Using EdSurvey to Analyze NCES Data: An Illustration of Analyzing NAEP Primer

Developed by Michael Lee, Paul Bailey, Ahmad Emad, Ting Zhang, Trang Nguyen, and Jiao Yu*†

February 21, 2020

Overview of the EdSurvey Package

National Assessment of Educational Progress (NAEP) datasets from the National Center for Education Statistics (NCES) require special statistical methods to analyze. Because of their scope and complexity, the `EdSurvey` package gives users functions to perform analyses that account for both complex sample survey designs and the use of plausible values.

The `EdSurvey` package also seamlessly takes advantage of the `LaF` package to read in data only when required for an analysis. Users with computers that have insufficient memory to read in entire NAEP datasets can still do analyses without having to write special code to read in just the appropriate variables. This situation is addressed directly in the `EdSurvey` package—behind the scenes and without any special tuning by the user.

Vignette Outline

This vignette will describe the basics of using the `EdSurvey` package for analyzing NAEP data as follows.

- Notes
  - Additional resources
  - Vignette notation
  - Software requirements
- Setting up the environment for analyzing NCES data
  - Installing and loading `EdSurvey`
  - Philosophy of Conducting Analyses Using the `EdSurvey` Package
  - Downloading data
  - Reading in data
  - Getting to know the data format
  - Removing special values
- Explore Variable Distributions with `summary2`
- Subsetting the data
- Retrieving data for further manipulation with `getData`

*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.

†The authors would like to thank Young Yee Kim, Yuqi Liao, Dan Sherman, and Qingshu Xie for reviewing this document, as well as Jiayi Li and Fei Liu for conducting quality control tests to verify the functions in this document.

    - Retrieving all variables in a dataset
    - Applying `rebindAttributes` to use `EdSurvey` functions with manipulated data frames
- Correlating variables with `cor.sdf`
    - Weighted correlations
    - Unweighted correlations
- Making a table with `edsurveyTable`
- Computing the percentages of students with `achievementLevels`
- Calculating percentiles with `percentile`
- Preparing an `edsurvey.data.frame.list`
    - Recoding variable names and levels using `recode.sdf` and `rename.sdf`
    - Combining several `edsurvey.data.frame` objects into a single object
    - Recommended workflow for standardizing variables in trend analyses
- Estimating the difference in two statistics with `gap`
    - Performing gap analysis and understanding the summary output
    - Gap analysis of achievement levels and percentiles
    - Gap analysis of jurisdictions
- Regression analysis with `lm.sdf`
- Multivariate regression with `mvrlm.sdf`
- Logistic regression analysis with `glm.sdf`, `logit.sdf`, and `probit.sdf`
    - oddsRatio
    - waldTest
- Quantile regression analysis with `rq.sdf`
- Mixed models with `mixed.sdf`
- Endnotes
    - Memory usage
    - Factors and factor analysis
    - Summary and next steps
    - Additional resources
    - Methodology resources
    - Reference

Vignette Notation

This vignette displays examples using notation for R console input and output. R console input will be displayed within a gray box:

```
inputCode <- c(2,"neat")
```

R console output will be displayed next to a double hash mark (`##`). Here is an example where the user types `inputCode` into the console and the code output R gives after the double hash marks:

```
inputCode
```

```
## [1] "2"    "neat"
```

Software Requirements

Unless you already have R version 3.3.0 or later, install the latest R version—which is available online at https://cran.r-project.org/. Users also may want to install RStudio desktop, which has an interface that many find easier to follow. RStudio is available online at https://www.rstudio.com/products/rstudio/download/.

Setting Up the Environment for Analyzing NCES Data

Installing and Loading EdSurvey

Inside R, run the following command to install `EdSurvey` as well as its package dependencies:

```
install.packages("EdSurvey")
```

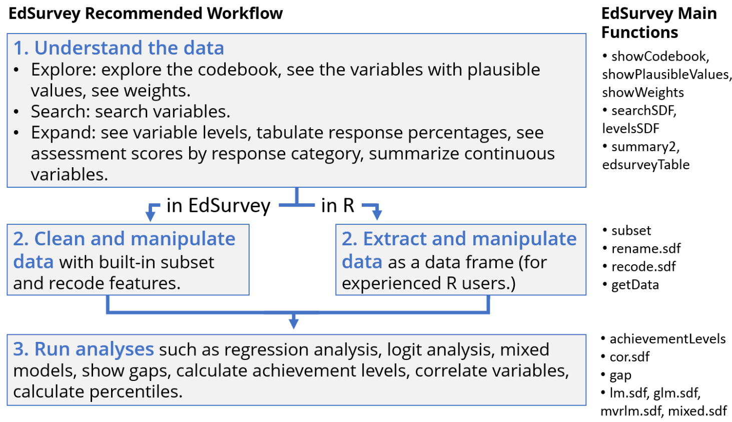
Once the package is successfully installed, `EdSurvey` can be loaded with the following command:

```
library(EdSurvey)
```

```
## Loading required package: car

## Loading required package: carData

## Loading required package: lfactors

## lfactors v1.0.4

## Registered S3 methods overwritten by 'lme4':
##   method                          from
##   cooks.distance.influence.merMod car
##   influence.merMod                car
##   dfbeta.influence.merMod         car
##   dfbetas.influence.merMod        car

## EdSurvey v2.4.0

##
## Attaching package: 'EdSurvey'

## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

Philosophy of Conducting Analyses Using the EdSurvey Package

Recognizing that researchers using R statistical software come with varying levels of experience, the `EdSurvey` package has provided multiple workflows to aid in this process of conducting survey analysis. The following graphic details the two recommended workflows:



The workflow has three sections:

1. Understanding the data
2. Preparing the data for analysis
3. Running the analysis

The phase in which the two methods diverge is the second section. The `EdSurvey` package provides functions for users to clean and manipulate their data, but experienced R programmers might prefer to extract and manipulate their data using other R methods or supplementary packages to do so; each method is supported for performing `EdSurvey` analytical functions.

Downloading Data

Although the bulk of this vignette will focus on NAEP data, `EdSurvey` includes a family of download and read functions for international studies, including the following:

- TIMSS: Trends in International Mathematics and Science Study and TIMSS Advanced (`downloadTIMSS`, `downloadTIMSSAdv`)
- PIRLS: Progress in International Reading Literacy Study (`downloadPIRLS`)
- ePIRLS: Electronic Progress in International Reading Literacy Study (`download_ePIRLS`)
- CIVED: The Civic Education Study 1999 and International Civic and Citizenship Study (`downloadCivEDICCS`)
- ICILS: International Computer and Information Literacy Study (`downloadICILS`)
- PISA: The Programme for International Student Assessment (`downloadPISA`)
- PIAAC: Programme for the International Assessment of Adult Competencies (`downloadPIAAC`)
- TALIS: Teaching and Learning International Survey (`downloadTALIS`)
- ECLS: Early Childhood Longitudinal Study (`downloadECLS_K`)
- HSLS: High School Longitudinal Study (`downloadHSLS`)
- ELS: Education Longitudinal Study (`downloadELS`)

For example, the `downloadTIMSS` function will download TIMSS data to a directory that the user specifies; for example, `"C:/Data"`. One also can manually download desirable survey data from their respective websites.

```
downloadTIMSS(years = 2015, root = "C:/", cache=FALSE)
```

For restricted datasets such as NAEP, please follow their restricted use instructions to save the whole intact data folder to a directory and read the data from there.

Reading in Data

Once the data have been prepared for your system, the read family of functions will open a connection to the specified data file to conduct your analysis. The read functions are as follows:

- TIMSS: Trends in International Mathematics and Science Study and TIMSS Advanced (`readTIMSS`, `readTIMSSAdv`)
- PIRLS: Progress in International Reading Literacy Study (`readPIRLS`)
- ePIRLS: Electronic Progress in International Reading Literacy Study (`read_ePIRLS`)
- CIVED: The Civic Education Study 1999 and International Civic and Citizenship Study (`readCivEDICCS`)
- ICILS: International Computer and Information Literacy Study (`readICILS`)
- PISA: The Programme for International Student Assessment (`readPISA`)
- PIAAC: Programme for the International Assessment of Adult Competencies (`readPIAAC`)
- TALIS: Teaching and Learning International Survey (`readTALIS`)
- ECLS: Early Childhood Longitudinal Study (`readECLS_K2011` and `readECLS_K1998`)
- HSLS: High School Longitudinal Study (`readHSLS`)
- ELS: Education Longitudinal Study (`readELS`)

For example, 2015 TIMSS data would be accessed by the `readTIMSS` function, selecting a data `path`, vector of `countries`, and `gradeLvl` of interest:

```
TIMSS15 <- readTIMSS(path = "C:/TIMSS2015/"), countries = c("usa"), gradeLvl = "4")
```

Each read function is unique given the differences across survey designs, but the functions typically follow a standard convention across functions for ease of use. To learn more about a particular read function, use `help(package = "EdSurvey")` to find the survey of interest and refer to its help documentation for guidance.

For NAEP, this is done using `EdSurvey`'s `readNAEP` function.

Vignette Sample NCES Dataset To follow along with this vignette, load the NAEP Primer dataset `M36NT2PM` and assign it the name `sdf` with this call:

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

Note that this command uses a somewhat unusual way of identifying a file path (the `system.file` function). Because the Primer data are bundled with the NAEPprimer package, the `system.file` function finds it regardless of where the package was installed on a machine. All other datasets are referred to by their system path.

NCES Dataset To load a unique NCES dataset for analysis, select the pathway to the DAT file in the NAEP assessment folder, which needs to be in the NCES standard folder directory titled `/Data`:

```
sdf2 <- readNAEP(filepath = '//.../Data/file.dat')
```

Note that the function recognizes the naming convention used by NCES for NAEP file names to determine which sample design and assessment information are attached to the resulting `edsurvey.data.frame`. The `readNAEP` function transparently accesses the necessary sample information and silently attaches it to the data.[1]

[1] The `EdSurvey` package uses the `.fr2` file in the `/Select/Parms` folder to assign this information to the `edsurvey.data.frame`.

It is possible that file pathways using special characters in your local directory could cause problems with reading data into R. Commonly used characters that require escapes include single quotation marks ('), double quotation marks ("), and backslashes (\). The most general solution to resolving these issues is adding an escape (i.e., the backslash key: \) before each character. For example, add an escape before the single quote used in `Nat'l`, as well as before each backslash as copied from a hypothetical windows file directory:

```
# original
"C:\2015 Nat'l Assessment Data\Data\file.dat"

# updated with escapes:
sdf2 <- readNAEP(filepath = "C:\\2015 Nat\'l Assessment Data\\Data\\file.dat")
```

An alternative option would involve using the `file.choose()` function to select the data file via a search window. The function opens your system's default file explorer to select a particular file. This file can be saved to an object, in this example `chosenFile`, which then can be read using `readNAEP`:

```
chosenFile <- file.choose()
sdf2 <- readNAEP(filepath = chosenFile)
```

Once read in, both student and school data from an NCES dataset can be analyzed and merged after loading the data into the R working environment. The `readNAEP` function is built to connect with the student data file, but it silently holds file formatting for the school dataset when read. More details on retrieving school variables for analysis will be outlined later in this vignette with the `getData` function.

Getting to Know the Data Format

Information about an `edsurvey.data.frame` can be obtained in multiple ways. To get general data information, simply call print by typing the name of the `data.frame` object (i.e., `sdf`) in the console.

```
sdf
```

```
## edsurvey.data.frame for 2005 NAEP (Mathematics) in USA
## Dimensions: 17606 rows and 302 columns.
##
## There is 1 full sample weight in this edsurvey.data.frame:
##   'origwt' with 62 JK replicate weights (the default).
##
##
## There are 6 subject scale(s) or subscale(s) in this edsurvey.data.frame:
## 'num_oper' subject scale or subscale with 5 plausible values.
##
## 'measurement' subject scale or subscale with 5 plausible values.
##
## 'geometry' subject scale or subscale with 5 plausible values.
##
## 'data_anal_prob' subject scale or subscale with 5 plausible values.
##
## 'algebra' subject scale or subscale with 5 plausible values.
##
## 'composite' subject scale or subscale with 5 plausible values (the
##   default).
```

```
##
##
## Omitted Levels: 'Multiple', 'NA', and 'Omitted'
##
## Default Conditions:
## tolower(rptsamp) == "reporting sample"
## Achievement Levels:
## Basic: 262
## Proficient: 299
## Advanced: 333
```

Some basic functions that work on a `data.frame`, such as `dim`, `nrow`, and `ncol`, also work on an `edsurvey.data.frame`.[2] They help check the dimensions of `sdf`.

```r
dim(x = sdf)
```

```
## [1] 17606   302
```

```r
nrow(x = sdf)
```

```
## [1] 17606
```

```r
ncol(x = sdf)
```

```
## [1] 302
```

The `colnames` function can be used to list all variable names in the data:

```r
colnames(x = sdf)
```

```
##   [1] "year"    "cohort"  "scrpsu"  "dsex"    "iep"     "lep"     "ell3"    "sdracem"
##   [9] "pared"   "b003501" "b003601" "b013801" "b017001" "b017101" "b018101" "b018201"
##  [17] "b017451" "m815401" "m815501" "m815601" "m815801" "m815701" "rptsamp" "repgrp1"
##  [25] "repgrp2" "jkunit"  "origwt"  "srwt01"  "srwt02"  "srwt03"  "srwt04"  "srwt05"
##  [33] "srwt06"  "srwt07"  "srwt08"  "srwt09"  "srwt10"  "srwt11"  "srwt12"  "srwt13"
##  [41] "srwt14"  "srwt15"  "srwt16"  "srwt17"  "srwt18"  "srwt19"  "srwt20"  "srwt21"
##  [49] "srwt22"  "srwt23"  "srwt24"  "srwt25"  "srwt26"  "srwt27"  "srwt28"  "srwt29"
##  [57] "srwt30"  "srwt31"  "srwt32"  "srwt33"  "srwt34"  "srwt35"  "srwt36"  "srwt37"
##  [65] "srwt38"  "srwt39"  "srwt40"  "srwt41"  "srwt42"  "srwt43"  "srwt44"  "srwt45"
##  [73] "srwt46"  "srwt47"  "srwt48"  "srwt49"  "srwt50"  "srwt51"  "srwt52"  "srwt53"
##  [81] "srwt54"  "srwt55"  "srwt56"  "srwt57"  "srwt58"  "srwt59"  "srwt60"  "srwt61"
##  [89] "srwt62"  "smsrswt" "mrps11"  "mrps12"  "mrps13"  "mrps14"  "mrps15"  "mrps21"
##  [97] "mrps22"  "mrps23"  "mrps24"  "mrps25"  "mrps31"  "mrps32"  "mrps33"  "mrps34"
## [105] "mrps35"  "mrps41"  "mrps42"  "mrps43"  "mrps44"  "mrps45"  "mrps51"  "mrps52"
## [113] "mrps53"  "mrps54"  "mrps55"  "mrpcm1"  "mrpcm2"  "mrpcm3"  "mrpcm4"  "mrpcm5"
## [121] "m075201" "m075401" "m075601" "m019901" "m066201" "m047301" "m046201" "m066401"
## [129] "m020101" "m067401" "m086101" "m047701" "m067301" "m048001" "m093701" "m086001"
## [137] "m051901" "m076001" "m046001" "m046101" "m067701" "m046701" "m046901" "m047201"
## [145] "m046601" "m046801" "m067801" "m066601" "m067201" "m068003" "m068005" "m068008"
## [153] "m068007" "m068006" "m093601" "m053001" "m047801" "m086301" "m085701" "m085901"
```

[2] Use `?function` in the R console to view documentation on base R and `EdSurvey` package functions (e.g., `?gsub` or `?lm.sdf`).

```
## [161] "m085601" "m085501" "m085801" "m019701" "m020001" "m046301" "m047001" "m046501"
## [169] "m066501" "m047101" "m066301" "m067901" "m019601" "m051501" "m047901" "m053101"
## [177] "m143601" "m143701" "m143801" "m143901" "m144001" "m144101" "m144201" "m144301"
## [185] "m144401" "m144501" "m144601" "m144701" "m144801" "m144901" "m145001" "m145101"
## [193] "m013431" "m0757cl" "m013131" "m091701" "m072801" "m091501" "m091601" "m073501"
## [201] "m052401" "m075301" "m072901" "m013631" "m075801" "m013731" "m013531" "m051801"
## [209] "m093401" "m093801" "m142001" "m142101" "m142201" "m142301" "m142401" "m142501"
## [217] "m142601" "m142701" "m142801" "m142901" "m143001" "m143101" "m143201" "m143301"
## [225] "m143401" "m143501" "m105601" "m105801" "m105901" "m106001" "m106101" "m106201"
## [233] "m106301" "m106401" "m106501" "m106601" "m106701" "m106801" "m106901" "m107001"
## [241] "m107101" "m107201" "m107401" "m107501" "m107601" "m109801" "m110001" "m110101"
## [249] "m110201" "m110301" "m110401" "m110501" "m110601" "m110701" "m110801" "m110901"
## [257] "m111001" "m111201" "m111301" "m111401" "m111501" "m111601" "m111801" "yrsexp"
## [265] "yrsmath" "t089401" "t088001" "t090801" "t090802" "t090803" "t090804" "t090805"
## [273] "t090806" "t087501" "t088301" "t088401" "t088501" "t088602" "t088603" "t088801"
## [281] "t088803" "t088804" "t088805" "t091502" "t091503" "t091504" "c052801" "c052802"
## [289] "c052804" "c052805" "c052806" "c052807" "c052808" "c052701" "c046501" "c044006"
## [297] "c044007" "c052901" "c053001" "c053101" "sscrpsu" "c052601"
```

To conduct a more powerful search of NAEP data variables, use the `searchSDF` function, which returns variable names and labels from an `edsurvey.data.frame` based on a character string. The user can specify which data source (either "student" or "school") the user would like to search. For example, the following call to `searchSDF` searches for the character string `"book"` in the `edsurvey.data.frame` and specifies the `fileFormat` to search the student data file:

```r
searchSDF(string = "book", data = sdf, fileFormat = "student")
```

```
##   variableName                                      Labels
## 1      b013801                               Books in home
## 2      t088804 Computer activities: Use a gradebook program
## 3      t091503     G8Math:How often use Geometry sketchbook
```

The levels and labels for each variable search via `searchSDF()` also can be returned by setting `levels = TRUE`:

```r
searchSDF(string = "book", data = sdf, fileFormat = "student", levels = TRUE)
```

```
## Variable: b013801
## Label: Books in home
## Levels (Lowest level first):
##      1. 0-10
##      2. 11-25
##      3. 26-100
##      4. >100
##      8. Omitted
##      0. Multiple
## Variable: t088804
## Label: Computer activities: Use a gradebook program
## Levels (Lowest level first):
##      1. Never or hardly ever
##      2. Once or twice/month
##      3. Once or twice a week
```

```
##      4. Almost every day
##      8. Omitted
##      0. Multiple
## Variable: t091503
## Label: G8Math:How often use Geometry sketchbook
## Levels (Lowest level first):
##      1. Never or hardly ever
##      2. Once or twice/month
##      3. Once or twice a week
##      4. Almost every day
##      8. Omitted
##      0. Multiple
```

The | (OR) operator can be used to search several strings simultaneously:

```
searchSDF(string="book|home|value", data=sdf)
```

```
##    variableName                                            Labels
## 1       b013801                                     Books in home
## 2       b017001                                 Newspaper in home
## 3       b017101                                  Computer at home
## 4       b018201        Language other than English spoken in home
## 5       b017451                         Talk about studies at home
## 6       m086101                              Read value from graph
## 7       m020001 Apply place value                              (R1)
## 8       m143601                       Solve for x given value of n
## 9       m142301                                Identify place value
## 10      t088804       Computer activities: Use a gradebook program
## 11      t088805  Computer activities: Post homework,schedule info
## 12      t091503           G8Math:How often use Geometry sketchbook
```

A vector of strings is used to search for variables that contain multiple strings, such as both "book" and "home"; each string is present in the variable label and can be used to filter the results:

```
searchSDF(string=c("book","home"), data=sdf)
```

```
##   variableName        Labels
## 1      b013801 Books in home
```

To return the levels and labels for a particular variable, use `levelsSDF()`:

```
levelsSDF(varnames = "b017451", data = sdf)
```

```
## Levels for Variable 'b017451' (Lowest level first):
##     1. Never or hardly ever (n=3837)
##     2. Once every few weeks (n=3147)
##     3. About once a week (n=2853)
##     4. 2 or 3 times a week (n=3362)
##     5. Every day (n=3132)
##     8. Omitted* (n=575)
##     0. Multiple* (n=9)
##     NOTE: * indicates an omitted level.
```

Access a full codebook using `showCodebook()`, retrieving the variable names, variable labels, and value labels of a survey. This function pairs well with the `View()` function to more easily explore a dataset:

```
View(showCodebook(sdf))
```

Basic information about plausible values and weights in an `edsurvey.data.frame` can be seen in the `print` function. The variables associated with plausible values and weights can be seen from the `showPlausibleValues` and `showWeights` functions, respectively, when the `verbose` argument is set to `TRUE`:

```
showPlausibleValues(data = sdf, verbose = TRUE)
```

```
## There are 6 subject scale(s) or subscale(s) in this edsurvey.data.frame:
## 'num_oper' subject scale or subscale with 5 plausible values.
##   The plausible value variables are: 'mrps11', 'mrps12', 'mrps13',
##   'mrps14', and 'mrps15'
##
## 'measurement' subject scale or subscale with 5 plausible values.
##   The plausible value variables are: 'mrps21', 'mrps22', 'mrps23',
##   'mrps24', and 'mrps25'
##
## 'geometry' subject scale or subscale with 5 plausible values.
##   The plausible value variables are: 'mrps31', 'mrps32', 'mrps33',
##   'mrps34', and 'mrps35'
##
## 'data_anal_prob' subject scale or subscale with 5 plausible values.
##   The plausible value variables are: 'mrps41', 'mrps42', 'mrps43',
##   'mrps44', and 'mrps45'
##
## 'algebra' subject scale or subscale with 5 plausible values.
##   The plausible value variables are: 'mrps51', 'mrps52', 'mrps53',
##   'mrps54', and 'mrps55'
##
## 'composite' subject scale or subscale with 5 plausible values (the
##   default).
##   The plausible value variables are: 'mrpcm1', 'mrpcm2', 'mrpcm3',
##   'mrpcm4', and 'mrpcm5'
```

```
showWeights(data = sdf, verbose = TRUE)
```

```
## There is 1 full sample weight in this edsurvey.data.frame:
##   'origwt' with 62 JK replicate weights (the default).
##     Jackknife replicate weight variables associated with the full sample
##     weight 'origwt':
##     'srwt01', 'srwt02', 'srwt03', 'srwt04', 'srwt05', 'srwt06', 'srwt07',
##     'srwt08', 'srwt09', 'srwt10', 'srwt11', 'srwt12', 'srwt13', 'srwt14',
##     'srwt15', 'srwt16', 'srwt17', 'srwt18', 'srwt19', 'srwt20', 'srwt21',
##     'srwt22', 'srwt23', 'srwt24', 'srwt25', 'srwt26', 'srwt27', 'srwt28',
##     'srwt29', 'srwt30', 'srwt31', 'srwt32', 'srwt33', 'srwt34', 'srwt35',
##     'srwt36', 'srwt37', 'srwt38', 'srwt39', 'srwt40', 'srwt41', 'srwt42',
##     'srwt43', 'srwt44', 'srwt45', 'srwt46', 'srwt47', 'srwt48', 'srwt49',
##     'srwt50', 'srwt51', 'srwt52', 'srwt53', 'srwt54', 'srwt55', 'srwt56',
##     'srwt57', 'srwt58', 'srwt59', 'srwt60', 'srwt61', and 'srwt62'
```

The functions getStratumVar and getPSUVar return the default stratum variable name or a PSU variable associated with a weight variable.

```
EdSurvey:::getStratumVar(data = sdf)
```

```
## [1] "repgrp1"
```

```
EdSurvey:::getPSUVar(data = sdf)
```

```
## [1] "jkunit"
```

These are particularly useful for accessing the variables associated with the weights in longitudinal surveys.

Removing Special Values

The EdSurvey package uses listwise deletion to remove special values in all analyses by default. For example, in the NAEP Primer data, the omitted levels are returned when `print(sdf)` is called: `Omitted Levels: 'Multiple', 'NA', 'Omitted'`. By default, these levels are excluded via listwise deletion. To use a different method, such as pairwise deletion, set `defaultConditions = FALSE` when running your analysis.

Explore Variable Distributions With summary2

The summary2 function produces both weighted and unweighted descriptive statistics for a variable. This functionality is particularly useful for gathering response information for survey variables when conducting data exploration. For NAEP data and other datasets that have a default weight variable, summary2 produces weighted statistics by default. If the specified variable is a set of plausible values, and the weightVar option is non-NULL, summary2 statistics account for both plausible values pooling and weighting.

```
summary2(sdf, "composite")
```

```
## Estimates are weighted using weight variable 'origwt'
##    Variable     N Weighted N   Min.  1st Qu.   Median     Mean  3rd Qu.    Max.
## 1 composite 16915   16932.46 126.11 251.9623 277.4784 275.8892 301.1835 404.184
##        SD NA's Zero-weights
## 1 36.5713    0            0
```

By specifying `weightVar = NULL`, the function prints out unweighted descriptive statistics for the selected variable or plausible values:

```
summary2(sdf, "composite", weightVar = NULL)
```

```
## Estimates are not weighted.
##   Variable     N   Min.  1st Qu. Median     Mean  3rd Qu.   Max.       SD NA's
## 1   mrpcm1 16915 130.53 252.0600 277.33 275.8606 300.7200 410.80 35.89864    0
## 2   mrpcm2 16915 124.16 252.2100 277.33 275.6399 300.6900 408.58 36.08483    0
## 3   mrpcm3 16915 115.09 252.0017 277.19 275.6570 300.5600 398.17 36.09278    0
## 4   mrpcm4 16915 137.19 252.4717 277.44 275.7451 300.5767 407.41 35.91078    0
## 5   mrpcm5 16915 123.58 252.4900 277.16 275.6965 300.5000 395.96 36.10905    0
```

For a categorical variable, the `summary2` function returns the weighted number of cases, the weighted percent, and the weighted standard error. For example, the variable `b017451` (frequency of students talking about studies at home) returns the following output:

```
summary2(sdf, "b017451")
```

```
## Estimates are weighted using weight variable 'origwt'
##                b017451    N Weighted N Weighted Percent Weighted Percent SE
## 1 Never or hardly ever 3837  3952.4529      23.34245648           0.4318975
## 2 Once every few weeks 3147  3190.8945      18.84483329           0.3740648
## 3    About once a week 2853  2937.7148      17.34960077           0.3414566
## 4  2 or 3 times a week 3362  3425.8950      20.23270282           0.3156289
## 5            Every day 3132  3223.8074      19.03921080           0.4442216
## 6              Omitted  575   194.3312       1.14768416           0.1272462
## 7             Multiple    9     7.3676       0.04351168           0.0191187
```

Note that by default, the `summary2` function includes omitted levels; to remove those, set `omittedLevels = TRUE`:

```
summary2(sdf, "b017451", omittedLevels = TRUE)
```

```
## Estimates are weighted using weight variable 'origwt'
##                b017451    N Weighted N Weighted Percent Weighted Percent SE
## 1 Never or hardly ever 3837   3952.453         23.62386           0.4367548
## 2 Once every few weeks 3147   3190.894         19.07202           0.3749868
## 3    About once a week 2853   2937.715         17.55876           0.3486008
## 4  2 or 3 times a week 3362   3425.895         20.47662           0.3196719
## 5            Every day 3132   3223.807         19.26874           0.4467063
```

Subsetting the Data

A subset of a dataset can be used with `EdSurvey` package functions. In this example, a summary table is created with `edsurveyTable` after filtering the sample to include only those students whose value in the `dsex` variable is male and race (as variable `sdracem`) is either values 1 or 3 (White or Hispanic). Both value levels and labels can be used in `EdSurvey` package functions.

```
sdfm <- subset(sdf, dsex == "Male" & (sdracem == 3 | sdracem == 1))
es2 <- edsurveyTable(formula = composite ~ dsex + sdracem, data = sdfm)
```

```
es2
```

Table 1: es2

dsex	sdracem	N	WTD_N	PCT	SE(PCT)	MEAN	SE(MEAN)
Male	White	5160	5035.169	76.11329	1.625174	287.6603	0.8995013
Male	Hispanic	1244	1580.192	23.88671	1.625174	260.8268	1.5822251

Retrieving Data for Further Manipulation With `getData`

Data can be extracted and manipulated using the function `getData`. The function `getData` takes an `edsurvey.data.frame` and returns a `light.edsurvey.data.frame` containing the requested variables by either specifying a set of variable names in `varnames` or entering a formula in `formula`.[3]

To access and manipulate data for `dsex` and `b017451` variables in `sdf`, call `getData`. In the following code, the `head` function is used, which reveals only the first few rows of the resulting data:

```
gddat <- getData(data = sdf, varnames = c("dsex","b017451"),
                 omittedLevels = TRUE)
head(gddat)
```

```
##     dsex              b017451
## 1   Male            Every day
## 2 Female    About once a week
## 3 Female            Every day
## 4   Male            Every day
## 6 Female Once every few weeks
## 7   Male  2 or 3 times a week
```

By default, setting `omittedLevels` to `TRUE` removes special values such as multiple entries or `NA`s. `getData` tries to help by dropping the levels of factors for regression, tables, and correlations that are not typically included in analysis.

Retrieving All Variables in a Dataset

To extract all data in an `edsurvey.data.frame`, define the `varnames` argument as `colnames(sdf)`, which will query all variables. Setting the arguments `omittedLevels` and `defaultConditions` to `FALSE` ensures that values that would normally be removed are included:

```
lsdf0 <- getData(data = sdf, varnames = colnames(sdf), addAttributes = TRUE,
                 omittedLevels = FALSE, defaultConditions = FALSE)
dim(lsdf0) # excludes the one school variable in the sdf
dim(sdf)
```

Once retrieved, this dataset can be used with all `EdSurvey` functions.

Applying `rebindAttributes` to Use `EdSurvey` Functions With Manipulated Data Frames

A helper function that pairs well with `getData` is `rebindAttributes`. This function allows users to reassign the attributes from a survey dataset to a data frame that might have had its attributes stripped during the manipulation process. Once attributes have been rebinded, all variables—including those outside the original dataset—are available for use in `EdSurvey` analytical functions.

For example, a user might want to run a linear model using `composite`, the default weight `origwt`, the variable `dsex`, and the categorical variable `b017451` recoded into a binary variable. To do so, we can return a portion of the `sdf` survey data as the `gddat` object. Next, use the `base` R function `ifelse` to conditionally recode the variable `b017451` by collapsing the levels `"Never or hardly ever"` and `"Once every few weeks"` into one level (`"Rarely"`) and all other levels into `"At least once a week"`.

[3] Use `?getData` for details on default `getData` arguments.

```r
gddat <- getData(data = sdf, varnames = c("dsex", "b017451", "origwt", "composite"),
                 omittedLevels = TRUE)
gddat$studyTalk <- ifelse(gddat$b017451 %in% c("Never or hardly ever",
                                               "Once every few weeks"),
                          "Rarely", "At least once a week")
```

From there, apply `rebindAttributes` from the attribute data `sdf` to the manipulated data frame `gddat`. The new variables are now available for use in `EdSurvey` analytical functions:

```r
gddat <- rebindAttributes(gddat, sdf)
lm2 <- lm.sdf(formula = composite ~ dsex + studyTalk, data = gddat)
summary(lm2)
```

```
##
## Formula: composite ~ dsex + studyTalk
##
## Weight variable: 'origwt'
## Variance method: jackknife
## JK replicates: 62
## Plausible values: 5
## jrrIMax: 1
## full data n: 17606
## n used: 16331
##
## Coefficients:
##                      coef        se        t    dof  Pr(>|t|)
## (Intercept)     281.69030   0.96690 291.3349 39.915 < 2.2e-16 ***
## dsexFemale       -2.89797   0.59549  -4.8665 52.433 1.081e-05 ***
## studyTalkRarely  -9.41418   0.79620 -11.8239 53.205 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared: 0.0168
```

Additional details on the features of the `getData` function appear in the vignette titled *Using the* ***`getData`*** *Function in EdSurvey*.

Correlating Variables With `cor.sdf`

The `EdSurvey` package features multiple correlation methods for data exploration and analysis that fully account for the complex sample design in NCES data by using the `cor.sdf` function.[4] These features include the following correlation procedures:

- Pearson product-moment correlations for continuous variables
- Spearman rank correlation for ranked variables
- Polyserial correlations for one categorical and one continuous variable
- Polychoric correlations for two categorical variables
- Correlations among plausible values of the subject scales and subscales (marginal correlation coefficients, which uses Pearson type)

[4] Use `?cor.sdf` for details on default `cor.sdf` arguments.

Weighted Correlations

In the following example, b013801, t088001, and the full sample weight origwt are read in to calculate the correlation using the Pearson method. Similar to other EdSurvey functions, the data are removed automatically from memory after the correlation is run.

```
cor_pearson <- cor.sdf(x = "b013801", y = "t088001", data = sdf,
                    method = "Pearson", weightVar = "origwt")
```

It is important to note the order of levels to ensure that the correlations are functioning as intended. Printing a correlation object will provide a condensed summary of the correlation details and the order of levels for each variable:

```
cor_pearson
```

```
## Method: Pearson
## full data n: 17606
## n used: 14492
##
## Correlation: -0.07269657
##
## Correlation Levels:
##   Levels for Variable 'b013801' (Lowest level first):
##     1. 0-10
##     2. 11-25
##     3. 26-100
##     4. >100
##   Levels for Variable 't088001' (Lowest level first):
##     1. Less than 3 hours
##     2. 3-4.9 hours
##     3. 5-6.9 hours
##     4. 7 hours or more
```

Variables in cor.sdf can be recoded and reordered. Variable levels and values can be redefined given the desired specifications. For example, b017451 and t088001 are correlated using the Pearson method, with the levels "2 or 3 times a week" and "Every day" of the variable b017451 being recoded to "Frequently" within a list of lists in the recode argument:

```
cor_recode <- cor.sdf(x = "b017451", y = "t088001", data = sdf,
                      method = "Pearson", weightVar = "origwt",
                      recode = list(b017451 = list(from = c("2 or 3 times a week", "Every day"),
                                                   to = c("Frequently"))))
cor_recode
```

```
## Method: Pearson
## full data n: 17606
## n used: 14468
##
## Correlation: -0.01949923
##
## Correlation Levels:
##   Levels for Variable 'b017451' (Lowest level first):
```

```
##     1. Never or hardly ever
##     2. Once every few weeks
##     3. About once a week
##     4. Frequently
##   Levels for Variable 't088001' (Lowest level first):
##     1. Less than 3 hours
##     2. 3-4.9 hours
##     3. 5-6.9 hours
##     4. 7 hours or more
```

Recoding can be useful when a level is very thinly populated (so it might merit combination with another level) or when changing the value label to something more appropriate for a particular analysis.

The variables `b017451` and `t088001` are correlated using the Pearson method in the following example, with the variable `t088001`'s values `"Less than 3 hours"`, `"3-4.9 hours"`, `"5-6.9 hours"`, `"7 hours or more"` being reordered to `"7 hours or more"`, `"5-6.9 hours"`, `"3-4.9 hours"`, `"Less than 3 hours"` within a list:

```
cor_reorder <- cor.sdf(x = "b017451", y = "t088001", data = sdf,
                       method = "Pearson", weightVar = "origwt",
                       reorder = list(t088001 = c("7 hours or more", "5-6.9 hours",
                                                  "3-4.9 hours", "Less than 3 hours")))
cor_reorder
```

```
## Method: Pearson
## full data n: 17606
## n used: 14468
##
## Correlation: 0.02048827
##
## Correlation Levels:
##   Levels for Variable 'b017451' (Lowest level first):
##     1. Never or hardly ever
##     2. Once every few weeks
##     3. About once a week
##     4. 2 or 3 times a week
##     5. Every day
##   Levels for Variable 't088001' (Lowest level first):
##     1. 7 hours or more
##     2. 5-6.9 hours
##     3. 3-4.9 hours
##     4. Less than 3 hours
```

Changing the order of the levels might be useful to modify a variable that is out of order or when reversing the orientation of a series. The `reorder` argument also is suitable when implemented in conjunction with recoded levels.

NOTE: As an alternative, recoding can be completed within `getData`. To see additional examples of recoding and reordering, use `?cor.sdf` in the R console.

The marginal correlation coefficient among plausible values of the subject scales and subscales can be calculated using the `cor.sdf` function and the Pearson method. The subject subscales `num_oper` and `algebra` are correlated in this example:

```r
cor3_mcc <- cor.sdf(x = "num_oper", y = "algebra", data = sdf, method = "Pearson")
cor3_mcc
```

```
## Method: Pearson
## full data n: 17606
## n used: 16915
##
## Correlation: 0.8924728
```

Use the `showPlausibleValues` function to return the plausible values of an `edsurvey.data.frame` for use in calculating the correlation coefficients between subject scales or subscales.

The `cor.sdf` function features multiple methods for data exploration and analysis using correlations. The following example shows the differences in correlation coefficients among the Pearson, Spearman, and polychoric methods using a `subset` of the `edsurvey.data.frame` data where `dsex == 1` (saved as the `sdf_dnf` object), `b017451`, `pared`, and the full sample weight `origwt`:

```r
sdf_dnf <- subset(sdf, dsex == 1)
cor_pearson <- cor.sdf(x = "b017451", y = "pared", data = sdf_dnf,
                    method = "Pearson", weightVar = "origwt")
cor_spearman <- cor.sdf(x = "b017451", y = "pared", data = sdf_dnf,
                    method = "Spearman", weightVar = "origwt")
cor_polychoric <- cor.sdf(x = "b017451", y = "pared", data = sdf_dnf,
                    method = "Polychoric", weightVar = "origwt")
```

```r
cbind(Correlation = c(Pearson = cor_pearson$correlation,
                    Spearman = cor_spearman$correlation,
                    Polychoric = cor_polychoric$correlation))
```

```
##            Correlation
## Pearson     0.08027069
## Spearman    0.06655288
## Polychoric  0.06972564
```

Plausible values for subject scales and subscales also can be correlated with variables using the `cor.sdf` function. In this case, the five plausible values for `composite`, the variable `b017451`, and the full sample weight `origwt` are read in to calculate the correlation coefficients using the Pearson, Spearman, and polyserial methods:

```r
cor_pearson2 <- cor.sdf(x = "composite", y = "b017451", data = sdf_dnf,
                    method = "Pearson", weightVar = "origwt")
cor_spearman2 <- cor.sdf(x = "composite", y = "b017451", data = sdf_dnf,
                    method = "Spearman", weightVar = "origwt")
cor_polyserial2 <- cor.sdf(x = "composite", y = "b017451", data = sdf_dnf,
                    method = "Polyserial", weightVar = "origwt")
```

```r
cbind(Correlation = c(Pearson = cor_pearson2$correlation,
                      Spearman = cor_spearman2$correlation,
                      Polyserial = cor_polyserial2$correlation))
```

```
##            Correlation
## Pearson      0.1031247
## Spearman     0.1148983
## Polyserial   0.1044407
```

Unweighted Correlations

The `cor.sdf` function also features the ability to perform correlations without accounting for weights. The `cor.sdf` function automatically accounts for the default sample weights of the NCES dataset read for analysis in `weightVar = "default"` but can be modified by setting `weightVar=NULL`. The following example shows the correlation coefficients of the Pearson and Spearman methods of the variables `pared` and `b017451` while excluding weights:

```
cor_pearson_unweighted <- cor.sdf(x = "b017451", y = "pared", data = sdf,
                                  method = "Pearson", weightVar = NULL)
cor_pearson_unweighted
```

```
## Method: Pearson
## full data n: 17606
## n used: 16278
##
## Correlation: 0.05316366
##
## Correlation Levels:
##   Levels for Variable 'b017451' (Lowest level first):
##     1. Never or hardly ever
##     2. Once every few weeks
##     3. About once a week
##     4. 2 or 3 times a week
##     5. Every day
##   Levels for Variable 'pared' (Lowest level first):
##     1. Did not finish H.S.
##     2. Graduated H.S.
##     3. Some ed after H.S.
##     4. Graduated college
##     5. I Don't Know
```

```
cor_spearman_unweighted <- cor.sdf(x = "b017451", y = "pared", data = sdf,
                                   method = "Spearman", weightVar = NULL)
cor_spearman_unweighted
```

```
## Method: Spearman
## full data n: 17606
## n used: 16278
##
## Correlation: 0.04283483
##
## Correlation Levels:
##   Levels for Variable 'b017451' (Lowest level first):
##     1. Never or hardly ever
##     2. Once every few weeks
##     3. About once a week
##     4. 2 or 3 times a week
##     5. Every day
##   Levels for Variable 'pared' (Lowest level first):
##     1. Did not finish H.S.
##     2. Graduated H.S.
##     3. Some ed after H.S.
```

```
##      4. Graduated college
##      5. I Don't Know
```

Making a Table with `edsurveyTable`

Summary tables can be created in the `EdSurvey` package using the `edsurveyTable` function. A call to `edsurveyTable`[5] with two variables, `dsex` and `b017451`, creates a table that shows the number, percentage, and NAEP mathematics performance scale scores of eighth-grade students by gender and frequency of talk about studies at home. Percentages add up to 100 within each gender.

```
es1 <- edsurveyTable(formula = composite ~ dsex + b017451, data = sdf,
                     jrrIMax = 1, varMethod = "jackknife")
```

This `edsurveyTable` is saved as the object `es1`, and the resulting table can be displayed by printing

```
es1$data
```

Table 2: es1

dsex	b017451	N	WTD_N	PCT	SE(PCT)	MEAN	SE(MEAN)
Male	Never or hardly ever	2350	2434.844	29.00978	0.6959418	270.8243	1.057078
Male	Once every few weeks	1603	1638.745	19.52472	0.5020657	275.0807	1.305922
Male	About once a week	1384	1423.312	16.95795	0.5057265	281.5612	1.409587
Male	2 or 3 times a week	1535	1563.393	18.62694	0.4811497	284.9066	1.546072
Male	Every day	1291	1332.890	15.88062	0.5872731	277.2597	1.795784
Female	Never or hardly ever	1487	1517.609	18.20203	0.5078805	266.7897	1.519020
Female	Once every few weeks	1544	1552.149	18.61630	0.4892491	271.2255	1.205528
Female	About once a week	1469	1514.403	18.16358	0.5782966	278.7502	1.719778
Female	2 or 3 times a week	1827	1862.502	22.33864	0.4844840	282.7765	1.404107
Female	Every day	1841	1890.918	22.67945	0.6553039	275.4628	1.219439

Note that we used the argument `jrrIMax` to indicate the maximum number of plausible values to be included when calculating sampling variance in the computation of the standard error of estimates, such as the following:

- Estimated scale scores
- Achievement levels
- Regression analysis of student performance 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 in the computation of the standard errors of estimates involving plausible values with the jackknife variance estimation method, as seen in the next example. The argument `jrrIMax` can be omitted to select the default. Higher values of `jrrIMax` leads to longer computing times but more accurate error estimates.[6] An alternative is to set `jrrIMax=Inf` to obtain the ideal estimation with the jackknife method.

The function also features variance estimation using the Taylor series method. By setting `varMethod = "Taylor"`, the same `edsurveyTable` call used in the previous example can return results using Taylor series variance estimation:

[5] Use `?edsurveyTable` for details on default `edsurveyTable` arguments.
[6] See the documentation for `lm.sdf` for details on the variance calculation.

```r
es1t <- edsurveyTable(formula = composite ~ dsex + b017451, data = sdf,
                      jrrIMax = 1, varMethod = "Taylor")
```

```r
es1t$data
```

Table 3: es1t

dsex	b017451	N	WTD_N	PCT	SE(PCT)	MEAN	SE(MEAN)
Male	Never or hardly ever	2350	2434.844	29.00978	0.6968466	270.8243	1.064411
Male	Once every few weeks	1603	1638.745	19.52472	0.5017827	275.0807	1.363576
Male	About once a week	1384	1423.312	16.95795	0.5060344	281.5612	1.417767
Male	2 or 3 times a week	1535	1563.393	18.62694	0.4810093	284.9066	1.513590
Male	Every day	1291	1332.890	15.88062	0.5866306	277.2597	1.789257
Female	Never or hardly ever	1487	1517.609	18.20203	0.5079071	266.7897	1.535320
Female	Once every few weeks	1544	1552.149	18.61630	0.4889362	271.2255	1.208797
Female	About once a week	1469	1514.403	18.16358	0.5787277	278.7502	1.739417
Female	2 or 3 times a week	1827	1862.502	22.33864	0.4846566	282.7765	1.386048
Female	Every day	1841	1890.918	22.67945	0.6554100	275.4628	1.242832

If the percentages do not add up to 100 at the desired level, an adjustment can be made in the `pctAggregationLevel` argument to change the aggregation level. By default, `pctAggregationLevel = 1`, indicating that the formula will be aggregated by each level the first variable in the call; in our previous example this is `dsex`. Setting `pctAggregationLevel = 0` aggregates by each level of each variable in the call:

```r
es2t <- edsurveyTable(formula = composite ~ dsex + b017451, data = sdf,
                      jrrIMax = 1, varMethod = "Taylor", pctAggregationLevel = 0)
```

```r
es2t$data
```

Table 4: es2t

dsex	b017451	N	WTD_N	PCT	SE(PCT)	MEAN	SE(MEAN)
Male	Never or hardly ever	2350	2434.844	14.553095	0.3742692	270.8243	1.064411
Male	Once every few weeks	1603	1638.745	9.794803	0.2649185	275.0807	1.363576
Male	About once a week	1384	1423.312	8.507154	0.2771855	281.5612	1.417767
Male	2 or 3 times a week	1535	1563.393	9.344421	0.2670878	284.9066	1.513590
Male	Every day	1291	1332.890	7.966700	0.2998687	277.2597	1.789257
Female	Never or hardly ever	1487	1517.609	9.070768	0.2986897	266.7897	1.535320
Female	Once every few weeks	1544	1552.149	9.277216	0.2498682	271.2255	1.208797
Female	About once a week	1469	1514.403	9.051606	0.2902747	278.7502	1.739417
Female	2 or 3 times a week	1827	1862.502	11.132198	0.2555007	282.7765	1.386048
Female	Every day	1841	1890.918	11.302039	0.3497829	275.4628	1.242832

The calculation of means and standard errors requires computation time that the user may not want to wait for. If you wish to simply see a table of the levels and the N sizes, you can set the `returnMeans` and `returnSepct` arguments to `FALSE` to omit those columns as follows:

```r
es1b <- edsurveyTable(formula = composite ~ dsex + b017451, data = sdf, jrrIMax = 1,
                      returnMeans = FALSE, returnSepct = FALSE)
```

In this `edsurveyTable`, the resulting table can be displayed by printing the object.

```r
es1b
```

Table 5: es1b

dsex	b017451	N	WTD_N	PCT
Male	Never or hardly ever	2350	2434.844	29.00978
Male	Once every few weeks	1603	1638.745	19.52472
Male	About once a week	1384	1423.312	16.95795
Male	2 or 3 times a week	1535	1563.393	18.62694
Male	Every day	1291	1332.890	15.88062
Female	Never or hardly ever	1487	1517.609	18.20203
Female	Once every few weeks	1544	1552.149	18.61630
Female	About once a week	1469	1514.403	18.16358
Female	2 or 3 times a week	1827	1862.502	22.33864
Female	Every day	1841	1890.918	22.67945

For more details on the arguments in the `edsurveyTable` function, look at the examples using

```r
?edsurveyTable
```

Computing the Percentages of Students With `achievementLevels`

The `achievementLevels` function[7] computes the percentages of students' achievement levels or benchmarks defined by an assessment including NAEP,International Association for the Evaluation of Educational Achievement (IEA) and Organisation for Economic Co-operation and Development (OECD) international studies such as TIMSS and PISA. Take NAEP as an example: each NAEP dataset's unique set of cutpoints for achievement levels (defined as ***Basic,*** ***Proficient,*** and ***Advanced***) is in the `EdSurvey` package. They can be accessed using the `showCutPoints` function:

```r
showCutPoints(data = sdf)
```

```
## Achievement Levels:
##   Basic:  262
##   Proficient:  299
##   Advanced:  333
```

The `achievementLevels` function applies the appropriate weights and 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*) and at or above each achievement level (*cumulative*), calculate the percentage distribution of students by achievement levels (discrete or cumulative) and selected

[7]Use `?achievementLevels` for details on default `achievementLevels` arguments.

characteristics (specified in `aggregateBy`), and compute the percentage distribution of students by selected characteristics within a specific achievement level.

The `achievementLevels` function can produce statistics by both discrete and cumulative achievement levels. By default, the `achievementLevels` function produces results only by discrete achievement levels; when the `returnCumulative` argument is set to `TRUE`, the function generates results by both discrete and cumulative achievement levels.

To compute overall results by achievement levels, use an NCES dataset's default plausible values in the `achievementVars` argument; in this case, they are the five or 20 plausible values for the subject composite scale.

```
aLev0 <- achievementLevels(achievementVars = c("composite"),
                           data = sdf, returnCumulative = TRUE)
```

```
aLev0$discrete
```

Table 6: aLev0$discrete

Level	N	wtdN	Percent	StandardError
Below Basic	5731.2	5779.5052	34.132690	0.9744207
At Basic	6695.6	6580.2181	38.861552	0.7115633
At Proficient	3666.0	3694.7565	21.820549	0.6342187
At Advanced	822.2	877.9837	5.185209	0.4007991

In the next example, the plausible values for `composite` and the variable `dsex` are used to calculate the achievement levels, which are aggregated by the variable `dsex` using `aggregateBy`.

```
aLev1 <- achievementLevels(achievementVars = c("composite", "dsex"), aggregateBy = "dsex",
                  data = sdf, returnCumulative = TRUE)
```

```
aLev1$discrete
```

Table 7: aLev1$discrete

Level	dsex	N	wtdN	Percent	StandardError
Below Basic	Male	2880.8	2865.6455	33.666050	1.0951825
At Basic	Male	3266.2	3236.4034	38.021772	0.9537470
At Proficient	Male	1877.2	1910.7861	22.448213	0.7257305
At Advanced	Male	461.8	499.1392	5.863965	0.5081607
Below Basic	Female	2850.4	2913.8597	34.604399	1.1154848
At Basic	Female	3429.4	3343.8146	39.710456	0.8650729
At Proficient	Female	1788.8	1783.9704	21.186066	0.8148916
At Advanced	Female	360.4	378.8444	4.499079	0.3888590

Note that each level of the `dsex` variable aggregates to 100 for the results by discrete achievement levels. The object `aLev1` created in this call to `achievementLevels` is a `list` with two `data.frame`s: one for the discrete results and the other for the cumulative results. In the previously described code, only the discrete levels are shown using `aLev1$discrete`. To show the cumulative results, change the specified `data.frame`. For example,

```r
aLev1$cumulative
```

Table 8: aLev1$cumulative

Level	dsex	N	wtdN	Percent	StandardError
Below Basic	Male	2880.8	2865.6455	33.666050	1.0951825
At or Above Basic	Male	5605.2	5646.3287	66.333950	1.0951825
At or Above Proficient	Male	2339.0	2409.9253	28.312178	0.8635866
At Advanced	Male	461.8	499.1392	5.863965	0.5081607
Below Basic	Female	2850.4	2913.8597	34.604399	1.1154848
At or Above Basic	Female	5578.6	5506.6295	65.395601	1.1154848
At or Above Proficient	Female	2149.2	2162.8149	25.685145	1.0073379
At Advanced	Female	360.4	378.8444	4.499079	0.3888590

The `aggregateBy` argument sums the percentage of students by discrete achievement level up to 100 at the most disaggregated level specified by the analytical variables and determines the order of aggregation. For example, when `dsex` and `iep` are used for analysis, `aggregateBy = c("dsex", "iep")` and `aggregateBy = c("iep", "dsex")` produce the same percentage but arrange the results in different ways depending on the order in the argument. When using `aggregateBy = c("iep", "dsex")`, the percentages add up to 100 within each category of `dsex` for each category of `iep`, respectively:

```r
achievementLevels(achievementVars = c("composite", "dsex", "iep"),
                  aggregateBy = c("iep", "dsex"), data = sdf)
```

```
##
## AchievementVars: composite, dsex, iep
## aggregateBy: iep, dsex
##
## Achievement Level Cutpoints:
## 262 299 333
##
## Plausible values: 5
## jrrIMax: 1
## Weight variable: 'origwt'
## Variance method: jackknife
## JK replicates: 62
## full data n: 17606
## n used: 16907
##
##
## Discrete
##          Level iep   dsex      N       wtdN    Percent StandardError
##    Below Basic Yes   Male  810.2  753.47862 66.4635116     2.0061208
##       At Basic Yes   Male  281.6  282.52828 24.9215056     2.0783210
##  At Proficient Yes   Male   72.8   85.69544  7.5590995     1.4614600
##    At Advanced Yes   Male    9.4   11.97026  1.0558833     0.7673700
##    Below Basic Yes Female  471.2  465.33346 76.4954517     2.9245271
##       At Basic Yes Female  108.8  106.71734 17.5430994     2.0864253
##  At Proficient Yes Female   31.2   34.36986  5.6500084     1.6430596
##    At Advanced Yes Female    2.8    1.89454  0.3114405     0.2601418
##    Below Basic  No   Male 2067.6 2111.69806 28.6261355     1.0630715
```

```
##       At Basic  No   Male 2982.6 2952.86086 40.0289211     1.0125447
##  At Proficient  No   Male 1804.4 1825.09062 24.7408909     0.7840337
##    At Advanced  No   Male  452.4  487.16896  6.6040524     0.5558956
##    Below Basic  No Female 2379.0 2448.49754 31.3451478     1.2051321
##       At Basic  No Female 3318.8 3236.55190 41.4336531     0.9207178
##  At Proficient  No Female 1757.4 1749.56228 22.3975264     0.8954779
##    At Advanced  No Female  356.8  376.79678  4.8236727     0.4233201
```

Notice that each unique value pair of the two variables (i.e., Yes + Male or No + Female) sums to 100 because of `aggregateBy`.

NOTE: It is not appropriate to aggregate the results by only one variable when more than one variable is used in the analysis. The same variables used in the analysis also need to be used in the argument `aggregateBy()`, but their order can be changed to obtain the desired results.

The `achievementLevels` function also can compute the percentage of students by selected characteristics within a specific achievement level. The object `aLev2` presents the percentage of students by sex within each achievement level (i.e., within each discrete and cumulative level).

```
aLev2 <- achievementLevels(achievementVars = c("composite", "dsex"),
                           aggregateBy = "composite",
                           data = sdf, returnCumulative = TRUE)
aLev2$discrete
```

```
##           Level   dsex      N      wtdN  Percent StandardError
## 1   Below Basic   Male 2880.8 2865.6455 49.58289      0.948680
## 2   Below Basic Female 2850.4 2913.8597 50.41711      0.948680
## 3      At Basic   Male 3266.2 3236.4034 49.18383      0.802051
## 4      At Basic Female 3429.4 3343.8146 50.81617      0.802051
## 5 At Proficient   Male 1877.2 1910.7861 51.71616      1.191306
## 6 At Proficient Female 1788.8 1783.9704 48.28384      1.191306
## 7   At Advanced   Male  461.8  499.1392 56.85063      2.007765
## 8   At Advanced Female  360.4  378.8444 43.14937      2.007765
```

```
aLev2$cumulative
```

```
##                    Level   dsex      N      wtdN  Percent StandardError
## 1            Below Basic   Male 2880.8 2865.6455 49.58289     0.9486800
## 2            Below Basic Female 2850.4 2913.8597 50.41711     0.9486800
## 3      At or Above Basic   Male 5605.2 5646.3287 50.62629     0.6131938
## 4      At or Above Basic Female 5578.6 5506.6295 49.37371     0.6131938
## 5 At or Above Proficient   Male 2339.0 2409.9253 52.70200     1.0576380
## 6 At or Above Proficient Female 2149.2 2162.8149 47.29800     1.0576380
## 7            At Advanced   Male  461.8  499.1392 56.85063     2.0077651
## 8            At Advanced Female  360.4  378.8444 43.14937     2.0077651
```

The percentage of students within a specific achievement level can be aggregated by one or more variables. For example, the percentage of students classified as English learners (`lep`) is aggregated by `dsex` within each achievement level:

```
aLev3 <- achievementLevels(achievementVars = c("composite", "dsex", "lep"),
                           aggregateBy = c("dsex", "composite"),
                           data = sdf, returnCumulative = TRUE)
aLev3$discrete
```

```
##             Level   dsex lep      N       wtdN    Percent StandardError
## 1    Below Basic   Male Yes  355.8  436.03778 15.2177175     1.6567089
## 2    Below Basic   Male  No 2523.8 2429.29192 84.7822825     1.6567089
## 3       At Basic   Male Yes  138.4  156.75146  4.8455620     0.7683430
## 4       At Basic   Male  No 3125.0 3078.19756 95.1544380     0.7683430
## 5  At Proficient   Male Yes   27.6   31.75786  1.6620312     0.5680123
## 6  At Proficient   Male  No 1849.6 1879.02820 98.3379688     0.5680123
## 7    At Advanced   Male Yes    1.2    0.75590  0.1514407     0.1793785
## 8    At Advanced   Male  No  460.6  498.38332 99.8485593     0.1976283
## 9    Below Basic Female Yes  334.2  422.06640 14.4853587     1.6957678
## 10   Below Basic Female  No 2515.4 2491.67850 85.5146413     1.6957678
## 11      At Basic Female Yes   96.4  102.80364  3.0744683     0.7676398
## 12      At Basic Female  No 3332.8 3240.98230 96.9255317     0.7676398
## 13 At Proficient Female Yes   19.2   22.69640  1.2722408     0.4289834
## 14 At Proficient Female  No 1769.6 1761.27402 98.7277592     0.4289834
## 15   At Advanced Female Yes    1.2    1.80846  0.4773622     0.7475650
## 16   At Advanced Female  No  359.2  377.03598 99.5226378     0.7919696
```

```r
aLev3$cumulative
```

```
##                      Level   dsex lep      N       wtdN    Percent StandardError
## 1              Below Basic   Male Yes  355.8  436.03778 15.2177175     1.6567089
## 2              Below Basic   Male  No 2523.8 2429.29192 84.7822825     1.6567089
## 3       At or Above Basic   Male Yes  167.2  189.26522  3.3528686     0.5358275
## 4       At or Above Basic   Male  No 5435.2 5455.60908 96.6471314     0.5358275
## 5  At or Above Proficient   Male Yes   28.8   32.51376  1.3491605     0.4574292
## 6  At or Above Proficient   Male  No 2310.2 2377.41152 98.6508395     0.4574292
## 7              At Advanced   Male Yes    1.2    0.75590  0.1514407     0.1793785
## 8              At Advanced   Male  No  460.6  498.38332 99.8485593     0.1976283
## 9              Below Basic Female Yes  334.2  422.06640 14.4853587     1.6957678
## 10             Below Basic Female  No 2515.4 2491.67850 85.5146413     1.6957678
## 11      At or Above Basic Female Yes  116.8  127.30850  2.3119254     0.5208318
## 12      At or Above Basic Female  No 5461.6 5379.29230 97.6880746     0.5208318
## 13 At or Above Proficient Female Yes   20.4   24.50486  1.1330078     0.4270294
## 14 At or Above Proficient Female  No 2128.8 2138.31000 98.8669922     0.4270294
## 15             At Advanced Female Yes    1.2    1.80846  0.4773622     0.7475650
## 16             At Advanced Female  No  359.2  377.03598 99.5226378     0.7919696
```

Finally, users can set unique cutpoints that override the standard values in the `EdSurvey` package by using the `cutpoints` argument. In the example that follows, `aLev1` uses the standard cutpoints of `c(262,299,333)` as shown in `showCutPoints` earlier, whereas `aLev4` uses `cutpoints = c(267,299,333)`, resulting in a higher threshold to reach the ***Basic*** category but leaving ***Proficient*** and ***Advanced*** unchanged:

```r
aLev4 <- achievementLevels(achievementVars = c("composite", "dsex"),
                           aggregateBy = "dsex",
                           data = sdf,
                           cutpoints = c(267, 299, 333),
                           returnCumulative = TRUE)

aLev4$discrete
```

```
##           Level   dsex      N      wtdN   Percent StandardError
## 1 Below Level 1   Male 3285.0 3262.6418 38.330025     1.2149501
```

```
## 2    At Level 1   Male 2862.0 2839.4071 33.357798     0.9636501
## 3    At Level 2   Male 1877.2 1910.7861 22.448213     0.7257305
## 4    At Level 3   Male  461.8  499.1392  5.863965     0.5081607
## 5 Below Level 1 Female 3284.8 3324.5956 39.482215     1.1460243
## 6    At Level 1 Female 2995.0 2933.0787 34.832640     0.7304983
## 7    At Level 2 Female 1788.8 1783.9704 21.186066     0.8148916
## 8    At Level 3 Female  360.4  378.8444  4.499079     0.3888590
```

```
aLev1$discrete
```

```
##           Level   dsex      N      wtdN   Percent StandardError
## 1   Below Basic   Male 2880.8 2865.6455 33.666050     1.0951825
## 2      At Basic   Male 3266.2 3236.4034 38.021772     0.9537470
## 3 At Proficient   Male 1877.2 1910.7861 22.448213     0.7257305
## 4   At Advanced   Male  461.8  499.1392  5.863965     0.5081607
## 5   Below Basic Female 2850.4 2913.8597 34.604399     1.1154848
## 6      At Basic Female 3429.4 3343.8146 39.710456     0.8650729
## 7 At Proficient Female 1788.8 1783.9704 21.186066     0.8148916
## 8   At Advanced Female  360.4  378.8444  4.499079     0.3888590
```

Changing the cutpoint for a particular achievement level will result in different distributions of student achievement. Notice that labels for the levels based on user-defined cutpoints are distinct from those based on NAEP-defined cutpoints; instead, labels are based on the range of values in the `cutpoints` argument.

Calculating Percentiles With `percentile`

The percentile function compares a numeric vector of percentiles in the range 0 to 100 for a data year. For example, to compare the NAEP Primer's subject composite scale at the 10th, 25th, 50th, 75th, and 90th percentiles, include these as integers in the `percentiles` argument:

```
pct1 <- percentile(variable = "composite", percentiles = c(10, 25, 50, 75, 90), data = sdf)
pct1
```

```
## Percentile
## Call: percentile(variable = "composite", percentiles = c(10, 25, 50,
##     75, 90), data = sdf)
## full data n: 17606
## n used: 16915
##
##  percentile estimate        se       df confInt.ci_lower confInt.ci_upper nsmall
##          10 227.7205 1.0555662 14.13296         225.2553         229.9806 1635.4
##          25 251.9623 1.0171720 15.15107         249.7341         253.9892 4189.8
##          50 277.4784 1.1374141 18.21071         275.7172         279.1877 8417.0
##          75 301.1835 0.9132983 25.17313         299.4246         302.8996 4138.2
##          90 321.9306 0.9035171 21.85127         319.9356         324.0352 1596.0
```

Preparing an `edsurvey.data.frame.list`

Whereas most functions in the `EdSurvey` package involve analyses using one dataset, an `edsurvey.data.frame.list` appends `edsurvey.data.frame` objects into one list for analysis. For example, four NAEP mathematics

assessments from different years can be combined into an `edsurvey.data.frame.list` to make a single call to analysis functions for ease of use in comparing, formatting, and/or plotting output data. Data from various countries in an international study can be integrated into an `edsurvey.data.frame.list` for further analysis.

To prepare an `edsurvey.data.frame.list` for gap analysis, it is necessary to ensure that variable information is consistent across each `edsurvey.data.frame`. When comparing groups across data years, it is not uncommon for variable names and labels to change. For example, some data years feature a split-sample design based on accommodations status, thereby containing differences in frequently used demographic variables between samples as well as across data years. Two useful functions in determining these inconsistencies are `searchSDF()` and `levelsSDF()`, which return variable names, labels, and levels based on a character string.

Recoding Variable Names and Levels Using `recode.sdf` and `rename.sdf`

To assist in the process of standardizing data for `edsurvey.data.frame`s, `light.edsurvey.data.frame`s, and `edsurvey.data.frame.list`s, the functions `recode.sdf()` and `rename.sdf()` are particularly handy.

Similar to the `recode` argument from the `cor.sdf()` section earlier in this vignette (and featured in many other functions), `recode.sdf()` accepts the levels of a variable as a vector from their current values to their new recoded value. For example, changing the lowest level of `b017451` from `"Never or hardly ever"` to `"Infrequently"` and the highest level from `"Every day"` to `"Frequently"`, will recode levels for that variable in our connection to `sdf`:

```
sdf2 <- recode.sdf(sdf,
                  recode=list(b017451=list(from=c("Never or hardly ever"),
                                           to=c("Infrequently")),
                              b017451=list(from=c("Every day"),
                                           to=c("Frequently"))
                              )
                  )
searchSDF("b017451", sdf2, levels = TRUE)
```

```
## Variable: b017451
## Label: Talk about studies at home
## Levels (Lowest level first):
##      2. Once every few weeks
##      3. About once a week
##      4. 2 or 3 times a week
##      8. Omitted
##      0. Multiple
##      9. Infrequently
##      10. Frequently
```

In addition, we can change the name of variables using `rename.sdf()`. The recoded variable `"b017451"` can be changed to a value that more effectively describes its contents, such as `"studyTalkFrequency"`:

```
sdf2 <- rename.sdf(sdf2, "b017451", "studytalkfrequency")
searchSDF("studytalkfrequency", sdf2, levels = TRUE)
```

```
## Variable: studytalkfrequency
## Label: Talk about studies at home
## Levels (Lowest level first):
```

```
##       2. Once every few weeks
##       3. About once a week
##       4. 2 or 3 times a week
##       8. Omitted
##       0. Multiple
##       9. Infrequently
##       10. Frequently
```

NOTE: The functions `rename.sdf()` and `recode.sdf()` do not permanently overwrite the variable information from your data source; they recode it only for the current connection to the data in R. The original file formatting can be retrieved by reconnecting to the data source via `readNAEP()`.

Combining Several `edsurvey.data.frame` Objects Into a Single Object

Once variables between each `edsurvey.data.frame` have been standardized, they are combined into an `edsurvey.data.frame.list` and are ready for analysis. In the following example, `sdf` is subset into four datasets, appended into an `edsurvey.data.frame.list`, and assigned unique labels:

```
# make four 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(datalist = list(sdfA, sdfB, sdfC, sdfD),
                                 labels = c("A locations","B locations",
                                            "C locations","D locations"))
```

This `edsurvey.data.frame.list` can now be analyzed in other `EdSurvey` functions.

Recommended Workflow for Standardizing Variables in Trend Analyses

Although the `EdSurvey` package features several methods to resolve inconsistencies across `edsurvey.data.frames`, the following approach is recommended:

1. Connect to each dataset using a read function such as `readNAEP()`.
2. Recode each discrepant variable name and level using `recode.sdf()` and `rename.sdf()`.
3. Combine datasets into one `edsurvey.data.frame.list` object.
4. Analyze trends using the `edsurvey.data.frame.list` object.

NOTE: It also is possible to retrieve and recode variables with the `getData` function; further details and examples of this method are discussed in the vignette titled *Using the `getData` Function in EdSurvey*.

Estimating the Difference in Two Statistics With `gap`

Gap analysis is a methodology that estimates the difference between two statistics (e.g., mean scores, achievement level percentages, percentiles, and student group percentages) for two groups in a population. A gap occurs when one group outperforms the other group, wherein the difference between the two statistics is statistically significant (i.e., the difference is larger than the margin of error).

In NAEP, the gap analysis can be comparisons between groups (e.g., male students vs. female students) by or across years, between jurisdictions (e.g., two states, district vs. home state, state vs. national public) by or

across years, or comparisons of the same group between years (e.g., male students in 2015 vs. male students in 2003). Independent tests with an alpha level of .05 are performed for most of these types of comparisons. For comparison between jurisdictions, a dependent test is used for the case in which one jurisdiction is contained in another (e.g., state vs. national public).

Note that NAEP typically tests two statistics (e.g., two groups or two years) at a time; if you want to test more than that, multiple comparison procedures should be applied, and your results will be more conservative than NAEP's reported results. For more information on gap analysis and multiple comparison, see *Drawing Inferences From NAEP Results*.

Performing Gap Analysis and Understanding the Summary Output

The following code uses an unexported function `copyDataToTemp` that generates fake data for use in examples.

```
set.seed(42)
year1 <- EdSurvey:::copyDataToTemp(f0 = "M32NT2PM")
year2 <- EdSurvey:::copyDataToTemp(f0 = "M40NT2PM")
```

The gap analysis function can perform comparisons between groups by or across years, of the same group between years, or of comparisons of the gaps between groups across years. The following example demonstrates the `gap` function, comparing the difference between the `dsex` variables using dummy datasets—`year1` and `year2`—appended into an `edsurvey.data.frame.list`:

```
mathList <- edsurvey.data.frame.list(datalist = list(year1, year2),
                                     labels = c("math year1", "math year2"))
mathGap <- gap(variable = "composite", data = mathList,
               groupA = dsex == "Male", groupB = dsex == "Female")
```

Each gap output contains a `data.frame` detailing the results of the analyses, which are returned using the following:

```
mathGap$results
```

```
##       labels estimateA estimateAse estimateB estimateBse   diffAB     covAB
## 1 math year1  277.2735    1.014495  274.9149    1.040229 2.358576 0.3694072
## 2 math year2  276.8393    1.056766  274.3754    1.114861 2.463876 0.7071580
##    diffABse diffABpValue    dofAB    diffAA covAA diffAAse diffAApValue    dofAA
## 1 1.1715210   0.05398869 27.44776        NA    NA       NA           NA       NA
## 2 0.9722933   0.01338217 74.24251 0.4342073     0 1.464908    0.7678634 65.12246
##      diffBB covBB diffBBse diffBBpValue    dofBB    diffABAB covABAB diffABABse
## 1        NA    NA       NA           NA       NA          NA      NA         NA
## 2 0.5395077     0 1.524792    0.7241024 117.9997 -0.1053004       0   1.522437
##   diffABABpValue  dofABAB sameSurvey
## 1             NA       NA         NA
## 2       0.945065 66.60037      FALSE
```

When the data argument is an `edsurvey.data.frame.list`, the summary results include the following information:

- the covariates and their respective means (`estimateA`/`estimateB`) and standard errors (`estimateAse`/`estimateBse`) across a variable (typically data years)

- the difference between the values of `estimateA` and `estimateB`, as well as its respective standard errors and *p*-value (each starting with `diffAB`)
- the difference between the values of `estimateA` across a variable compared with the reference dataset, as well as its respective standard errors and *p*-value (each starting with `diffAA`)
- the difference within the values of `estimateB` across a variable compared with the reference dataset, as well as its respective standard errors and *p*-value (each starting with `diffBB`)
- the difference between the difference of `estimateA` and `estimateB` across a variable compared with the reference dataset, as well as its respective standard errors and *p*-value (each starting with `diffABAB`)
- the value `sameSurvey`, which indicates if a line in the data output uses the same survey as the reference line (a logical: `TRUE`/`FALSE`)

For example, in `mathGap$results`:

- The gap in mean mathematics scores between the `dsex` variables in year 1 (`diffAB`) is `2.2009456`.
- The gap in mean mathematics scores within the `dsex` variables across data years where `groupA = "Male"` (`diffAA`) is `0.6268042`.
- The gap in mean mathematics scores within the `dsex` variables across data years where `groupB = "Female"` (`diffBB`) is `-0.7962217`.
- The gap in mean mathematics scores between the `dsex` variables across data years (`diffABAB`) is `1.423026`.

In addition to the summary results, the gap output also contains a `data.frame` of percentage gaps, in a format matching the previous results `data.frame`. This is returned by using the following:

```
mathGap$percentage
```

```
##       labels     pctA    pctAse     pctB    pctBse    diffAB      covAB diffABse
## 1 math year1 50.31267 0.7732424 49.68733 0.7732424 0.6253492 -0.5979038 1.546485
## 2 math year2 51.04887 0.7316137 48.95113 0.7316137 2.0977415 -0.5352586 1.463227
##   diffABpValue    dofAB     diffAA covAA diffAAse diffAApValue    dofAA    diffBB
## 1    0.6884604 34.19288         NA    NA       NA           NA       NA        NA
## 2    0.1569309 59.25477 -0.7361961     0 1.064501    0.4911036 83.97934 0.7361961
##   covBB diffBBse diffBBpValue    dofBB  diffABAB covABAB diffABABse diffABABpValue
## 1    NA       NA           NA       NA        NA      NA         NA             NA
## 2     0 1.064501    0.4911036 83.97934 -1.472392       0   2.129002      0.4911036
##    dofABAB
## 1       NA
## 2 83.97934
```

Gap Analysis of Achievement Levels and Percentiles

Gap analysis also may be performed across achievement levels and percentiles by specifying the values in the `achievementLevel` or `percentiles` arguments, respectively. Using our previous `edsurvey.data.frame.list` object (`mathList`), setting `achievementLevel=c("Basic", "Proficient", "Advanced")` will perform comparisons between groups by and across years for each achievement level value.

```
mathALGap <- gap(variable = "composite", data = mathList,
                 groupA = dsex == "Male", groupB = dsex == "Female",
                 achievementLevel = c("Basic", "Proficient", "Advanced"))
mathALGap$results
```

```
##         achievementLevel     labels estimateA estimateAse estimateB estimateBse
## 1      At or Above Basic math year1 66.870662   1.2335219 65.100350   1.3290706
## 2      At or Above Basic math year2 66.008354   1.4761274 64.212592   1.2702205
## 3 At or Above Proficient math year1 28.719053   1.2778962 25.546317   1.1150117
## 4 At or Above Proficient math year2 28.469990   1.0605786 25.852175   1.2256185
## 5             At Advanced math year1  6.111211   0.6868165  4.509459   0.5578078
## 6             At Advanced math year2  5.833023   0.7136854  4.353453   0.4823550
##     diffAB      covAB  diffABse diffABpValue    dofAB    diffAA covAA  diffAAse
## 1 1.770311 0.47346693 1.5300559   0.25305547 47.43947        NA    NA        NA
## 2 1.795762 0.80498819 1.4773070   0.22899798 58.93240 0.8623073     0 1.9236757
## 3 3.172735 0.16683642 1.5945522   0.05430082 35.80025        NA    NA        NA
## 4 2.617815 0.63726122 1.1629469   0.02759747 68.48216 0.2490623     0 1.6606763
## 5 1.601751 0.09886164 0.7649465   0.04228707 42.36482        NA    NA        NA
## 6 1.479570 0.11276154 0.7186725   0.04512671 46.64694 0.2781884     0 0.9904867
##   diffAApValue     dofAA     diffBB covBB  diffBBse diffBBpValue     dofBB
## 1           NA        NA         NA    NA        NA           NA        NA
## 2    0.6559148  49.66853  0.8877581     0 1.8384474    0.6302057 102.66886
## 3           NA        NA         NA    NA        NA           NA        NA
## 4    0.8810447 115.55769 -0.3058583     0 1.6569224    0.8538879 109.77788
## 5           NA        NA         NA    NA        NA           NA        NA
## 6    0.7795513  79.05949  0.1560067     0 0.7374387    0.8331024  66.60928
##      diffABAB covABAB diffABABse diffABABpValue   dofABAB sameSurvey
## 1          NA      NA         NA             NA        NA         NA
## 2 -0.02545076       0   2.126854      0.9904753 104.21223      FALSE
## 3          NA      NA         NA             NA        NA         NA
## 4  0.55492055       0   1.973586      0.7793705  73.18939      FALSE
## 5          NA      NA         NA             NA        NA         NA
## 6  0.12218164       0   1.049587      0.9075937  87.93697      FALSE
```

Similarly, setting `percentiles = c(10, 25, 50, 75, 90)` will perform comparisons between groups by and across years for each percentile value.

```
mathPercentilesGap <- gap(variable = "composite", data = mathList,
                          groupA = dsex == "Male", groupB = dsex == "Female",
                          percentiles = c(10, 25, 50, 75, 90))
mathPercentilesGap$results
```

```
##    percentiles     labels estimateA estimateAse estimateB estimateBse   diffAB
## 1           10 math year1  228.5492   0.9667854  227.0247   2.2177507 1.524483
## 2           10 math year2  227.9186   2.4995864  226.2819   2.2006815 1.636691
## 3           25 math year1  253.0893   0.9471983  250.9874   1.5062090 2.101920
## 4           25 math year2  252.2491   1.3755823  250.0428   1.3677150 2.206282
## 5           50 math year1  278.2843   1.3730106  276.9812   0.9264616 1.303117
## 6           50 math year2  278.0656   1.6903894  276.3454   1.3722702 1.720201
## 7           75 math year1  302.8779   1.3682276  299.6181   1.1878050 3.259775
## 8           75 math year2  302.7776   0.8726368  299.9592   1.0668784 2.818378
## 9           90 math year1  324.2191   1.7859545  320.1731   1.2427261 4.046013
## 10          90 math year2  324.3229   1.8148749  319.6371   1.3593064 4.685793
##         covAB diffABse diffABpValue    dofAB     diffAA covAA diffAAse diffAApValue
## 1  0.12874954 2.365501  0.524380292 28.76492         NA    NA       NA           NA
## 2  1.59047562 2.812469  0.567514040 18.81623  0.6305664     0 2.680038    0.8155427
## 3  0.06906226 1.740036  0.231540228 63.44991         NA    NA       NA           NA
## 4  0.05717917 1.910108  0.253858464 47.38973  0.8401989     0 1.670153    0.6168345
```

```
## 5  0.14110321 1.568848  0.411462617 37.37699         NA    NA       NA           NA
## 6  1.27774827 1.478190  0.250443398 46.73963  0.2186734     0 2.177745    0.9203339
## 7  0.07671289 1.769040  0.073719644 35.58427         NA    NA       NA           NA
## 8  0.39727222 1.051275  0.009435823 60.84149  0.1002895     0 1.622819    0.9508935
## 9  0.13587502 2.112404  0.075499842 14.41527         NA    NA       NA           NA
## 10 0.47512103 2.047252  0.026239504 51.30331 -0.1038355     0 2.546253    0.9676270
##       dofAA     diffBB covBB diffBBse diffBBpValue     dofBB    diffABAB covABAB
## 1        NA         NA    NA       NA           NA        NA          NA      NA
## 2  30.88172  0.7427746     0 3.124327    0.8129968  53.19276 -0.1122082       0
## 3        NA         NA    NA       NA           NA        NA          NA      NA
## 4  57.62887  0.9445612     0 2.034529    0.6434062 106.40957 -0.1043623       0
## 5        NA         NA    NA       NA           NA        NA          NA      NA
## 6  63.31403  0.6357575     0 1.655735    0.7024377  56.57640 -0.4170841       0
## 7        NA         NA    NA       NA           NA        NA          NA      NA
## 8  72.08366 -0.3411076     0 1.596593    0.8316963  49.70196  0.4413971       0
## 9        NA         NA    NA       NA           NA        NA          NA      NA
## 10 52.23293  0.5359443     0 1.841761    0.7717175  90.91000 -0.6397798       0
##    diffABABse diffABABpValue   dofABAB sameSurvey
## 1          NA             NA        NA         NA
## 2    3.674993      0.9757890  41.32595      FALSE
## 3          NA             NA        NA         NA
## 4    2.583842      0.9678588 104.78313      FALSE
## 5          NA             NA        NA         NA
## 6    2.155534      0.8470519  81.70437      FALSE
## 7          NA             NA        NA         NA
## 8    2.057834      0.8308790  60.72581      FALSE
## 9          NA             NA        NA         NA
## 10   2.941682      0.8288465  43.44324      FALSE
```

Gap Analysis of Jurisdictions

Comparisons of district, state, and national jurisdictions also can be performed using the `gap` function. The `NAEPprimer` data package does not contain jurisdiction level variables, such as `fips`; therefore, examples cannot be shown in this vignette; instead, the following code scripts are to be used as a reference:

```r
# comparisons of two states
mathStateGap <- gap(variable = "composite", data = mathList,
                    fips == "California", fips == "Virginia")

# comparisons of state to all public schools in nation
mathList <- subset(mathList, schtyp2 == "Public")
mathStateNationGap <- gap(variable = "composite", data = mathList,
                          fips == "California", schtyp2 == "Public")

# comparisons of district to state
mathStateDistrictGap <- gap("composite", data = mathList,
                            distcod == "Los Angeles", fips == "California")
```

Regression Analysis With `lm.sdf`

After the data are read in with the `EdSurvey` package, a linear model can be fit to fully account for the complex sample design used for an NCES data by using `lm.sdf`.

The option `jrrIMax` is omitted in the following example; therefore, the default jackknife variance estimator is used. Also, an explicit weight variable is not set, so the `lm.sdf` function uses a default weight for the full sample in the analysis. For instance, `origwt` is the default weight in NAEP.

The data are read in and analyzed by the `lm.sdf` function—in this case, `dsex`, `b017451`, the five plausible values for `composite`, and the full sample weight `origwt`. By default, variance is estimated using the jackknife method, so the following call reads in the jackknife replicate weights:[8]

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

```
##
## Formula: composite ~ dsex + b017451
##
## Weight variable: 'origwt'
## Variance method: jackknife
## JK replicates: 62
## Plausible values: 5
## jrrIMax: 1
## full data n: 17606
## n used: 16331
##
## Coefficients:
##                                  coef        se        t    dof  Pr(>|t|)
## (Intercept)                 270.41112   1.02443 263.9615 54.670 < 2.2e-16 ***
## dsexFemale                   -2.95858   0.60423  -4.8965 54.991 8.947e-06 ***
## b017451Once every few weeks   4.23341   1.18327   3.5777 57.316 0.0007131 ***
## b017451About once a week     11.22612   1.25854   8.9200 54.683 2.983e-12 ***
## b0174512 or 3 times a week   14.94591   1.18665  12.5951 72.582 < 2.2e-16 ***
## b017451Every day              7.52998   1.30846   5.7549 48.470 5.755e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared: 0.0224
```

After the regression is run, the data are automatically removed from memory. By default, `lm.sdf` uses "treatment contrasts," where one level is dropped from the regression. This cannot be changed, but the omitted and comparison groups can be changed with the `relevels` argument. In the following example, "Female" is omitted from the analysis for the variable `dsex`:

```
lm1f <- lm.sdf(formula = composite ~ dsex + b017451, data = sdf,
               relevels = list(dsex = "Female"))
summary(lm1f)
```

```
##
## Formula: composite ~ dsex + b017451
##
## Weight variable: 'origwt'
## Variance method: jackknife
## JK replicates: 62
## Plausible values: 5
## jrrIMax: 1
```

[8] Use `?lm.sdf` for details on default `lm.sdf` arguments.

```
## full data n: 17606
## n used: 16331
##
## Coefficients:
##                                  coef        se        t    dof  Pr(>|t|)
## (Intercept)                 267.45254   1.13187 236.2919 76.454 < 2.2e-16 ***
## dsexMale                      2.95858   0.60423   4.8965 54.991 8.947e-06 ***
## b017451Once every few weeks   4.23341   1.18327   3.5777 57.316 0.0007131 ***
## b017451About once a week     11.22612   1.25854   8.9200 54.683 2.983e-12 ***
## b0174512 or 3 times a week   14.94591   1.18665  12.5951 72.582 < 2.2e-16 ***
## b017451Every day              7.52998   1.30846   5.7549 48.470 5.755e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared: 0.0224
```

Note that the coefficient on `dsex` changed from negative in the previous run to positive of the exact same magnitude, whereas none of the other coefficients (aside from the intercept) changed; this is the expected result. The change results from the switch of the reference gender from "Male" in the first regression model to "Female" in the second regression model. The `lm.sdf` function features variance estimation using both the jackknife and Taylor series variance estimation methods by setting the `varMethod` argument to the desired technique.

The standardized regression coefficient also can be returned by adding `src = TRUE` into the summary call to your regression model object:

```
summary(lm1f, src=TRUE)
```

```
##
## Formula: composite ~ dsex + b017451
##
## Weight variable: 'origwt'
## Variance method: jackknife
## JK replicates: 62
## Plausible values: 5
## jrrIMax: 1
## full data n: 17606
## n used: 16331
##
## Coefficients:
##                                   coef         se        t    dof   Pr(>|t|) stdCoef
## (Intercept)                 2.6745e+02 1.1319e+00 236.2919 76.454 0.0000e+00      NA
## dsexMale                    2.9586e+00 6.0423e-01   4.8965 54.991 8.9474e-06  0.0407
## b017451Once every few weeks 4.2334e+00 1.1833e+00   3.5777 57.316 7.1311e-04  0.0458
## b017451About once a week    1.1226e+01 1.2585e+00   8.9200 54.683 2.9834e-12  0.1175
## b0174512 or 3 times a week  1.4946e+01 1.1866e+00  12.5951 72.582 0.0000e+00  0.1659
## b017451Every day            7.5300e+00 1.3085e+00   5.7549 48.470 5.7550e-07  0.0817
##                                stdSE
## (Intercept)                       NA
## dsexMale                    0.008313 **
## b017451Once every few weeks 0.012791 *
## b017451About once a week    0.013175 *
## b0174512 or 3 times a week  0.013175 *
## b017451Every day            0.014200 *
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared: 0.0224
```

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.

Multivariate Regression With `mvrlm.sdf`

A multivariate regression model can be fit to fully account for the complex sample design used for NCES data by using `mvrlm.sdf`. This function implements an estimator that correctly handles multiple dependent variables that are numeric (such as plausible values), which allows for variance estimation using the jackknife replication method.

The vertical line symbol `|` separates dependent variables on the left-hand side of formula. In the following example, a multivariate regression is fit with two subject scales as the outcome variables (`algebra` and `geometry`) by two predictor variables signifying gender and a survey item concerning the ability to identify the best unit of area (`dsex` and `m072801`):

```
mvrlm1 <- mvrlm.sdf(algebra | geometry ~ dsex + m072801, data = sdf)
summary(mvrlm1)
```

```
##
## Formula: algebra | geometry ~ dsex + m072801
##
## jrrIMax:
## Weight variable: 'origwt'
## Variance method:
## JK replicates: 62
## full data n: 17606
## n used: 3287
##
## Coefficients:
##
## algebra
##                          coef        se         t    dof  Pr(>|t|)
## (Intercept)         258.32980   2.38447 108.33839 42.729 < 2.2e-16 ***
## dsexFemale            6.94298   1.51265   4.58995 49.897 3.021e-05 ***
## m072801B *           24.78260   2.23171  11.10475 67.935 < 2.2e-16 ***
## m072801C             11.75561   2.97489   3.95162 64.737 0.0001945 ***
## m072801D            -12.88466   6.55887  -1.96446 12.131 0.0728026 .
## m072801E              1.96793   5.38314   0.36557 21.275 0.7182938
## m072801Not Reached -33.52297  17.46008  -1.91998 10.968 0.0812328 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## geometry
##                            coef         se          t     dof  Pr(>|t|)
```

```
## (Intercept)        255.351767   2.368025 107.833211 33.7224 < 2.2e-16 ***
## dsexFemale           5.407780   1.584977   3.411898 35.8676  0.001613 **
## m072801B *          22.369806   2.212790  10.109321 57.1693 2.442e-14 ***
## m072801C             8.850143   3.647400   2.426425 51.3747  0.018796 *
## m072801D            -9.260011   5.873402  -1.576601 12.8849  0.139113
## m072801E            -0.185649   5.919666  -0.031361 23.9251  0.975242
## m072801Not Reached -31.782791  23.915420  -1.328966  5.1159  0.240046
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual correlation matrix:
##
##          algebra geometry
## algebra     1.00     0.85
## geometry    0.85     1.00
##
## Multiple R-squared by dependent variable:
##
##  algebra geometry
##   0.0926   0.0858
```

The `mvrlm.sdf` documentation provides examples to compare the regression outputs. See `?mvrlm.sdf` for an overview of additional details that can be accessed through components of the returned object. In addition, the vignette titled *Statistical Methods Used in EdSurvey* goes into further detail by describing estimation of the reported statistics.

Logistic Regression Analysis With `glm.sdf`, `logit.sdf`, and `probit.sdf`

A logistic regression model can be fit to fully account for the complex sample design used for NCES data by using `glm.sdf`, `logit.sdf`, and `probit.sdf`. These functions predict *binary* outcomes from a set of predictor variables factoring in appropriate weights and variance estimates.

Although some variables might already be binary, the function `I()` can be used to specify the desired outcome level for a nonbinary variable. A logistic regression can be run exploring the impact of gender (`dsex`) on the number of books at home (`b013801`) with the level matching `">100"` as the outcome level:

```
logit1 <- logit.sdf(I(b013801 %in% ">100") ~ dsex,
                    weightVar = 'origwt', data = sdf)
summary(logit1)
```

```
##
## Formula: b013801 ~ dsex
## Family: binomial (logit)
##
## Weight variable: 'origwt'
## Variance method: jackknife
## JK replicates: 62
## full data n: 17606
## n used: 16359
##
## Coefficients:
```

```
##                    coef         se          t    dof  Pr(>|t|)
## (Intercept)  -0.920421   0.046355 -19.855835 60.636 < 2.2e-16 ***
## dsexFemale    0.178274   0.050129   3.556331 54.578 0.0007863 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The log odds of having more than 100 books at home (versus less than or equal to 100 books) increases by `0.178274` for female students compared with male students.

Logistic regression results can be further interpreted with the assistance of the `oddsRatio` and `waldTest` functions.

oddsRatio

The `oddsRatio` helper function allows for the conversion of coefficients from an `EdSurvey` logit regression model to odds ratios. Odds ratios are useful for understanding the real likelihood of an event occurring based on a transformation to the log odds returned in a logistic model.

In `EdSurvey`, odds ratios can be returned by specifying the logistic model object (`logit1`)

```r
oddsRatio(logit1)
```

```
##                    OR      2.5%     97.5%
## (Intercept) 0.3983511 0.3621585 0.4345438
## dsexFemale  1.1951531 1.0777266 1.3125797
```

The odds of having more than 100 books at home (versus less than or equal to 100 books) increases by `1.1951531` for female students compared with male students.

waldTest

The `waldTest` function allows the user to test composite hypotheses—hypotheses with multiple coefficients involved—even when the data include plausible values. Because there is no likelihood test for plausible values or residuals, the Wald test fills the role of the likelihood ratio test, ANOVA, and F-test.

Wald tests can be run by specifying the model and coefficients. The 2nd coefficient in our `logit1` model object (`Female`) is tested in the following example:

```r
waldTest(model = logit1, coefficients = 2)
```

```
## Wald test:
## ----------
## H0:
## dsexFemale = 0
##
## Chi-square test:
## X2 = 12.6, df = 1, P(> X2) = 0.00038
##
## F test:
## W = 12.6, df1 = 1, df2 = 62, P(> W) = 0.00073
```

To learn more about conducting Wald tests, consult the vignette titled *Methods and Overview of Using EdSurvey for Running Wald Tests* at the AIR website.

Quantile Regression Analysis with `rq.sdf`

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

To conduct quantile regression at a given tau value (by default, tau is set as 0.5), specify using the `tau` argument (in this example `tau = 0.8`); all other arguments are otherwise consistent with `lm.sdf`, except for `returnVarEstInputs`, `returnNumberOfPSU`, and `standardizeWithSamplingVar`, which are not available.

```
rq1 <- rq.sdf(composite ~ dsex + b017451, data=sdf, tau = 0.8)
summary(rq1)
```

```
##
## Formula: composite ~ dsex + b017451
##
## tau: 0.8
## jrrIMax: 1
## Weight variable: 'origwt'
## Variance method: jackknife
## JK replicates: 62
## full data n: 17606
## n used: 16331
##
## Coefficients:
##                                coef       se        t    dof  Pr(>|t|)
## (Intercept)                299.7680   1.8103 165.5883 29.389 < 2.2e-16 ***
## dsexFemale                  -4.6280   1.2908  -3.5852 58.617 0.0006868 ***
## b017451Once every few weeks   6.5880   1.9086   3.4518 46.045 0.0012041 **
## b017451About once a week     12.4800   2.2959   5.4359 67.782 8.032e-07 ***
## b0174512 or 3 times a week   16.5420   2.4616   6.7201 29.867 1.943e-07 ***
## b017451Every day             12.7420   1.6932   7.5253 50.343 8.717e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

Mixed Models With `mixed.sdf`

The `EdSurvey` package features the functionality of estimating mixed-effects models accounting for plausible values and survey weights. The `EdSurvey` package fits a weighted mixed model, also known as a weighted multilevel or hierarchical linear model using the `WeMix` package.

This example illustrates how the user might implement the student-level weighting when using a survey (NAEP in this example) that does not have a weighting scheme previously implemented.

```
# Subset data to a sample of interest
sdf2 <- subset(sdf, scrpsu < 500)
```

```r
# Extract variables of interest to a light.edsurvey.data.frame
lsdf <- getData(sdf2, c("composite","dsex","b017451","scrpsu","origwt","smsrswt"),
                addAttributes=TRUE)

# Transform weights using your method (Note that this method is not recommended for NAEP)
lsdf$pwt1 <- lsdf$origwt/lsdf$smsrswt
lsdf$pwt2 <- lsdf$smsrswt

m1 <- mixed.sdf(composite ~ dsex + b017451 + (1|scrpsu), data=lsdf,
                weightVar = c('pwt1', 'pwt2'))
```

```
## Some weights are larger at higher levels. This could be the result of scaling. However, if the weigh
```

```r
summary(m1)
```

```
## Call:
## mixed.sdf(formula = composite ~ dsex + b017451 + (1 | scrpsu),
##     data = lsdf, weightVars = c("pwt1", "pwt2"))
##
## Formula: composite ~ dsex + b017451 + (1 | scrpsu)
##
## Plausible Values: 5
## Number of Groups:
##   Group Var Observations Level
## 1    scrpsu           22     2
## 2  Residual          492     1
##
## Variance terms:
##                    variance Std. Error Std.Dev.
## scrpsu.(Intercept) 558.6111  221.73775 23.63495
## Residual           876.7564   92.42563 29.61007
##
## Fixed Effects:
##                             Estimate Std. Error t value
## (Intercept)                 266.7950     8.1996 32.5374
## dsexFemale                   -1.1788     2.9982 -0.3932
## b017451Once every few weeks   2.1730     6.9541  0.3125
## b017451About once a week      9.8088     4.4724  2.1932
## b0174512 or 3 times a week   10.8633     6.0979  1.7815
## b017451Every day              6.7917     7.3655  0.9221
##
## Intraclass Correlation= 0.389
```

For further guidance and use cases for mixed-effects models in `EdSurvey`, see the vignette titled *Methods Used for Estimating Mixed-Effects Models in EdSurvey*. For examples of how NCES recommends using weighted mixed-effects models, as well as their summary of the mathematical background and description of hierarchical linear model's insufficiency in this case, see Appendix D in the NCES working paper on analysis of TIMSS data at *Using TIMSS to Analyze Correlates of Performance Variation in Mathematics*.

Endnotes

Memory Usage

Because many NCES databases have hundreds of columns and hundreds of thousands of rows, the `EdSurvey` package allows users to subset data and run regressions without storing it in the global environment. Alternatively, the `getData` function retrieves `light.edsurvey.data.frames` into the global environment, which can be costly to memory usage.

This package uses the `LaF` package to read in only the necessary data when needed for an analysis. Instead of storing all the data in memory, only some "header" information is stored as well as a link to the file in question. When the user calls a function, only the data needed for that function is read in. It works seamlessly and reduces the memory requirements for a user's machine.

Factors and Factor Analysis

R uses the concept of factors for data storage, which is a separate concept from factor analysis. In the case of the R storage method, it is simply a way of enforcing that valid data labels are the only labels that are used.

Summary and Next Steps

This vignette covered the basics of the `EdSurvey` package, such as preparing the R environment for analysis, creating summary tables with `edsurveyTable`, running linear regression models with `lm.sdf`, correlating variables with `cor.sdf`, and retrieving data for manipulation with the `getData` function. Aspects of the package relating to memory usage also were considered.

If you are interested in manipulating the `EdSurvey` data in a similar manner as other `data.frame`s, consult the vignette titled *Using the `getData` Function in EdSurvey*.

For a full list of `EdSurvey` functions and documentation, use the R help viewer:

```
help(package = "EdSurvey")
```

Additional Resources

Supplementary vignettes are available to assist in analyzing NCES data. Note that some of them are written with NAEP Primer data as examples, whereas others are relevant to international assessment or longitudinal data.

Several vignettes are available to assist in analyzing NCES data:

- *Using EdSurvey to Analyze NCES Data: An Illustration of Analyzing NAEP Primer* is an introduction to the basics of using the `EdSurvey` package for analyzing NCES data, using the NAEP Primer as an example. The vignette covers topics such as preparing the R environment for processing, creating summary tables, running linear regression models, and correlating variables.
- *Exploratory Data Analysis on NCES Data* provides examples of conducting exploratory data analysis on NAEP data.
- *Calculating Adjusted* p-*Values From EdSurvey Results* describes the basics of adjusting *p*-values to account for multiple comparisons.

- *Using the `getData` Function in EdSurvey* describes the use of the `EdSurvey` package when extensive data manipulation is required before analysis.
- *Using EdSurvey to Analyze NAEP Data With and Without Accommodations* provides an overview of the use of NAEP data with accommodations and describes methods used to analyze these data.
- *Using EdSurvey to Analyze TIMSS Data* is an introduction to the methods used in the analysis of large-scale educational assessment programs such as TIMSS using the `EdSurvey` package. The vignette covers topics such as preparing the R environment for processing, creating summary tables, running linear regression models, and correlating variables.
- *Using EdSurvey to Analyze ECLS-K:2011 Data* is an introduction to the methods used in the analysis of the large-scale child development study Early Childhood Longitudinal Study, Kindergarten Class of 2010-11 (ECLS-K:2011) using the `EdSurvey` package. The vignette covers topics such as preparing the R environment for processing, creating summary tables, running linear regression models, and correlating variables.
- *Using EdSurvey for Trend Analysis* describes the methods used in the `EdSurvey` package to conduct analyses of statistics that change across time in large-scale educational studies.
- *Producing LaTeX Tables From edsurveyTable Results With edsurveyTable2pdf* details the creation of pdf summary tables from summary results using the `edsurveyTable2pdf` function.

Methodology Resources

Documents that describe the statistical methodology used in the `EdSurvey` package include the following:

- *Statistical Methods Used in EdSurvey* details the estimation of the statistics in the `lm.sdf`, `achievementLevel`, and `edsurveyTable` functions.
- *Analyses Using Achievement Levels Based on Plausible Values* describes the methodological approaches for analyses using NAEP achievement levels.
- *Methods Used for Gap Analysis in EdSurvey* covers the methods comparing the gap analysis results of the `EdSurvey` package to the NAEP Data Explorer.
- *Methods Used for Estimating Percentiles in EdSurvey* describes the methods used to estimate percentiles.
- *Methods Used for Estimating Mixed-Effects Models in EdSurvey* describes the methods used to estimate mixed-effects models with plausible values and survey weights and how to fit different types of mixed-effects models using the `EdSurvey` package.
- *Methods and Overview of Using EdSurvey for Multivariate Regression* details the estimation of multivariate regression models using `mvrlm.sdf`.
- *Methods and Overview of Using EdSurvey for Running Wald Tests* describes the use of the Wald test to jointly test regression coefficients estimated using `lm.sdf` and `glm.sdf`.

Reference

Lee, M. D., Bailey, P. D., & Emad, A. (2018). *Using the `getData` Function in EdSurvey*. Washington, DC: American Institutes for Research. Retrieved from https://www.air.org/sites/default/files/EdSurvey-getData.pdf