

Using the `getData` Function in EdSurvey

Michael Lee, Paul Bailey, and Ahmad Emad^{*†}

May 4, 2018

The `EdSurvey` package gives users functions to efficiently analyze education survey data. Although the package allows for rudimentary data manipulation and analysis, this vignette shows how to use both `EdSurvey` and base R functions to edit data before processing. By calling the function `getData()`, one can extract a `light.edsurvey.data.frame`: a `data.frame`-like object containing requested variables, weights, and plausible values. This `light.edsurvey.data.frame` can be manipulated in the same manner as other `data.frame` objects but also can be used with packaged `EdSurvey` functions.

Note:

Users who wish to analyze the data with limited memory usage or without making manipulations should consult the vignette titled *Using EdSurvey to Analyze NCES Data: An Illustration of Analyzing NAEP Primer*.¹

This vignette details the following information: First, how to prepare the environment for processing, then how to retrieve the data of interest, followed by ways in which the data can be manipulated in both base R and with `EdSurvey` functions. With this knowledge, a user will be able to fit a unique `light.edsurvey.data.frame` to a summary table and a linear regression model. Two sample workflows will finish the vignette and synthesize the process of using the `EdSurvey` package.

Setting Up the Environment

Before processing begins, load the `EdSurvey` package and the National Assessment of Educational Progress (NAEP) data to be analyzed. The `readNAEP` function will connect to the `EdSurvey` database for analysis by linking to its folder storage location.

```
## Loading required package: lfactors
## lfactors v1.0.4
## EdSurvey v2.0.0
##
## Attaching package: 'EdSurvey'
## The following objects are masked from 'package:base':
##
##   cbind, rbind
library(EdSurvey)
sdf <- readNAEP(filepath = '../Data/file.dat')
```

To follow along with this vignette, load the NAEP Primer data set `M36NT2PM`, assigned to `sdf`, from the package directory using `system.file`:

^{*}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, Qingshu Xie, Jiao Yu, and Ting Zhang for reviewing the `getData` functions and this vignette, as well as Dan Sherman for reviewing this document.

¹The vignette is available online at <https://www.air.org/sites/default/files/EdSurvey.pdf>.

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

This allows access to the NAEP Primer data to demonstrate EdSurvey functions.

Retrieve the Data

Data can be retrieved from the selected file using the `getData` function, which includes several powerful parameters to customize the retrieval of data. The three retrieval methods are (a) calling variable names, (b) providing a formula, and (c) merging files on unique variables. We detail the three methods as follows.

1. Provide Variable Names to `getData()` Function

First, get the names of the weight and other variables that will be used.² For details on specifying and searching for particular arguments from a database, consult the **Getting to Know the Data Format** section in the vignette titled *Using EdSurvey to Analyze NCES Data: An Illustration of Analyzing NAEP Primer*.³ Then you can select the variables and weight(s) you wish to call:

```
gddat <- getData(data = sdf, varnames = c('composite', 'dsex', 'b017451', 'origwt'),  
               addAttributes = TRUE, omittedLevels = FALSE)
```

In this example, `getData` extracts the following:

- two variables, `dsex` and `b017451`
- five plausible values associated with `composite`
- the weight for this data frame: `origwt`

A few important things to note:

1. `addAttributes` is set to `TRUE` so that the object (`gddat`) returned by this call to `getData` can be passed to the EdSurvey package functions. This argument is `FALSE` by default.
2. All jackknife replicate weights are returned automatically (`srwt01` to `srwt62`).
3. `omittedLevels` is set to `FALSE` so that variables with special values (such as multiple entries or NAs) can still be returned by `getData` and manipulated by the user. The default setting (i.e., `omittedLevels = TRUE`) removes these values from factors that are not typically included in regression analysis and cross-tabulation.

2. Extract the Variables From a Formula

The `getData` function can extract variable names embedded in a formula. The arguments `formula = composite ~ dsex + b017451` and `varnames = "origwt"` tell `getData` to extract the necessary subject scale, the outcome variables used in the formula, and the default weight. The `addAttributes` argument is important for use in further functions; setting it to `TRUE` passes the resulting `light.edsurvey.data.frame` to all functions that require an `edsurvey.data.frame`. Setting `defaultConditions = TRUE` uses the default conditions stored in the `edsurvey.data.frame` to subset the data, in this case subsetting the `edsurvey.data.frame` on the reporting sample.⁴

```
gddat <- getData(data = sdf, formula = composite ~ dsex + b017451, varnames = "origwt",  
               addAttributes = TRUE, defaultConditions = TRUE)
```

²Consult `?getData` or the appendix of the vignette titled *Using EdSurvey 2.0.0 to Analyze NAEP Data: An Illustration of Analyzing NAEP Primer* for details on default `getData` arguments.

³View documentation on `searchSDF()`, `showPlausibleValues()`, and `showWeights()`, in particular.

⁴Use `print` to view the default conditions in an `edsurvey.data.frame`.

Note that in the following code, the `head` function is used, focusing on columns 1 through 7. This reveals that we have retrieved the requested variables by viewing the first few rows of the resulting data:

```
head(x = gddat[,1:7])

##      dsex          b017451 mrpcm1 mrpcm2 mrpcm3 mrpcm4 mrpcm5
## 1  Male          Every day 318.01 303.68 296.61 328.97 315.70
## 2 Female    About once a week 288.43 283.93 280.45 290.03 286.23
## 3 Female          Every day 342.72 338.03 329.48 352.46 342.26
## 4  Male          Every day 348.76 321.79 327.87 333.35 327.32
## 6 Female    Once every few weeks 278.44 245.08 263.00 277.50 285.04
## 7  Male    2 or 3 times a week 327.95 338.59 328.07 334.07 320.02
```

Manipulate the Data

Basic manipulation of data is possible without having to use `getData` to extract a `light.edsurvey.data.frame`. Users who wish to analyze the data without making complicated manipulations should consult the vignette titled *Using EdSurvey to Analyze NCES Data: An Illustration of Analyzing NAEP Primer*.

However, more complicated manipulations require extracting data using `getData`. We list two examples here:

The base R function `gsub` allows users to substitute one string for another.⁵ The following step recodes “Every day” to “Seven days a week”:

```
# 1. Recode a Column Based on a String

gddat$b017451 <- gsub(pattern = "Every day", replacement = "Seven days a week",
  x = gddat$b017451)
head(x = gddat$b017451)
```

```
## [1] "Seven days a week"    "About once a week"    "Seven days a week"
## [4] "Seven days a week"    "Once every few weeks" "2 or 3 times a week"
```

The base R function `subset` allows users to subset vectors, matrices, or data frames that meet conditions. In the following example, users create a subsample of students who talk about studies at home (variable `b017451`) “2 or 3 times a week” or “About once a week,” assigned to the object `df`:

```
# 2. Subset the Data Based on a String

df <- subset(gddat, b017451 == "2 or 3 times a week" | b017451 == "About once a week")
head(x = df[,1:7])
```

```
##      dsex          b017451 mrpcm1 mrpcm2 mrpcm3 mrpcm4 mrpcm5
## 2  Female    About once a week 288.43 283.93 280.45 290.03 286.23
## 7   Male    2 or 3 times a week 327.95 338.59 328.07 334.07 320.02
## 8  Female    2 or 3 times a week 275.68 286.68 283.13 280.78 295.56
## 10  Male    2 or 3 times a week 308.04 288.12 298.10 295.60 285.40
## 11 Female    2 or 3 times a week 314.69 291.48 296.68 287.79 298.49
## 12 Female    2 or 3 times a week 318.00 322.98 316.06 318.25 309.46
```

Because the `EdSurvey` package functions accept both value levels and labels, the same subset can be made using value levels:

```
# 2. Subset the Data Based on a String

gddat <- getData(data = sdf, varnames = c("composite", "dsex", "b017451",
  "c052601", "origwt"), addAttributes = TRUE)
```

⁵Use `?function` in the R console to view documentation on base R and `EdSurvey` package functions (e.g., `?gsub` or `?lm.sdf`).

```
df <- subset(gddat, b017451 == 4 | b017451 == 3)
head(x = df[,1:7])
```

```
##      dsex          b017451      c052601 mrpcm1 mrpcm2 mrpcm3 mrpcm4
## 3  Female  About once a week 6 to 10 percent 288.43 283.93 280.45 290.03
## 8   Male  2 or 3 times a week 6 to 10 percent 327.95 338.59 328.07 334.07
## 1  Female  2 or 3 times a week 6 to 10 percent 275.68 286.68 283.13 280.78
## 12  Male  2 or 3 times a week 6 to 10 percent 308.04 288.12 298.10 295.60
## 13 Female  2 or 3 times a week 6 to 10 percent 314.69 291.48 296.68 287.79
## 14 Female  2 or 3 times a week 6 to 10 percent 318.00 322.98 316.06 318.25
```

Use EdSurvey Functions on Unique `light.edsurvey.data.frames`

After manipulating the data, you can use a `light.edsurvey.data.frame` with any EdSurvey function. Most notably, `light.edsurvey.data.frames` can create `edsurveyTables` using `edsurveyTable` and run regressions by the `lm.sdf` function.

edsurveyTable

The following example creates an `edsurveyTable` using the manipulated `light.edsurvey.data.frame` (named `gddat`), the variables `dsex` and `b017451`, the five plausible values for `composite`, and the default weight `origwt`:⁶

```
es2 <- edsurveyTable(formula = composite ~ dsex + b017451,
                     weightVar = "origwt", data = gddat)
```

Table 1: Table es2

dsex	b017451	N	WTD_N	PCT	SE(PCT)	MEAN	SE(MEAN)
Male	Never or hardly ever	2171	2276.820	28.99585	0.7044670	270.8526	1.090086
Male	Once every few weeks	1489	1535.884	19.55985	0.5538779	275.6296	1.357837
Male	About once a week	1293	1339.204	17.05508	0.5278360	281.7165	1.449683
Male	2 or 3 times a week	1424	1454.934	18.52893	0.5158073	284.7212	1.661465
Male	Every day	1203	1245.385	15.86028	0.5824622	277.8021	1.929363
Female	Never or hardly ever	1383	1425.512	18.24810	0.5115641	266.7741	1.555760
Female	Once every few weeks	1419	1454.837	18.62349	0.5134568	271.5970	1.295964
Female	About once a week	1379	1450.724	18.57084	0.5789385	279.3023	1.660139
Female	2 or 3 times a week	1697	1737.825	22.24604	0.5070853	282.8398	1.459509
Female	Every day	1686	1742.940	22.31153	0.6531813	275.7997	1.321104

lm.sdf

To generate a linear model using `light.edsurvey.data.frame`, the included arguments from the previous example, as well as the weight `origwt`, are passed through the `lm.sdf` function:⁷

⁶Consult `?edsurveyTable` or the appendix of the vignette titled *Using EdSurvey to Analyze NCES Data: An Illustration of Analyzing NAEP Primer* for details on default `edsurveyTable` arguments.

⁷Consult `?lm.sdf` or the appendix of the vignette titled *Using EdSurvey to Analyze NCES Data: An Illustration of Analyzing NAEP Primer* for details on default `lm.sdf` arguments.

```
lm2 <- lm.sdf(formula = composite ~ dsex + b017451, weightVar = "origwt", data = gddat)
summary(lm2)
```

```
##
## Formula: composite ~ dsex + b017451
##
## jrrIMax: 1
## Weight variable: 'origwt'
## Variance method: jackknife
## JK replicates: 62
## full data n: 17606
## n used: 15144
##
## Coefficients:
##              coef          se          t    dof Pr(>|t|)
## (Intercept)  270.40708  1.05390 256.57683 51.496 < 2.2e-16
## dsexFemale   -2.92147  0.61554  -4.74615 53.963 1.565e-05
## b017451Once every few weeks  4.68200  1.16792   4.00882 55.188 0.0001848
## b017451About once a week    11.57319  1.26477   9.15045 49.005 3.519e-12
## b0174512 or 3 times a week  14.88024  1.23890  12.01083 77.130 < 2.2e-16
## b017451Every day           7.93104  1.28155   6.18864 50.501 1.074e-07
##
## (Intercept)          ***
## dsexFemale           ***
## b017451Once every few weeks ***
## b017451About once a week    ***
## b0174512 or 3 times a week  ***
## b017451Every day          ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared:  0.0224
```

Contrasts from treatment groups also can be omitted from a linear model by stating the variable name in the `relevels` argument. In this example, values with `dsex="Female"` are withheld from the regression. Use the base R function `summary` to view details about the linear model.

```
lm3 <- lm.sdf(formula = composite ~ dsex + b017451, data = gddat,
              relevels = list(dsex = "Female"))
summary(lm3)
```

```
##
## Formula: composite ~ dsex + b017451
##
## jrrIMax: 1
## Weight variable: 'origwt'
## Variance method: jackknife
## JK replicates: 62
## full data n: 17606
## n used: 15144
##
## Coefficients:
##              coef          se          t    dof Pr(>|t|)
## (Intercept)  267.48561  1.11204 240.53502 65.757 < 2.2e-16
## dsexMale     2.92147  0.61554   4.74615 53.963 1.565e-05
```

```
## b017451Once every few weeks 4.68200 1.16792 4.00882 55.188 0.0001848
## b017451About once a week 11.57319 1.26477 9.15045 49.005 3.519e-12
## b0174512 or 3 times a week 14.88024 1.23890 12.01083 77.130 < 2.2e-16
## b017451Every day 7.93104 1.28155 6.18864 50.501 1.074e-07
##
## (Intercept) ***
## dsexMale ***
## b017451Once every few weeks ***
## b017451About once a week ***
## b0174512 or 3 times a week ***
## b017451Every day ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared: 0.0224
```

cor.sdf

Users might generate a correlation to explore a manipulated `light.edsurvey.data.frame`. The marginal correlation coefficient among plausible values of the subject scales and subscales can be calculated on a `light.edsurvey.data.frame` object `eddat` using the `cor.sdf` function and the Pearson method. In this example, the variable `dsex=="Female"` subsets our `light.edsurvey.data.frame` to calculate the correlation between the subject subscales `num_oper` and `algebra` using the default weight `origwt`:⁸

```
eddat <- getData(data = sdf, varnames = c("num_oper", "algebra", "dsex", 'origwt'),
                addAttributes = TRUE, omittedLevels = FALSE)

eddat <- subset(eddat, dsex == "Female")
cor2 <- cor.sdf(x = "num_oper", y = "algebra", weightVar = "origwt",
               data = eddat, method = "Pearson")

cor2
```

```
## Method: Pearson
## full data n: 17606
## n used: 8429
##
## Correlation: 0.8917132
```

Sample Workflow

The following are two sequences in which the `EdSurvey` package can be implemented to gather information from NAEP data:

Example 1: Recode One Variable

A possible workflow might consist of analyzing student's mathematics performance by major racial/ethnic groups and a student's individualized education program (IEP) status. A sample data manipulation might include recoding the variable for race/ethnicity:

⁸Consult `?cor.sdf` or the appendix of the vignette titled *Using EdSurvey to Analyze NCES Data: An Illustration of Analyzing NAEP Primer* for details on default `cor.sdf` arguments.

```
rsdf <- getData(data = sdf, varnames = c(all.vars(composite ~ sdracem + iep),"origwt"),
  addAttributes = TRUE)
```

Note that `addAttributes = TRUE` so that the object (`rsdf`) returned by this call to `getData` can be passed to the `EdSurvey` package functions. Because the focus of interest is on the performance of major racial groups, some smaller racial groups need to be combined. The variable `sdracem` then is recoded to keep White, Black, Hispanic, and Asian/Pacific Islander values unchanged and combines the remaining students of other racial groups as one group: "Other." Use the base R function `unique` to view details about the recoded variable `sdracem`.

```
rsdf$sdracem <- gsub(pattern = "Amer Ind/Alaska Natv|Other",
  replacement = "Other", x = rsdf$sdracem)
unique(x = rsdf$sdracem)
```

```
## [1] "White"           "Asian/Pacific Island" "Hispanic"
## [4] "Other"            "Black"
```

Now run a regression using the `composite`, the default weight `origwt`, as well as the variables `iep` and the recoded `sdracem`:

```
lm4 <- lm.sdf(formula = composite ~ iep + sdracem, weightVar = "origwt", data = rsdf)
summary(lm4)
```

```
##
## Formula: composite ~ iep + sdracem
##
## jrrIMax: 1
## Weight variable: 'origwt'
## Variance method: jackknife
## JK replicates: 62
## full data n: 17606
## n used: 16907
##
## Coefficients:
##              coef          se          t      dof Pr(>|t|)
## (Intercept)  254.63418   3.26147  78.07337  21.216 < 2.2e-16 ***
## iepNo        37.11369   1.31150  28.29858  23.233 < 2.2e-16 ***
## sdracemBlack -33.38347   3.18841 -10.47026  17.470 5.958e-09 ***
## sdracemHispanic -29.42032   3.56676  -8.24846  21.013 5.003e-08 ***
## sdracemOther  -15.72074   4.31091  -3.64674  36.836 0.0008157 ***
## sdracemWhite  -0.40070   3.18288  -0.12589  14.811 0.9015100
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared:  0.2602
```

Alternatively, this also could be completed within one `getData` call. The `sdracem` variables changed are passed through the `recode` argument from their current values to their new recoded value.

```
eddat <- getData(data = rsdf,
  varnames = c(all.vars(composite ~ sdracem + iep),"origwt"),
  recode = list(sdracem = list(from = c("Amer Ind/Alaska Natv|Other"),
    to = c("Other"))),
  addAttributes = TRUE)
```

```
## Warning in recode.sdf(sdf, recode): When recoding, could not find the
## level(s) "Amer Ind/Alaska Natv|Other" in the variable "sdracem".
```

This produces the same linear model:

```
lm5 <- lm.sdf(formula = composite ~ iep + sdracem, weightVar = "origwt", data = eddat)
summary(lm5)
```

```
##
## Formula: composite ~ iep + sdracem
##
## jrrIMax: 1
## Weight variable: 'origwt'
## Variance method: jackknife
## JK replicates: 62
## full data n: 17606
## n used: 16907
##
## Coefficients:
##              coef          se          t    dof Pr(>|t|)
## (Intercept)  254.63418   3.26147  78.07337 21.216 < 2.2e-16 ***
## iepNo        37.11369   1.31150  28.29858 23.233 < 2.2e-16 ***
## sdracemBlack -33.38347   3.18841 -10.47026 17.470 5.958e-09 ***
## sdracemHispanic -29.42032  3.56676  -8.24846 21.013 5.003e-08 ***
## sdracemOther  -15.72074  4.31091  -3.64674 36.836 0.0008157 ***
## sdracemWhite  -0.40070   3.18288  -0.12589 14.811 0.9015100
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared:  0.2602
```

Example 2: Linear Regression Using Multiple Variables

Another example involves subsetting multiple variables with special values. Let's look at the values for English learners, gender, students with IEPs, and their composite mathematics performance.

```
gddat <- getData(data = sdf,
                 varnames = c(all.vars(composite ~ lep + dsex + iep), "origwt"),
                 addAttributes = TRUE, omittedLevels = FALSE)
```

By setting `omittedLevels = FALSE`, special values are included in the `light.edsurvey.data.frame`. Users can view the unique instances of the variables `lep`, `dsex`, and `iep` by using the base R function `unique`:

```
unique(x = gddat[,c("lep", "dsex", "iep")])
```

```
##      lep  dsex  iep
## 1      No  Male   No
## 2      No Female   No
## 16     Yes  Male   No
## 21     No  Male   Yes
## 29     No Female   Yes
## 65     Yes Female   No
## 140    Yes Female   Yes
## 226    Yes  Male   Yes
## 1403 Omitted  Male <NA>
## 1405     No Female <NA>
## 1419     No  Male <NA>
## 1422 Omitted Female   No
```



```
## 1456    Yes    Male <NA>
## 3622 Omitted    Male    No
```

It is easy to notice that omitted values have been included in the `lep` and `iep` columns. Let's start by recoding the values.

```
gddat=subset(gddat,iep %in% c("No", "Yes"))
gddat=subset(gddat,lep %in% c("No", "Yes"))
unique(x = gddat[,c("lep","dsex","iep")])
```

```
##      lep  dsex iep
## 1     No   Male  No
## 2     No  Female  No
## 16    Yes   Male  No
## 21    No   Male  Yes
## 29    No  Female  Yes
## 65    Yes  Female  No
## 140   Yes  Female  Yes
## 226   Yes   Male  Yes
```

Now that we have finished subsetting the variables, we can run the regression:

```
lm6 <- lm.sdf(composite ~ lep + dsex + iep, weightVar = "origwt", gddat)
summary(lm6)
```

```
##
## Formula: composite ~ lep + dsex + iep
##
## jrrIMax: 1
## Weight variable: 'origwt'
## Variance method: jackknife
## JK replicates: 62
## full data n: 17606
## n used: 16904
##
## Coefficients:
##              coef          se          t    dof Pr(>|t|)
## (Intercept) 211.03272    2.91863   72.30543  20.477 < 2.2e-16 ***
## lepNo       35.80224    2.42207   14.78169  10.911 1.465e-08 ***
## dsexFemale  -4.26358    0.64376   -6.62290  56.950 1.354e-08 ***
## iepNo       37.51960    1.60437   23.38583  21.042 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared: 0.1586
```

Example 3: Linear Regression Using a New Variable

Users can add their own variables to a `light.edsurvey.data.frame` and analyze them with `EdSurvey` functions. In this example, a researcher plans to create a new variable “t08880a” labeled “computer activities” by summing computer-use variables in the Primer data. First, the researcher retrieves the four variables for computer-use, the five plausible values for `composite`, and the default weight `origwt` to create a `light.edsurvey.data.frame`:

```
comp <- getData(data = sdf, varnames = c("composite", "t088801", "t088803",
                                         "t088804", "t088805", "origwt"), addAttributes = TRUE)
```

Then add the new variable (which we will call `t08880a`) to the object `comp`. The base function `sapply` applies a function over a vector—in this case coercing our vector of four variables for computer-use to numeric values using `as.numeric`. This capability is necessary because the `EdSurvey` package stores variables as `lfactors`, where both levels and labels are stored for each value.

```
comp_vars <- c("t088801", "t088803", "t088804", "t088805")
comp[,comp_vars] <- sapply(X = comp[,comp_vars], FUN = as.numeric)
comp$t08880a <- comp$t088801 + comp$t088803 + comp$t088804 + comp$t088805
names(comp)
```

```
## [1] "t088801" "t088803" "t088804" "t088805" "mrpcm1" "mrpcm2" "mrpcm3"
## [8] "mrpcm4" "mrpcm5" "srwt01" "srwt02" "srwt03" "srwt04" "srwt05"
## [15] "srwt06" "srwt07" "srwt08" "srwt09" "srwt10" "srwt11" "srwt12"
## [22] "srwt13" "srwt14" "srwt15" "srwt16" "srwt17" "srwt18" "srwt19"
## [29] "srwt20" "srwt21" "srwt22" "srwt23" "srwt24" "srwt25" "srwt26"
## [36] "srwt27" "srwt28" "srwt29" "srwt30" "srwt31" "srwt32" "srwt33"
## [43] "srwt34" "srwt35" "srwt36" "srwt37" "srwt38" "srwt39" "srwt40"
## [50] "srwt41" "srwt42" "srwt43" "srwt44" "srwt45" "srwt46" "srwt47"
## [57] "srwt48" "srwt49" "srwt50" "srwt51" "srwt52" "srwt53" "srwt54"
## [64] "srwt55" "srwt56" "srwt57" "srwt58" "srwt59" "srwt60" "srwt61"
## [71] "srwt62" "origwt" "t08880a"
```

Now that the computer-use variable has been created, we can run the regression:

```
comp_lm <- lm.sdf(formula = composite ~ t08880a, weightVar = "origwt", data = comp)
summary(comp_lm)
```

```
##
## Formula: composite ~ t08880a
##
## jrrIMax: 1
## Weight variable: 'origwt'
## Variance method: jackknife
## JK replicates: 62
## full data n: 17606
## n used: 14518
##
## Coefficients:
##           coef          se          t    dof Pr(>|t|)
## (Intercept) 264.66851    3.03007  87.34726 26.735 < 2.2e-16 ***
## t08880a      1.41686    0.31283   4.52915 25.740 0.0001189 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared: 0.0104
```

Important Data Manipulation Notes

Memory Usage

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

The base R function `object.size` provides an estimate of the memory that is being used to store an R object. Computations using objects stored in the global environment are markedly costlier to memory than those made directly from the EdSurvey database:

```
object.size(gddat <- getData(data = sdf,
                             varnames = c('composite', 'dsex', 'b017451', 'origwt'),
                             addAttributes = TRUE, omittedLevels = FALSE))
```

```
## 9509584 bytes
```

```
object.size(lm7 <- lm.sdf(formula = composite ~ dsex + b017451,
                           weightVar='origwt', data = gddat))
```

```
## 8589256 bytes
```

```
object.size(lm8 <- lm.sdf(formula = composite ~ dsex + b017451,
                           weightVar='origwt', data = sdf))
```

```
## 4047944 bytes
```

Although a manipulated `light.edsurvey.data.frame` requires nearly 10 MB of working memory to store both the `light.edsurvey.data.frame` and the regression model object (`lm7`), the resulting object of the same computation made directly from the EdSurvey database (`lm8`) holds only 5–7 kB. It is a good practice to remove unnecessary values saved in the global environment; because we have stored many large data objects, let's remove these before moving on.

```
rm(df, gddat, eddat, rsdf)
```

Some operating systems continue to hold the memory usage even after removing an object. R will clean up your global environment automatically, but a forced garbage cleanup also can be employed:

```
gc()
```

Forgetting to Include a Column Variable

The EdSurvey package will give a warning when a column is missing when creating a summary table or when running regression:

```
gddat <- getData(data = sdf,
                 varnames = c(all.vars(composite ~ lep + dsex + iep), "origwt"),
                 addAttributes = TRUE, omittedLevels = FALSE)
lm9 <- lm.sdf(formula = composite ~ lep + dsex + iep + b017451, data = gddat)
```

```
## Using default weight variable 'origwt'
```

```
## Error in getData(sdf, c(all.vars(formula), wgt), ..., includeNaLabel = TRUE)
## The following variable names are required for this call
## and are not on the incoming data 'b017451'.
```

The solution is simple: Edit the call to `getData` to include the variable and re-run the linear model.

```
gddat <- getData(data = sdf,
                 varnames = c(all.vars(composite ~ lep + dsex + iep + b017451), "origwt"),
                 addAttributes = TRUE, omittedLevels = FALSE)
lm9 <- lm.sdf(formula = composite ~ lep + dsex + iep + b017451, data = gddat)
lm9
```

```
##                (Intercept)                lepNo
```

##	207.356989	35.278034
##	dsexFemale	iepNo
##	-5.285498	36.170641
##	b017451Once every few weeks	b017451About once a week
##	3.254744	9.210189
##	b0174512 or 3 times a week	b017451Every day
##	12.659496	6.808825

Reference

Lee, M. D., Bailey, P. D., Emad, A., Zhang, T., Nguyen, T. M., & Yu, J. (2018). *Using EdSurvey to analyze NCES data: An illustration of analyzing NAEP primer*. Washington, DC: American Institutes for Research. Retrieved from <https://www.air.org/sites/default/files/EdSurvey.pdf>