
                     varMethod="Taylor")
summary(logit1t)

logit2 <- logit.sdf(I(composite >= 300) ~ dsex + b013801, data=sdf)
summary(logit2)

logit3 <- glm.sdf(I(composite >= 300) ~ dsex + b013801, data=sdf,
                  family=quasibinomial(link="logit"))

# Wald test for joint hypothesis that all coefficients in b013801 are zero
waldTest(logit3, "b013801")

summary(logit3)

## End(Not run)
```

hasPlausibleValue *Plausible Value Test*

Description

Returns a value indicating if this variable has associated plausible values in an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`.

Usage

```
hasPlausibleValue(var, data)
```

Arguments

`var`	a character indicating the variable in question
`data`	an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`

Details

This function returns `TRUE` only when the variable passed to it is the name for a set of plausible values but not if it is an individual plausible value from such a set. Thus, on the NAEP Primer, `composite` has plausible values (and so `TRUE` would be returned by this function), but any of the plausible values or variable names defined in the actual data (such as `"mrpcm1"` or `"dsex"`) are not.

Value

a Boolean (or vector when `var` is a vector) indicating if each element of `var` has plausible values associated with it

Author(s)

Michael Lee and Paul Bailey

Examples

```
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package = "NAEPprimer"))

# TRUE
hasPlausibleValue(var="composite", data=sdf)

# FALSE
hasPlausibleValue(var="dsex", data=sdf)
```

isWeight *Weight Test*

Description

Returns logical values indicating whether a vector of variables is a weight for an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`.

Usage

```
isWeight(var, data)
```

Arguments

`var`	a character vector of variables
`data`	an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`

Details

Note that this function returns `TRUE` only when the `var` element is the name of the weight used for making estimates but not if it is one of the individual jackknife replicates.

Value

a logical vector of values indicating if each element of `var` is a weight

Author(s)

Michael Lee and Paul Bailey

Examples

```
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package = "NAEPprimer"))

# TRUE
isWeight(var="origwt", data=sdf)

# FALSE
isWeight(var="dsex", data=sdf)
```

levelsSDF *Print Levels and Labels*

Description

Retrieve the levels and labels of a variable from an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`.

Usage

```
levelsSDF(varnames, data, showOmitted = TRUE, showN = TRUE)
```

Arguments

`varnames`	a vector of character strings to search for in the database connection object (`data`)
`data`	an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`
`showOmitted`	a Boolean indicating if omitted levels should be shown
`showN`	a Boolean indicating if (unweighted) *n*-sizes should be shown for each response level

Author(s)

Michael Lee and Paul Bailey

Examples

```
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package = "NAEPprimer"))

# search variables in the sdf
levelsSDF(varnames="pared", data=sdf)

# search multiple variables
levelsSDF(varnames=c("pared","ell3"), data=sdf)

# search multiple variables in a light.edsurvey.data.frame with recodes
df2 <- getData(data=sdf, varnames=c("dsex", "t088301"),
               recode=list(t088301=list(from=c("Yes, available","Yes, I have access"),
                                        to=c("Yes")),
                           t088301=list(from=c("No, have no access"),
                                        to=c("No"))),
               addAttributes=TRUE)
levelsSDF(varnames=c("dsex","t088301"), data=df2)
```

lm.sdf *EdSurvey Linear Models*

Description

Fits a linear model that uses weights and variance estimates appropriate for the data.

Usage

```
lm.sdf(formula, data, weightVar = NULL, relevels = list(),
              varMethod = c("jackknife", "Taylor"), jrrIMax = 1,
              omittedLevels = TRUE, defaultConditions = TRUE, recode = NULL,
              returnVarEstInputs = FALSE, returnNumberOfPSU = FALSE,
              standardizeWithSamplingVar = FALSE)
```

Arguments

`formula`	a `formula` for the linear model. See `lm`. If *y* is left blank, the default subject scale or subscale variable will be used. (You can find the default using `showPlausibleValues`.) If *y* is a variable for a subject scale or subscale (one of the names shown by `showPlausibleValues`), then that subject scale or subscale is used.
`data`	an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`
`weightVar`	a character indicating the weight variable to use (see Details). The `weightVar` must be one of the weights for the `edsurvey.data.frame`. If `NULL`, it uses the default for the `edsurvey.data.frame`.
`relevels`	a list. Used to change the contrasts from the default treatment contrasts to the treatment contrasts with a chosen omitted group (the reference group). The name of each element should be the variable name, and the value should be the group to be omitted (the reference group).
`varMethod`	a character set to “jackknife” or “Taylor” that indicates the variance estimation method to be used. See Details.
`jrrIMax`	a numeric value; when using the jackknife variance estimation method, the default estimation option, `jrrIMax=1`, uses the sampling variance from the first plausible value as the component for sampling variance estimation. The `Vjrr` term (see *Statistical Methods Used in EdSurvey*) can be estimated with any number of plausible values, and values larger than the number of plausible values on the survey (including `Inf`) will result in all plausible values being used. Higher values of `jrrIMax` lead to longer computing times and more accurate variance estimates.
`omittedLevels`	a logical value. When set to the default value of `TRUE`, drops those levels of all factor variables that are specified in an `edsurvey.data.frame`. Use `print` on an `edsurvey.data.frame` to see the omitted levels.

`defaultConditions`
: a logical value. When set to the default value of `TRUE`, uses the default conditions stored in an `edsurvey.data.frame` to subset the data. Use `print` on an `edsurvey.data.frame` to see the default conditions.

`recode`
: a list of lists to recode variables. Defaults to `NULL`. Can be set as `recode=list(var1 = list(from= c("a", "b", "c"), to= "d"))`. See Examples.

`returnVarEstInputs`
: a logical value set to `TRUE` to return the inputs to the jackknife and imputation variance estimates, which allow for the computation of covariances between estimates.

`returnNumberOfPSU`
: a logical value set to `TRUE` to return the number of primary sampling units (PSUs)

`standardizeWithSamplingVar`
: a logical value indicating if the standardized coefficients should have the variance of the regressors and outcome measured with sampling variance. Defaults to `FALSE`.

Details

This function implements an estimator that correctly handles left-hand side variables that are either numeric or plausible values and allows for survey sampling weights and estimates variances using the jackknife replication method. The vignette titled *Statistical Methods Used in EdSurvey* describes estimation of the reported statistics.

Regardless of the variance estimation, the coefficients are estimated using the sample weights according to the sections “Estimation of Weighted Means When Plausible Values Are Not Present” or “Estimation of Weighted Means When Plausible Values Are Present,” depending on if there are assessment variables or variables with plausible values in them.

How the standard errors of the coefficients are estimated depends on the value of `varMethod` and the presence of plausible values (assessment variables), But once it is obtained, the *t* statistic is given by

$$t = \frac{\hat{\beta}}{\sqrt{\mathrm{var}(\hat{\beta})}}$$

where $\hat{\beta}$ is the estimated coefficient and $\mathrm{var}(\hat{\beta})$ is the variance of that estimate.

The **coefficient of determination (*R*-squared value)** is similarly estimated by finding the average *R*-squared using the average across the plausible values.

Standardized regression coefficients: Standardized regression coefficients can be returned in a call to `summary`, by setting the argument `src` to `TRUE`. See Examples.

By default, the standardized coefficients are calculated using standard deviations of the variables themselves, including averaging the standard deviation across any plausible values. When `standardizeWithSamplingVar` is set to `TRUE`, the variance of the standardized coefficient is calculated similar to a regression coefficient and therefore includes the sampling variance in the variance estimate of the outcome variable.

Variance estimation of coefficients: All variance estimation methods are shown in the vignette titled *Statistical Methods Used in EdSurvey*. When `varMethod` is set to the `jackknife` and the predicted value does not have plausible values, the variance of the coefficients is estimated according to the section “Estimation of Standard Errors of Weighted Means When Plausible Values Are Not Present, Using the Jackknife Method.”

When plausible values are present and `varMethod` is `jackknife`, the variance of the coefficients is estimated according to the section “Estimation of Standard Errors of Weighted Means When Plausible Values Are Present, Using the Jackknife Method.”

When plausible values are not present and `varMethod` is `Taylor`, the variance of the coefficients is estimated according to the section “Estimation of Standard Errors of Weighted Means When Plausible Values Are Not Present, Using the Taylor Series Method.”

When plausible values are present and `varMethod` is `Taylor`, the variance of the coefficients is estimated according to the section “Estimation of Standard Errors of Weighted Means When Plausible Values Are Present, Using the Taylor Series Method.”

Value

An `edsurvey.lm` with the following elements:

`call`	the function call
`formula`	the formula used to fit the model
`coef`	the estimates of the coefficients
`se`	the standard error estimates of the coefficients
`Vimp`	the estimated variance from uncertainty in the scores (plausible value variables)
`Vjrr`	the estimated variance from sampling
`M`	the number of plausible values
`varm`	the variance estimates under the various plausible values
`coefm`	the values of the coefficients under the various plausible values
`coefmat`	the coefficient matrix (typically produced by the summary of a model)
`r.squared`	the coefficient of determination
`weight`	the name of the weight variable
`npv`	the number of plausible values
`jrrIMax`	the `jrrIMax` value used in computation
`njk`	the number of the jackknife replicates used; set to `NA` when Taylor series variance estimates are used
`varMethod`	one of `Taylor series` or the `jackknife`
`residuals`	residuals from the average regression coefficients
`PV.residuals`	residuals from the by plausible value coefficients
`PV.fitted.values`	fitted values from the by plausible value coefficients
`B`	imputation variance covariance matrix, before multiplication by (M+1)/M
`U`	sampling variance covariance matrix

`rbar`	average relative increase in variance; see van Buuren (2012, eq. 2.29)
`nPSU`	number of PSUs used in calculation
`n0`	number of rows on an `edsurvey.data.frame` before any conditions were applied
`nUsed`	number of observations with valid data and weights larger than zero
`data`	data used for the computation
`Xstdev`	standard deviations of regressors, used for computing standardized regression coefficients when `standardizeWithSamplingVar` is set to `FALSE` (the default)
`varSummary`	the result of running `summary2` (unweighted) on each variable in the regression
`varEstInputs`	when `returnVarEstInputs` is `TRUE`, this element is returned. These are used for calculating covariances with `varEstToCov`.
`standardizeWithSamplingVar`	when `standardizeWithSamplingVar` is set to `TRUE`, this element is returned. Calculates the standard deviation of the standardized regression coefficients like any other variable.

Testing

Of the common hypothesis tests for joint parameter testing, only the Wald test is widely used with plausible values and sample weights. As such, it replaces, if imperfectly, the Akaike Information Criteria (AIC), the likelihood ratio test, chi-squared, and analysis of variance (ANOVA, including F-tests). See `waldTest` or the vignette titled *Methods and Overview of Using EdSurvey for Running Wald Tests*.

Author(s)

Paul Bailey

References

Binder, D. A. (1983). On the variances of asymptotically normal estimators from complex surveys. *International Statistical Review*, *51*(3), 279–292.

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys*. New York, NY: Wiley.

van Buuren, S. (2012). *Flexible imputation of missing data*. New York, NY: CRC Press.

Weisberg, S. (1985). *Applied linear regression* (2nd ed.). New York, NY: Wiley.

See Also

`lm`

Examples

```
## Not run:
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package = "NAEPprimer"))

# by default uses jackknife variance method using replicate weights
```

```
lm1 <- lm.sdf(composite ~ dsex + b017451, data=sdf)
lm1

# the summary function displays detailed results
summary(lm1)

# to show standardized regression coefficients
summary(lm1, src=TRUE)

# to specify a variance method, use varMethod
lm2 <- lm.sdf(composite ~ dsex + b017451, data=sdf, varMethod="Taylor")
lm2
summary(lm2)

# use relevel to set a new omitted category
lm3 <- lm.sdf(composite ~ dsex + b017451, data=sdf, relevels=list(dsex="Female"))
summary(lm3)
# test of a simple joint hypothesis
waldTest(lm3, "b017451")

# use recode to change values for specified variables
lm4 <- lm.sdf(composite ~ dsex + b017451, data=sdf,
              recode=list(b017451=list(from=c("Never or hardly ever",
                                              "Once every few weeks",
                                              "About once a week"),
                                       to=c("Infrequently")),
                          b017451=list(from=c("2 or 3 times a week","Every day"),
                                       to=c("Frequently"))))
# Note: "Infrequently" is the dropped level for the recoded b017451
summary(lm4)


## End(Not run)
```

---

merge *EdSurvey Merge*

Description

Takes a `data.frame` or a `light.edsurvey.data.frame` and merges with a `light.edsurvey.data.frame`.

Usage

```
## S3 method for class 'light.edsurvey.data.frame'
merge(x, y, ...)
```

Arguments

`x`	a `light.edsurvey.data.frame`. The attributes of the resulting `light.edsurvey.data.frame` are taken from `x`.

`y`	either a `light.edsurvey.data.frame` or a `data.frame`
`...`	arguments to be passed to `merge`

Value

a `light.edsurvey.data.frame` with the same attributes as `x`

Author(s)

Trang Nguyen

See Also

`merge`

Examples

```
# read in NAEP primer data
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package = "NAEPprimer"))
lsdf <- getData(data=sdf, varnames=c("dsex", "b017451"), addAttributes = TRUE)
df <- data.frame(dsex = c("Male","Female"), dsex2 = c("Boy","Girl"))

# merging a light.edsurvey.data.frame with a data.frame
# returns a light.edsurvey.data.frame object
merged_lsdf <- merge(lsdf,df, by = "dsex")
class(merged_lsdf) #  "light.edsurvey.data.frame" "data.frame"
head(merged_lsdf) # shows merge results

# merging behaves similarly to base::merge
df2 <- data.frame(dsex = c("Male","Female"), b017451 = c(1,2))
merged_lsdf2 <- merge(lsdf,df2, by = "dsex")
names(merged_lsdf2) # "dsex"      "b017451.x" "b017451.y"
head(merged_lsdf2) # shows merge results
```

`mixed.sdf` *EdSurvey Mixed-Effects Model*

Description

Fits a linear weighted mixed-effects model.

Usage

```
mixed.sdf(
  formula,
  data,
  weightVars = NULL,
  weightTransformation = TRUE,
  recode = NULL,
  defaultConditions = TRUE,
  tolerance = 0.01,
  nQuad = NULL,
  verbose = 0,
  family = NULL,
  centerGroup = NULL,
  centerGrand = NULL,
  fast = FALSE,
  ...
)
```

Arguments

`formula`	a `formula` for the multilevel regression or mixed model. See Examples and the vignette titled *Methods Used for Estimating Mixed-Effects Models in EdSurvey* for more details on how to specify a mixed model. If *y* is left blank, the default subject scale or subscale variable will be used. (You can find the default using `showPlausibleValues`.) If *y* is a variable for a subject scale or subscale (one of the names shown by `showPlausibleValues`), then that subject scale or subscale is used.
`data`	an `edsurvey.data.frame` or a `light.edsurvey.data.frame`
`weightVars`	character vector indicating weight variables for corresponding levels to use. The `weightVar` must be the weights for the `edsurvey.data.frame`. The weight variables must be in the order of level (from lowest to highest level).
`weightTransformation`	a logical value to indicate whether the function should standardize weights before using it in the multilevel model. If set to `TRUE`, the function will look up standard weight transformation methods often used for a specific survey. Weight transformation can be found in the vignette titled *Methods Used for Estimating Mixed-Effects Models in EdSurvey*. If set to `FALSE` or if the survey of the specified `data` does not have a standard weight transformation method, raw weights will be used.
`recode`	a list of lists to recode variables. Defaults to `NULL`. Can be set as `recode=list(var1 = list(from= c("a", "b", "c"), to= "d"))`. See Examples in `lm.sdf`.
`defaultConditions`	a logical value. When set to the default value of `TRUE`, uses the default conditions stored in an `edsurvey.data.frame` to subset the data. Use `print` on an `edsurvey.data.frame` to see the default conditions.
`tolerance`	depreciated, no effect
`nQuad`	depreciated, no effect
`verbose`	an integer; when set to `1`, it will print out the brief progress of the function `mix.sdf`. Users can use these traced messages for further diagnosis. When set to `2`, it will print out the detailed progress, including temporary estimates during the optimization. Defaults to `0`, which will run the function without output.
`family`	this argument is depreciated; please use the `WeMix` package's `mix` function directly for binomial models.

`centerGroup`	a list in which the name of each element is the name of the aggregation level, and the element is a formula of variable names to be group mean centered. For example, to group mean center gender and age within the group student: `list("student"= ~gender+age)`. Defaults to `NULL`, which means predictors are not adjusted by group centering. See Examples in the `WeMix` function `mix`.
`centerGrand`	a formula of variable names to be grand mean centered. For example, to center the variable education by overall mean of education: `~education`. Defaults to `NULL`, which means predictors are not adjusted by grand centering.
`fast`	depreciated, no effect
`...`	other potential arguments to be used in `mix`

Details

This function uses the `mix` call in the `WeMix` package to fit mixed models. When the outcome does not have plausible values, the variance estimator directly from the `mix` function is used; these account for covariance at the top level of the model specified by the user.

When the outcome has plausible values, the coefficients are estimated in the same way as in `lm.sdf`, that is, averaged across the plausible values. In addition, the variance of the coefficients is estimated as the sum of the variance estimate from the `mix` function and the imputation variance. The formula for the imputation variance is, again, the same as for `lm.sdf`, with the same estimators as in the vignette titled *Statistical Methods Used in EdSurvey*. In the section “Estimation of Standard Errors of Weighted Means When Plausible Values Are Present, Using the Jackknife Method” in the formula for V_{imp}, the variance and estimates of the variance components are estimated with the same formulas as the regression coefficients.

Value

A `mixedSdfResults` object with the following elements:

`call`	the original call used in `mixed.sdf`
`formula`	the formula used to fit the model
`coef`	a vector of coefficient estimates
`se`	a vector with the standard error estimates of the coefficients and the standard error of the variance components
`vars`	estimated variance components of the model
`levels`	the number of levels in the model
`ICC`	the intraclass correlation coefficient of the model
`npv`	the number of plausible values
`ngroups`	a `data.frame` that includes the number of observations for each group
`n0`	the number of observations in the original data
`nused`	the number of observations used in the analysis

If the formula does not involve plausible values, the function will return the following additional elements:

`lnlf`	the likelihood function

`lnl`	the log-likelihood of the model

If the formula involves plausible values, the function will return the following additional elements:

`Vimp`	the estimated variance from uncertainty in the scores
`Vjrr`	the estimated variance from sampling

Author(s)

Paul Bailey, Trang Nguyen, and Claire Kelley

References

Rabe-Hesketh, S., & Skrondal, A. (2006). Multilevel modelling of complex survey data. *Journal of the Royal Statistical Society: Series A (Statistics in Society), 169*(4), 805–827.

See Also

WeMix `mix` function and `lm.sdf`

Examples

```
## Not run:
# save TIMSS 2015 data to ~/TIMSS/2015
downloadTIMSS(root="~/", years=2015)
fin <- readTIMSS("~/TIMSS/2015", countries="fin", gradeLvl=4)
# uses all plausible values
mix1 <- mixed.sdf(mmat ~ itsex + (1|idschool), data = fin,
                  weightVar=c("totwgt","schwgt"), weightTransformation=FALSE)
summary(mix1)
# uses only one plausible value
mix2 <- mixed.sdf(asmmat01 ~ itsex + (1|idschool), data = fin,
                  weightVar=c("totwgt","schwgt"), weightTransformation=FALSE)
summary(mix2)

## End(Not run)
```

`mvrlm.sdf` *Multivariate Regression*

Description

Fits a multivariate linear model that uses weights and variance estimates appropriate for the `edsurvey.data.frame`.

Usage

```
mvrlm.sdf(
  formula,
  data,
  weightVar = NULL,
  relevels = list(),
  jrrIMax = 1,
  omittedLevels = TRUE,
  defaultConditions = TRUE,
  recode = NULL,
  returnVarEstInputs = FALSE,
  estMethod = "OLS"
)
```

Arguments

`formula`	a `Formula` package `Formula` for the linear model. See `Formula`; left-hand side variables are separated with vertical pipes (`|`). See Examples.
`data`	an `edsurvey.data.frame` or an `edsurvey.data.frame.list`
`weightVar`	character indicating the weight variable to use (see Details). The `weightVar` must be one of the weights for the `edsurvey.data.frame`. If `NULL`, uses the default for the `edsurvey.data.frame`.
`relevels`	a list. Used to change the contrasts from the default treatment contrasts to treatment contrasts with a chosen omitted group (the reference group). To do this, the user puts an element on the list with the same name as a variable to change contrasts on and then make the value for that list element equal to the value that should be the omitted group (the reference group).
`jrrIMax`	a numeric value; when using the jackknife variance estimation method, the default estimation option, `jrrIMax=1`, uses the sampling variance from the first plausible value as the component for sampling variance estimation. The V_{jrr} term (see *Statistical Methods Used in EdSurvey*) can be estimated with any number of plausible values, and values larger than the number of plausible values on the survey (including `Inf`) will result in all plausible values being used. Higher values of `jrrIMax` lead to longer computing times and more accurate variance estimates.
`omittedLevels`	a logical value. When set to the default value of `TRUE`, drops those levels of all factor variables that are specified in `edsurvey.data.frame`. Use `print` on an `edsurvey.data.frame` to see the omitted levels.
`defaultConditions`	a logical value. When set to the default value of `TRUE`, uses the default conditions stored in `edsurvey.data.frame` to subset the data. Use `print` on an `edsurvey.data.frame` to see the default conditions.
`recode`	a list of lists to recode variables. Defaults to `NULL`. Can be set as `recode = list(var1= list(from=c("a","b","c"), to ="d"))`.

`returnVarEstInputs`	a logical value. Set to `TRUE` to return the inputs to the jackknife and imputation variance estimates, which allow for computation of covariances between estimates.
`estMethod`	a character value indicating which estimation method to use. Default is `OLS`; other option is `GLS`.

Details

This function implements an estimator that correctly handles multiple left-hand side variables that are either numeric or plausible values, allows for survey sampling weights, and estimates variances using the jackknife replication method. The vignette titled *Statistical Methods Used in EdSurvey* describes estimation of the reported statistics.

The **coefficients** are estimated using the sample weights according to the section "Estimation of Weighted Means When Plausible Values Are Not Present" or the section "Estimation of Weighted Means When Plausible Values Are Present," depending on if there are assessment variables or variables with plausible values in them.

The **coefficient of determination (R-squared value)** is similarly estimated by finding the average R-squared using the sample weights for each set of plausible values.

Variance estimation of coefficients: All variance estimation methods are shown in the vignette titled *Statistical Methods Used in EdSurvey*.

When the predicted value does not have plausible values, the variance of the coefficients is estimated according to the section "Estimation of Standard Errors of Weighted Means When Plausible Values Are Not Present, Using the Jackknife Method."

When plausible values are present, the variance of the coefficients is estimated according to the section "Estimation of Standard Errors of Weighted Means When Plausible Values Are Present, Using the Jackknife Method."

For more information on the specifics of multivariate regression, see the vignette titled Methods and Overview of Using EdSurvey for Multivariate Regression.

Value

An `edsurvey.mvrlm` with elements:

`call`	the function call
`formula`	the formula used to fit the model
`coef`	the estimates of the coefficients
`se`	the standard error estimates of the coefficients
`Vimp`	the estimated variance caused by uncertainty in the scores (plausible value variables)
`Vjrr`	the estimated variance caused by sampling
`M`	the number of plausible values
`varm`	the variance estimates under the various plausible values
`coefm`	the values of the coefficients under the various plausible values

`coefmat`	the coefficient matrix (typically produced by the summary of a model)
`r.squared`	the coefficient of determination
`weight`	the name of the weight variable
`npv`	the number of plausible values
`njk`	the number of the jackknife replicates used
`varEstInputs`	When `returnVarEstInputs` is `TRUE`, this element is returned. These are used for calculating covariances with `varEstToCov`.
`residuals`	residuals for each of the PV models
`fitted.values`	model fitted values
`residCov`	residual covariance matrix for dependent variables
`residPV`	residuals for each dependent variable
`inputs`	coefficient estimation input matrices
`n0`	full data *n*
`nUsed`	*n* used for model
`B`	imputation variance-covariance matrix, before multiplication by (M+1)/M
`U`	sampling variance-covariance matrix

Author(s)

Alex Lishinski and Paul Bailey

See Also

the stats package `lm`, `lm.sdf`

Examples

```
## Not run:
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package = "NAEPprimer"))

# use | symbol to separate dependent variables in the left-hand side of formula
mvrlm.fit <- mvrlm.sdf(algebra | geometry ~ dsex + m072801, jrrIMax = 5, data = sdf)

# print method returns coefficients, as does coef method
mvrlm.fit
coef(mvrlm.fit)

# for more detailed results, use summary:
summary(mvrlm.fit)

# details of model can also be accessed through components of the returned object; for example:

# coefficients (one column per dependent variable)
mvrlm.fit$coef
# coefficient table with standard errors and p-values (1 table per dependent variable)
```

```
mvrlm.fit$coefmat
# R-squared values (one per dependent variable)
mvrlm.fit$r.squared
# residual covariance matrix
mvrlm.fit$residCov

# model residuals and other details are available as well

# show the structure of the residuals objects
str(mvrlm.fit$residuals)
str(mvrlm.fit$residPV)

# dependent variables can have plausible values or not (or a combination)

mvrlm.fit <- mvrlm.sdf(composite | mrps22 ~ dsex + m072801, data = sdf, jrrIMax = 5)
summary(mvrlm.fit)

mvrlm.fit <- mvrlm.sdf(algebra | geometry | measurement ~ dsex + m072801, data = sdf, jrrIMax = 5)
summary(mvrlm.fit)

mvrlm.fit <- mvrlm.sdf(mrps51 | mrps22 ~ dsex + m072801, data = sdf, jrrIMax = 5)
summary(mvrlm.fit)

# hypotheses about coefficient restrictions can also be tested using the Wald test

mvr <- mvrlm.sdf(algebra | geometry ~ dsex + m072801, data = sdf)

hypothesis <- c("geometry_dsexFemale = 0", "algebra_dsexFemale = 0")

# test statistics based on the F and chi-squared distribution are available
linearHypothesis(mvr, hypothesis = hypothesis, test = "F")
linearHypothesis(mvr, hypothesis = hypothesis, test = "Chisq")

## End(Not run)
```

oddsRatio *Odds Ratios for edsurveyGlm Models*

Description

Converts coefficients from `edsurveyGlm` logit regression model to odds ratios.

Usage

```
oddsRatio(model)
```

Arguments

`model`	an `edsurveyGlm` model

Value

An `oddsRatio.edsurveyGlm` object with the following elements:

`OR`	odds ratio coefficient estimates
`2.5%`	lower bound 95% confidence interval
`97.5%`	upper bound 95% confidence interval

`percentile` *EdSurvey Percentiles*

Description

Calculates the percentiles of a numeric variable in an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`.

Usage

```
percentile(
  variable,
  percentiles,
  data,
  weightVar = NULL,
  jrrIMax = 1,
  varMethod = c("jackknife", "Taylor"),
  alpha = 0.05,
  omittedLevels = TRUE,
  defaultConditions = TRUE,
  recode = NULL,
  returnVarEstInputs = FALSE,
  returnNumberOfPSU = FALSE,
  pctMethod = c("unbiased", "simple"),
  confInt = TRUE
)
```

Arguments

`variable`	the character name of the variable to percentiles computed, typically a subject scale or subscale
`percentiles`	a numeric vector of percentiles in the range of 0 to 100 (inclusive)
`data`	an `edsurvey.data.frame` or an `edsurvey.data.frame.list`
`weightVar`	a character indicating the weight variable to use.
`jrrIMax`	a numeric value; when using the jackknife variance estimation method, the default estimation option, `jrrIMax=1`, uses the sampling variance from the first plausible value as the component for sampling variance estimation. The V_{jrr}

term (see *Statistical Methods Used in EdSurvey*) can be estimated with any number of plausible values, and values larger than the number of plausible values on the survey (including `Inf`) will result in all plausible values being used. Higher values of `jrrIMax` lead to longer computing times and more accurate variance estimates.

`varMethod` a character set to `jackknife` or `Taylor` that indicates the variance estimation method used when constructing the confidence intervals. The jackknife variance estimation method is always used to calculate the standard error.

`alpha` a numeric value between 0 and 1 indicating the confidence level. An `alpha` value of 0.05 would indicate a 95% confidence interval and is the default.

`omittedLevels` a logical value. When set to the default value of `TRUE`, drops those levels of all factor variables that are specified in `achievementVars` and `aggregatBy`. Use `print` on an `edsurvey.data.frame` to see the omitted levels.

`defaultConditions`
a logical value. When set to the default value of `TRUE`, uses the default conditions stored in an `edsurvey.data.frame` to subset the data. Use `print` on an `edsurvey.data.frame` to see the default conditions.

`recode` a list of lists to recode variables. Defaults to `NULL`. Can be set as `recode=list(var1= list(from= c("a", "b", "c"), to= "d"))`.

`returnVarEstInputs`
a logical value set to `TRUE` to return the inputs to the jackknife and imputation variance estimates which allows for the computation of covariances between estimates.

`returnNumberOfPSU`
a logical value set to `TRUE` to return the number of primary sampling units (PSUs)

`pctMethod` one of “unbiased” or “simple”; unbiased produces a weighted median unbiased percentile estimate, whereas simple uses a basic formula that matches previously published results.

`confInt` a Boolean indicating if the confidence interval should be returned

Details

Percentiles, their standard errors, and confidence intervals are calculated according to the vignette titled *Statistical Methods Used in EdSurvey*. The standard errors and confidence intervals are based on separate formulas and assumptions.

The Taylor series variance estimation procedure is not relevant to percentiles because percentiles are not continuously differentiable.

Value

The return type depends on whether the class of the `data` argument is an `edsurvey.data.frame` or an `edsurvey.data.frame.list`.

The data argument is an edsurvey.data.frame When the `data` argument is an `edsurvey.data.frame`, `percentile` returns an S3 object of class `percentile`. This is a `data.frame` with typical attributes (`names`, `row.names`, and `class`) and additional attributes as follows:

`n0`	number of rows on `edsurvey.data.frame` before any conditions were applied
`nUsed`	number of observations with valid data and weights larger than zero
`nPSU`	number of PSUs used in the calculation
`call`	the call used to generate these results

The columns of the `data.frame` are as follows:

`percentile`	the percentile of this row
`estimate`	the estimated value of the percentile
`se`	the jackknife standard error of the estimated percentile
`df`	degrees of freedom
`confInt.ci_lower`	the lower bound of the confidence interval
`confInt.ci_upper`	the upper bound of the confidence interval
`nsmall`	the number of units with more extreme results, averaged across plausible values

When the `confInt` argument is set to `FALSE`, the confidence intervals are not returned.

The data argument is an edsurvey.data.frame.list When the `data` argument is an `edsurvey.data.frame.list`, `percentile` returns an S3 object of class `percentileList`. This is a data.frame with a `call` attribute. The columns in the `data.frame` are identical to those in the previous section, but there also are columns from the `edsurvey.data.frame.list`.

`covs`	a column for each column in the `covs` value of the `edsurvey.data.frame.list`. See Examples.

When `returnVarEstInputs` is `TRUE`, an attribute `varEstInputs` also is returned that includes the variance estimate inputs used for calculating covariances with `varEstToCov`.

Author(s)

Paul Bailey

References

Hyndman, R. J., & Fan, Y. (1996). Sample quantiles in statistical packages. *American Statistician*, *50*, 361–365.

Examples

```
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package="NAEPprimer"))

# get the median of the composite
percentile("composite", 50, sdf)

## Not run:
# get several percentiles
percentile("composite", c(0,1,25,50,75,99,100), sdf)
```

```
# build an edsurvey.data.frame.list
sdfA <- subset(sdf, scrpsu %in% c(5,45,56))
sdfB <- subset(sdf, scrpsu %in% c(75,76,78))
sdfC <- subset(sdf, scrpsu %in% 100:200)
sdfD <- subset(sdf, scrpsu %in% 201:300)

sdfl <- edsurvey.data.frame.list(list(sdfA, sdfB, sdfC, sdfD),
                                 labels=c("A locations",
                                          "B locations",
                                          "C locations",
                                          "D locations"))
# this shows how these datasets will be described:
sdfl$covs

percentile("composite", 50, sdfl)
percentile("composite", c(25, 50, 75), sdfl)

## End(Not run)
```

print.achievementLevels

Print AchievementLevels Results

Description

Prints details of discrete and cumulative achievement levels calculated using weights and variance estimates appropriate for the `edsurvey.data.frame`.

Usage

```
## S3 method for class 'achievementLevels'
print(x, printCall = TRUE, printDiscrete = TRUE, printCumulative = TRUE, ...)
```

Arguments

`x`	an `achievementLevels` object
`printCall`	a logical value; by default (`TRUE`), prints details about plausible values and weights used for calculating achievement levels
`printDiscrete`	a logical value; by default (`TRUE`), prints discrete achievement levels if they are present in `x`
`printCumulative`	a logical value; by default (`TRUE`), prints cumulative achievement levels if they are present in `x`
`...`	these arguments are not passed anywhere and are included only for compatibility

Author(s)

Huade Huo and Ahmad Emad

`print.edsurvey.data.frame` *EdSurvey Metadata Summary*

Description

Prints metadata regarding an `edsurvey.data.frame` or an `edsurvey.data.frame.list`

Usage

```
## S3 method for class 'edsurvey.data.frame'
print(x, printColnames = FALSE, ...)
```

Arguments

`x`	an `edsurvey.data.frame` or an `edsurvey.data.frame.list`
`printColnames`	a logical value; set to `TRUE` to see all column names in the `edsurvey.data.frame` or the `edsurvey.data.frame.list`
`...`	these arguments are not passed anywhere and are included only for compatibility

Author(s)

Michael Lee and Paul Bailey

`print.gap` *Gap Analysis Printing*

Description

Prints labels and a results vector of a gap analysis.

Usage

```
## S3 method for class 'gap'
print(x, ..., printPercentage = TRUE)

## S3 method for class 'gapList'
print(x, ..., printPercentage = TRUE)
```

Arguments

`x`	an `R` object representing a gap of class `gap` or `gapList`
`...`	these arguments are not passed anywhere and are included only for compatibility
`printPercentage`	a logical value set to `TRUE` to request printing of the percentage in the groups. Defaults to `TRUE`.

Author(s)

Paul Bailey

`readBB_2001` *Connect to B&B 2000–2001 Data*

Description

Opens a connection to a Baccalaureate & Beyond 2000–2001 data file and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readBB_2001(
  b01DER_Filepath,
  b01WT_FilePath,
  masterTxtFilepath,
  forceReread = FALSE,
  verbose = TRUE
)
```

Arguments

`b01DER_Filepath`	a character value to the main study-derived analytical data file (`B01DER.dat`). Located within the ECBW/Data Folder.
`b01WT_FilePath`	a character value to the study weight data file (`B01WT.dat`). Located within the `ECBW/Data` Folder.
`masterTxtFilepath`	a character value of the master format file (`master.txt`) that contains the meta-data for the `b01DER_Filepath` and `b01WT_FilePath`. Located in the `ECBW` folder.
`forceReread`	a logical value to force rereading of all processed data. The default value of `FALSE` will speed up the read function by using existing read-in data already processed.
`verbose`	a logical value that will determine if you want verbose output while the `readBB_2001` function is running to indicate processing progress. The default value is `TRUE`.

Details

Reads the `masterTxtFilepath` file to parse the `b01DER_Filepath` and `b01WT_FilePath` files to an `edsurvey.data.frame`. This function creates two cached data files will be saved in the same directory and filename as the `b01DER_Filepath` file for the B&B 2000–2001 longitudinal survey. The two cached data files will have file extensions of .txt and .meta.

Value

an `edsurvey.data.frame` for the B&B 2000–2001 longitudinal dataset.

Author(s)

Tom Fink

See Also

`readECLS_K2011`, `readNAEP`, and `getData`

Examples

```
## Not run:
setwd("C:/B&B2001/ecbw/B01")

dervFP <- file.path(getwd(), "Data", "B01DER.dat")
wgtFP <- file.path(getwd(), "Data", "B01WT.dat")
mstFP <- file.path(getwd(), "master.txt")

# with verbose output as default
bb2001 <- readBB_2001(dervFP, wgtFP, mstFP)

# silent output
bb2001 <- readBB_2001(dervFP, wgtFP, mstFP, verbose=FALSE)

# force cache update
bb2001 <- readBB_2001(dervFP, wgtFP, mstFP, forceReread = TRUE)

## End(Not run)
```

---

readBB_2003 *Connect to B&B 1993–2003 Data*

Description

Opens a connection to a Baccalaureate & Beyond 1993–2003 data file and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readBB_2003(
  b03DAS_Filepath,
  b03WEIGHT_FilePath,
  masterTxtFilepath,
  forceReread = FALSE,
  verbose = TRUE
)
```

Arguments

`b03DAS_Filepath`	a character value to the main study-derived analytical data file (`B03DAS.dat`). Located within the `ECBW/Data` Folder.
`b03WEIGHT_FilePath`	a character value to the study weight data file (`B03WEIGHT.dat`). Located within the `ECBW/Data` Folder.
`masterTxtFilepath`	a character value of the master format file (`master.txt`) that contains the meta-data for the `b03DAS_Filepath` and `b03WEIGHT_FilePath`. Located in the `ECBW` folder.
`forceReread`	a logical value to force rereading of all processed data. The default value of `FALSE` will speed up the read function by using existing read-in data already processed.
`verbose`	a logical value that will determine if you want verbose output while the `readBB_2003` function is running to indicate processing progress. The default value is `TRUE`.

Details

Reads in the specified `masterTxtFilepath` file to parse the `b03DAS_Filepath` and `b03WEIGHT_FilePath` files for the B&B 1993–2003 longitudinal survey to an `edsurvey.data.frame`. The two created cached data files will be saved in the same directory and filename as the `b03DAS_Filepath` file, having new file extensions of .txt and .meta.

Value

an `edsurvey.data.frame` for the B&B 1993–2003 longitudinal dataset.

Author(s)

Tom Fink

See Also

`readECLS_K2011`, `readNAEP`, and `getData`

Examples

```
## Not run:
setwd("C:/B&B2003/ecbw/B03")

dasFP <- file.path(getwd(), "Data", "B03DAS.dat")
wgtFP <- file.path(getwd(), "Data", "B03WEIGHT.dat")
mstFP <- file.path(getwd(), "master.txt")

# with verbose output as default
bb2003 <- readBB_2003(dasFP, wgtFP, mstFP)

# silent output
```

```
bb2003 <- readBB_2003(dasFP, wgtFP, mstFP, verbose=FALSE)

# force data cache update
bb2003 <- readBB_2003(dasFP, wgtFP, mstFP, forceReread = TRUE)

## End(Not run)
```

---

readBB_2012 *Connect to B&B 2008–2012 Data*

Description

Opens a connection to a Baccalaureate & Beyond 2008–2012 data file and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readBB_2012(
  path,
  csvFilename = "b12derived_datafile.csv",
  formatFilename = "b12derived_format.txt",
  metadataFilename = "b12derived_metadata.txt"
)
```

Arguments

`path`	a character value to the root directory path containing the `csvFilename`, `formatFilename`, and `metadataFilename` data files.
`csvFilename`	a character value of the derived data file (.csv) containing the raw B&B 2008–2012 data.
`formatFilename`	a character value of the format file (.txt) that describes the layout of the `csvFilename`.
`metadataFilename`	a character value of the metadata file (.txt) that describes additional metadata of the `csvFilename`.

Details

Reads in the specified `csvFilename` file for the B&B 2008–2012 longitudinal survey to an `edsurvey.data.frame`.

Value

an `edsurvey.data.frame` for the B&B 2008–2012 longitudinal dataset.

Author(s)

Tom Fink

See Also

readECLS_K2011, readNAEP, and getData

Examples

```
## Not run:
# using default filenames
bb <- readBB_2012(path = "C:/BB/2012")
dim(bb)

# parameters specified without default filenames
bb <- readBB_2012(path = getwd(),
                  csvFilename = "renamedData.csv",
                  formatFilename = "renamedFormat.txt",
                  metadataFilename = "renamedMeta.txt")
dim(bb)

## End(Not run)
```

`readBPS_1994` *Beginning Postsecondary Students (BPS:90/94) Data*

Description

Opens a connection to a BPS 1990 to 1994 cohort data file and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readBPS_1994(
  b94DAT_Filepath,
  masterTxtFilepath,
  forceReread = FALSE,
  verbose = TRUE
)
```

Arguments

`b94DAT_Filepath`	a character value of the file path to the student BPS:90/94 (.dat) file
`masterTxtFilepath`	a character value of the file path to the `master.txt` file contained within the electronic codebook directory
`forceReread`	a logical value to force rereading of all processed data. The default value of `FALSE` will speed up the read function by using existing read-in data already processed.
`verbose`	a logical value that will determine if you want verbose output while the `readHSB_Sophomore` function is running to indicate processing progress. The default value is `TRUE`.

Details

Reads in the student BPS 1990 to 1994 cohort data file to an `edsurvey.data.frame`.

Value

An `edsurvey.data.frame` for the BPS longitudinal dataset.

Author(s)

Sun-joo Lee

See Also

`readBB_2001`, `readNAEP`, and `getData`

Examples

```
## Not run:
setwd("C:/BPS1994/")

datFP <- file.path(getwd(), "BPS94", "STUDENT.DAT")
mstFP <- file.path(getwd(), "ECBW", "BP4", "MASTER.TXT")

#with verbose output as default
bps94 <- readBPS_1994(datFP, mstFP)

#silent output
bps94 <- readBPS_1994(datFP, mstFP, verbose=FALSE)

#force data cache update
bps94 <- readBPS_1994(datFP, mstFP, forceReread = TRUE)

## End(Not run)
```

`readBPS_2001` *Beginning Postsecondary Students (BPS:96/01) Data*

Description

Opens a connection to a BPS 1996 to 2001 cohort data file and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readBPS_2001(
  path = getwd(),
  csvFilename = "bps01derived_datafile.csv",
  formatFilename = "bps01derived_format.txt",
  metadataFilename = "bps01derived_metadata.txt"
)
```

Arguments

`path`	a character value to the full directory path to the BPS extracted data files
`csvFilename`	a character value of the derived data file (.csv) containing the raw BPS:96/01 data
`formatFilename`	a character value of the format file (.txt) that describes the layout of the `csvFilename`
`metadataFilename`	a character value of the metadata file (.txt) that describes additional metadata of the `csvFilename`

Details

Reads in the unzipped derived data files from the BPS longitudinal database of the BPS 1996 to 2001 cohort.

Value

An `edsurvey.data.frame` for the BPS longitudinal dataset.

Author(s)

Tom Fink

See Also

`readBB_2012`, `readNAEP`, and `getData`

Examples

```
## Not run:
# using default filenames
bps01 <- readBPS_2001(path = "C:/BPS/2001")
dim(bps01)

# parameters specified without default filenames
bps01 <- readBPS_2001(path = getwd(),
                      csvFilename = "renamedData.csv",
                      formatFilename = "renamedFormat.txt",
                      metadataFilename = "renamedMeta.txt")
dim(bps01)

## End(Not run)
```

readBPS_2009 *Beginning Postsecondary Students (BPS:04/09) Data*

Description

Opens a connection to a BPS 2004 to 2009 cohort data file and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readBPS_2009(
  path = getwd(),
  csvFilename = "bps09derived_datafile.csv",
  formatFilename = "bps09derived_format.txt",
  metadataFilename = "bps09derived_metadata.txt"
)
```

Arguments

`path`	a character value to the full directory path to the BPS extracted data files
`csvFilename`	a character value of the derived data file (.csv) containing the raw BPS:04/09 data
`formatFilename`	a character value of the format file (.txt) that describes the layout of the `csvFilename`
`metadataFilename`	a character value of the metadata file (.txt) that describes additional metadata of the `csvFilename`

Details

Reads in the unzipped derived data files from the BPS longitudinal database of the BPS 2004 to 2009 cohort.

Value

An `edsurvey.data.frame` for the BPS longitudinal dataset.

Author(s)

Tom Fink

See Also

`readBB_2012`, `readNAEP`, and `getData`

Examples

```
## Not run:
# using default filenames
bps09 <- readBPS_2009(path = "C:/BPS/2009")
dim(bps09)

# parameters specified without default filenames
bps09 <- readBPS_2009(path = getwd(),
                      csvFilename = "renamedData.csv",
                      formatFilename = "renamedFormat.txt",
                      metadataFilename = "renamedMeta.txt")
dim(bps09)

## End(Not run)
```

`readBPS_2014` *Beginning Postsecondary Students (BPS:12/14) Data*

Description

Opens a connection to a BPS 2012 to 2014 cohort data file and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readBPS_2014(
  path = getwd(),
  csvFilename = "bps14derived_datafile.csv",
  formatFilename = "bps14derived_format.txt",
  metadataFilename = "bps14derived_metadata.txt"
)
```

Arguments

`path`	a character value to the full directory path to the BPS extracted data files
`csvFilename`	a character value of the derived data file (.csv) containing the raw BPS:12/14 data
`formatFilename`	a character value of the format file (.txt) that describes the layout of the `csvFilename`
`metadataFilename`	a character value of the metadata file (.txt) that describes additional metadata of the `csvFilename`

Details

Reads in the unzipped files downloaded from the BPS 2012 to 2014 cohort longitudinal database.

Value

An `edsurvey.data.frame` for the BPS longitudinal dataset.

Author(s)

Sun-joo Lee

See Also

`readBB_2012`, `readNAEP`, and `getData`

Examples

```
## Not run:
# using default filenames
bps14 <- readBPS_2014(path = "C:/BPS/2012")
dim(bps14)

# parameters specified without default filenames
bps14 <- readBPS_2014(path = getwd(),
                  csvFilename = "renamedData.csv",
                  formatFilename = "renamedFormat.txt",
                  metadataFilename = "renamedMeta.txt")
dim(bps14)

## End(Not run)
```

`readBTLS` *Connect to BTLS Data*

Description

Opens a connection to the Beginning Teacher Longitudinal Study (BTLS) waves 1 through 5 data file and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readBTLS(dat_FilePath, spss_FilePath, verbose = TRUE)
```

Arguments

`dat_FilePath`	a character value to the full path of the BTLS fixed-width (.dat) data file
`spss_FilePath`	a character value to the full path of the SPSS syntax file to process the `dat_FilePath`
`verbose`	a logical value that will determine if you want verbose output while the `readBTLS` function is running to indicate processing progress (the default value is `TRUE`)

Details

Reads the `spss_FilePath` file to parse the `dat_FilePath` to an `edsurvey.data.frame`. There is no cached data because the `dat_FilePath` format already is in fixed-width format.

Value

an `edsurvey.data.frame` for the BTLS waves 1 to 5 longitudinal dataset.

Author(s)

Tom Fink

See Also

`readECLS_K2011`, `readNAEP`, and `getData`

Examples

```
## Not run:

  fld <- "C:/EdSurveyData/BTLS"
  datPath <- file.path(fld, "ASCII Data File", "BTLS2011_12.dat")
  spsPath <- file.path(fld, "Input Syntax for Stata and SPSS", "BTLS2011_12.sps")

  #read in the data to an edsurvey.data.frame
  btls <- readBTLS(datPath, spsPath, verbose = TRUE)

  dim(btls)

## End(Not run)
```

`readCivEDICCS` *Connect to ICCS and CivED Data*

Description

Opens a connection to an ICCS or CivEd (1999) data file and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readCivEDICCS(
  path,
  countries,
  dataSet = c("student", "teacher"),
  gradeLvl = c("8", "9", "12"),
  forceReread = FALSE,
  verbose = TRUE
)
```

Arguments

`path`	a character value of the full directory to the ICCS/CivED extracted SPSS (.sav) set of data
`countries`	a character vector of the country/countries to include using the three-digit International Organization for Standardization (ISO) country code. A list of country codes can be found on Wikipedia at `https://en.wikipedia.org/wiki/ISO_3166-1#Current_codes` or other online sources. Consult the *ICCS/CivED User Guide* to help determine what countries are included within a specific testing year of ICCS/CivED. To select all countries, use a wildcard value of `*`.
`dataSet`	a character value of either `student` or `teacher` to indicate which set of data is returned. The student-level and teacher-level datasets cannot both be returned at the same time, unlike other IEA datasets. Note: The CivED 1999 study also included student-to-teacher data for Grade 8. Specifying `dataSet="student"` and `gradeLvl=8` will include both the `student` and `teacher` data in the resulting `edsurvey.data.frame`.
`gradeLvl`	a character value of the grade level to return • **8** = eighth grade (the default if not specified) • **9** = ninth grade • **12** = 12th grade (for CivED 1999 only)
`forceReread`	a logical value to force rereading of all processed data. The default value of `FALSE` will speed up the `readCivEDICCS` function by using existing read-in data already processed.
`verbose`	a logical value to either print or suppress status message output. The default value is `TRUE`.

Details

Reads in the unzipped files downloaded from the international database(s) using the IEA Study Data Repository. Data files require the SPSS data file (.sav) format using the default filenames.

When using the `getData` function with a CivED or ICCS study `edsurvey.data.frame`, the requested data variables are inspected, and it handles any necessary data merges automatically. The `school` data always will be returned merged to the `student` data, even if only `school` variables are requested. If a 1999 CivED Grade 8 `edsurvey.data.frame` with `teacher` data variables is requested by the `getData` call, it will cause `teacher` data to be merged. Many `students` can be linked to many `teachers`, which varies widely between countries, and not all countries contain `teacher` data.

Calling the `dim` function for a CivED 1999 Grade 8 `edsurvey.data.frame` will result in the row count as if the `teacher` dataset was merged. This row count will be considered the `full data N` of the `edsurvey.data.frame`, even if no `teacher` data were included in an analysis. The column count returned by `dim` will be the count of unique column variables across all data levels.

Value

an `edsurvey.data.frame` for a single specified country or an `edsurvey.data.frame.list` if multiple countries specified

Author(s)

Tom Fink

See Also

readNAEP, readTIMSS, getData, and downloadCivEDICCS

Examples

```
## Not run:
eng <- readCivEDICCS("C:/ICCS2009/Gr8", countries = c("eng"),
                     gradeLvl = 8, dataSet = "student")
gg <- getData(eng, c("famstruc", "totwgts", "civ"))
head(gg)
edsurveyTable(civ ~ famstruc, eng)

## End(Not run)
```

---

readECLS_K1998 *Connect to ECLS–K 1998 Data*

Description

Opens a connection to an ECLS–K 1998 data file and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readECLS_K1998(
  path = getwd(),
  filename = "eclsk_98_99_k8_child_v1_0.dat",
  layoutFilename = "Layout_k8_child.txt",
  forceReread = FALSE,
  verbose = TRUE
)
```

Arguments

`path`	a character value to the full directory path(s) to the ECLS–K-extracted fixed-width-format (.dat) set of data files
`filename`	a character value of the name of the fixed-width (.dat) data file in the specified `path` to be read
`layoutFilename`	a character value of the filename of either the ASCII (.txt) layout file of the `filename` within the specified `path` or a character value of the filename of the SPSS syntax (.sps) layout file of the `filename` within the specified `path`

`forceReread`	a logical value to force rereading of all processed data. The default value of `FALSE` will speed up the read function by using existing read-in data already processed.
`verbose`	a logical value that will determine if you want verbose output while the `readECLS-K1998` function is running to indicate processing progress. The default value is `TRUE`.

Details

Reads in the unzipped files downloaded from the ECLS–K 1998 longitudinal dataset(s) to an `edsurvey.data.frame`. The ECLS–K 1998–99 study consisted of three distinct separate datasets that cannot be combined: (1) Child Grades K–8 Data, (2) School Base-Year Data, and (3) Teacher Base-Year Data. The `filename` and `layoutFilename` arguments default to the corresponding Child K–8 default filenames.

Value

an `edsurvey.data.frame` for the ECLS–K 1998 longitudinal dataset

Author(s)

Tom Fink

See Also

`readECLS_K2011`, `readNAEP`, `getData`, `downloadECLS_K`

Examples

```
## Not run:
# read-in student file with defaults
eclsk_df <- readECLS_K1998(path="C:/ECLS_K/1998") #using defaults
d <- getData(eclsk_df, c("childid", "gender", "race"))
summary(d)

# read-in with parameters specified
eclsk_df <- readECLS_K1998(path = "C:/ECLS_K1998",
                           filename = "eclsk_98_99_k8_child_v1_0.dat",
                           layoutFilename = "Layout_k8_child.txt",
                           verbose = TRUE,
                           forceReread = FALSE)

## End(Not run)
```

`readECLS_K2011` *Connect to ECLS–K 2011 Data*

Description

Opens a connection to an ECLS–K 2011 data file and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readECLS_K2011(
  path = getwd(),
  filename = "childK4p.dat",
  layoutFilename = "ECLSK2011_K4PUF.sps",
  forceReread = FALSE,
  verbose = TRUE
)
```

Arguments

`path`	a character value to the full directory path(s) to the ECLS–K 2010–11 extracted fixed-with-format (.dat) set of data files
`filename`	a character value of the name of the fixed-width (.dat) data file in the specified `path` to be read
`layoutFilename`	a character value of the filename of either the ASCII (.txt) layout file of the `filename` within the specified `path` or a character value of the filename of the SPSS syntax (.sps) layout file of the `filename` within the specified `path`
`forceReread`	a logical value to force rereading of all processed data. The default value of `FALSE` will speed up the read function by using existing read-in data already processed.
`verbose`	a logical value that will determine if you want verbose output while the `readECLS--K2011` function is running to indicate processing progress. The default value is `TRUE`.

Details

Reads in the unzipped files downloaded from the ECLS–K 2010–11 longitudinal dataset.

Value

an `edsurvey.data.frame` for the ECLS–K 2010–11 longitudinal dataset

Author(s)

Tom Fink

See Also

readECLS_K1998, readNAEP, getData, and downloadECLS_K

Examples

```
## Not run:
# read-in student file with defaults
eclsk_df <- readECLS_K2011(path="C:/ECLS_K/2011") #using defaults
d <- getData(eclsk_df, c("childid", "c1hgt1", "c1wgt1"))
summary(d)

# read-in with parameters specified
eclsk_df <- readECLS_K2011(path = "C:/ECLS_K2011",
                           filename = "childK4p.dat",
                           layoutFilename = "ECLSK2011_K4PUF.sps",
                           forceReread = FALSE,
                           verbose = TRUE)

## End(Not run)
```

---

readELS *Connect to Education Longitudinal Study (ELS:2002) Data*

Description

Opens a connection to an ELS data file and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readELS(
  path = getwd(),
  filename = "els_02_12_byf3pststu_v1_0.sav",
  wgtFilename = ifelse(filename == "els_02_12_byf3pststu_v1_0.sav",
    "els_02_12_byf3stubrr_v1_0.sav", NA),
  forceReread = FALSE,
  verbose = TRUE
)
```

Arguments

path	a character value to the directory path of the extracted set of data files and layout files.
filename	a character value of the name of the SPSS (.sav) data file in the specified path to be read.

`wgtFilename`	a character value of the name of the associated balanced repeated replication (BRR) weight SPSS (.sav) data file in the specified `path` to be read. This argument is applicable only for the student-level data, which contains a separate data file containing the weight replicate information. If using default filenames (recommended), then you shouldn't need to specify this parameter because it will inspect the `filename` argument. For data files with no BRR weight file associated, specify a value of `NULL` or `NA`.
`forceReread`	a logical value to force rereading of all processed data. The default value of `FALSE` will speed up the read function by using existing read-in data already processed.
`verbose`	a logical value that will determine if you want verbose output while the `readELS` function is running to indicate processing progress. The default value is `TRUE`.

Details

Reads in the unzipped files downloaded from the ELS longitudinal dataset(s) to an `edsurvey.data.frame`. The ELS 2002 study consisted of four distinct separate datasets that cannot be combined:

- Student: bas -year through follow-up three (default)
- School: base year through follow-up one
- Institution: follow-up two
- Institution: follow-up three

Value

an `edsurvey.data.frame` for the ELS longitudinal dataset

Author(s)

Tom Fink

See Also

`readECLS_K2011`, `readNAEP`, `getData`, and `downloadECLS_K`

Examples

```
## Not run:
# read-in student file including weight file as default
setwd("C:/ELS2002")
els_df <- readELS() #student level with weights)
d <- getData(els_df, c("stu_id", "bysex", "bystlang"))
summary(d)

# read-in with parameters specified (student level with weights)
els_wgt_df <- readELS(path = "C:/ELS2002",
                            filename = "els_02_12_byf3pststu_v1_0.dat",
                            layoutFilename = "Layout_BYF3PSTSTU.PRI.txt",
                            wgtFilename = "els_02_12_byf3stubrr_v1_0.dat",
                              wgtLayoutFilename = "Layout_BYF3BRRPST.PRI.txt",
                              verbose = TRUE,
                              forceReread = FALSE)

# read-in with parameters specified (school level, no weights)
els_sch_df <- readELS(path = "C:/ELS2002",
                      filename = "els_02_12_byf1sch_v1_0.dat",
                      layoutFilename = "Layout_BYF1SCH.txt",
                      wgtFilename = NA,
                      wgtLayoutFilename = NA,
                      verbose = TRUE,
                      forceReread = FALSE)


## End(Not run)
```

`readHSB_Senior` *Connect to HS&B Study Senior Data*

Description

Opens a connection to a High School & Beyond 1980–1986 Senior cohort data file and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readHSB_Senior(
  HSR8086_PRI_FilePath,
  HSR8086_SASSyntax_Path,
  forceReread = FALSE,
  verbose = TRUE
)
```

Arguments

`HSR8086_PRI_FilePath`	a character value to the main study-derived analytical data file (HSR8086_REV.PRI). Located within the `REVISED_ASCII` Folder.
`HSR8086_SASSyntax_Path`	a character value to the SAS syntax file for parsing the `HSR8086_REV.PRI` data file. Located within the `SAS_EXTRACT_LOGIC` Folder.
`forceReread`	a logical value to force rereading of all processed data. The default value of `FALSE` will speed up the read function by using existing read-in data already processed.
`verbose`	a logical value that will determine if you want verbose output while the `readHSB_Senior` function is running to indicate processing progress. The default value is `TRUE`.

Details

Reads in the specified `HSR8086_SASSyntax_Path` file to parse the `HSR8086_PRI_FilePath` file. A cached data file and metadata file will be saved in the same directory and filename as the `HSR8086_PRI_FilePath` file, having new file extensions of .txt and .meta, respectively.

Please note the original source `repcode` variable has been split into two variables named `repcode_str` for the stratum value and `repcode_psu` for the primary sampling unit (PSU) value in the resulting cache data.

Value

an `edsurvey.data.frame` for the HS&B Senior 1980–1986 longitudinal dataset

Author(s)

Tom Fink

See Also

`readECLS_K2011`, `readNAEP`, and `getData`

Examples

```
## Not run:
wrkFld <- "C:/HSB/SENIOR"

dataPath <- file.path(wrkFld, "REVISED_ASCII", "HSR8086_REV.PRI")
sasPath <- file.path(wrkFld, "SAS_EXTRACT_LOGIC", "HSBsr_READ_HSR8086.SAS")

# with verbose output as default
hsbSR <- readHSB_Senior(dataPath, sasPath)

# silent output
hsbSR <- readHSB_Senior(dataPath, sasPath, verbose = FALSE)

# force cache update
hsbSR <- readHSB_Senior(dataPath, sasPath, forceReread = TRUE)

## End(Not run)
```

`readHSB_Sophomore` *Connect to HS&B Study Sophomore Data*

Description

Opens a connection to a High School & Beyond 1980–1992 Sophomore cohort data file and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readHSB_Sophomore(
  HSO8092_PRI_FilePath,
  HSO8092_SASSyntax_Path,
  forceReread = FALSE,
  verbose = TRUE
)
```

Arguments

`HSO8092_PRI_FilePath`	a character value to the main study-derived analytical data file (HSO8092_REV.PRI). Located within the `REVISED_ASCII` folder.
`HSO8092_SASSyntax_Path`	a character value to the SAS syntax file for parsing the `HSO8092_REV.PRI` data file. Located within the `SAS_EXTRACT_LOGIC` folder.
`forceReread`	a logical value to force rereading of all processed data. The default value of `FALSE` will speed up the read function by using existing read-in data already processed.
`verbose`	a logical value that will determine if you want verbose output while the `readHSB_Sophomore` function is running to indicate processing progress. The default value is `TRUE`.

Details

Reads in the specified `HSO8092_SASSyntax_Path` file to parse the `HSO8092_PRI_FilePath` file. A cached data file and metadata file will be saved in the same directory and filename as the `HSO8092_PRI_FilePath` file, having new file extensions of .txt and .meta, respectively.

Please note the original source `repcode` variable has been split into two variables named `repcode_str` for the stratum value and `repcode_psu` for the primary sampling unit (PSU) value in the resulting cache data.

Value

an `edsurvey.data.frame` for the HS&B Sophomore 1980–1992 longitudinal dataset

Author(s)

Tom Fink

See Also

`readECLS_K2011`, `readNAEP`, and `getData`

Examples

```
## Not run:
wrkFld <- "C:/HSB/SOPHOMORE"

dataPath <- file.path(wrkFld, "REVISED_ASCII", "HSO8092_REV.PRI")
```

```
sasPath <- file.path(wrkFld, "SAS_EXTRACT_LOGIC", "HSBso_READ_HSO8092.SAS")

# with verbose output as default
hsbSO <- readHSB_Sophomore(dataPath, sasPath)

# silent output
hsbSO <- readHSB_Sophomore(dataPath, sasPath, verbose = FALSE)

# force cache update
hsbSO <- readHSB_Sophomore(dataPath, sasPath, forceReread = TRUE)

## End(Not run)
```

readHSLS — *Connect to High School Longitudinal Study 2009 (HSLS:2009) Data*

Description

Opens a connection to an HSLS data file and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readHSLS(
  path = getwd(),
  filename = "hsls_16_student_v1_0.sav",
  wgtFilename = NA,
  forceReread = FALSE,
  verbose = TRUE
)
```

Arguments

`path`	a character value to the full directory path(s) to the HSLS extracted SPSS (.sav) set of data files
`filename`	a character value of the name of the SPSS (.sav) datafile to be read
`wgtFilename`	a character value of the name of the associated BRR weight SPSS (.sav) data file in the specificed `path` to be read. This argument is only applicable for the restricted-use student level data, which contains a separate data-file containing the weight replicate information. For data files with no balanced repeated replication (BRR) weight file associated, specify a value of `NULL` or `NA`.
`forceReread`	a logic value to force a rereading of all processed data. The default value of `FALSE` speeds up the `readHSLS` function by using existing read-in data already processed.
`verbose`	a logical value set to `TRUE` for verbose output that indicates progress

Details

Reads in the unzipped files downloaded from the HSLS longitudinal dataset.

Value

an `edsurvey.data.frame` for the HSLS longitudinal dataset

Note

The SPSS (.sav) format is preferred over the fixed-width-format (.dat) ASCII file format at this time relating to value label issues identified with the ASCII layout specifications.

Author(s)

Tom Fink

See Also

`readECLS_K2011`, `readNAEP`, and `getData`

Examples

```
## Not run:
# use function default values at working directory
hsls <- readHSLS("C:/HSLS/2009")

# specify parameters with verbose output
hsls <- readHSLS(path="C:/HSLS/2009",
                 filename = "hsls_16_student_v1_0.sav",
                 forceReread = FALSE,
                 verbose = TRUE)

# specify parameters silent output
hsls <- readHSLS(path="C:/HSLS/2009",
                 filename = "hsls_16_student_v1_0.sav",
                 forceReread = FALSE,
                 verbose = FALSE)

## End(Not run)
```

`readICILS` *Connect to ICILS Data*

Description

Opens a connection to an ICILS data file residing on the disk and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readICILS(
  path,
  countries,
  dataSet = c("student", "teacher"),
  forceReread = FALSE,
  verbose = TRUE
)
```

Arguments

`path`	a character value to the full directory path to the ICILS extracted SPSS (.sav) set of data
`countries`	a character vector of the country/countries to include using the three-digit ISO country code. A list of country codes can be found on Wikipedia at `https://en.wikipedia.org/wiki/ISO_3166-1#Current_codes` or other online sources. Consult the *ICILS User Guide* to help determine what countries are included within a specific testing year of ICILS. To select all countries, use a wildcard value of `*`.
`dataSet`	a character value of either `student` (the default if not specified) or `teacher` to indicate which set of data is returned. The student-level and teacher-level datasets cannot both be returned at the same time, unlike other IEA datasets.
`forceReread`	a logical value to force rereading of all processed data. The default value of `FALSE` will speed up the `readICILS` function by using existing read-in data already processed.
`verbose`	a logical value to either print or suppress status message output. The default value is `TRUE`.

Details

Reads in the unzipped files downloaded from the ICILS international dataset(s) using the IEA Study Data Repository. Data files require the SPSS data file (.sav) format using the default filenames.

Value

an `edsurvey.data.frame` for a single specified country or an `edsurvey.data.frame.list` if multiple countries specified

Author(s)

Tom Fink and Jeppe Bundsgaard (updated for 2018)

See Also

`readNAEP`, `readTIMSS`, and `getData`

Examples

```
## Not run:
pol <- readICILS("C:/ICILS/2013", countries = "pol", dataSet = "student")
gg <- getData(pol, c("idstud", "cil", "is1g18b"))
head(gg)
edsurveyTable(cil ~ is1g18b, pol)

## End(Not run)
```

`readNAEP` *Connect to NAEP Data*

Description

Opens a connection to an NAEP data file residing on the disk and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readNAEP(
  path,
  defaultWeight = "origwt",
  defaultPvs = "composite",
  omittedLevels = c("Multiple", NA, "Omitted"),
  frPath = NULL
)
```

Arguments

`path`	a character value indicating the full filepath location and name of the (.dat) data file
`defaultWeight`	a character value that indicates the default weight specified in the resulting `edsurvey.data.frame`. Default value is `origwt` if not specified.
`defaultPvs`	a character value that indicates the default plausible value specified in the resulting `edsurvey.data.frame`. Default value is `composite` if not specified.
`omittedLevels`	a character vector indicating which factor levels/labels should be excluded. When set to the default value of `c('Multiple',NA,'Omitted')`, adds the vector to the `edsurvey.data.frame`.
`frPath`	a character value indicating the location of the `fr2` parameter layout file included with the data companion to parse the specified `filepath` data file

Details

The function uses the `frPath` file layout (.fr2) data to read in the fixed-width data file (.dat) and builds the `edsurvey.data.frame`.

Value

An edsurvey.data.frame containing the following elements:

userConditions	a list containing all user conditions set using the subset.edsurvey.data.frame method
defaultConditions	the default conditions to be applied to the edsurvey.data.frame
data	an LaF object containing a connection to the student dataset on disk
dataSch	an LaF object containing a connection to the school dataset on disk
dataTch	not applicable for NAEP data; returns NULL
weights	a list containing the weights found on the edsurvey.data.frame
pvvar	a list containing the plausible values found on the edsurvey.data.frame
subject	the subject of the dataset contained in the edsurvey.data.frame
year	the year of assessment of the dataset contained in the edsurvey.data.frame
assessmentCode	the code of the dataset contained in the edsurvey.data.frame
dataType	the type of data (whether student or school) contained in the edsurvey.data.frame
gradeLevel	the grade of the dataset contained in the edsurvey.data.frame
achievementLevels	default NAEP achievement cutoff scores
omittedLevels	the levels of the factor variables that will be omitted from the edsurvey.data.frame
fileFormat	a data.frame containing the parsed information from the student .fr2 file associated with the data
fileFormatSchool	a data.frame containing the parsed information from the school .fr2 file associated with the data
fileFormatTeacher	not applicable for NAEP data; returns NULL
survey	the type of survey data contained in the edsurvey.data.frame

Author(s)

Tom Fink and Ahmad Emad

Examples

```
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package = "NAEPprimer"))
sdf

## Not run:
# To read in an NCES file first set the directory to the ~/Data subfolder,
# then read in the appropriate .dat file:
setwd("location/of/Data")
sdf <- readNAEP(path="M36NT2PM.dat")
```

```
# Or read in the .dat file directly through the folder pathway:
sdf <- readNAEP(path="location/of/Data/M36NT2PM.dat")

## End(Not run)
```

---

readNLS72 *Connect to NLS-72 Study Data*

Description

Opens a connection to a National Longitudinal Study of the High School Class of 1972 (NLS-72) cohort data file and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readNLS72(
  NLS7286_PRI_FilePath,
  NLS7286_SASSyntax_Path,
  forceReread = FALSE,
  verbose = TRUE
)
```

Arguments

`NLS7286_PRI_FilePath`	a character value to the main study derived analyticial data file (NLS7286_REV.PRI). Located within the `REVISED_ASCII` folder.
`NLS7286_SASSyntax_Path`	a character value to the SAS syntax file for parsing the `NLS7286_REV.PRI` data file. Located within the `SAS_EXTRACT_LOGIC` folder.
`forceReread`	a logical value to force rereading of all processed data. The default value of `FALSE` will speed up the read function by using existing read-in data already processed.
`verbose`	a logical value that will determine if you want verbose output while the `readNLS72` function is running to indicate processing progress. The default value is `TRUE`.

Details

Reads in the specified `NLS7286_SASSyntax_Path` file to parse the `NLS7286_PRI_FilePath` file. A cached data file and metadata file will be saved in the same directory and filename as the `NLS7286_PRI_FilePath` file, having new file extensions of .csv and .meta, respectively.

Value

an `edsurvey.data.frame` for the National Longitudinal Study of the High School Class of 1972 longitudinal dataset

Author(s)

Tom Fink

See Also

readHSB_Sophomore, readHSB_Senior, and getData

---

readPIAAC *Connect to PIAAC Data*

Description

Opens a connection to a PIAAC data file and returns an edsurvey.data.frame with information about the file and data.

Usage

```
readPIAAC(path, countries, forceReread = FALSE, verbose = TRUE)
```

Arguments

path	a character value to the full directory path to the PIAAC .csv files and Microsoft Excel codebook
countries	a character vector of the country/countries to include using the three-digit ISO country code. A list of country codes can be found in the PIAAC codebook or https://en.wikipedia.org/wiki/ISO_3166-1#Current_codes. If files are downloaded using downloadPIAAC, a country dictionary text file can be found in the filepath. You can use * to indicate all countries available.
forceReread	a logical value to force rereading of all processed data. Defaults to FALSE. Setting forceReread to be TRUE will cause PIAAC data to be reread and increase the processing time.
verbose	a logical value that will determine if you want verbose output while the function is running to indicate the progress. Defaults to TRUE.

Details

Reads in the unzipped .csv files downloaded from the PIAAC dataset using the OECD repository (http://www.oecd.org/skills/piaac/). Users can use downloadPIAAC to download all required files automatically.

Value

an edsurvey.data.frame for a single specified country or an edsurvey.data.frame.list if multiple countries specified

Author(s)

Trang Nguyen

References

Organisation for Economic Co-operation and Development. (2016). *Technical report of the survey of adult skills (PIAAC)* (2nd ed.). Paris, France: Author. Retrieved from `http://www.oecd.org/skills/piaac/PIAAC_Technical_Report_2nd_Edition_Full_Report.pdf`

See Also

`getData` and `downloadPIAAC`

Examples

```
## Not run:
# the following call returns an edsurvey.data.frame to PIAAC for Canada
can <- readPIAAC("C:/PIAAC", countries = "can")

# extract a data.frame with a few variables
gg <- getData(can, c("c_d05","ageg10lfs"))
head(gg)

# conduct an analysis on the edsurvey.data.frame
edsurveyTable(~ c_d05 + ageg10lfs, data = can)

## End(Not run)
```

readPIRLS *Connect to PIRLS Data*

Description

Opens a connection to a PIRLS data file and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readPIRLS(path, countries, forceReread = FALSE, verbose = TRUE)
```

Arguments

`path`	a character value to the full directory path to the PIRLS extracted SPSS (.sav) set of data

`countries`	a character vector of the country/countries to include using the three-digit ISO country code. A list of country codes can be found on Wikipedia at `https://en.wikipedia.org/wiki/ISO_3166-1#Current_codes` or other online sources. Consult the *PIRLS User Guide* to help determine what countries are included within a specific testing year of PIRLS. To select all countries, use a wildcard value of `*`.
`forceReread`	a logical value to force rereading of all processed data. The default value of `FALSE` will speed up the `readPIRLS` function by using existing read-in data already processed.
`verbose`	a logical value to either print or suppress status message output. The default value is `TRUE`.

Details

Reads in the unzipped files downloaded from the PIRLS international database(s) using the IEA Study Data Repository. Data files require the SPSS data file (.sav) format using the default filenames.

A PIRLS `edsurvey.data.frame` includes three distinct data levels:

- student
- school
- teacher

When the `getData` function is called using a PIRLS `edsurvey.data.frame`, the requested data variables are inspected, and it handles any necessary data merges automatically. The `school` data always will be returned merged to the `student` data, even if only `school` variables are requested. If `teacher` variables are requested by the `getData` call, it will cause `teacher` data to be merged. Many `students` can be linked to many `teachers`, which varies widely between countries.

Please note that calling the `dim` function for a PIRLS `edsurvey.data.frame` will result in the row count as if the `teacher` dataset was merged. This row count will be considered the `full data N` of the `edsurvey.data.frame`, even if no `teacher` data were included in an analysis. The column count returned by `dim` will be the count of unique column variables across all three data levels.

Value

an `edsurvey.data.frame` for a single specified country or an `edsurvey.data.frame.list` if multiple countries specified

Author(s)

Tom Fink

See Also

`readNAEP`, `readTIMSS`, `getData`, and `downloadPIRLS`

Examples

```
## Not run:
nor <- readPIRLS("C:/PIRLS2011", countries = c("nor"))
gg <- getData(nor, c("itsex", "totwgt", "rrea"))
head(gg)
edsurveyTable(rrea ~ itsex, nor)

## End(Not run)
```

readPISA *Connect to PISA Data*

Description

Opens a connection to a PISA data file and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readPISA(
  path,
  database = "INT",
  countries,
  cognitive = "score",
  forceReread = FALSE,
  verbose = TRUE
)
```

Arguments

`path`	a character vector to the full directory path(s) to the PISA-extracted fixed-width files and SPSS control files (.txt).
`database`	a character to indicate a selected database. Must be one of `INT` (general database that most people use), `CBA` (computer-based database in PISA 2012 only), or `FIN` (financial literacy database in PISA 2012 only). Defaults to `INT`.
`countries`	a character vector of the country/countries to include using the three-digit ISO country code. A list of country codes can be found in the PISA codebook or `https://en.wikipedia.org/wiki/ISO_3166-1#Current_codes`. If files are downloaded using `downloadPISA`, a country dictionary text file can be found in the filepath.
`cognitive`	one of `none`, `score`, or `response`. Default is `score`. The PISA database often has three student files: student questionnaire, cognitive item response, and scored cognitive item response. The first file is used as the main student file with student background information. Users can choose whether to merge `score` or `response` data into the main file or not (if `none`).

`forceReread`	a logical value to force rereading of all processed data. Defaults to `FALSE`. Setting `forceReread` to be `TRUE` will cause PISA data to be reread and increase processing time.
`verbose`	a logical value that will determine if you want verbose output while the function is running to indicate progress. Defaults to `TRUE`.

Details

Reads in the unzipped files downloaded from the PISA database using the OECD Repository (`http://www.oecd.org/pisa/`). Users can use `downloadPISA` to download all required files. Student questionnaire files (with weights and plausible values) are used as main files, which are then merged with cognitive, school, and parent files (if available).

The average first-time processing time for 1 year and one database for all countries is 10–15 minutes. If `forceReread` is set to be `FALSE`, the next time this function is called will take only 5–10 seconds.

Value

an `edsurvey.data.frame` for a single specified country or an `edsurvey.data.frame.list` if multiple countries are specified

Author(s)

Tom Fink, Trang Nguyen, and Paul Bailey

References

Organisation for Economic Co-operation and Development. (2017). *PISA 2015 technical report*. Paris, France: OECD Publishing. Retrieved from `http://www.oecd.org/pisa/data/2015-technical-report/`

See Also

`getData` and `downloadPISA`

Examples

```
## Not run:
# the following call returns an edsurvey.data.frame to
# PISA 2012 International Database for Singapore
sgp2012 <- readPISA(path = "C:/PISA/2012", database = "INT", countries = "sgp")

# extract a data.frame with a few variables
gg <- getData(sgp2012, c("cnt","read","w_fstuwt"))
head(gg)

# conduct an analysis on the edsurvey.data.frame
edsurveyTable(read ~ st04q01 + st20q01, data = sgp2012)

## End(Not run)
```

readTALIS *Connect to TALIS Data*

Description

Opens a connection to a TALIS data file and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readTALIS(
  path,
  countries,
  isced = "b",
  dataLevel = "teacher",
  forceReread = FALSE,
  verbose = TRUE
)
```

Arguments

`path`	a character vector to the full directory path(s) to the TALIS SPSS files (.sav)
`countries`	a character vector of the country/countries to include using the three-digit ISO country code. A list of country codes can be found in the TALIS codebook, or you can use `https://en.wikipedia.org/wiki/ISO_3166-1#Current_codes`. You can use `*` to indicate all countries available.
`isced`	a character value that is one of `a`, `b`, or `c`. `a` stands for *Primary Level*, `b` is for *Lower Secondary Level*, and `c` is for *Upper Secondary Level*. Default to `b`.
`dataLevel`	a character value that indicates which data level to be used. It can be `teacher` (the default) or `school`.
`forceReread`	a logical value to force rereading of all processed data. Defaults to `FALSE`. Setting `forceReread` to be `TRUE` will cause PISA data to be reread and increase processing time.
`verbose`	a logical value that will determine if you want verbose output while the function is running to indicate the progress. Defaults to `TRUE`.

Details

Reads in the unzipped files downloaded from the TALIS database using the OECD Repository (`https://www.oecd.org/education/talis/`). If `dataLevel` is set to be `teacher`, it treats the teacher data file as the main dataset and merges school data into teacher data for each country. If `dataLevel` is `school`, it uses only the school data file. To conduct a school-level analysis with teacher variables, it is recommended that users aggregate teacher-level data first before merging it to school files.

Value

an edsurvey.data.frame for a single specified country or an edsurvey.data.frame.list if multiple countries specified

Author(s)

Paul Bailey and Trang Nguyen

References

Organisation for Economic Co-operation and Development. (2014). *TALIS 2013 technical report*. Paris, France: Author. Retrieved from http://www.oecd.org/education/school/TALIS-technical-report-2013.pdf

See Also

getData and downloadTALIS

Examples

```
## Not run:
# the following call returns an edsurvey.data.frame to TALIS 2013
# for US teacher-level data at secondary level
usa2013 <- readTALIS(path = "C:/TALIS/2013", isced = "b",
                     dataLevel = "teacher", countries = "usa")

# extract a data.frame with a few variables
gg <- getData(usa2013, c("tt2g05b", "tt2g01"))
head(gg)

# conduct an analysis on the edsurvey.data.frame
edsurveyTable(tt2g05b ~ tt2g01, data = usa2013)

## End(Not run)
```

---

readTIMSS *Connect to TIMSS Data*

Description

Opens a connection to a TIMSS data file and returns an edsurvey.data.frame with information about the file and data.

Usage

```
readTIMSS(path, countries, gradeLvl, forceReread = FALSE, verbose = TRUE)
```

Arguments

`path`	a character vector to the full directory path(s) to the TIMSS extracted SPSS (.sav) set of data
`countries`	a character vector of the country/countries to include using the three-digit ISO country code. A list of country codes can be found on Wikipedia at `https://en.wikipedia.org/wiki/ISO_3166-1#Current_codes` or other online sources. Consult the *TIMSS User Guide* documentation to help determine what countries are included within a specific testing year of TIMSS and for country code definitions. To select all countries available, use a wildcard value of `*`.
`gradeLvl`	a character value to indicate the specific grade level you wish to return • **4** = fourth grade (the default if not specified) • **8** = eighth grade
`forceReread`	a logical value to force rereading of all processed data. The default value of `FALSE` will speed up the `readTIMSS` function by using existing read-in data already processed.
`verbose`	a logical value to either print or suppress status message output. The default value is `TRUE`.

Details

Reads in the unzipped files downloaded from the TIMSS international database(s) using the IEA Study Data Repository. Data files require the SPSS data file (.sav) format using the default filenames.

A TIMSS `edsurvey.data.frame` includes three distinct data levels:

- student
- school
- teacher

When the `getData` function is called using a TIMSS `edsurvey.data.frame`, the requested data variables are inspected, and it handles any necessary data merges automatically. The `school` data always will be returned merged to the `student` data, even if only `school` variables are requested. If `teacher` variables are requested by the `getData` call, it will cause `teacher` data to be merged. Many `students` can be linked to many `teachers`, which varies widely between countries.

Please note that calling the `dim` function for a TIMSS `edsurvey.data.frame` will result in the row count as if the `teacher` dataset was merged. This row count will be considered the `full data N` of the `edsurvey.data.frame`, even if no `teacher` data were included in an analysis. The column count returned by `dim` will be the count of unique column variables across all three data levels.

Beginning with TIMSS 2015, a `numeracy` dataset was designed to assess mathematics at the end of the primary school cycle for countries where most children are still developing fundamental mathematics skills. The `numeracy` dataset is handled automatically for the user and is included within the fourth-grade dataset `gradeLvl=4`. Most `numeracy` countries have a `4th grade` dataset in addition to their `numeracy` dataset, but some do not. For countries that have both a `numeracy` and a `4th grade` dataset, the two datasets are combined into one `edsurvey.data.frame` for that country. Data variables missing from either dataset are kept, with `NA` values inserted for the dataset records where that variable did not exist. Data variables common to both datasets are kept as a single data variable, with records retaining their original values from the source dataset. Consult the *TIMSS User Guide* for further information.

Value

an `edsurvey.data.frame` for a single specified country or an `edsurvey.data.frame.list` if multiple countries specified

Author(s)

Tom Fink

See Also

`readNAEP`, `getData`, and `downloadTIMSS`

Examples

```
## Not run:
# single country specified
fin <- readTIMSS("C:/TIMSS/2015", countries = c("fin"), gradeLvl = 4)
gg <- getData(fin, c("asbg01", "totwgt", "srea"))
head(gg)
edsurveyTable(srea ~ asbg01, fin)

# multiple countries returned as edsurvey.data.frame.list, specify all countries with '*' argument
timss2011 <- readTIMSS("C:/TIMSS/2011", countries="*", gradeLvl = 8, verbose = TRUE)
# print out edsurvey.data.frame.list covariates
timss2011$covs

## End(Not run)
```

---

readTIMSSAdv *Connect to TIMSS Advanced Data*

Description

Opens a connection to a TIMSS Advanced data file and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
readTIMSSAdv(
  path,
  countries,
  subject = c("math", "physics"),
  forceReread = FALSE,
  verbose = TRUE
)
```

Arguments

`path`	a character vector to the full directory path to the TIMSS Advanced extracted SPSS (.sav) set of data
`countries`	a character vector of the country/countries to include using the three-digit ISO country code. A list of country codes can be found on Wikipedia at `https://en.wikipedia.org/wiki/ISO_3166-1#Current_codes` or other online sources. Consult the *TIMSS Advanced User Guide* to help determine what countries are included within a specific testing year of TIMSS Advanced. To select all countries, use a wildcard value of `*`.
`subject`	a character value to indicate if you wish to import the `math` or `physics` dataset. Only one subject can be read in at a time.
`forceReread`	a logical value to force rereading of all processed data. The default value of `FALSE` will speed up the `readTIMSSAdv` function by using existing read-in data already processed.
`verbose`	a logical value to either print or suppress status message output. The default value is `TRUE`.

Details

Reads in the unzipped files downloaded from the TIMSS Advanced international database(s) using the IEA Study Data Repository. Data files require the SPSS data file (.sav) format using the default filenames.

A TIMSS Advanced `edsurvey.data.frame` includes three distinct data levels:

- student
- school
- teacher

When the `getData` function is called using a TIMSS Advanced `edsurvey.data.frame`, the requested data variables are inspected, and it handles any necessary data merges automatically. The `school` data always will be returned merged to the `student` data, even if only `school` variables are requested. If `teacher` variables are requested by the `getData` call it will cause the `teacher` data to be merged. Many `students` can be linked to many `teachers`, which varies widely between countries.

Please note that calling the `dim` function for a TIMSS Advanced `edsurvey.data.frame` will result in the row count as if the `teacher` dataset was merged. This row count will be considered the `full data N` of the `edsurvey.data.frame`, even if no `teacher` data were included in an analysis. The column count returned by `dim` will be the count of unique column variables across all three data levels.

Value

an `edsurvey.data.frame` for a single specified country or an `edsurvey.data.frame.list` if multiple countries specified

Author(s)

Tom Fink

See Also

readNAEP, readTIMSS, getData, and downloadTIMSSAdv

Examples

```
## Not run:
swe <- readTIMSSAdv("C:/TIMSSAdvanced/Math/2015",
                    countries = c("swe"), subject = "math")
gg <- getData(swe, c("itsex", "totwgt", "malg"))
head(gg)
edsurveyTable(malg ~ itsex, swe)

## End(Not run)
```

`read_ePIRLS` *Connect to ePIRLS Data*

Description

Opens a connection to an ePIRLS data file and returns an `edsurvey.data.frame` with information about the file and data.

Usage

```
read_ePIRLS(path, countries, forceReread = FALSE, verbose = TRUE)
```

Arguments

`path`	a character value to the full directory path to the ePIRLS extracted SPSS (.sav) set of data
`countries`	a character vector of the country/countries to include using the three-digit ISO country code. A list of country codes can be found on Wikipedia at `https://en.wikipedia.org/wiki/ISO_3166-1#Current_codes` or other online sources. Consult the *ePIRLS User Guide* to help determine what countries are included within a specific testing year of ePIRLS. To select all countries, use a wildcard value of `*`.
`forceReread`	a logical value to force rereading of all processed data. The default value of `FALSE` will speed up the `read_ePIRLS` function by using existing read-in data already processed.
`verbose`	a logical value to either print or suppress status message output. The default value is `TRUE`.

Details

Reads in the unzipped files downloaded from the ePIRLS international database(s) using the IEA Study Data Repository. Data files require the SPSS data file (.sav) format using the default filenames.

An ePIRLS `edsurvey.data.frame` includes three distinct data levels:

- student
- school
- teacher

When the `getData` function is called using an ePIRLS `edsurvey.data.frame`, the requested data variables are inspected, and it handles any necessary data merges automatically. The `school` data always will be returned merged to the `student` data, even if only `school` variables are requested. If `teacher` variables are requested by the `getData` call, it will cause `teacher` data to be merged. A `student` can be linked to many `teachers`, which varies widely between countries.

Please note that calling the `dim` function for an ePIRLS `edsurvey.data.frame` will result in the row count as if the `teacher` dataset was merged. This row count will be considered the `full data N` of the `edsurvey.data.frame`, even if no `teacher` data were included in an analysis. The column count returned by `dim` will be the count of unique column variables across all three data levels.

Value

an `edsurvey.data.frame` for a single specified country or an `edsurvey.data.frame.list` if multiple countries are specified

Author(s)

Tom Fink

See Also

`readNAEP`, `readTIMSS`, `getData`, and `download_ePIRLS`

Examples

```
## Not run:
usa <- read_ePIRLS("C:/ePIRLS/2016", countries = c("usa"))
gg <- getData(usa, c("itsex", "totwgt", "erea"))
head(gg)
edsurveyTable(erea ~ itsex, usa)

## End(Not run)
```

`rebindAttributes` *Copy Data Frame Attributes*

Description

Many R functions strip attributes from data frame objects. This function assigns the attributes from the `attributeData` argument to the data frame in the `data` argument.

Usage

```
rebindAttributes(data, attributeData)
```

Arguments

`data`	a `data.frame`
`attributeData`	an `edsurvey.data.frame` or `light.edsurvey.data.frame` that contains the desired attributes

Value

a `data.frame` with a class of a `light.edsurvey.data.frame` containing all elements of data and the attributes (except `names` and `row.names`) from `attributeData`

Author(s)

Paul Bailey and Trang Nguyen

Examples

```
## Not run:
require(dplyr)
PISA2012 <- readPISA(path = paste0(edsurveyHome, "PISA/2012"),
                     database = "INT",
                     countries = "ALB", verbose=TRUE)
ledf <- getData(data = PISA2012, varnames = c("cnt", "oecd", "w_fstuwt",
                                              "st62q04", "st62q11",
                                              "st62q13", "math"),
                omittedLevels = FALSE, addAttributes = TRUE)

omittedLevels <- c('Invalid', 'N/A', 'Missing', 'Miss', 'NA', '(Missing)')
for (i in c("st62q04", "st62q11", "st62q13")) {
  ledf[,i] <- factor(ledf[,i], exclude=omittedLevels)
}

# after applying some dplyr functions, the "light.edsurvey.data.frame" becomes just "data.frame"
PISA2012_ledf <- ledf %>%
  rowwise() %>%
  mutate(avg_3 = mean(c(st62q04, st62q11, st62q13), na.rm = TRUE)) %>%
  ungroup() %>%
  rebindAttributes(PISA2012) # could also be called with ledf
class(PISA2012_ledf)
# again, a light.edsurvey.data.frame
lma <- lm.sdf(math ~ avg_3,data=PISA2012_ledf)
summary(lma)

PISA2012_ledf <- ledf %>%
  rowwise() %>%
  mutate(avg_3 = mean(c(st62q04, st62q11, st62q13), na.rm = TRUE)) %>%
  ungroup() %>%
  rebindAttributes(ledf) # return attributes and make a light.edsurvey.data.frame
# again a light.edsurvey.data.frame
lma <- lm.sdf(math ~ avg_3,data=PISA2012_ledf)
summary(lma)

## End(Not run)
```

recode.sdf *Recode Levels Within Variables*

Description

Recodes variables in an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`.

Usage

```
recode.sdf(x, recode)
```

Arguments

`x`	an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`
`recode`	a list of recoding rules. See Examples for the format of recoding rules.

Value

an object of the same class as `x` with the `recode` added to it

Author(s)

Trang Nguyen and Paul Bailey

Examples

```
## Not run:
# filepath argument will vary by operating system conventions
usaG4.15 <- readTIMSS("C:/TIMSS2015", "usa", 4)
d <- getData(usaG4.15, "itsex")
summary(d) #show details: MALE/FEMALE
usaG4.15 <- recode.sdf(usaG4.15,
                          recode = list(itsex=list(from=c("MALE"),
                                                   to=c("BOY")),
                                        itsex=list(from=c("FEMALE"),
                                                   to=c("GIRL"))))

d <- getData(usaG4.15, "itsex") #apply recode
summary(d) #show details: BOY/GIRL

## End(Not run)
```

rename.sdf *Modify Variable Names*

Description

Renames variables in an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`. This function often is used when users want to conduct a gap analysis across years but variable names differ across two years of data.

Usage

```
rename.sdf(x, oldnames, newnames, avoid_duplicated = TRUE)
```

Arguments

`x`	an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`
`oldnames`	a character vector of old variable names
`newnames`	a character vector of new variable names to replace the corresponding old names
`avoid_duplicated`	a logical value to indicate whether to avoid renaming the variable if the corresponding new name already exists in the data. Defaults to `TRUE`.

Details

All variable names are coerced to lowercase to comply with the `EdSurvey` standard.

Value

an object of the same class as `x` with new variable names

Author(s)

Trang Nguyen

See Also

`gap`

Examples

```
## Not run:
usaG4.15 <- readTIMSS("C:/TIMSS2015", "usa", 4)
usaG4.15.renamed <- rename.sdf(usaG4.15,
                               c("itsex", "mmat"),
                               c("gender", "math_overall"))
lm1 <- lm.sdf(math_overall ~ gender, data = usaG4.15.renamed)
summary(lm1)

## End(Not run)
```

`rq.sdf` *EdSurvey Quantile Regression Models*

Description

Fits a quantile regression model that uses weights and variance estimates appropriate for the data.

Usage

```
rq.sdf(
  formula,
  data,
  tau = 0.5,
  weightVar = NULL,
  relevels = list(),
  jrrIMax = 1,
  omittedLevels = TRUE,
  defaultConditions = TRUE,
  recode = NULL,
  returnNumberOfPSU = FALSE,
  ...
)
```

Arguments

`formula`	a `formula` for the quantile regression model. See `rq` in the `quantreg` package. If *y* is left blank, the default subject scale or subscale variable will be used. (You can find the default using `showPlausibleValues`.) If *y* is a variable for a subject scale or subscale (one of the names shown by `showPlausibleValues`), then that subject scale or subscale is used.
`data`	an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`
`tau`	the quantile to be estimated. The value could be set between 0 and 1 with a default of 0.5.
`weightVar`	a character indicating the weight variable to use. The `weightVar` must be one of the weights for the `edsurvey.data.frame`. If `NULL`, it uses the default for the `edsurvey.data.frame`.

`relevels`	a list. Used to change the contrasts from the default treatment contrasts to the treatment contrasts with a chosen omitted group (the reference group). The name of each element should be the variable name, and the value should be the group to be omitted (the reference group).
`jrrIMax`	when using the jackknife variance estimation method, the default estimation option, `jrrIMax=1`, uses the sampling variance from the first plausible value as the component for sampling variance estimation. The V_{jrr} term can be estimated with any number of plausible values, and values larger than the number of plausible values on the survey (including `Inf`) will result in all plausible values being used. Higher values of `jrrIMax` lead to longer computing times and more accurate variance estimates.
`omittedLevels`	a logical value. When set to the default value of `TRUE`, drops those levels of all factor variables that are specified in an `edsurvey.data.frame`. Use `print` on an `edsurvey.data.frame` to see the omitted levels.
`defaultConditions`	a logical value. When set to the default value of `TRUE`, uses the default conditions stored in an `edsurvey.data.frame` to subset the data. Use `print` on an `edsurvey.data.frame` to see the default conditions.
`recode`	a list of lists to recode variables. Defaults to `NULL`. Can be set as `recode=list(var1 = list(from= c("a", "b", "c"), to= "d"))`.
`returnNumberOfPSU`	a logical value set to `TRUE` to return the number of primary sampling units (PSUs)
`...`	additional parameters passed from `rq`

Details

The function computes an estimate on the `tau`-th conditional quantile function of the response, given the covariates, as specified by the formula argument. Like `lm.sdf()`, the function presumes a linear specification for the quantile regression model (i.e., that the formula defines a model that is linear in parameters). Unlike `lm.sdf()`, the jackknife is the only applicable variance estimation method used by the function.

For further details on quantile regression models and how they are implemented in R, see Koenker and Bassett (1978), Koenker (2005), and the vignette from the `quantreg` package— accessible by `vignette("rq",package="quantreg")`—on which this function is built.

For further details on how left-hand side variables, survey sampling weights, and estimated variances are correctly handled, see `lm.sdf` or the vignette titled *Statistical Methods Used in EdSurvey*.

Value

An `edsurvey.rq` with the following elements:

`call`	the function call
`formula`	the formula used to fit the model
`tau`	the quantile to be estimated
`coef`	the estimates of the coefficients

`se`	the standard error estimates of the coefficients
`Vimp`	the estimated variance from uncertainty in the scores (plausible value variables)
`Vjrr`	the estimated variance from sampling
`M`	the number of plausible values
`varm`	the variance estimates under the various plausible values
`coefm`	the values of the coefficients under the various plausible values
`coefmat`	the coefficient matrix (typically produced by the summary of a model)
`weight`	the name of the weight variable
`npv`	the number of plausible values
`njk`	the number of the jackknife replicates used; set to `NA` when Taylor series variance estimates are used

Author(s)

Trang Nguyen, Paul Bailey, and Yuqi Liao

References

Binder, D. A. (1983). On the variances of asymptotically normal estimators from complex surveys. *International Statistical Review*, *51*(3), 279–292.

Johnson, E. G., & Rust, K. F. (1992). Population inferences and variance estimation for NAEP data. *Journal of Education Statistics*, *17*(2), 175–190.

Koenker, R. W., & Bassett, G. W. (1978). Regression quantiles, *Econometrica, 46,* 33–50.

Koenker, R. W. (2005). *Quantile regression*. Cambridge, UK: Cambridge University Press.

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys*. New York, NY: Wiley.

See Also

`rq`

Examples

```
## Not run:
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package = "NAEPprimer"))

# conduct quantile regression at a given tau value (by default, tau is set to be 0.5)
rq1 <- rq.sdf(composite ~ dsex + b017451, data=sdf, tau = 0.8)
summary(rq1)

## End(Not run)
```

SD *EdSurvey Standard Deviation*

Description

Calculate the standard deviation of a numeric variable in an `edsurvey.data.frame`.

Usage

```
SD(
  data,
  variable,
  weightVar = NULL,
  jrrIMax = 1,
  varMethod = "jackknife",
  omittedLevels = TRUE,
  defaultConditions = TRUE,
  recode = NULL,
  targetLevel = NULL,
  jkSumMultiplier = 1,
  returnVarEstInputs = FALSE
)
```

Arguments

`data`	an `edsurvey.data.frame`, an `edsurvey.data.frame.list`, or a `light.edsurvey.data.frame`
`variable`	character vector of variable names
`weightVar`	character weight variable name. Default is the default weight of `data` if it exists. If the given survey data do not have a default weight, the function will produce unweighted statistics instead. Can be set to `NULL` to return unweighted statistics.
`jrrIMax`	a numeric value; when using the jackknife variance estimation method, the default estimation option, `jrrIMax=1`, uses the sampling variance from the first plausible value as the component for sampling variance estimation. The `Vjrr` term (see *Statistical Methods Used in EdSurvey*) can be estimated with any number of plausible values, and values larger than the number of plausible values on the survey (including `Inf`) will result in all plausible values being used. Higher values of `jrrIMax` lead to longer computing times and more accurate variance estimates.
`varMethod`	deprecated parameter; gap always uses the jackknife variance estimation
`omittedLevels`	a logical value. When set to `TRUE`, drops those levels of the specified `variable`. Use print on an `edsurvey.data.frame` to see the omitted levels. Defaults to `FALSE`.
`defaultConditions`	a logical value. When set to the default value of `TRUE`, uses the default conditions stored in an `edsurvey.data.frame` to subset the data. Use `print` on an `edsurvey.data.frame` to see the default conditions.

`recode`	a list of lists to recode variables. Defaults to `NULL`. Can be set as `recode = list(var1 = list(from = c("a","b","c"), to = "d"))`.
`targetLevel`	a character string. When specified, calculates the gap in the percentage of students at `targetLevel` in the `variable` argument, which is useful for comparing the gap in the percentage of students at a survey response level.
`jkSumMultiplier`	when the jackknife variance estimation method—or balanced repeated replication (BRR) method—multiplies the final jackknife variance estimate by a value, set `jkSumMultiplier` to that value. For an `edsurvey.data.frame`, or a `light.edsurvey.data.frame`, the recommended value can be recovered with `EdSurvey::getAttributes(myData, "jkSumMultiplier")`.
`returnVarEstInputs`	a logical value set to `TRUE` to return the inputs to the jackknife and imputation variance estimates, which allows for the computation of covariances between estimates.

Value

`SD` returns:

mean the mean assessment score for `variable`, calculated according to the vignette titled Statistical Methods Used in EdSurvey

std the standard deviation of the `mean`

stdSE the standard error of the `std`

sd the degrees of freedom of the `std`

When `returnVarEstInputs` is `TRUE`, an attribute `varEstInputs` also is returned that includes the variance estimate inputs used for calculating covariances with `varEstToCov`.

Author(s)

Paul Bailey and Huade Huo

Examples

```
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package="NAEPprimer"))

# get standard deviation for Male's composite score
SD(data = subset(sdf, dsex == "Male"), variable = "composite")

# get several standard deviations

# build an edsurvey.data.frame.list
sdfA <- subset(sdf, scrpsu %in% c(5,45,56))
sdfB <- subset(sdf, scrpsu %in% c(75,76,78))
sdfC <- subset(sdf, scrpsu %in% 100:200)
sdfD <- subset(sdf, scrpsu %in% 201:300)
```

```
sdfl <- edsurvey.data.frame.list(list(sdfA, sdfB, sdfC, sdfD),
                                 labels=c("A locations",
                                          "B locations",
                                          "C locations",
                                          "D locations"))

# this shows how these datasets will be described:
sdfl$covs

SD(data = sdfl, variable = "composite")
```

---

searchSDF *EdSurvey Codebook Search*

Description

Retrieves variable names and labels for an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list` using character string matching.

Usage

```
searchSDF(string, data, fileFormat = NULL, levels = FALSE)
```

Arguments

`string`	a vector of character strings to search for in the database connection object (`data`). The function will search the codebook for a matching character string using regular expressions. When a string has several elements, all must be present for a variable to be returned.
`data`	an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`
`fileFormat`	a character string indicating the data source to search for variables. The default `NULL` argument searches all codebooks.
`levels`	a logical value; set to `TRUE` to return a snapshot of the levels in an `edsurvey.data.frame`

Value

a `data.frame` that shows the variable names, labels, and levels (if applicable) from an `edsurvey.data.frame` or a `light.edsurvey.data.frame` based on a matching character string

Author(s)

Michael Lee and Paul Bailey

Examples

```
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package="NAEPprimer"))

# search both the student and school files by a character string
searchSDF(string="book", data=sdf)

# use the `|` (OR) operator to search several strings simultaneously
searchSDF(string="book|home|value", data=sdf)

# use a vector of strings to search for variables that contain multiple strings,
# such as both "book" and "home"
searchSDF(string=c("book","home"), data=sdf)

# search only the student files by a character string
searchSDF(string="algebra", data=sdf, fileFormat="student")

# search both the student and school files and return a glimpse of levels
searchSDF(string="value", data=sdf, levels=TRUE)

# save the search as an object to return a full data.frame of search
ddf <- searchSDF(string="value", data=sdf, levels=TRUE)
ddf
```

---

showCodebook *Summary Codebook*

Description

Retrieves variable names, variable labels, and value labels for an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`.

Usage

```
showCodebook(
  data,
  fileFormat = NULL,
  labelLevels = FALSE,
  includeRecodes = FALSE
)
```

Arguments

`data`	an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`
`fileFormat`	a character string indicating the data source to search for variables. The default `NULL` argument searches all available codebooks in the database connection object.

`labelLevels`	a logical value; set to `TRUE` to return a snapshot of the label levels in an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`. When set to `FALSE` (the default), label levels are removed.
`includeRecodes`	a logical value; set to `TRUE` to return value labels that have been recoded in an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`. When set to `FALSE` (the default), only the original value labels are included in the returned `data.frame`.

Value

a `data.frame` that shows the variable names, variable labels, value labels, value levels (if applicable), and the file format data source from an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`

Author(s)

Michael Lee and Paul Bailey

Examples

```
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package="NAEPprimer"))

# search both the student and school files, returning levels for variable values
showCodebook(sdf, c("student","school"), labelLevels = TRUE, includeRecodes = FALSE)

# return codebook information for the student file, excluding variable value levels,
# including recoded variables
sdf <- recode.sdf(sdf, recode = list(dsex = list(from = c("Male"), to = c("MALE"))))
showCodebook(sdf, c("student"), labelLevels = FALSE, includeRecodes = TRUE)

# return codebook information for the student file, including variable value levels
# and recoded variables
showCodebook(sdf, c("student","school"), labelLevels = FALSE, includeRecodes = TRUE)

# return codebook information for all codebooks in an edsurvey.data.frame; commonly use View()
## Not run:
View(showCodebook(sdf))

## End(Not run)
```

`showCutPoints` *Retrieve Achievement Level Cutpoints*

Description

Retrieves a summary of the achievement level cutpoints for a selected study represented in an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`.

Usage

```
showCutPoints(data)
```

Arguments

`data`	an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`

Author(s)

Michael Lee and Paul Bailey

Examples

```
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("exttdata/data", "M36NT2PM.dat", package="NAEPprimer"))

# show the cut points
showCutPoints(data=sdf)
```

`showPlausibleValues` *Plausible Value Variable Names*

Description

Prints a summary of the subject scale or subscale and the associated variables for their plausible values for an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`.

Usage

```
showPlausibleValues(data, verbose = FALSE)
```

Arguments

`data`	an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`
`verbose`	a logical value; set to `TRUE` to get the variable names for plausible values. Otherwise, prints only the subject scale or subscale names for variables that use plausible values.

Author(s)

Michael Lee and Paul Bailey

Examples

```
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package="NAEPprimer"))

# show the plausible values
showPlausibleValues(data=sdf, verbose=TRUE)
```

showWeights *Retrieve Weight Variables*

Description

Prints a summary of the weights in an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`.

Usage

```
showWeights(data, verbose = FALSE)
```

Arguments

`data`	an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`
`verbose`	a logical value; set to TRUE to print the complete list of jackknife replicate weights associated with each full sample weight; otherwise, prints only the full sample weights

Author(s)

Michael Lee and Paul Bailey

Examples

```
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package="NAEPprimer"))

# show the weights
showWeights(sdf, TRUE)
```

subset *EdSurvey Subset*

Description

Subsets an `edsurvey.data.frame`, an `edsurvey.data.frame.list`, or a `light.edsurvey.data.frame`.

Usage

```
## S3 method for class 'edsurvey.data.frame'
subset(x, subset, ..., inside = FALSE)
```

Arguments

`x`	an `edsurvey.data.frame`, an `edsurvey.data.frame.list`, or a `light.edsurvey.data.frame`
`subset`	a logical expression indicating elements or rows to keep
`...`	not used; included only for compatibility
`inside`	set to `TRUE` to prevent the `substitute` condition from being called on it (see Details)

Details

Any variables defined on condition that are not references to column names on the `edsurvey.data.frame` and are part of the environment where `subset.edsurvey.data.frame` was called will be evaluated in the environment from which `subset.edsurvey.data.frame` was called. Similar to the difficulty of using subset within a function call because of the call to substitute on condition, this function is difficult to use (with `inside` set to the default value of `FALSE`) inside another function call. See Examples for how to call this function from within another function.

Value

an object of the same class as `x`

Author(s)

Paul Bailey and Trang Nguyen

References

Wickham, H. (2014). *Advanced R*. Boca Raton, FL: Chapman & Hall/CRC.

Examples

```
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package = "NAEPprimer"))

# table to compare to subsequent tables with subsets
edsurveyTable(composite ~ dsex, data=sdf, returnMeans=FALSE, returnSepct=FALSE)

# subset to just males
newsdf <-  subset(x=sdf, subset= dsex == "Male")
# table of dsex after subset
edsurveyTable(composite ~ dsex, data=newsdf, returnMeans=FALSE, returnSepct=FALSE)

# Variable names that are not in the sdf get resolved in the parent frame.
# practically, that means that the following two subset
# calls sdfM1 and sdfM2 do the same thing
male_var <- "Male"
sdfM1 <- subset(x=sdf, subset= dsex == male_var)
sdfM2 <- subset(x=sdf, subset= dsex == "Male")
table(getData(data=sdfM1, varnames="dsex"))
table(getData(data=sdfM2, varnames="dsex"))
```

```
# variable can also be resolved as members of lists
genders <- c("Male", "Female","not a sex level")
sdfn <- subset(x=sdf, subset= dsex == genders[2])
table(getData(data=sdfn, varnames="dsex"))

# variables can also be subset using %in%
sdfM3 <- subset(x=sdf, subset= dsex %in% c("Male", "not a sex level"))
table(getData(data=sdfM3, varnames="dsex"))

# if you need to call a name on the sdf dynamically, you can use as.name
dsex_standin <- as.name("dsex")
sdfM4 <- subset(x=sdf, subset= dsex_standin == "Male")
table(getData(data=sdfM4, varnames="dsex"))

# Here is an example of how one might want to call
# subset from within a function or loop.
# First, define a few variables to use dynamically
rhs_vars <- c("dsex", "b017451")
lvls <- c("Male", "Female")

# create a parsed condition
cond <- parse(text=paste0(rhs_vars[1], " == \"",lvls[1],"\""))[[1]]

# when inside=TRUE a parsed condition can be passed to subset
dsdf <- subset(x=sdf, subset=cond, inside=TRUE)

# check the result
table(getData(data=dsdf, varnames="dsex"))

# returns data, but uses substantial memory
## Not run:
head(sdf[c("origwt","m145101")])

# subset an edsurvey.data.frame.list
sdfA <- subset(sdf, scrpsu %in% c(5,45,56))
sdfB <- subset(sdf, scrpsu %in% c(75,76,78))
sdfC <- subset(sdf, scrpsu %in% 100:200)
sdfD <- subset(sdf, scrpsu %in% 201:300)

# construct an edsurvey.data.frame.list from these four datasets
sdfl <- edsurvey.data.frame.list(list(sdfA, sdfB, sdfC, sdfD),
                                 labels=c("A locations",
                                          "B locations",
                                          "C locations",
                                          "D locations"))

sdfl2 <- subset(sdfl, dsex=="Male")
# the number of rows in each element of the sdfl
nrow(sdfl)
# the number of rows after subsetting each element to just the Males
nrow(sdfl2)

## End(Not run)
```

summary2 *Summarize edsurvey.data.frame Variables*

Description

Summarizes `edsurvey.data.frame` variables.

Usage

```
summary2(
  data,
  variable,
  weightVar = attr(getAttributes(data, "weights"), "default"),
  omittedLevels = FALSE
)
```

Arguments

`data`	an `edsurvey.data.frame`, an `edsurvey.data.frame.list`, or `light.edsurvey.data.frame`
`variable`	character vector of variable names
`weightVar`	character weight variable name. Default is the default weight of `data` if it exists. If the given survey data do not have a default weight, the function will produce unweighted statistics instead. Can be set to `NULL` to return unweighted statistics.
`omittedLevels`	a logical value. When set to `TRUE`, drops those levels of the specified `variable`. Use print on an `edsurvey.data.frame` to see the omitted levels. Defaults to `FALSE`.

Value

summary of weighted or unweighted statistics of a given variable in an `edsurvey.data.frame`

For categorical variables, the summary results are a crosstab of all variables and include the following:

`[variable name]`	level of the variable in the column name that the row regards. There is one column per element of `variable`.
`N`	number of cases for each category. Weighted N also is produced if users choose to produce weighted statistics.
`Percent`	percentage of each category. Weighted percent also is produced if users choose to produce weighted statistics.
`SE`	standard error of the percentage statistics

For continuous variables, the summary results are by variable and include the following:

`Variable`	name of the variable the row regards
`N`	total number of cases (both valid and invalid cases)

`Min.`	smallest value of the variable
`1st Qu.`	first quantile of the variable
`Median`	median value of the variable
`Mean`	mean of the variable
`3rd Qu.`	third quantile of the variable
`Max.`	largest value of the variable
`SD`	standard deviation or weighted standard deviation
`NA's`	number of `NA` in variable and in weight variables
`Zero-weights`	number of zero-weight cases if users choose to produce weighted statistics

If the weight option is chosen, the function produces weighted percentile and standard deviation. Refer to the vignette titled *Statistical Methods Used in EdSurvey* and the vignette titled *Methods Used for Estimating Percentiles in EdSurvey* for how the function calculates these statistics (with and without plausible values).

Author(s)

Paul Bailey and Trang Nguyen

See Also

`percentile`

Examples

```
## Not run:
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package = "NAEPprimer"))

# print out summary of weighted statistics of a continuous variable
summary2(sdf, "composite")
# print out summary of weighted statistics of a variable, including omitted levels
summary2(sdf, "b017451", omittedLevels = FALSE)
# make a crosstab
summary2(sdf, c("b017451", "dsex"), omittedLevels = FALSE)

# print out summary of unweighted statistics of a variable
summary2(sdf, "composite", weightVar = NULL)

## End(Not run)
```

updatePlausibleValue *Update Plausible Value Variable Names*

Description

Changes the name used to refer to a set of plausible values from `oldVar` to `newVar` in an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`.

Usage

```
updatePlausibleValue(oldVar, newVar, data)
```

Arguments

`oldVar`	a character value indicating the existing name of the variable
`newVar`	a character value indicating the new name of the variable
`data`	an `edsurvey.data.frame`, a `light.edsurvey.data.frame`, or an `edsurvey.data.frame.list`

Value

an object of the same class as the `data` argument, with the name of the plausible value updated from `oldVar` to `newVar`

Author(s)

Michael Lee and Paul Bailey

See Also

`getPlausibleValue` and `showPlausibleValues`

Examples

```
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package="NAEPprimer"))

# get the PVs before
showPlausibleValues(sdf)
sdf2 <- updatePlausibleValue("composite", "overall", sdf)
showPlausibleValues(sdf2)
lm1 <- lm.sdf(overall ~ b017451, data=sdf2)
summary(lm1)
```

`varEstToCov` *Covariance Estimation*

Description

When the variance of a derived statistic (e.g., a difference) is required, the covariance between the two statistics must be calculated. This function uses results generated by various functions (e.g., a `lm.sdf`) to find the covariance between two statistics.

Usage

```
varEstToCov(varEstA, varEstB = varEstA, varA, varB = varA, jkSumMultiplier)
```

Arguments

`varEstA`	a list of two `data.frame`s returned by a function after the `returnVarEstInputs` argument was turned on. The statistic named in the `varA` argument must be present in each `data.frame`.
`varEstB`	a list of two `data.frame`s returned by a function after the `returnVarEstInputs` argument was turned on. The statistic named in the `varA` argument must be present in each `data.frame`. When the same as `varEstA`, the covariance is within one result.
`varA`	a character that names the statistic in the `varEstA` argument for which a covariance is required
`varB`	a character that names the statistic in the `varEstB` argument for which a covariance is required
`jkSumMultiplier`	when the jackknife variance estimation method—or balanced repeated replication (BRR) method—multiplies the final jackknife variance estimate by a value, set `jkSumMultiplier` to that value. For an `edsurvey.data.frame` or a `light.edsurvey.data.frame`, the recommended value can be recovered with `EdSurvey::getAttributes(myData, "jkSumMultiplier")`.

Details

These functions are not vectorized, so `varA` and `varB` must contain exactly one variable name.

The method used to compute the covariance is in the vignette titled *Statistical Methods Used in EdSurvey*

The method used to compute the degrees of freedom is in the vignette titled *Statistical Methods Used in EdSurvey* in the section "Estimation of Degrees of Freedom."

Value

a numeric value; the jackknife covariance estimate

Author(s)

Paul Bailey

Examples

```
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package = "NAEPprimer"))

# estimate a regression
lm1 <- lm.sdf(composite ~ dsex + b017451, sdf, returnVarEstInputs=TRUE)
summary(lm1)
# estimate the covariance between two regression coefficients
# note that the variable names are parallel to what they are called in lm1 output
covFEveryDay <- varEstToCov(lm1$varEstInputs,
                            varA="dsexFemale",
                            varB="b017451Every day",
                       jkSumMultiplier=EdSurvey:::getAttributes(sdf, "jkSumMultiplier"))
# the estimated difference between the two coefficients
# note: unname prevents output from being named after the first coefficient
unname(coef(lm1)["dsexFemale"] - coef(lm1)["b017451Every day"])
# the standard error of the difference
# uses the formula SE(A-B) = sqrt(var(A) + var(B) - 2*cov(A,B))
sqrt(lm1$coefmat["dsexFemale", "se"]^2
     + lm1$coefmat["b017451Every day", "se"]^2
     - 2 * covFEveryDay)
```

`waldTest` *Wald Tests*

Description

Tests on coefficient(s) of `edsurveyGlm` and `edsurveyLm` models.

Usage

```
waldTest(model, coefficients, H0 = NULL)
```

Arguments

`model`	an `edsurveyGlm` and `edsurveyLm`
`coefficients`	coefficients to be tested, by name or position in `coef` vector. See Details.
`H0`	reference values to test coefficients against, default = 0

Details

When plausible values are present, likelihood ratio tests cannot be used. However, the Wald test can be used to test estimated parameters in a model, with the null hypothesis being that a parameter(s) is equal to some value(s). In the default case where the null hypothesis value of the parameters is 0, if the test fails to reject the null hypothesis, removing the variables from the model will not substantially harm the fit of that model. Alternative null hypothesis values also can be specified with the `H0` argument. See Examples.

Coefficients to test can be specified by an integer (or integer vector) corresponding to the order of coefficients in the summary output. Coefficients also can be specified using a character vector, to specify coefficient names to test. The name of a factor variable can be used to test all levels of that variable.

This test produces both chi-square and F-tests; their calculation is described in the vignette titled *Methods and Overview of Using EdSurvey for Running Wald Tests*.

Value

An `edsurveyWaldTest` object with the following elements:

`Sigma`	coefficient covariance matrix
`coefficients`	indices of the coefficients tested
`H0`	null hypothesis values of coefficients tested
`result`	result object containing the values of the chi-square and F-tests
`hypoMatrix`	hypothesis matrix used for the Wald Test

Author(s)

Alex Lishinski and Paul Bailey

References

Diggle, P. J., Liang, K.-Y., & Zeger, S. L. (1994). *Analysis of longitudinal data*. Oxford, UK: Clarendon Press.

Draper, N. R., & Smith, H. (1998). *Applied regression analysis*. New York, NY: Wiley.

Fox, J. (1997). *Applied regression analysis, linear models, and related methods*. Thousand Oaks, CA: SAGE.

[Institute for Digital Research and Education. (n.d.). FAQ: How are the likelihood ratio, Wald, and LaGrange multiplier (score) tests different and/or similar?](https://stats.idre.ucla.edu/other/mult-pkg/faq/general/faqhow-are-the-likelihood-ratio-wald-and-lagrange-multiplier-score-tests-different-andor-similar/). Los Angeles: University of California at Los Angeles. Retrieved from https://stats.idre.ucla.edu/other/mult-pkg/faq/general/faqhow-are-the-likelihood-ratio-wald-and-lagrange-multiplier-score-tests-different-andor-similar/

Korn, E., & Graubard, B. (1990). Simultaneous testing of regression coefficients with complex survey data: Use of Bonferroni t statistics. *The American Statistician*, *44*(4), 270–276.

Examples

```
## Not run:
# read in the example data (generated, not real student data)
sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package = "NAEPprimer"))

# example with glm model
myLogit <- logit.sdf(dsex ~ b017451 + b003501, data = sdf, returnVarEstInputs = T)

# single coefficient integer
waldTest(model = myLogit, coefficients = 2)

# set of coefficients integer vector
waldTest(model = myLogit, coefficients = 2:5)

# specify levels of factor variable to test
waldTest(myLogit, c("b017451Every day", "b017451About once a week"))

# specify all levels of factor variable to test
waldTest(myLogit, "b017451")

# example with lm model
fit <- lm.sdf(composite ~ dsex + b017451, data = sdf, returnVarEstInputs = T)

waldTest(model = fit, coefficients = "b017451")

# examples with alternative (nonzero) null hypothesis values
waldTest(model = myLogit, coefficients = 2, H0 = 0.5)

waldTest(model = myLogit, coefficients = 2:5, H0 = c(0.5, 0.6, 0.7, 0.8))

waldTest(model = myLogit, coefficients = "b017451", H0 = c(0.5, 0.6, 0.7, 0.8))

waldTest(model = myLogit, coefficients = c("b017451Every day", "b017451About once a week"),
         H0 = c(0.1, 0.2))

## End(Not run)
```

Index

$.edsurvey.data.frame
    (edsurvey.data.frame), 28
$<-.edsurvey.data.frame
    (edsurvey.data.frame), 28

achievementLevels, *4*, 4, *42*
append.edsurvey.data.frame.list
    (edsurvey.data.frame.list), 32
as.data.frame, *8*, 8

cbind, *9*, 9
contourPlot, 10
cor.sdf, *4*, 11

dim.edsurvey.data.frame, 14
DoFCorrection, 15
download_ePIRLS, 27, *114*
downloadCivEDICCS, 16, *89*
downloadECLS_K, 17, *90*, *92, 93*
downloadELS, 18
downloadHSLS, 19
downloadICILS, 20
downloadPIAAC, 21, *103, 104*
downloadPIRLS, 22, *105*
downloadPISA, 23, *106, 107*
downloadTALIS, 24, *109*
downloadTIMSS, 25, *111*
downloadTIMSSAdv, 26, *113*

EdSurvey-package, 4
edsurvey.data.frame, 28, *48*
edsurvey.data.frame.list, *31*, 32
edsurveyTable, *4*, 35, *39*, *42*
edsurveyTable2pdf, 38

gap, *4*, *31*, 40, *117*
getAttributes (edsurvey.data.frame), 28
getData, *4*, *12*, *31*, 47, *78, 79*, *81–84*, *86, 87*,
    89, 90, *92, 93*, *95, 96*, *98, 99*,
    103–105, *107*, *109*, *111*, *113, 114*
getPlausibleValue, 50, *132*
getPSUVar (edsurvey.data.frame), 28
getStratumVar (edsurvey.data.frame), 28
getWeightJkReplicates, 51
glm (glm.sdf), 52
glm.sdf, 52

hasPlausibleValue, *50*, 56

isWeight, 57

levelsSDF, 58
lm (lm.sdf), 59
lm.sdf, *4*, *12*, *15*, *36*, *42*, 59, *67*, *70*, *119*, *133*
logit.sdf, *4*
logit.sdf (glm.sdf), 52

merge, 63
mixed.sdf, *4*, 64
mvrlm.sdf, 67

oddsRatio, 71

percentile, *4*, *42*, 72, *131*
print.achievementLevels, 75
print.edsurvey.data.frame, 76
print.gap, 76
print.gapList (print.gap), 76
probit.sdf (glm.sdf), 52

rbind, *9*
rbind (cbind), 9
read_ePIRLS, *4*, *27*, 113
readBB_2001, 77, *82*
readBB_2003, 78
readBB_2012, 80, *83, 84*, *86*
readBPS_1994, 81
readBPS_2001, 82
readBPS_2009, 84
readBPS_2014, 85
readBTLS, 86
readCivEDICCS, *4*, *17*, 87

readECLS_K1998, *18*, 89, *92*
readECLS_K2011, *4*, *18*, *78, 79*, *81*, *87*, *90*, 91, *93*, *95, 96*, *98*
readELS, *19*, 92
readHSB_Senior, 94, *103*
readHSB_Sophomore, 95, *103*
readHSLS, *19*, 97
readICILS, *4*, *20*, 98
readNAEP, *4*, *35*, *78, 79*, *81–84*, *86, 87*, *89, 90*, *92, 93*, *95, 96*, *98*, *99*, 100, *105*, *111*, *113, 114*
readNLS72, 102
readPIAAC, *4*, 103
readPIRLS, *4*, *22*, 104
readPISA, *4*, *24*, 106
readTALIS, *4*, *24*, 108
readTIMSS, *4*, *25*, *89*, *99*, *105*, 109, *113, 114*
readTIMSSAdv, *4*, *26*, 111
rebindAttributes, *31, 32*, *48, 49*, 115
recode.sdf, *12*, 116
rename.sdf, 117
rq *(*rq.sdf*)*, 118
rq.sdf, *4*, 118

SD, 121
searchSDF, 123
setAttributes *(*edsurvey.data.frame*)*, 28
showCodebook, 124
showCutPoints, 125
showPlausibleValues, *5*, *35*, *51*, *59*, *65*, *118*, 126, *132*
showWeights, *5*, *35*, 127
subset, 127
subset.edsurvey.data.frame, *49*
summary2, *4*, 130

updatePlausibleValue, *51*, 132

varEstToCov, *37*, *44*, *55*, *62*, *70*, *74*, *122*, 133

waldTest, *55*, *62*, 134