
Analyzing NAEP and TIMSS Data with Direct Estimation using the R packages EdSurvey and Dire

Presenters: Paul Bailey, Ting Zhang, Michael Lee & Sinan Yavuz

April 2022

Workshop Goal

Provide participants with an overview of the plausible values and direct estimation methods used to analyze national and international large-scale assessment data using the R package **EdSurvey** and **Dire**.

Follow along in [edsurvey_part2_Script.R](#)

Outline of EdSurvey Workshop - Part 2

1. Descriptive statistics
2. Hands-on practice
3. Direct estimation with EdSurvey and Dire
4. Hands-on practice

Data Processing

- First, load the **EdSurvey** package and read in the data

```
# to load the package
```

```
library(EdSurvey)
```

```
library(Dire)
```

NAEP Primer:

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

Summary statistics

Summary statistics

summary2() produces both weighted and unweighted descriptive statistics for a variable. **summary2()** takes four following arguments in order:

- **data** : an **EdSurvey** object.
- **variable** : name of the variable you want to produce statistics on.
- **weightVar** : name of the weight variable; or **NULL** if users want to produce unweighted statistics.
- **omittedLevels** : if **TRUE**, the function will remove omitted levels for the specified variable before producing descriptive statistics. If **FALSE**, the function will include omitted levels in the output statistics.

Summary statistics

For a continuous variable (i.e., composite Math score):

- For NAEP data and other datasets that have default weight variable, `summary2` produces weighted statistics by default. If specified, `variable` is a plausible value and weight option is selected, `summary2` statistics account for both plausible value pooling and weighting.

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

```
## Estimates are weighted using the weight variable 'origwt'
```

```
##   Variable      N Weighted N   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.      SD NA's Zero weights
## 1 composite 16915   16932.46 126.11 251.9626 277.4784 275.8892 301.1827 404.184 36.5713    0          0
```

Summary statistics

For a continuous variable (i.e., composite Math score):

- By specifying `weightVar = NULL`, the function prints out unweighted descriptive statistics for `variable`, or each plausible value if `variable` is a plausible value name.

```
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

Summary statistics

For a categorical variable (i.e., frequency of students talking about studies at home):

- By default, `omittedLevels` is set to `FALSE`. That is, the function includes omitted levels of the variable `b017451` in the output statistics.

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

```
## Estimates are weighted using the 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

Summary statistics

For a categorical variable (i.e., frequency of students talking about studies at home):

- By specifying `omittedLevels = TRUE`, the function removes omitted levels out of the output statistics.

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

```
## Estimates are weighted using the 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

Cross tabulation

edsurveyTable(): creates a summary table of outcome and categorical variables. There are 3 important arguments as followed:

- **formula**: typically written as **a ~ b + c**, in which:
 - **a**: a continuous variable (optional) that the function will return weighted mean on.
 - **b** and **c**: categorical variable(s) that the function will run cross-tabulation on; multiple crosstab categorical variables can be separated using **+** symbol.
- **data**: an **EdSurvey** object
- **pctAggregationLevel**: a numeric value (i.e., 0, 1, 2) that indicates the level of aggregation in the cross-tabulation result's percentage column.

Cross tabulation

- Summary table of NAEP composite mathematics performance scale scores (**composite**) of 8th grade students by two student factors:
 - **dsex**: gender
 - **b017451**: frequency of talk about studies at home

```
es1 <- edsurveyTable(composite ~ dsex + b017451, data = sdf)
```

- **pctAggregationLevel** is by default set to **NULL** (or **1**). That is, the **PCT** column adds up to 100 within each level of the first categorical variable **dsex**.

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

Cross tabulation

- By specifying `pctAggregationLevel = 0`, the `PCT` column adds up to 100 across the entire sample.

```
es2 <- edsurveyTable(composite ~ dsex + b017451, data = sdf, pctAggr
```

dsex	b017451	N	WTD_N	PCT	SE(PCT)	MEAN	SE(MEAN)
Male	Never or hardly ever	2350	2434.844	14.553095	0.3738531	270.8243	1.057078
Male	Once every few weeks	1603	1638.745	9.794803	0.2651368	275.0807	1.305922
Male	About once a week	1384	1423.312	8.507154	0.2770233	281.5612	1.409587
Male	2 or 3 times a week	1535	1563.393	9.344421	0.2670298	284.9066	1.546072
Male	Every day	1291	1332.890	7.966700	0.3000579	277.2597	1.795784
Female	Never or hardly ever	1487	1517.609	9.070768	0.2984443	266.7897	1.519020
Female	Once every few weeks	1544	1552.149	9.277216	0.2498498	271.2255	1.205528
Female	About once a week	1469	1514.403	9.051606	0.2899668	278.7502	1.719778
Female	2 or 3 times a week	1827	1862.502	11.132198	0.2552321	282.7765	1.404107
Female	Every day	1841	1890.918	11.302039	0.3497982	275.4628	1.219439

Cross tabulation

- Related Documentation - [EdSurvey-LaTeXtables.pdf](#), Chap 5.4.1, [EdSurvey Book](#)

Self-Reflection - edsurveyTable

Ask yourself: Use EdSurvey functions to create a summary table using `edsurveyTable` with these parameters:

- overall math performance across subscales (`composite`)
- variable that has to do with IEP status
- variable that has to do with number of books at home

Self-Reflection - edsurveyTable

Scenario Result:

```
edexercise <- edsurveyTable(composite ~ iep + b013801,  
                             weightVar = 'origwt', data = sdf)
```

```
edexercise
```

```
##  
## Formula: composite ~ iep + b013801  
##  
## Plausible values: 5  
## jrrIMax: 1  
## Weight variable: 'origwt'  
## Variance method: jackknife  
## JK replicates: 62  
## full data n: 17606  
## n used: 16351  
##  
##  
## Summary Table:  
##   iep b013801    N    WTD_N    PCT    SE(PCT)    MEAN    SE(MEAN)  
##   Yes    0-10   304   297.1972  17.33406  1.0388812  226.1623  2.3075125  
##   Yes   11-25   430   429.6252  25.05794  1.4034976  231.8103  2.3796081  
##   Yes  26-100   517   530.9539  30.96795  1.5297784  249.2306  2.4682667  
##   Yes   >100   457   456.7507  26.64004  1.6556494  257.6787  2.8205193  
##   No    0-10  1720  1890.3037  12.56502  0.4765198  257.6975  1.2861579  
##   No   11-25  2936  3170.9954  21.07789  0.5632689  266.0401  0.9908671  
##   No  26-100  5330  5350.4978  35.56524  0.6242526  281.5820  0.8305656  
##   No   >100  4657  4632.3807  30.79185  0.8511616  296.2606  1.0533164
```


Linear Regression with PVs

Linear Regression with PVs - `lm.sdf()`

`lm.sdf()` : fits a linear model formula using sampling weights and a variance estimation method. The format is:

```
myfit <- lm.sdf(formula, data, weightVar, varMethod, relevels)
```

- `formula`: model to be fit.
- `data`: data frame containing the data to be used in fitting the model.
- `weightVar`: indicates the weight variable to use.
- `varMethod`: the variance estimation method (Jackknife or Taylor series) with the Jackknife as the default.
- `relevels`: is used when the user wants to change the reference level of a categorical variable.

Linear Regression with PVs - `lm.sdf()`

The resulting object (`myfit` in this case) is a list containing extensive information about the fitted model.

Formula notation is typically written as:

```
Y ~ X1 + X2 + ... + Xk
```

- The `~` separates the response variable on the left from the predictor variables on the right.
- The `+` sign separates the predictor variables.

Regressions with PVs - lm.sdf()

$$\text{Composite} = \beta_0 +$$

$$\beta_1 \text{Freq. of talk about studies at home} + \epsilon$$

```
lm1 <- lm.sdf(composite ~ dsex + b013801,  
              weightVar = 'origwt', data = sdf)  
summary(lm1)
```

```
##  
## Formula: composite ~ dsex + b013801  
##  
## Weight variable: 'origwt'  
## Variance method: jackknife  
## JK replicates: 62  
## Plausible values: 5  
## jrrIMax: 1  
## full data n: 17606  
## n used: 16359  
##  
## Coefficients:  
##
```

Self-Reflection - lm.sdf

Ask yourself: Use EdSurvey functions to perform a regression with multiple predictors using `lm.sdf` using these parameters:

- overall math performance across subscales (`composite`)
- variable that has to do with computers at home
- variable that has to do with language other than English spoken in home

Self-Reflection - lm.sdf

Scenario Result:

```
lmexercise2 <- lm.sdf(composite ~ b017101 + b018201,
                      weightVar = 'origwt', data = sdf)

summary(lmexercise2)

##
## Formula: composite ~ b017101 + b018201
##
## Weight variable: 'origwt'
## Variance method: jackknife
## JK replicates: 62
## Plausible values: 5
## jrrIMax: 1
## full data n: 17606
## n used: 15884
##
## Coefficients:
##
##              coef          se          t    dof  Pr(>|t|)
## (Intercept)    281.95112    0.80871 348.64281 43.827 < 2.2e-16 ***
## b017101No      -22.44306    1.36521 -16.43932 42.935 < 2.2e-16 ***
## b018201Once in a while    0.63672    0.90717   0.70188 61.423   0.4854
## b018201Half the time     -7.32985    1.58448  -4.62604 50.514 2.624e-05 ***
## b018201All or most of time -12.61417    1.27458  -9.89675 29.860 6.121e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared: 0.0658
```

Direct estimation with EdSurvey and Dire

Direct Estimation with EdSurvey - mml.sdf

- The `mml.sdf` function in `EdSurvey` enables marginal maximum likelihood estimation (MML) of linear models for NAEP and TIMMS.
- `mml.sdf` was designed to automate weighting and complex design calculation with simple steps. The direct estimation can be done in `EdSurvey` with a simplified operation via the `mml.sdf` function
- Item parameters, scoring, scaling and weighting information were grabbed from existing NAEP documents, and multiple procedures were streamlined and calculated behind the scene automatically.
- Plausible values (PVs) can be drawing from the latent distribution with `drawPVs`.

Direct Estimation with EdSurvey - mml.sdf

```
mmlA <- mml.sdf(composite ~ dsex + b013801, data=sdf)
```

```
## Warning in mml.sdf(composite ~ dsex + b013801, data = sdf): These items were in the assessment, but not in your data: m0732c1,  
## m073601, m092401, m092601, m141301, m141901, m012231, m012431, m013331, m019201, m020901, m021001, m051701, m052501, m067001,  
## m073001, m073101, m073301, m091901, m092201, m140401, m140501, m140601, m140701, m140801, m140901, m141001, m141101, m141201,  
## m141401, m141501, m141601, m141701, and m141801
```

- **formula:** this is the conditioning model
 - **dsex:** student gender
 - **b013801:** books in the home
- **data:** the data set

Direct Estimation with EdSurvey - mml.sdf

```
summary(mmlA)
```

```
## Call:
## mml.sdf(formula = composite ~ dsex + b013801, data = sdf)
## Summary Call:
## summary.mml.sdf(object = mmlA)
##
## Summary:
##           Estimate      StdErr  t.value
## (Intercept)   253.39797    1.36948 185.0325
## dsexFemale    -4.03996    0.67148  -6.0165
## b01380111-25   10.59273    1.48387   7.1386
## b01380126-100  27.50233    1.37309  20.0295
## b013801>100   42.69997    1.35193  31.5845
## b013801Omitted 18.32500    3.79930   4.8233
## b013801Multiple 3.61187   19.04067   0.1897
##
## Residual Variance Estimate:
##           Estimate StdErr
## Population SD 34.72184    NA
##
## Convergence = converged
## Iterations = 48, 46, 45, 42, 40
```

Scoring

- shows how each item it changed from raw data to a score

```
scard <- getAttributes(sdf, "scoreCard")  
subset(scard, key=="m107101")
```

```
##           key  answer score  
## 1117 m107101      A      0  
## 1118 m107101      B      0  
## 1119 m107101      C      0  
## 1120 m107101    D *      1  
## 1121 m107101      E      0  
## 1122 m107101 Omitted      8  
## 1124 m107101 Multiple      8
```

IRT Parameters

- shows how each item it changed from raw data to a score

```
dparam <- getAttributes(sdf, "dichotParamTab")  
head(dparam)
```

##	ItemID	subtest	slope	difficulty	guessing	D	missingValue	missingCode
##	7657 m012231	number	0.51	-3.36	0.17	1.7	c	8
##	7658 m012431	number	0.78	-0.08	0.10	1.7	c	8
##	7659 m013131	data	0.64	0.91	0.00	1.7	c	8
##	7660 m013331	measurement	1.09	-1.21	0.26	1.7	c	8
##	7661 m013431	number	0.94	-0.24	0.11	1.7	c	8
##	7662 m013531	number	0.92	1.03	0.11	1.7	c	8

Test Data

```
getAttributes(sdf, "testData")
```

##	subtest	location	scale	subtestWeight
## 229	algebra	281.79	35.64	0.30
## 228	data	281.91	40.37	0.15
## 227	geometry	275.64	33.22	0.20
## 226	measurement	275.86	46.57	0.15
## 225	number	277.11	37.78	0.20

- Shows scaling information by construct

Draw PVs with EdSurvey - drawPVs

- The `drawPVs` requires two data sources:
 - an `edsurvey.data.frame` (in this case, the primer data `sdf`), and
 - a fit from a call to `mml.sdf` or a `summary` of `mml.sdf` call (i.e., `mmlA` from the example).
- The `npv` argument specifies the number of PVs for the scale.

```
sdf2 <- drawPVs(sdf, mmlA, npv=20L)
```

```
## Warning in (function (data, varnames = NULL, drop = FALSE, dropUnusedLevels = TRUE, : U  
## are multiples of the label 'Correct'.
```

```
## Warning in (function (data, varnames = NULL, drop = FALSE, dropUnusedLevels = TRUE, : U  
## are multiples of the label 'Correct'.
```

```
## Calculating posterior distribution for construct algebra (1 of 5)
```

```
## Calculating posterior distribution for construct data (2 of 5)
```

Use new PVs with EdSurvey - drawPVs

- The new plausible values variables end with `_dire`
- these can be used to fit a regression with any combination of conditioning variables
- variables must be included in the conditioning model (`mml.sdf` call) for the estimator to be unbiased

```
lm2 <- lm.sdf(composite_dire ~ b013801, data=sdf2)
summary(lm2)
```

```
##
```

```
## Formula: composite_dire ~ b013801
```

```
##
```

```
## Weight variable: 'origwt'
```

```
## Variance method: jackknife
```


Use new PVs with EdSurvey - drawPVs

- Variables not included in the conditioning model (`mml.sdf` call) will have biased estimates
- this includes interaction terms

```
lm1a <- lm.sdf(composite ~ b018201, data=sdf2)
summary(lm1a)$coefmat
```

##	coef	se	t	dof	Pr(> t)
## (Intercept)	279.3510325	0.9015424	309.8589995	32.41016	0.000000e+00
## b018201Once in a while	0.7718548	0.9499216	0.8125458	60.62978	4.196576e-01
## b018201Half the time	-9.0098638	1.6034921	-5.6189012	37.47070	1.981582e-06
## b018201All or most of time	-14.5362514	1.3103503	-11.0934085	27.26864	1.294259e-11

```
lm1b <- lm.sdf(composite_dire ~ b018201, data=sdf2)
summary(lm1b)$coefmat
```

##	coef	se	t	dof	Pr(> t)
## (Intercept)	278.463321	0.6742498	412.997246	43.57473	0.000000e+00
## b018201Once in a while	1.360532	1.0774220	1.262766	61.48771	2.114412e-01
## b018201Half the time	-6.630013	1.6339818	-4.057581	50.63910	1.716528e-04
## b018201All or most of time	-10.439672	1.2572012	-8.303899	55.75191	2.531460e-11

Self-Reflection - mml.sdf

Ask yourself: Use EdSurvey functions to perform Direct Estimation with multiple predictors using `mml.sdf` using these parameters:

- algebra math performance across subscales (`composite`)
- variable that has to do with attendance/absence
- variable(s) that has to do with effort on the test

Self-Reflection - mml.sdf

Scenario Result:

```
mmlExercisel <- mml.sdf(algebra ~ b018101 + m815401 + m815501, data = sdf)

## Warning in mml.sdf(algebra ~ b018101 + m815401 + m815501, data = sdf): These items were in the assessment, but not in your data
## m0732c1, m073601, m092401, m092601, m141301, m141901, m012231, m012431, m013331, m019201, m020901, m021001, m051701, m051801,
## m067001, m073001, m073101, m073301, m091901, m092201, m140401, m140501, m140601, m140701, m140801, m140901, m141001, m141101,
## m141201, m141401, m141501, m141601, m141701, and m141801

summary(mmlExercisel)

## Call:
## mml.sdf(formula = algebra ~ b018101 + m815401 + m815501, data = sdf)
## Summary Call:
## summary.mml.sdf(object = mmlExercisel)
""
```

Data Synthesis Example

Direct Estimation with Dire

The `Dire` package enables MML linear model and PVs generation for assessment data.

It is flexible for data linking or data frame expanding (e.g., adding Principle Component variables).

Multiple steps required:

- Process the data, link datasets if needed
- Prepare necessary arguments for item, location, scale and weight parameters
- Run MML regression
- Summary of the fitted results with the Taylor series method
- Draw PVs

Data Processing and Exploration

The `NAEPDataSimulation` package includes a simulated NAEP-like dataset. This dataset was generated by using the NAEP 2015 Mathematics 8 grade item parameters and block design.

```
require(NCESDataLike)
```

```
## Loading required package: NCESDataLike
```

```
#Simulated NAEP-like
```

```
sN1 <- readNAEP(system.file("extdata/data", "M46NT2PM.dat",  
                           package = "NCESDataLike"))
```

```
cd <- showCodebook(sN1)
```

Data Linking

- We will merge a data from **The US Census Bureau** and the simulated NAEP-like data.
- However, the possibilities are endless as long as there is a common variable to combine datasets from different resources.
- From the selection of data provided by The US Census Bureau we choose the following variable from **American Community Survey (ACS) 5-Year Data (2009-2020) dataset**:

B06011_001E: Median income in the past 12 months (in 2019 inflation-adjusted dollars) by place of birth in the United States

- The dataset and all other variables can be found here.
<https://api.census.gov/data/2019/acs/acs5/variables.html>

Data Linking - Download the external dataset

Because the dataset is publicly available it can be directly downloaded as follows:

```
url="https://api.census.gov/data/2019/acs/acs5?get=NAME,B06011_001E&
temp <- tempfile()
download.file(url , temp)
AcsDt <- read.table(temp, sep="," ,header = TRUE)
unlink(temp)
head(AcsDt)
```

```
##      X..NAME B06011_001E state zip.code.tabulation.area. X
## 1 [ZCTA5 01001      36257    25                01001] NA
## 2 [ZCTA5 01002      17716    25                01002] NA
## 3 [ZCTA5 01003       4054    25                01003] NA
## 4 [ZCTA5 01005      39944    25                01005] NA
## 5 [ZCTA5 01007      43144    25                01007] NA
## 6 [ZCTA5 01008      41458    25                01008] NA
```


Data Linking - Clean up the ACS data

- Before merging, we need to clean the brackets and replace the missing values with `NA`. Additionally, we will divide the median income variable by 10,000 for quicker convergence and more readable coefficients.

```
#remove the opening bracket on the first column
AcsDt[,1] <- gsub(pattern="\[", replacement="", x= AcsDt[,1])
#remove the last empty column
AcsDt$X <- NULL
#remove the bracket from the last column
AcsDt[,ncol(AcsDt)] <- gsub(pattern="\]",
                           replacement="", x= AcsDt[,ncol(AcsDt)])
AcsDt$B06011_001E[AcsDt$B06011_001E==-6666666666] <- NA
AcsDt$B06011_001E <- as.numeric(AcsDt$B06011_001E)

## Warning: NAs introduced by coercion

AcsDt$B06011_001E_10K <- AcsDt$B06011_001E/10000
```

Data Linking - The ACS data

- Here is the cleaned ACS data

```
head(AcsDt)
```

```
##           X..NAME B06011_001E state zip.code.tabulation.area. B06011_001E_10K
## 1 ZCTA5 01001          36257    25                01001          3.6257
## 2 ZCTA5 01002          17716    25                01002          1.7716
## 3 ZCTA5 01003           4054    25                01003          0.4054
## 4 ZCTA5 01005          39944    25                01005          3.9944
## 5 ZCTA5 01007          43144    25                01007          4.3144
## 6 ZCTA5 01008          41458    25                01008          4.1458
```

```
tail(AcsDt)
```

```
##           X..NAME B06011_001E state zip.code.tabulation.area. B06011_001E_10K
## 33115 ZCTA5 00736           NA    72                00736           NA
## 33116 ZCTA5 00907           NA    72                00907           NA
## 33117 ZCTA5 00786           NA    72                00786           NA
## 33118 ZCTA5 00694           NA    72                00694           NA
## 33119 ZCTA5 00631           NA    72                00631           NA
## 33120 ZCTA5 00926           NA    72                00926           NA
```

Data Linking - Clean up and merge

- We will merge this data with the simulated data by using the zip code variable.

```
items <- cd$variableName[grep("item m", cd$Labels)]
simdf <- EdSurvey::getData(sNl, varnames = c("idvar", "dsex", "pared",
                                             "repgrpl", "jkunit", "o
                                             items), omittedLevels =
linkedData <- merge(simdf, AcsDt, by.x = "zip",
                    by.y = "zip.code.tabulation.area.", all.x = TRUE)
linkAtt <- rebindAttributes(linkedData, sNl)
```

MML Model - mml

- `mml` function can estimate a linear model via marginal maximum likelihood.

```
fit <- mml.sdf(composite ~ B06011_001E_10K + dsex + pared,  
              data = linkAtt, weightVar='origwt', idVar="idvar")
```

```
## Warning in mml.sdf(composite ~ B06011_001E_10K + dsex + pared, data = linkAtt, : m152602, m2372c1, m3498c1 did not have  
## parameters for the max score and will be taken out of analysis.  
## Warning in mml.sdf(composite ~ B06011_001E_10K + dsex + pared, data = linkAtt, : Removing 350 rows with NAs from analysis
```

MML Model - Summary of the model

summary(fit)

```
## Warning in getVarTaylor(object = obj, H_B_prime = H_B_prime0[block[[i]], : Of the 25 strata, 1 strata have only one PSU. All
## strata with only one PSU are excluded from variance estimation. See the "singletonFix" argument for other options.

## Warning in getVarTaylor(object = obj, H_B_prime = H_B_prime0[block[[i]], : Of the 25 strata, 1 strata have only one PSU. All
## strata with only one PSU are excluded from variance estimation. See the "singletonFix" argument for other options.

## Warning in getVarTaylor(object = obj, H_B_prime = H_B_prime0[block[[i]], : Of the 25 strata, 1 strata have only one PSU. All
## strata with only one PSU are excluded from variance estimation. See the "singletonFix" argument for other options.

## Warning in getVarTaylor(object = obj, H_B_prime = H_B_prime0[block[[i]], : Of the 25 strata, 1 strata have only one PSU. All
## strata with only one PSU are excluded from variance estimation. See the "singletonFix" argument for other options.

## Warning in getVarTaylor(object = obj, H_B_prime = H_B_prime0[block[[i]], : Of the 25 strata, 1 strata have only one PSU. All
## strata with only one PSU are excluded from variance estimation. See the "singletonFix" argument for other options.

## Call:
## mml.sdf(formula = composite ~ B06011_001E_10K + dsex + pared,
##       data = linkAtt, weightVar = "origwt", idVar = "idvar")
## Summary Call:
## summary.mml.sdf(object = fit)
##
## Summary:
##               Estimate      StdErr t.value
## (Intercept)    279.97052    3.02943  92.4170
## B06011_001E_10K      0.91175    0.91408   0.9975
## dsexFemale        0.73909    0.81260   0.9095
## paredGraduated HS   -0.25162    1.95133  -0.1289
## paredSome ed after HS  1.93552    1.58346   1.2223
## paredGraduated college 1.57703    1.82919   0.8621
## paredI don't know    0.79383    1.63351   0.4860
## paredOmitted        1.29527    1.46602   0.8835
## paredMultiple       0.53092    1.99222   0.2665
```

Drawing PVs

- Three of the generated items didn't get enough responses. So we removed them from the score card.

```
'%!in%' <- function(x,y)!('%in%'(x,y))
fit$sCard <- fit$sCard[fit$sCard$key %!in%
                        c("m152602", "m2372c1", "m3498c1"),]
PVs <- drawPVs(linkAtt, fit, npv = 20L)

## Calculating posterior distribution for construct algebra (1 of 5)
## Calculating posterior distribution for construct data (2 of 5)
## Calculating posterior distribution for construct geometry (3 of 5)
## Calculating posterior distribution for construct measurement (4 of 5)
## Calculating posterior distribution for construct number (5 of 5)
## Calculating posterior correlation between construct algebra and data (1 of 10)
## Calculating posterior correlation between construct algebra and geometry (2 of 10)
## Calculating posterior correlation between construct algebra and measurement (3 of 10)
## Calculating posterior correlation between construct algebra and number (4 of 10)
## Calculating posterior correlation between construct data and geometry (5 of 10)
## Calculating posterior correlation between construct data and measurement (6 of 10)
## Calculating posterior correlation between construct data and number (7 of 10)
## Calculating posterior correlation between construct geometry and measurement (8 of 10)
## Calculating posterior correlation between construct geometry and number (9 of 10)
## Calculating posterior correlation between construct measurement and number (10 of 10)
## Generating plausible values.
```

Drawing PVs

Finally, here is the drawn plausible values. We can see the first plausible values of each subscale and the composite score.

```
PVs[1:5,c("algebra_dire1", "composite_dire1")]
```

##	algebra_dire1	composite_dire1
## 1	292.2637	287.8699
## 2	280.7014	293.9316
## 3	314.1876	296.7939
## 4	236.8094	233.3035
## 5	258.3922	249.1563

Self Reflection

1. Select a variables from ACS
2. Write your url
3. Download your ACS data
4. Clean up and scale your variable as needed
5. Merge with the simulated NAEP-like dataset
6. Build your own model (`composite ~ ACSvariable + simulatedDataVariable`)
7. Summarize the results
8. Draw new PVs

Self Reflection - Step 1: Select a variables

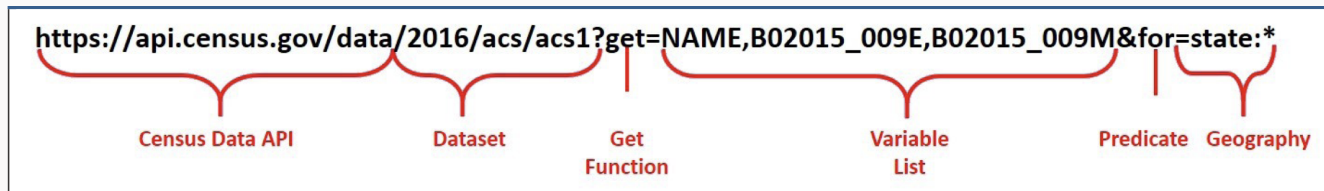
- Select a variable of your interest below:
 - B01001_001E (population)
 - B10063_002E : Household with grandparents living with grandchildren:
 - B19001_017E : Past 12 months income \$200,000 or more
 - B27001_014E : Male 26 to 34 years with no health insurance coverage
 - B28010_007E : No Computer
 - B28011_008E : No Internet access
 - C16001_002E : Speak only English

For more variables: **from**

<https://api.census.gov/data/2019/acs/acs5/variables.html>

Self Reflection - Step 2: Write your **url**

- Add your variable(s) to your *url* link



Source and more information

- Here is the set of links:

```
url1 = "https://api.census.gov/data/2019/acs/acs5?get=NAME,B01001_00
url2 = "https://api.census.gov/data/2019/acs/acs5?get=NAME,B01001_00
url3 = "https://api.census.gov/data/2019/acs/acs5?get=NAME,B01001_00
url4 = "https://api.census.gov/data/2019/acs/acs5?get=NAME,B01001_00
url5 = "https://api.census.gov/data/2019/acs/acs5?get=NAME,B01001_00
url6 = "https://api.census.gov/data/2019/acs/acs5?get=NAME,B01001_00
```

Self Reflection - Step 3. Download your ACS data

- Change your *url* name below (if you selected a different variable)

```
temp <- tempfile()
download.file(url6 , temp)
AcsDt6 <- read.table(temp, sep="," ,header = TRUE)
```

```
## Warning in scan(file = file, what = what, sep = sep, quote = quote, dec = dec, : number of items read is not a multiple of the
## number of columns
```

```
unlink(temp)
head(AcsDt6)
```

```
##      X..NAME B01001_001E C16001_002E state zip.code.tabulation.area. X
## 1 [ZCTA5 25245          600          600    54                25245] NA
## 2 [ZCTA5 25268          964          834    54                25268] NA
## 3 [ZCTA5 25286         1700         1667    54                25286] NA
## 4 [ZCTA5 25303         6764         6109    54                25303] NA
## 5 [ZCTA5 25311        10964        10090    54                25311] NA
## 6 [ZCTA5 25419        11062         9830    54                25419] NA
```

Self Reflection - Step 4. Clean your ACS data and scale

```
AcsDt6[,1] <- gsub(pattern="\\[", replacement="", x= AcsDt6[,1])
AcsDt6$X <- NULL
AcsDt6[,ncol(AcsDt6)] <- gsub(pattern="\\]", replacement="",
                                x= AcsDt6[,ncol(AcsDt6)])

summary(AcsDt6)
```

##	X..NAME	B01001_001E	C16001_002E	state	zip.code.tabulation.area.
##	Length:33120	Min. : 0.0	Min. : 0	Min. : 1.00	Length:33120
##	Class :character	1st Qu.: 705.8	1st Qu.: 610	1st Qu.:18.00	Class :character
##	Mode :character	Median : 2801.0	Median : 2365	Median :30.00	Mode :character
##		Mean : 9903.3	Mean : 7221	Mean :29.89	
##		3rd Qu.: 13475.2	3rd Qu.:10196	3rd Qu.:42.00	
##		Max. :128294.0	Max. :77013	Max. :72.00	

Self Reflection - Step 4. Clean your ACS data and scale

```
AcsDt6$EngSpeakers <- AcsDt6$C16001_002E/AcsDt6$B01001_001E  
summary(AcsDt6$EngSpeakers)
```

##	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
##	0.0000	0.8105	0.8945	0.8380	0.9325	1.0000	344

Self Reflection - Step 5. Merge with the simulated NAEP-like dataset

- Here we merge two datasets.

```
linkedData6 <- merge(simdf, AcsDt6, by.x = "zip",  
                    by.y = "zip.code.tabulation.area.",  
                    all.x = TRUE)
```

- Don't forget to rebind attributes of the `edsurvey.data.frame`.

```
linkAtt6 <- rebindAttributes(linkedData6, sN1)
```

Self Reflection - Step 6. Build your own model

- Here is an example: `composite ~ ACSvariable + simulatedDataVariable`

```
fitSR <- mml.sdf(composite ~ EngSpeakers + dsex + pared,  
                 data = linkAtt6, weightVar='origwt', idVar="idvar")
```

```
## Warning in mml.sdf(composite ~ EngSpeakers + dsex + pared, data = linkAtt6, : m152602, m2372c1, m3498c1 did not have enough  
## parameters for the max score and will be taken out of analysis.  
## Warning in mml.sdf(composite ~ EngSpeakers + dsex + pared, data = linkAtt6, : Removing 250 rows with NAs from analysis.
```

Self Reflection - Step 7. Summarize the results

```
summary(fitSR)
```

```
## Call:
## mml.sdf(formula = composite ~ EngSpeakers + dsex + pared, data = linkAtt6,
##   weightVar = "origwt", idVar = "idvar")
## Summary Call:
## summary.mml.sdf(object = fitSR)
##
## Summary:
##               Estimate      StdErr t.value
## (Intercept)    282.49142    5.31272  53.1727
## EngSpeakers      0.27328    5.88009   0.0465
## dsexFemale       0.62971    0.80396   0.7833
## paredGraduated HS -0.53478    2.00344  -0.2669
## paredSome ed after HS  1.72625    1.63548   1.0555
## paredGraduated college  1.61946    1.76090   0.9197
## paredI don't know   0.81701    1.63138   0.5008
## paredOmitted       1.39149    1.41960   0.9802
## paredMultiple      0.47635    1.93305   0.2464
##
## Residual Variance Estimate:
##               Estimate StdErr
## Population SD  39.1455    NA
```


Self Reflection - Step 8. Draw new PVs

```
fitSR$sCard <- fitSR$sCard[fitSR$sCard$key %!in%  
                           c("m152602", "m2372c1", "m3498c1"),]  
pvSR <- drawPVs(linkAtt6, fitSR, npv= 20L)
```

```
## Calculating posterior distribution for construct algebra (1 of 5)  
## Calculating posterior distribution for construct data (2 of 5)  
## Calculating posterior distribution for construct geometry (3 of 5)  
## Calculating posterior distribution for construct measurement (4 of 5)  
## Calculating posterior distribution for construct number (5 of 5)  
## Calculating posterior correlation between construct algebra and data (1 of 10)  
## Calculating posterior correlation between construct algebra and geometry (2 of 10)  
## Calculating posterior correlation between construct algebra and measurement (3 of 10)  
## Calculating posterior correlation between construct algebra and number (4 of 10)  
## Calculating posterior correlation between construct data and geometry (5 of 10)  
## Calculating posterior correlation between construct data and measurement (6 of 10)  
## Calculating posterior correlation between construct data and number (7 of 10)  
## Calculating posterior correlation between construct geometry and measurement (8 of 10)  
## Calculating posterior correlation between construct geometry and number (9 of 10)  
## Calculating posterior correlation between construct measurement and number (10 of 10)  
## Generating plausible values.
```

Wrap Up

Learning EdSurvey

- Reading vignettes provided in training materials

```
vignette("introduction", package="EdSurvey")
```

- R help

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

- [EdSurvey eBook](#)
- [EdSurvey Website](#)
- [EdSurvey Github](#)
- [NAEP Data Training workshop](#)

Under development

- Package is still under development
 - Subsequent releases of the EdSurvey package will provide additional functionality for NAEP linking errors and direct estimation.
- Your feedback is important to us!

Contact Information

EdSurvey Package Help

- EdSurvey.help@air.org

EdSurvey Package Help on NCES.ed.gov

- <http://nces.ed.gov/nationsreportcard/contactus.aspx>

Ting Zhang

- tzhang@air.org

Paul Bailey

- pbailey@air.org

Emmanuel Sikali

- Emmanuel.Sikali@ed.gov