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

NAEP
NATIONAL ASSESSMENT OF EDUCATIONAL PROGRESS

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

NAEP
NATIONAL ASSESSMENT OF EDUCATIONAL PROGRESS

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

Cross tabulation

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

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

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

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

NAEP
NATIONAL ASSESSMENT OF EDUCATIONAL PROGRESS

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:
##                    coef         se         t    dof  Pr(>|t|)
```

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

NAEP
NATIONAL ASSESSMENT OF EDUCATIONAL PROGRESS

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.sdf`.

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: m141901,
## m0732cl, m092601, m092401, m141301, m073601, m140501, m140901, m141501, m052501, m067001, m051701, m140701, m141601, m092201,
## m140601, m141201, m141401, m141701, m021001, m020901, m140401, m140801, m141001, m013331, m073301, m019201, m141101, m141801,
## m012231, m073001, m073101, m012431, and m091901
```

- **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)
```

Draw PVs with EdSurvey `drawPVs.sdf`

- The `drawPVs.sdf` 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, : Updating labels on 'm144901' because there
## are multiples of the label 'Correct'.
## Warning in (function (data, varnames = NULL, drop = FALSE, dropUnusedLevels = TRUE, : Updating labels on 'm145101' because there
## are multiples of the label 'Correct'.
## 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)
```

Use new PVs with EdSurvey

`drawPVs.sdf`

- 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
## JK replicates: 62
## Plausible values: 20
## jrrIMax: 1
## full data n: 17606
## n used: 16359
##
## Coefficients:
##                                                    Estimate Std. Error t value Pr(>|t|)
```

Use new PVs with EdSurvey

`drawPVs.sdf`

- 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)                278.307508 0.6401081 434.782025 42.78501 0.000000e+00
## b018201Once in a while       1.484049 0.9804717   1.513607 57.16122 1.356356e-01
## b018201Half the time        -6.299519 1.7496515  -3.600442 43.61601 8.078577e-04
## b018201All or most of time -10.527413 1.4245251  -7.390121 42.48531 3.778315e-09
```

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:

```
mmlExercise1 <- 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:
## m141901, m0732cl, m092601, m092401, m141301, m073601, m140501, m140901, m141501, m052501, m067001, m051701, m140701, m141601,
## m092201, m140601, m141201, m141401, m141701, m021001, m020901, m140401, m140801, m141001, m013331, m073301, m019201, m141101,
## m141801, m012231, m073001, m073101, m012431, and m091901
```

```
summary(mmlExercise1)
```

```
## Call:
## mml.sdf(formula = algebra ~ b018101 + m815401 + m815501, data = sdf)
## Summary Call:
## summary.mml.sdf(object = mmlExercise1)
##
## Summary:
##                            Estimate   StdErr  t.value
## (Intercept)                289.6188   1.6197 178.8093
## b0181011-2 days             -6.6006   1.1096  -5.9488
## b0181013-4 days            -15.9040   1.5078 -10.5478
## b0181015-10 days           -19.4088   1.7553 -11.0572
## b018101More than 10 days   -35.7046   3.3311 -10.7186
## b018101Omitted             -11.2818   5.3001  -2.1286
## b018101Multiple            -23.5953  12.7314  -1.8533
## m815401As hard as others   -10.9665   1.0572 -10.3731
## m815401Harder than others  -13.5255   1.6639  -8.1287
## m815401Much harder         -25.8999   3.1016  -8.3504
## m815401Omitted              -8.9882   6.5721  -1.3676
## m815401Multiple              4.7606  14.0996   0.3376
## m815501Tried about as hard   6.0804   1.1804   5.1509
## m815501Tried harder         -4.1539   1.7166  -2.4198
```

Data Synthesis example

Direct Estimation with Dire

NAEP
NATIONAL ASSESSMENT OF EDUCATIONAL PROGRESS

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(NAEPDataSimulation)
simNAEP <- NAEPDataSimulation::NAEPlikeDt
dim(simNAEP)
```

```
## [1] 1000  346
```

```
simNAEP[1:6,1:6]
```

```
##      idVar schtyp2              drace10         ell3             pared            b017451
## 1 10000001  Public   Asian, not Hispanic  Formerly ELL       I don't know          Every day
## 2 10000002  Public  Native Am/Pac Island       Omitted           Multiple   2-3 times a week
## 3 10000003  Public   Asian, not Hispanic       Omitted       I don't know           Multiple
## 4 10000004  Public  Hispanic of any race  Formerly ELL           Multiple           Multiple
## 5 10000005  Public  Hispanic of any race  Formerly ELL           Multiple  About once a week
## 6 10000006  Public  >1 race,not Hispanic  Formerly ELL  Graduated college          Every day
```

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&for=zip
temp <- tempfile()
download.file(url , temp)
AcsDt <- 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(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 readible coefficients.

```
#remove the opening bracket on the first column
AcsDt[,1] <- gsub(pattern="\\[", replacement="", x= AcsDt[,1])
#remove the last empty column
AcsDt$X <- NULL
#remove the bracket from the last column
AcsDt[,ncol(AcsDt)] <- gsub(pattern="\\]", replacement="", x= AcsDt[,ncol(A
AcsDt$B06011_001E[AcsDt$B06011_001E==-666666666] <- NA
AcsDt$B06011_001E <- as.numeric(AcsDt$B06011_001E)
```

```
## Warning: NAs introduced by coercion
```

Data Linking - The ACS data

- Here is the cleaned ACS data

```
head(AcsDt)
```

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

```
tail(AcsDt)
```

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

Data Linking - Clean up and merge

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

```
linkedData <- merge(simNAEP, AcsDt, by.x = "zip",
                    by.y = "zip.code.tabulation.area.", all.x = TRUE)
linkedData$B06011_001E_10K <- linkedData$B06011_001E/10000
linkedData[1:6,1:8]
```

```
##     zip    idVar schtyp2              drace10         ell3             pared              b017451   m820901
## 1 02053 10000321  Public  Native Am/Pac Island      Omitted  Did not finish HS Never or hardly ever  Multiple
## 2 02053 10000322  Public   Asian, not Hispanic  Formerly ELL           Multiple            Every day     Agree
## 3 02053 10000323  Public  Afric Amer, not Hisp       Not ELL  Graduated college             Multiple  Multiple
## 4 02053 10000324  Public  Afric Amer, not Hisp  Formerly ELL  Graduated college     2-3 times a week   Omitted
## 5 02053 10000325  Public  Hispanic of any race       Not ELL  Graduated college              Omitted   Omitted
## 6 02053 10000326  Public  Hispanic of any race       Not ELL       I don't know    About once a week  Multiple
```

```
linkedData[1:6,344:350]
```

```
##   measurement_dire20 number_dire20 composite_dire20     X..NAME B06011_001E state B06011_001E_10K
## 1           413.5530      320.5881         350.2271 ZCTA5 02053       56219    25          5.6219
## 2           343.2232      294.2513         290.8688 ZCTA5 02053       56219    25          5.6219
## 3           279.1026      349.1320         301.5277 ZCTA5 02053       56219    25          5.6219
## 4           187.6923      337.9166         304.8722 ZCTA5 02053       56219    25          5.6219
## 5           298.0790      280.7946         266.2336 ZCTA5 02053       56219    25          5.6219
## 6           321.1388      253.9735         287.4320 ZCTA5 02053       56219    25          5.6219
```

MML Model Preparation

1. Gather item parameters
   - use `NAEPirtparams::parameters` and `naepParamTabs`
2. Provide the test scale information
   - use `NAEPirtparams::transformations`
3. Reshape student data to a long format
   - use `reshape`

MML Model Preparation - Item Parameters

The most convenient way to acquire NAEP item parameters is to call `NAEPirtparams` package. From this package item parameters can be gathered as follows:

```
require(NAEPirtparams)
param <- NAEPirtparams::parameters
item_par <- param[param$level == 8 & param$subject == "Mathematics" & par
head(item_par)
```

```
##          source level levelType  NAEPid assessmentCode accommodations subtest     subject year       a        b       c d1 d2 d3
## 18043 data-file     8     grade m350201       National                algebra Mathematics 2015 0.72419 -0.08700 0.12483 NA NA NA
## 18047 data-file     8     grade m351501       National                algebra Mathematics 2015 0.90420  0.36616 0.11865 NA NA NA
## 18050 data-file     8     grade m348801       National                algebra Mathematics 2015 0.98045  0.41474 0.22152 NA NA NA
## 18054 data-file     8     grade m151701       National                algebra Mathematics 2015 1.25033 -0.03616 0.22043 NA NA NA
## 18056 data-file     8     grade m151901       National                algebra Mathematics 2015 0.72176  0.21992 0.05703 NA NA NA
## 18058 data-file     8     grade m152101       National                algebra Mathematics 2015 0.82808  0.73585 0.19510 NA NA NA
##       d4 d5
## 18043 NA NA
## 18047 NA NA
```

MML Model Preparation - Item Parameters 2

```
paramTabs <- naepParamTabs(item_par)
polyParamTab <- paramTabs$polyParamTab
dichotParamTab <- paramTabs$dichotParamTab
dichotParamTab$test <- 'composite'
polyParamTab$test <- 'composite'
polyParamTab$scorePoints <- apply(polyParamTab[,c('d1','d2','d3','d4','d5
head(polyParamTab)
```

```
##        ItemID subtest   slope itemLocation       d1       d2       d3       d4 d5   D missingValue missingCode      test
## 19448 m233601 algebra 0.40052     -0.96127 -1.99458  1.99458       NA       NA NA 1.7            c           8 composite
## 19454 m152602 algebra 0.76739      0.40144  1.64744 -2.50488  0.85744       NA NA 1.7            c           8 composite
## 19459 m352301 algebra 0.63122     -0.30693 -1.76434  1.76434       NA       NA NA 1.7            c           8 composite
## 19461 m355301 algebra 0.32615     -1.69379 -2.65159  2.65159       NA       NA NA 1.7            c           8 composite
## 19464 m237901 algebra 0.53773     -0.08510  0.09204 -0.09204       NA       NA NA 1.7            c           8 composite
## 19467 m2372cl algebra 0.53084     -0.14531  0.38116  1.04697 -0.99900 -0.42914 NA 1.7            c           8 composite
##       scorePoints
## 19448           2
```

MML Model Preparation - Scaling Arguments

Another important piece of information is the location, scale and weight parameters of the each subscale of Mathematics subject. This information can also be received from using `NAEPirtparams` package as follows

```
transf <- NAEPirtparams::transformations
transFilter <- transf[transf$level== 8 & transf$year== 2015 & transf$subjec
head(transFilter)
```

```
##        source level assessmentCode accommodations     subtest     subject year   scale location subtestWeight
## 462 data-file     8       National                    algebra Mathematics 2015 38.3117 289.9699          0.30
## 461 data-file     8       National                       data Mathematics 2015 41.2000 285.6097          0.15
## 460 data-file     8       National                   geometry Mathematics 2015 33.2916 282.1575          0.20
## 459 data-file     8       National                measurement Mathematics 2015 45.4775 281.4212          0.15
## 458 data-file     8       National                     number Mathematics 2015 35.7337 281.1218          0.20
```

MML Model Preparation - Scaling Arguments 2

- Note that, because of the selected "Mathematics" subject has subscales the user should add the combined test name, which we use "composite".

```
testScale <- transFilter[,c('subtest','location','scale', 'subtestWeight')]
testScale <- testScale[!is.na(testScale$subtestWeight),]
testScale$test <- 'composite'
testScale
```

```
##         subtest location   scale subtestWeight      test
## 462     algebra 289.9699 38.3117          0.30 composite
## 461        data 285.6097 41.2000          0.15 composite
## 460    geometry 282.1575 33.2916          0.20 composite
## 459 measurement 281.4212 45.4775          0.15 composite
## 458      number 281.1218 35.7337          0.20 composite
```

MML Model Prep - Student Responses

- After forming the `dichotParamTab`, `polyParamTab` and `testScale`, the final step is the preparation of a "long" formatted item responses.
- In the simulated data item responses are already provided, however these are in a "wide" format. These item responses can be reshaped using the following lines:

```
itemNames <- c(dichotParamTab$ItemID, polyParamTab$ItemID)
stuItems <- simNAEP[,c("idVar", itemNames)]
stuItems <- reshape(simNAEP[,c("idVar", itemNames)], idvar = "idVar", direc
                    timevar = "key", times = itemNames,
                    varying = itemNames)
```

MML Model Prep - Student Responses

- After forming the `dichotParamTab`, `polyParamTab` and `testScale`, the final step is the preparation of a "long" formatted item responses.
- In the simulated data item responses are already provided, however these are in a "wide" format. These item responses can be reshaped using the following lines:

```
head(stuItems)
```

```
##                    idVar     key score
## 10000001.m350201 10000001 m350201    NA
## 10000002.m350201 10000002 m350201     0
## 10000003.m350201 10000003 m350201    NA
## 10000004.m350201 10000004 m350201    NA
## 10000005.m350201 10000005 m350201    NA
## 10000006.m350201 10000006 m350201    NA
```

MML Model - `mml`

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

```
fit <- mml(composite ~  B06011_001E_10K + dsex + pared,
          stuItems = stuItems,
          stuDat = linkedData,
          idVar = "idVar",
          dichotParamTab = dichotParamTab,
          polyParamTab =  polyParamTab,
          testScale = testScale,
          strataVar="repgrp1", PSUVar="jkunit",
          weightVar = "origwt",
          fast = TRUE,
          multiCore = FALSE,
          verbose=2)
```

MML Model - Summary of the model

.codeScroll25[

```
summary(fit)
```

```
## Warning in getVarTaylor(object = obj, H_B_prime = H_B_prime0[block[[i]], : Of the 7 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 7 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 7 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 7 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 7 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(formula = composite ~ B06011_001E_10K + dsex + pared, stuItems = stuItems,
##     stuDat = linkedData, idVar = "idVar", dichotParamTab = dichotParamTab,
##     polyParamTab = polyParamTab, testScale = testScale, weightVar = "origwt",
##     multiCore = FALSE, strataVar = "repgrp1", PSUVar = "jkunit",
##     fast = TRUE, verbose = 2)
## Summary Call:
## summary.mmlCompositeMeans(object = fit)
##
## Summary:
##                          Estimate    StdErr t.value
## (Intercept)             281.99895   8.23542 34.2422
## B06011_001E_10K          -0.16618   1.58184 -0.1051
## dsex Female               0.76614   4.06602  0.1884
## pared Graduated HS        1.97577   7.12334  0.2774
## pared Some ed after HS   -3.92339   7.61297 -0.5154
```

Drawing PVs

```
PVs <- drawPVs(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$data[1:5, 1:7]
```

```
##          id algebra_dire1 data_dire1 geometry_dire1 measurement_dire1 number_dire1 composite_dire1
## 1  10000001      296.9079   233.9566       289.3078          386.1770     330.2497        306.0039
## 21 10000002      276.2170   192.1293       271.2700          197.4266     190.0973        233.5719
## 41 10000003      329.2267   334.8914       315.2246          259.1249     301.8601        311.2874
## 61 10000004      299.9391   266.9001       300.0631          250.3687     337.1283        295.0104
## 81 10000005      280.9615   295.9477       316.6059          303.2775     270.7084        291.6351
```

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. Prepare your data for `Dire`
7. Build your own model (`composite ~ ACSvariable + simulatedDataVariable`)
8. Estimate your regression model
9. Summarize the results

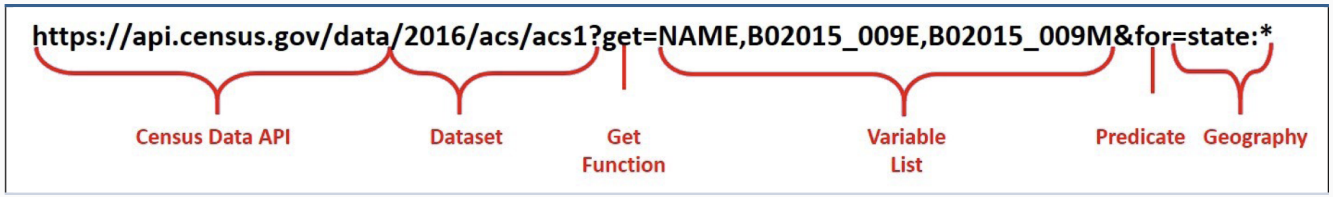
Self Reflection - Step 1: Select a variables

- 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_001E,B100
url2 = "https://api.census.gov/data/2019/acs/acs5?get=NAME,B01001_001E,B190
url3 = "https://api.census.gov/data/2019/acs/acs5?get=NAME,B01001_001E,B270
url4 = "https://api.census.gov/data/2019/acs/acs5?get=NAME,B01001_001E,B280
url5 = "https://api.census.gov/data/2019/acs/acs5?get=NAME,B01001_001E,B280
url6 = "https://api.census.gov/data/2019/acs/acs5?get=NAME,B01001_001E,C160
```

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)
AcsDt <- 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(AcsDt)
```

```
##        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

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

```
##    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

```
AcsDt$EngSpeakers <- AcsDt$C16001_002E/AcsDt$B01001_001E
summary(AcsDt$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

```
linkedData <- merge(simNAEP, AcsDt, by.x = "zip",
                    by.y = "zip.code.tabulation.area.", all.x = TRUE)
dim(linkedData)
```

```
## [1] 1000  351
```

```
linkedData[1:6,1:8]
```

```
##     zip    idVar schtyp2               drace10          ell3              pared              b017451   m820901
## 1 02053 10000321  Public  Native Am/Pac Island       Omitted  Did not finish HS Never or hardly ever  Multiple
## 2 02053 10000322  Public   Asian, not Hispanic  Formerly ELL           Multiple            Every day     Agree
## 3 02053 10000323  Public  Afric Amer, not Hisp       Not ELL  Graduated college             Multiple  Multiple
## 4 02053 10000324  Public  Afric Amer, not Hisp  Formerly ELL  Graduated college     2-3 times a week   Omitted
## 5 02053 10000325  Public  Hispanic of any race       Not ELL  Graduated college             Omitted   Omitted
## 6 02053 10000326  Public  Hispanic of any race       Not ELL       I don't know    About once a week  Multiple
```

```
linkedData[1:6,344:350]
```

```
##   measurement_dire20 number_dire20 composite_dire20     X..NAME B01001_001E C16001_002E state
## 1           413.5530      320.5881         350.2271 ZCTA5 02053       13325       11781    25
## 2           343.2232      294.2513         290.8688 ZCTA5 02053       13325       11781    25
## 3           279.1026      349.1320         301.5277 ZCTA5 02053       13325       11781    25
## 4           187.6923      337.9166         304.8722 ZCTA5 02053       13325       11781    25
## 5           298.0790      280.7946         266.2336 ZCTA5 02053       13325       11781    25
## 6           321.1388      253.9735         287.4320 ZCTA5 02053       13325       11781    25
```

Self Reflection - Step 6. Prepare your data for `Dire`

```
require(NAEPirtparams)
require(NAEPDataSimulation)
param <- NAEPirtparams::parameters
item_par <- param[param$level == 8 & param$subject == "Mathematics" & par
polyParamTab <- naepParamTabs(item_par)$polyParamTab
dichotParamTab <- naepParamTabs(item_par)$dichotParamTab
polyParamTab$scorePoints <- apply(polyParamTab[,c('d1','d2','d3','d4','d5
transf <- NAEPirtparams::transformations
transFilter <- transf[transf$level== 8 & transf$year== 2015 & transf$subj
testScale <- transFilter[,c('subtest','location','scale', 'subtestWeight'
testScale$test <- polyParamTab$test <- dichotParamTab$test <- 'composite'
itemNames <- c(dichotParamTab$ItemID, polyParamTab$ItemID)
stuItems <- simNAEP[,c("idVar", itemNames)]
```

Self Reflection - Step 6. Build your own model

```
composite ~ ACSvariable + simulatedDataVariable
```

```
fit <- mml(composite ~  EngSpeakers + dsex,
             stuItems = stuItems, stuDat = linkedData,
             idVar = "idVar", dichotParamTab = dichotParamTab,
             polyParamTab =  polyParamTab,
             testScale = testScale,
             strataVar="repgrp1", PSUVar="jkunit",
             weightVar = "origwt", verbose=2)
```

```
## Estimating construct "algebra"
## Calculating likelihood function.
## design matrix condition number = 122.024
## Initial optimization with Quasi-Newton.
## iter   10 value 6903738.004832
## final  value 6903738.004832
```

Self Reflection - Step 7. Summarize the results

.codeScroll25[

```r
summary(fit)
```

```
## Warning in getVarTaylor(object = obj, H_B_prime = H_B_prime0[block[[i]], : Of the 7 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 7 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 7 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 7 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 7 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(formula = composite ~ EngSpeakers + dsex, stuItems = stuItems,
##     stuDat = linkedData, idVar = "idVar", dichotParamTab = dichotParamTab,
##     polyParamTab = polyParamTab, testScale = testScale, weightVar = "origwt",
##     strataVar = "repgrp1", PSUVar = "jkunit", verbose = 2)
## Summary Call:
## summary.mmlCompositeMeans(object = fit)
##
## Summary:
##             Estimate   StdErr t.value
## (Intercept) 282.2449   9.3318 30.2455
## EngSpeakers  -1.0047  10.9230 -0.0920
```

Wrap Up

NAEP
NATIONAL ASSESSMENT OF EDUCATIONAL PROGRESS

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