Using EdSurvey 1.0.5 to Analyze NAEP Data: An Illustration of Analyzing NAEP Primer

Developed by Paul Bailey, Ahmad Emad, Michael Lee, Ting Zhang, Qingshu Xie, & Jiao Yu[*†]

April 26, 2017

Overview of the EdSurvey Package

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

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

Vignette outline

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

- Preparing the R environment for processing
- Accessing details about data of interest
- Creating summary tables using the `edsurveyTable` function
- Computing the percentages of students by achievement levels with the `achievementLevels` function
- Running linear regression models using the `lm.sdf` function
- Correlating variables with the `cor.sdf` function
- Retrieving data for manipulation by the user using the `getData` function

Additional resources

There are two supplementary vignettes in the package to assist in analyzing NCES data. *Using the getData Function in EdSurvey 1.0.5 to Manipulate the NAEP Primer Data* describes using the `EdSurvey` package in situations where extensive data manipulation is performed before analysis. The other, *Using the EdSurvey Package to Analyze NAEP Data With and Without Accommodations*, provides an overview of NAEP data with accommodations and describes methods used to analyze this data.

---

*This publication was prepared for NCES under Contract No. ED-IES-12-D-0002 with 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 and Dan Sherman for reviewing this document, as well as Jiayi Li and Fei Liu for conducting quality control tests to verify the functions in this document.

Vignette notation

This vignette displays examples using notation for R console input and output. Console input will be displayed within a grey 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 3.2.0 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 Analysis of 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: lfactors
```

```
## lfactors v1.0.1
```

```
## EdSurvey v1.0.5
```

Reading in data

The first step to running an analysis is reading in the data. This is done using `EdSurvey`'s `readNAEP` function.

Vignette sample NCES data set:

To follow along with this vignette, load the NAEP Primer data set `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 is bundled with the NAEPprimer package, the `system.file` function finds it regardless of where the package was installed on the machine. All other data sets will have to be referred to by their system path.

NCES data set

To load a unique NCES data set 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('//.../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 is attached to the resulting `edsurvey.data.frame`. The `readNAEP` function transparently accesses the necessary sample information and silently attaches it to the data.[1]

Both student and school data from a NCES data set 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 data set when read. More details on retrieving school variables for analysis will be outlined in this vignette with the `getData` function.

Getting to know the data format

There are several ways to get information about an `edsurvey.data.frame`. 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 with 17606 rows and 302 columns.
##
## There are 1 full sample weight(s) 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', 'Omitted'
```

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

```
##
##
## Default Conditions:
##   tolower(rptsamp) == "reporting sample"
## Achievement Levels:
##   Basic:      262
##   Proficient: 299
##   Advanced:   333
##
## Survey: NAEP
```

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

```
dim(sdf)
```

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

```
nrow(sdf)
```

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

```
ncol(sdf)
```

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

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

```
names(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"
```

[2] Use ?*function* in the R console to view documentation on base R and `EdSurvey` package functions. For example `?gsub` or `?lm.sdf`.

```
## [153] "m068007" "m068006" "m093601" "m053001" "m047801" "m086301" "m085701" "m085901"
## [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, this call to `searchSDF` searches for the character string `"book"` in the `edsurvey.data.frame` and specifies the `fileformat` to search the student data file:

```
searchSDF("book", 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 variables search via `searchSDF()` can also be returned by setting `levels = TRUE`:

```
searchSDF("book", 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
```

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

```
levelsSDF("b017451", sdf)
```

```
## 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
##     8. Omitted
##     0. Multiple
```

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(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. They are:
##     'mrps11' 'mrps12' 'mrps13' 'mrps14' 'mrps15'
##
##   'measurement' subject scale or subscale with 5 plausible values. They are:
##     'mrps21' 'mrps22' 'mrps23' 'mrps24' 'mrps25'
##
##   'geometry' subject scale or subscale with 5 plausible values. They are:
##     'mrps31' 'mrps32' 'mrps33' 'mrps34' 'mrps35'
##
##   'data_anal_prob' subject scale or subscale with 5 plausible values. They are:
##     'mrps41' 'mrps42' 'mrps43' 'mrps44' 'mrps45'
##
##   'algebra' subject scale or subscale with 5 plausible values. They are:
##     'mrps51' 'mrps52' 'mrps53' 'mrps54' 'mrps55'
##
##   'composite' subject scale or subscale with 5 plausible values (the default). They are:
##     'mrpcm1' 'mrpcm2' 'mrpcm3' 'mrpcm4' 'mrpcm5'
```

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

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

Removing Special Values

The EdSurvey package uses listwise deletion to remove special values in all of it's 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.

Making a Table

Summary tables can be created in the `EdSurvey` package using the `edsurveyTable` function. A call to `edsurveyTable`[3] 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(composite ~ dsex + b017451, sdf,
                     jrrIMax=1, varMethod="jackknife")
```

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

```
es1$data
```

Table 1: 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

[3]Consult the appendix or `?edsurveyTable` for details on default `edsurveyTable` arguments.

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 standard error of estimates, such as

- estimated scale scores,
- achievement levels, and
- regression analysis of student performance using jackknife variance estimation.

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.[4] An alternative is to set `jrrIMax=Inf` to obtain the ideal estimation with 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:

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

```
es1t$data
```

Table 2: 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 to up to 100 at the desired level, an adjustment can be made in the `pctAggregationLevel` argument to change to the level that they need to add up to 100.

Calculation of means and standard errors require 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:

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

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

[4] See the documentation for `lm.sdf` for details on the variance calculation.

```
es1b
```

Table 3: 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

```
?edsurveyTable
```

Achievement Level Analysis

The `achievementLevels` function[5] computes the percentages of students by achievement levels defined by NAEP. Each NAEP data set's unique set of cut points for achievement levels (defined as ***Basic***, ***Proficient***, and ***Advanced***) is provided in the `EdSurvey` package. They can be accessed using the `showCutPoints` function:

```
showCutPoints(sdf)
```

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

The `achievementLevels` function applies 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*), at or above each achievement level (*cumulative*), calculate the percentage distribution of students by achievement levels (discrete or cumulative) and selected 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 `achivementLevels` function only produces the results 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 the NAEP data set's default plausible values in the `achievementVars` argument; in this case, they are the 5 or 20 plausible values for the subject composite scale.

[5]Consult the appendix or `?achievementLevels` for details on default `achievementLevels` arguments.

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

```
aLev0$discrete
```

Table 4: 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(c("composite", "dsex"), aggregateBy="dsex",
                  data=sdf, returnCumulative=TRUE)
```

```
aLev1$discrete
```

Table 5: 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 cumulative. 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:

```
aLev1$cumulative
```

Table 6: 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

Level	dsex	N	wtdN	Percent	StandardError
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, as well as determining 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 percentages, but arrange the results in different ways, depending on order in the argument. When using `aggregateBy = c("dsex", "iep")`, the percentages add up to 100 within each category of `iep` for each category of `dsex`, respectively; when using `aggregateBy = c("iep", "dsex")`, the percentages add up to 100 within each category of `dsex` for each category of `iep`, respectively.

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

```
##
## AchievementVars: composite, dsex, iep
## aggregateBy: dsex, iep
##
## Achievement Level Cutpoints:
## 262 299 333
##
## jrrIMax: 1
## Weight variable: 'origwt'
## Variance method: jackknife
## JK replicates: 62
## full data n: 17606
## n used: 16907
##
##
## Discrete
##          Level   dsex iep      N       wtdN    Percent StandardError
##    Below Basic   Male Yes  810.2  753.47862 66.4635116     2.0061208
##       At Basic   Male Yes  281.6  282.52828 24.9215056     2.0783210
##  At Proficient   Male Yes   72.8   85.69544  7.5590995     1.4614600
##    At Advanced   Male Yes    9.4   11.97026  1.0558833     0.7673700
##    Below Basic   Male  No 2067.6 2111.69806 28.6261355     1.0630715
##       At Basic   Male  No 2982.6 2952.86086 40.0289211     1.0125447
##  At Proficient   Male  No 1804.4 1825.09062 24.7408909     0.7840337
##    At Advanced   Male  No  452.4  487.16896  6.6040524     0.5558956
##    Below Basic Female Yes  471.2  465.33346 76.4954517     2.9245271
##       At Basic Female Yes  108.8  106.71734 17.5430994     2.0864253
##  At Proficient Female Yes   31.2   34.36986  5.6500084     1.6430596
##    At Advanced Female Yes    2.8    1.89454  0.3114405     0.2601418
##    Below Basic Female  No 2379.0 2448.49754 31.3451478     1.2051321
##       At Basic Female  No 3318.8 3236.55190 41.4336531     0.9207178
##  At Proficient Female  No 1757.4 1749.56228 22.3975264     0.8954779
##    At Advanced Female  No  356.8  376.79678  4.8236727     0.4233201
```

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

```
##
## AchievementVars: composite, dsex, iep
## aggregateBy: iep, dsex
##
## Achievement Level Cutpoints:
## 262 299 333
##
## 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 due to `aggregateBy`.

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

The `achievementLevels` function can also 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 levels).

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

```
##           Level   dsex      N      wtdN  Percent StandardError
```

```
## 1   Below Basic Female 2850.4 2913.8597 50.41776     0.9486797
## 2   Below Basic   Male 2880.8 2865.6455 49.58224     0.9486797
## 3      At Basic Female 3429.4 3343.8146 50.81662     0.8020508
## 4      At Basic   Male 3266.2 3236.4034 49.18338     0.8020508
## 5 At Proficient Female 1788.8 1783.9704 48.28434     1.1913055
## 6 At Proficient   Male 1877.2 1910.7861 51.71566     1.1913055
## 7   At Advanced Female  360.4  378.8444 43.16691     2.0076502
## 8   At Advanced   Male  461.8  499.1392 56.83309     2.0076502
```

```
aLev2$cumulative
```

```
##                    Level   dsex      N      wtdN  Percent StandardError
## 1            Below Basic Female 2850.4 2913.8597 50.41776     0.9486797
## 2            Below Basic   Male 2880.8 2865.6455 49.58224     0.9486797
## 3      At or Above Basic Female 5578.6 5506.6295 49.37388     0.6131937
## 4      At or Above Basic   Male 5605.2 5646.3287 50.62612     0.6131937
## 5 At or Above Proficient Female 2149.2 2162.8149 47.29676     1.0576369
## 6 At or Above Proficient   Male 2339.0 2409.9253 52.70324     1.0576369
## 7            At Advanced Female  360.4  378.8444 43.16691     2.0076502
## 8            At Advanced   Male  461.8  499.1392 56.83309     2.0076502
```

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 ELL (`lep`) is aggregated by `dsex` within each achievement level:

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

```
##            Level   dsex lep      N       wtdN    Percent StandardError
## 9    Below Basic   Male  No 2523.8 2429.29192 84.7819777     1.6567088
## 10   Below Basic   Male Yes  355.8  436.03778 15.2180223     1.6567088
## 11      At Basic   Male  No 3125.0 3078.19756 95.1552181     0.7683424
## 12      At Basic   Male Yes  138.4  156.75146  4.8447819     0.7683424
## 13 At Proficient   Male  No 1849.6 1879.02820 98.3361448     0.5680079
## 14 At Proficient   Male Yes   27.6   31.75786  1.6638552     0.5680079
## 15   At Advanced   Male  No  460.6  498.38332 99.8482894     0.1976280
## 16   At Advanced   Male Yes    1.2    0.75590  0.1517106     0.1793098
## 1    Below Basic Female  No 2515.4 2491.67850 85.5147337     1.6957678
## 2    Below Basic Female Yes  334.2  422.06640 14.4852663     1.6957678
## 3       At Basic Female  No 3332.8 3240.98230 96.9257657     0.7676397
## 4       At Basic Female Yes   96.4  102.80364  3.0742343     0.7676397
## 5  At Proficient Female  No 1769.6 1761.27402 98.7279927     0.4289833
## 6  At Proficient Female Yes   19.2   22.69640  1.2720073     0.4289833
## 7    At Advanced Female  No  359.2  377.03598 99.5214056     0.7919682
## 8    At Advanced Female Yes    1.2    1.80846  0.4785944     0.7473271
```

```
aLev3$cumulative
```

```
##                     Level   dsex lep      N       wtdN    Percent StandardError
```

```
## 9             Below Basic   Male  No 2523.8 2429.29192 84.7819777     1.6567088
## 10            Below Basic   Male Yes  355.8  436.03778 15.2180223     1.6567088
## 11      At or Above Basic   Male  No 5435.2 5455.60908 96.6473565     0.5358274
## 12      At or Above Basic   Male Yes  167.2  189.26522  3.3526435     0.5358274
## 13 At or Above Proficient   Male  No 2310.2 2377.41152 98.6506740     0.4574292
## 14 At or Above Proficient   Male Yes   28.8   32.51376  1.3493260     0.4574292
## 15            At Advanced   Male  No  460.6  498.38332 99.8482894     0.1976280
## 16            At Advanced   Male Yes    1.2    0.75590  0.1517106     0.1793098
## 1             Below Basic Female  No 2515.4 2491.67850 85.5147337     1.6957678
## 2             Below Basic Female Yes  334.2  422.06640 14.4852663     1.6957678
## 3       At or Above Basic Female  No 5461.6 5379.29230 97.6882679     0.5208317
## 4       At or Above Basic Female Yes  116.8  127.30850  2.3117321     0.5208317
## 5  At or Above Proficient Female  No 2128.8 2138.31000 98.8665821     0.4270291
## 6  At or Above Proficient Female Yes   20.4   24.50486  1.1334179     0.4270291
## 7             At Advanced Female  No  359.2  377.03598 99.5214056     0.7919682
## 8             At Advanced Female Yes    1.2    1.80846  0.4785944     0.7473271
```

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

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

aLev4$discrete
```

```
##        Level   dsex      N      wtdN   Percent StandardError
## 5      < 267   Male 3285.0 3262.6418 38.330025     1.2149501
## 6 [267, 299)   Male 2862.0 2839.4071 33.357798     0.9636501
## 7 [299, 333)   Male 1877.2 1910.7861 22.448213     0.7257305
## 8     >= 333   Male  461.8  499.1392  5.863965     0.5081607
## 1      < 267 Female 3284.8 3324.5956 39.482215     1.1460243
## 2 [267, 299) Female 2995.0 2933.0787 34.832640     0.7304983
## 3 [299, 333) Female 1788.8 1783.9704 21.186066     0.8148916
## 4     >= 333 Female  360.4  378.8444  4.499079     0.3888590
```

```
aLev1$discrete
```

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

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

Regression Analysis

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

Note that the option `jrrIMax` is left out in the following example; therefore, the default jackknife variance estimator is used. Also, note that an explicit weight variable is not set, so the `lm.sdf` function uses `origwt`, the default, as the weight for the full sample in the analysis.

The data is 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:[6]

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

```
##
## Formula: composite ~ dsex + b017451
##
## Weight variable: 'origwt'
## Variance method: jackknife
## JK replicates: 62
## full data n: 17606
## n used: 16331
##
## Coefficients:
##                                  coef        se        t  Pr(>|t|)
## (Intercept)                  270.41112   1.02443 263.9615 < 2.2e-16 ***
## dsexFemale                    -2.95858   0.60423  -4.8965 7.307e-06 ***
## b017451Once every few weeks    4.23341   1.18327   3.5777 0.0006795 ***
## b017451About once a week      11.22612   1.25854   8.9200 1.020e-12 ***
## b0174512 or 3 times a week    14.94591   1.18665  12.5951 < 2.2e-16 ***
## b017451Every day               7.52998   1.30846   5.7549 2.878e-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 is 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 group can be changed with the `relevels` argument. In the following example, "Female" is omitted from the analysis for the variable `dsex`:

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

[6] Consult the appendix or `?lm.sdf` for details on default `lm.sdf` arguments.

```
##
## Formula: composite ~ dsex + b017451
##
## Weight variable: 'origwt'
## Variance method: jackknife
## JK replicates: 62
## full data n: 17606
## n used: 16331
##
## Coefficients:
##                                  coef        se        t  Pr(>|t|)
## (Intercept)                  267.45254   1.13187 236.2919 < 2.2e-16 ***
## dsexMale                       2.95858   0.60423   4.8965 7.307e-06 ***
## b017451Once every few weeks    4.23341   1.18327   3.5777 0.0006795 ***
## b017451About once a week      11.22612   1.25854   8.9200 1.020e-12 ***
## b0174512 or 3 times a week    14.94591   1.18665  12.5951 < 2.2e-16 ***
## b017451Every day               7.52998   1.30846   5.7549 2.878e-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 exactly same magnitude, whereas none of the other coefficients (aside from the intercept) changed at all–this is the expected result. The change is due to 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.

Correlation Analysis

The `EdSurvey` package features multiple correlation methods for data exploration and analysis that fully accounts for the complex sample design in NAEP data by using the `cor.sdf` function.[7] This includes 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, and
- Correlations among plausible values of the subject scales and subscales (marginal correlation coefficients, which uses Pearson type).

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 is removed automatically from memory after the correlation is run.

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

[7] Consult the appendix or `?cor.sdf` for details on default `cor.sdf` arguments.

It's important to take note of the order of levels to ensure that correlations are functioning as intended. Printing a correlation object will provide a condensed summary of 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 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("b017451","t088001", 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 that 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("b017451","t088001", 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 levels might be useful to modify a variable that is out of order or when reversing the orientation of a series. The `reorder` argument is also suitable when implemented in conjunction with recoded levels.

NOTE: As an alternative, recoding also can be completed within `getData`, a function detailed later in the vignette. 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:

```
cor3_mcc <- cor.sdf("num_oper", "algebra", 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`[8] of the `edsurvey.data.frame` data where `dsex == 1` (saved as the `sdf_dnf` object), `b017451`, `pared`, and the full sample weight `origwt`:

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

```
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:

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

```
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 data set read for

[8] `subset` will be further detailed in this vignette, or use `?subset` to access function documentation.

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("b017451", "pared", 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("b017451", "pared", 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
```

Subsetting the Data

A subset of a data set 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(composite ~ dsex + sdracem, sdfm)
```

```
es2
```

Table 7: 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

Getting a data.frame for Further Manipulation

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 requested variables by either specifying a set of variable names in `varnames` or by entering a formula in `formula`.[9]

To access and manipulate all data for `dsex` and `b017451` variables in `sdf` can be returned by calling `getData`. Note that in the following code, the `head` function is used. This reveals only the first few rows of the resulting data:

```
gddat <- getData(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.

Merging student and school data sets

After loading the `edsurvey.data.frame` object into the R working environment, both student and school data from a NCES data set can be analyzed and merged using `getData`. To retrieve school variables, include them in the vector of variable names in the `getData` call and specify the key linking variables in the student

[9] Consult the appendix or `?getData` for details on default `getData` arguments.

and school data sets within the arguments `schoolMergeVarStudent` and `schoolMergeVarSchool`. In this example, `getData` calls the variables `dsex` and `b017451` from the student data file, as well as the variable `c052601` from the school data file, merging the student variable key `scrpsu` on the school variable key `sscrpsu`:

```
gddat2 <- getData(sdf, varnames=c("dsex","b017451","c052601"),
                  schoolMergeVarStudent="scrpsu", schoolMergeVarSchool="sscrpsu")
head(gddat2)
```

```
##     dsex              b017451         c052601
## 2   Male            Every day 6 to 10 percent
## 3 Female    About once a week 6 to 10 percent
## 4 Female            Every day 6 to 10 percent
## 5   Male            Every day 6 to 10 percent
## 7 Female Once every few weeks 6 to 10 percent
## 8   Male  2 or 3 times a week 6 to 10 percent
```

Retrieving all variables in a data set

To extract *all* of the data in an `edsurvey.data.frame`, define the `varnames` argument as `names(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(sdf, varnames=names(sdf), addAttributes=TRUE,
                 omittedLevels=FALSE, defaultConditions=FALSE)
dim(lsdf0) # excludes the one school variable in the sdf
```

```
## [1] 17606   301
```

```
dim(sdf)
```

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

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

Additional details on the features of the `getData` function are included in the *Using the getData Function in EdSurvey 1.0.5 to Manipulate the NAEP Primer Data* vignette.

Notes

Memory usage

Since many NCES databases have hundreds of columns and hundreds of thousand 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 it is needed for an analysis. Instead of storing all of 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. This 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 *Using the getData Function in `EdSurvey` 1.0.5 to Manipulate the NAEP Primer Data* vignette.

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

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

Appendix

1. achievementLevels

- **achievementVars:** Character vector indicating variables to be included in the achievement levels table, potentially with a subject scale or subscale. When the subject scale or subscale is omitted, then the default subject scale or subscale is used. You can find the default composite scale and all subscales using the function `showPlausibleValues`.
- **aggregateBy:** Character vector specifying variables to aggregate achievement levels by. The percentage column sums up to 100 for all levels of all variables specified here. When set to defaut of `NULL`, the percentage column sums up to 100 for all levels of all variables specified in in `achievementVars`.
- **data:** An `edsurvey.data.frame`.
- **cutpoints:** Numeric vector indicating cut points for *Basic*, *Proficient*, *Advanced*. Set to standard NAEP cut points by default.
- **returnDiscrete:** Logical indicating if discrete achievement levels should be returned. Defaults to `TRUE`.
- **returnCumulative:** Logical indicating if cumulative achievement levels should be returned. Defaults to `FALSE`.
- **weightVar:** Character indicating the weight variable to use.
- **jrrIMax:** Numeric value. When using the jackknife variance estimation method, the V_{jrr} term can be estimated with any positive number of plausible values and is estimated on the first of the lower of the number of available plausible values and `jrrIMax`. Because of this, when `jrrIMax` is set to Inf, all of the plausible values will be used. Higher values of `jrrIMax` lead to longer computing times and more accurate variance estimates.
- **schoolMergeVarStudent:** A character variable name from the student file used to merge student and school data files. Set to `NULL` by default.
- **schoolMergeVarSchool:** A character variable name from the school file used to merge student and school data files. Set to `NULL` by default.
- **omittedLevels:** A logical value set to `TRUE` indicating that drops those levels of all factor variables that are specified in `edsurvey.data.frame`. Use `print` on an `edsurvey.data.frame` to see the omitted levels.

- **defaultConditions:** A logical value set to `TRUE` that uses the default conditions stored in `edsurvey.data.frame` to subset the data. Use `print` on an `edsurvey.data.frame` to see the default conditions.
- **recode:** A list of lists to recode variables. Defaults to `NULL`. Can be set as `recode = list(var1= list(from=c("a","b","c"), to ="d"))`. See examples using `?cor.sdf`.

2. cor.sdf

- **x:** A character variable name from the `data` to be correlated with `y`.
- **y:** A character variable name from the `data` to be correlated with `x`.
- **data:** Object of class `edsurvey.data.frame`.
- **method:** Character string indicating which correlation coefficient (or covariance) is to be computed. One of "Pearson" (default), "Spearman," "Polychoric," or "Polyserial."
- **weightVar:** Character indicating the weight variable to use.
- **reorder:** A list to reorder variables. Defaults to `NULL`. Can be set as `reorder = list(var1=c("a","b","c"), var2=c("4","3","2","1"))`. See examples using `?cor.sdf`.
- **schoolMergeVarStudent:** A character variable name from the student file used to merge student and school data files. Set to `NULL` by default.
- **schoolMergeVarSchool:** A character variable name from the school file used to merge student and school data files. Set to `NULL` by default.
- **omittedLevels:** A logical value set to `TRUE` indicating that drops those levels of all factor variables that are specified in `edsurvey.data.frame`. Use `print` on an `edsurvey.data.frame` to see the omitted levels.
- **defaultConditions:** A logical value set to `TRUE` that uses the default conditions stored in `edsurvey.data.frame` to subset the data. Use `print` on an `edsurvey.data.frame` to see the default conditions.
- **recode:** A list of lists to recode variables. Defaults to `NULL`. Can be set as `recode = list(var1= list(from=c("a","b","c"), to ="d"))`. See examples using `?cor.sdf`.

3. edsurveyTable

- **formula:** Object of class `formula`, potentially with a subject scale or subscale on the left hand side, and variable(s) for tabulation on the right hand side.
- **data:** Object of class `edsurvey.data.frame`.
- **weightVar:** Character string indicating the weight variable to use, defaults to `NULL`.
- **jrrIMax:** Integer indicating the maximum number of plausible values to include when calculating the variance term. Defaults to 1.
- **pctAggregationLevel:** The percentage variable sums up to 100 for the first `pctAggregationLevel` columns. Defaults to `NULL`.
- **returnMeans:** A logical value set to `TRUE` to get the `MEAN` and `SE(MEAN)` columns in the returned table.
- **returnSepct:** A logical value set to `TRUE` to get the `SEPCT` column in the returned table.
- **drop:** A logical value set to `FALSE` indicating that when a single column is returned, it is still represented as a `data.frame` and is not converted to a vector.
- **schoolMergeVarStudent:** A character variable name from the student file used to merge student and school data files. Set to `NULL` by default.
- **schoolMergeVarSchool:** A character variable name from the school file used to merge student and school data files. Set to `NULL` by default.
- **omittedLevels:** A logical value set to `TRUE` indicating that drops those levels of all factor variables that are specified in `edsurvey.data.frame`. Use `print` on an `edsurvey.data.frame` to see the omitted levels.

- **defaultConditions:** A logical value set to `TRUE` that uses the default conditions stored in `edsurvey.data.frame` to subset the data. Use `print` on an `edsurvey.data.frame` to see the default conditions.
- **recode:** A list of lists to recode variables. Defaults to `NULL`. Can be set as `recode = list(var1= list(from=c("a","b","c"), to ="d"))`. See examples using `?getData`.

4. getData

- **data:** Object of class `edsurvey.data.frame`.
- **varnames:** A character vector of variable names that will be returned. When both `varnames` and a `formula` are specified, variables associated with both are returned. Set to `NULL` by default.
- **drop:** A logical value set to `FALSE` indicating that when a single column is returned, it is still represented as a `data.frame` and is not converted to a vector.
- **schoolMergeVarStudent:** A character variable name from the student file used to merge student and school data files. Set to `NULL` by default.
- **schoolMergeVarSchool:** A character variable name from the school file used to merge student and school data files. Set to `NULL` by default.
- **dropUnusedLevels:** A logical value set to `TRUE` that drops unused levels of all factor variables.
- **omittedLevels:** A logical value set to `TRUE` indicating that drops those levels of all factor variables that are specified in `edsurvey.data.frame`. Use `print` on an `edsurvey.data.frame` to see the omitted levels.
- **defaultConditions:** A logical value set to `TRUE` that uses the default conditions stored in `edsurvey.data.frame` to subset the data. Use `print` on an `edsurvey.data.frame` to see the default conditions.
- **formula:** A `formula`. When included, `getData` returns data associated with all variables of the `formula`. When both `varname` and a `formula` are specified, the variables associated with both are returned. Set to `NULL` by default.
- **recode:** A list of lists to recode variables. Defaults to `NULL`. Can be set as `recode = list(var1= list(from=c("a","b","c"), to ="d"))`. See examples using `?getData`.
- **includeNaLabel:** A logical value that when set to `TRUE` returns literal `NA` values. When set to `FALSE` (the default), it returns factor levels coded as `NA`.
- **addAttributes:** A logical value set to `TRUE` that returns a `light.edsurvey.data.frame` for use in calls to other functions that usually would require an `edsurvey.data.frame`.
- **returnJKreplicates:** A logical value indicating if jackknife replicate weights be returned. Defaults to `TRUE`.

5. lm.sdf

- **formula:** Object of class `formula`, potentially with a subject scale or subscale on the left hand side, and variable(s) for tabulation on the right hand side.
- **data:** Object of class `edsurvey.data.frame`.
- **weightVar:** Character indicating the weight variable to use.
- **relevels:** Object of class list. Used to change the contrasts from the default treatment contrasts to treatment contrasts with a chosen omitted group, defaults to `NULL`. See examples using `?lm.sdf`.
- **varMethod:** A character set to "jackknife" or "Taylor" that indicates the variance estimation method to be used.
- **jrrIMax:** Integer indicating the maximum number of plausible values to include when calculating the variance term, defaults to 1.
- **schoolMergeVarStudent:** A character variable name from the student file used to merge student and school data files. Set to `NULL` by default.
- **schoolMergeVarSchool:** A character variable name from the school file used to merge student and school data files. Set to `NULL` by default.

- **omittedLevels:** A logical value set to `TRUE` indicating that drops those levels of all factor variables that are specified in `edsurvey.data.frame`. Use `print` on an `edsurvey.data.frame` to see the omitted levels.
- **defaultConditions:** A logical value set to `TRUE` that uses the default conditions stored in `edsurvey.data.frame` to subset the data. Use `print` on an `edsurvey.data.frame` to see the default conditions.
- **recode:** A list of lists to recode variables. Defaults to `NULL`. Can be set as `recode = list(var1= list(from=c("a","b","c"), to ="d"))`. See examples using `?getData`.