STRATEGIC DATA PROJECT

CONNECT: DATA LINKING GUIDE FOR HUMAN CAPITAL

SDP TOOLKIT

FOR EFFECTIVE DATA USE IN EDUCATION AGENCIES

www.gse.harvard.edu/sdp/toolkit

Toolkit Documents

An Introduction to the SDP Toolkit for Effective Data Use

Identify: Data Specification Guide

Clean: Data Building Guide for Human Capital

Connect: Data Linking Guide for Human Capital

Analyze: Human Capital Analysis Guide

Adopt: Coding Style Guide

SDP Stata Glossary

VERSION: 1.0

Last Modified: April 4th, 2014

Connect: SDP Data Linking Guide for Human Capital

Now that you have identified and cleaned data, you will merge it together to create an analysis file.

Purpose

Connect links the data elements that have been processed in the preceding tasks of **Clean** into two related analysis files, one at the student-teacher-year level and another at the teacher-year level. These files will allow you to execute analyses inspired by the SDP Human Capital Diagnostic to examine patterns of teacher recruitment, placement, development, evaluation, and retention.

After completing **Connect**, you will have:

- produced a student-teacher-year file, `Student_Teacher_Year_Analysis,` and teacher-year file, `Teacher_Year_Analysis`,
- merged disparate data files to create these analysis files in support of **Analyze**, and
- generated a series of additional variables for later use in **Analyze**.

Note: This guide references **Identify** and requires output from **Clean**. To move through **Connect**, you should review these stages of the toolkit.

Data and Structure

Connect consists of four major steps to build `Student_Teacher_Year_Analysis` and `Teacher_Year_Analysis` files that include a measure of teacher effectiveness.

1. **Merge** together student, teacher, class, and school information using the output files from **Clean** to provide a complete picture of class enrollment for students.
2. **Restrict** the sample to tested grades and core subjects with students who have a single teacher and classes of plausible size, and ultimately produce a file unique by student and school year.
3. **Generate** key variables required for the `Teacher_Year_Analysis` files, such as whether a teacher is a new hire, late hire, novice, or veteran.
4. **Understand** how to incorporate one possible measure of teacher of effectiveness, value-added estimates, into your analysis. Then **Merge** these estimates or a measure of your own choosing onto the `Student_Teacher_Year_Analysis` and `Teacher_Year_Analysis` files.

Note: Step 4 does not teach you to produce value-added estimates yourself. We encourage you you to use the measures of teacher effectiveness used by your own agency, such as principal evaluations, student surveys, student growth percentiles, or some combination of student learning measures. For the purposes of following this toolkit, the sample data includes value-added estimates that have already been calculated for you.

At the end of Step 4, you will have all the variables necessary to complete all sections of Analyze using the `Student_Teacher_Year_Analysis` and `Teacher_Year_Analysis files.`

To summarize, **Connect** consists of four major steps to build the `Student_Teacher_Year_Analysis` and `Teacher_Year_Analysis` file s from files from **Clean**, denoted by the suffix `_Clean.dta` in the leftmost column below.

Files needed in **Connect**:	Step in Connect:	Output file name:

`Core_Courses_Clean.dta`
`Student_Class_Enrollment_Clean.dta`
`Student_Test_Scores_Clean.dta`
`Student_School_Year_Clean.dta`
`Student_Attributes_Clean.dta`
`Staff_School_Year_Clean.dta`
`Staff_Attributes_Clean.dta`
`Staff_Certifications_Clean.dta`
`School_Clean.dta`
	1. Merge	`Connect_Step1.dta`

`Connect_Step1.dta`	**2. Restrict**	`Connect_Step2.dta`

`Connect_Step1.dta`
`Connect_Step2.dta`
`School_Clean.dta`
	3. Generate	`Teacher_Year_Analysis.dta`
		`Student_Teacher_Year_Analysis.dta`

`Teacher_Year_Analysis.dta` `Connect_TEM.dta` (or your own teacher effectiveness data)	**4. Understand and Merge**	`Teacher_Year_Analysis.dta` (now including teacher effectiveness data)

Throughout **Connect** the term "merge" indicates that two files will be linked. Merging allows you to combine datasets horizontally and add new columns (or variables) from one dataset to another based on identifier(s) present in both. To merge, you must start with a dataset "loaded" in memory.

At the end of **Connect**, you will have generated the additional variables (shaded in **green**) necessary for the `Student_Teacher_Year_Analysis` and `Teacher_Year_Analysis` files. A preview of these files is below:

Teacher_Year_Analysis		
tid	race_ethnicity	🟩 t_latehire
school_year	birth_date	🟩 t_novice
school_code	t_male	🟩 t_veteran
school_name	t_certification_pathway	🟩 t_veteran_newhire
elementary	t_race_ethnicity	🟩 t_novice_ever
middle	t_birth_date	🟩 t_stay
high	t_'raceeth'	🟩 t_transfer
alternative	t_adv_degree	🟩 t_other_agency_job
school_lvl	t_certification_code1	🟩 t_leave
t_job_code	t_certification_code2	🟩 t_retire
t_experience	t_certification_code3	🟩 tre_'subj'
t_degree	t_esl	🟩 current_tre_'subj'
t_is_teacher	t_nbct	🟩 curr2year_tre_'subj'
t_hire_date	t_sped	🟩 tre_'subj'_'yr'
t_termination_date	t_newhire	🟩 tre_'subj'_'yr_range'
male		🟩 school_poverty_quartile

Notes to Condensed Variable List
`subj' = math, ela
`yr' = 2008, 2009, 2010, 2011
`yr_range' = 2008and2009, 2009and2010, 2010and2011
`raceeth' = afam, asian, hisp, naam, white, mult, racemiss

Student_Teacher_Year_Analysis			
sid	t_degree	tre_'subj'_'yr'	_CLstd_scaled_score_'subj'_tm1
school_year	t_adv_degree	tre_'subj'_'yr_range'	_CLstd_scaled_score_'subj'_tm1_sd
grade_level	t_certification_code1	s_iep	_CLnumber_students_'subj'
cid_'subj'	t_certification_code2	s_gifted	_CLpct_missing_std_'subj'_tm1
school_code	t_certification_code3	s_'subgroup'	_COmean_s_ever_iep
school_name	t_esl	s_race_ethnicity	_COmean_s_ever_gifted
school_poverty_quartile	t_nbct	s_'raceeth'	_COmean_s_'subgroup'
elementary	t_sped	s_'lunchstatus'	_COmean_s_'raceeth'
middle	t_newhire	scaled_score_'subj'	_COmean_s_'lunchstatus'
high	t_latehire	std_scaled_score_'subj'	_COstd_scaled_score_'subj'_tm1
alternative	t_novice	scaled_score_'subj'_tm1	_COstd_scaled_score_'subj'_tm1_sd
school_lvl	t_veteran	std_scaled_score_'subj'_tm1	_COnumber_students
tid_'subj'	t_veteran_newhire	qrt_'subj'	_COpct_missing_std_'subj'_tm1
t_birth_date	t_novice_ever	qrt_'subj'_tm1	male
t_race_ethnicity	t_stay	language_version_'subj'	race_ethnicity
t_'raceeth'	t_transfer	language_version_'subj'_tm1	certification_pathway
t_is_teacher	t_other_agency_job	_CLmean_s_ever_iep_'subj'	t_male
t_job_code	t_leave	_CLmean_s_ever_gifted_'subj'	t_certification_pathway
t_hire_date	tre_'subj'	_CLmean_s_'subgroup'_'subj'	t_retire
t_termination_date	current_tre_'subj'	_CLmean_s_'raceeth"_'subj'	
t_experience	curr2year_tre_'subj'	_CLmean_s_'lunchstatus'_'subj'	

Notes to Condensed Variable List

`subj' = math, ela
`yr' = 2008, 2009, 2010, 2011
`yr_range' = 2008and2009, 2009and2010, 2010and2011
`raceeth' = afam, asian, hisp, naam, white, mult, racemiss
`lunchstatus' = fulllunch, reducedlunch, freelunch, misslunch
`subgroup' = ell, retained, move_nonstruct, move_struct, absence_high, absence_miss, male

Step Components

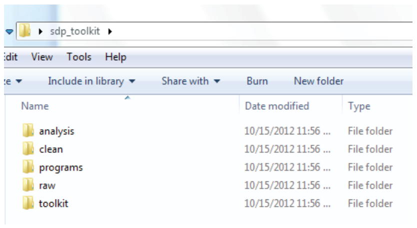
For each step, you will find the following:

- **Purpose:** an overview of each step;
- **Files needed:** data elements or files required to complete each step; and
- **After this step:** an overview of "output" generated by each step.

Also, throughout **Connect**, you will find Stata code to explain each of the substeps. Code appears in blue boxes, like the sample below:

```
foreach subj in math ela {
  egen qrt_`subj' = rowmax(qrt_*_`subj'_*)
}

drop qrt_*_*_*
```

Infrastructure



Name	Date modified	Type
analysis	10/15/2012 11:56 ...	File folder
clean	10/15/2012 11:56 ...	File folder
programs	10/15/2012 11:56 ...	File folder
raw	10/15/2012 11:56 ...	File folder
toolkit	10/15/2012 11:56 ...	File folder

In the infrastructure folder you unzipped from **www.gse.harvard.edu/sdp/toolkit**, look for the files in the **clean** folder and the file `Connect.do` in the **programs** folder. The files in the **clean** folder are those you produced via **Clean** and that we have also provided for you. The do files provide a shell for you to fill in the four steps of **Connect**. Doing so will allow you to produce the `Student_Teacher_Year_Analysis` and `Teacher_Year_Analysis` files, which will be saved to the **analysis** folder.

As always, if you would like additional support from the friendly SDP team, please email us at **sdp@gse.harvard.edu**.

STEP 1: Merge all files together

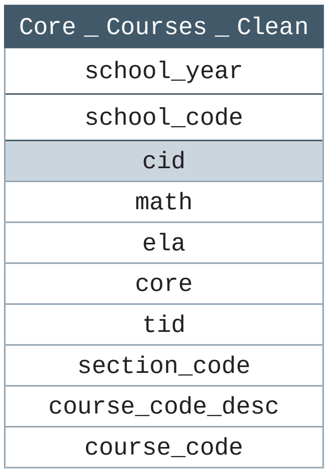
Purpose: Merge together output files from **Clean** to provide a complete picture of class enrollment for students.

Files needed: `Core_Courses_Clean.dta`, `Student_Class_Enrollment_Clean.dta`, `Student_Test_Scores_Clean.dta`, `Student_School_Year_Clean.dta`, `Student_Attributes_Clean.dta`, `Staff_School_Year_Clean.dta`, `Staff_Attributes_Clean.dta`, and `Staff_Certifications_Clean.dta`, `School_Clean.dta` in **Clean**

After this step, you will have merged together your data from **Clean** to form a complete picture of class enrollment in your agency.

```
if $step_1_merge == 1 {
```

1.1 Load the Core_Courses_Clean output file and ensure its uniqueness by class ID (cid).



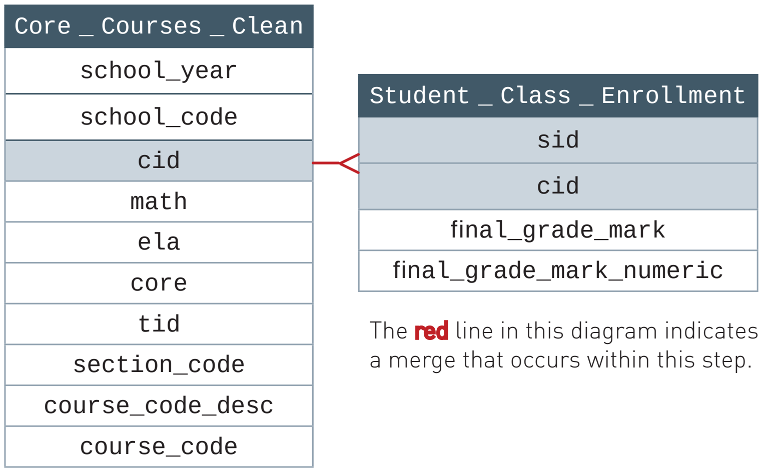
Note the structure of the file. Thanks to Student Task 4, the file is unique by `cid` (highlighted in blue).

```
use ${clean}/Core_Courses_Clean.dta, clear

isid cid
```

1.2 Merge the Student_Class_Enrollment_Clean output file using the cid as an identifier.

Since a class in `Core_Courses_Clean` will have multiple students in `Student_Class_Enrollment`, merge these files with a 1:m (one-to-many) relationship.

Keep only the student IDs (`sid`) and `cid` from the student enrollment file by using the `keepusing` option in the `merge` command.



The **red** line in this diagram indicates a merge that occurs within this step.

```
merge 1:m cid using "${clean}/Student_
Class_Enrollment_Clean.dta", gen(_m_class_
enrollment) keepusing(sid cid)
```

Drop duplicate observations and ensure the result's uniqueness by `sid` and `cid`.

```
duplicates drop
isid sid cid
```

1. Merge

1.3 Load Student_Test_Scores_Clean. Create a quartile variable that corresponds to each test subject. Create variables for prior-year test results.

Preserve your current working file and load `Student_Test_Scores_Clean`, dropping composite scores, which are not used in the later parts of the toolkit.

```
preserve

    use "${clean}/Student_Test_Scores_Clean.dta", clear
    drop *composite
```

Student _ Test _ Scores _ Clean
sid
school_year
grade_level
scaled_score_math
scaled_score_ela
scaled_score_composite
std_ scaled_score_math
std_ scaled_score_ela
std_scaled_score_composite
language_version_ela
language_version_math

Next, loop through each subject (`subj`), `school_year`, and `grade_level` using the `levelsof` command to get a list of values to loop through.

Quartile each test using the `xtile` command, setting the number of quantiles, `nq`, to 4. (Test scores for each subject, school year, and grade level must be quartiled separately, since they are scaled and scored independently of one another.) As a side note, be sure to create test score quartiles before restricting the sample in any way. Failure to do so creates quartile values that do not properly correspond to test scores.

```
foreach subj in math ela {

  quietly: levelsof school_year, local(year_list)
  foreach yr in `year_list' {

    quietly: levelsof grade_level, local(grade_list)
    foreach gr in `grade_list' {

      xtile qrt_`gr'_`subj'_`yr' = scaled_score_`subj'
        if school_year == `yr' & grade_level == `gr', nq(4)
    }
  }
}
```

Make sure each test score has only one quartile value by using the `rowmax()` function of the `egen` command to condense the variables into one variable per subject quartile.

Drop the variables no longer needed and save the new test scores file as a tempfile.

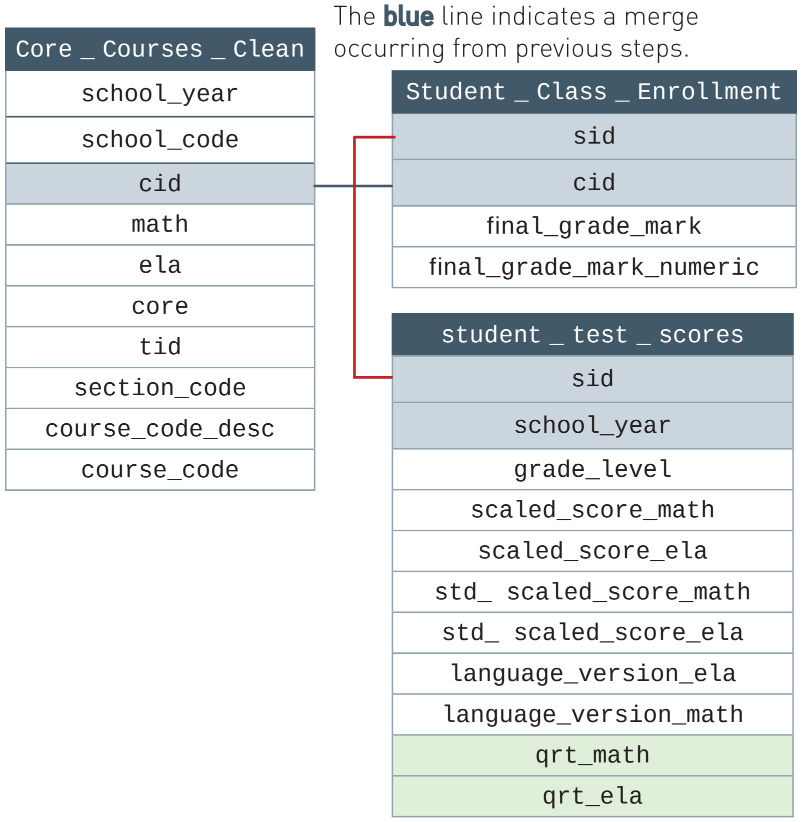
Having created the `student_test_scores` tempfile, you can now restore the previous dataset to work again with the working file.

```
foreach subj in math ela {
  egen qrt_`subj' = rowmax(qrt_*_`subj'_*)
}


foreach var of varlist scaled* std* qrt* language* {
bys sid: gen `var'_tm1 = L.`var'
}

tempfile student_test_scores
save `student_test_scores'

restore
```

1. Merge

1.4 Merge the `student_test_scores' tempfile to the working file you previously had in memory.

The **blue** line indicates a merge occurring from previous steps.



Core _ Courses _ Clean
school_year
school_code
cid
math
ela
core
tid
section_code
course_code_desc
course_code

Student _ Class _ Enrollment
sid
cid
final_grade_mark
final_grade_mark_numeric

student _ test _ scores
sid
school_year
grade_level
scaled_score_math
scaled_score_ela
std_ scaled_score_math
std_ scaled_score_ela
language_version_ela
language_version_math
qrt_math
qrt_ela

Since the working file contains all of a given student's classes, the student may appear more than once, necessitating an m:1 (many-to-one) merge with `sid` and `school_year` as identifiers. Ensure that the file is still unique by `sid` and `cid`.

```
merge m:1 sid school_year using
`student_test_scores', gen(_m_test_
scores)

isid sid cid
```

1.5 Load Student_School_Year_Clean and rename all student variables with an 's_' prefix.

Preserve your current working file and load `Student_School_Year_Clean`, dropping composite scores, which are not used in the later parts of the toolkit.

```
preserve

  use "${clean}/Student_School_Year_
Clean.dta", clear
```

Student _ School _ Year _ Clean
sid
school_year
grade_level
frpl
iep
ell
gifted
retained
move_nonstruct
move_struct
absence_high
absence_miss

Rename each variable, except `sid` and `school_year`, with an s_ prefix to distinguish student variables from teacher variables that will be merged later on in the process.

```
  foreach var of varlist grade_level
days* absence* {
    rename `var' s_`var'
  }
```

Save the data as the tempfile `student_school_year` and restore the working file.

```
  tempfile student_school_year
  save `student_school_year'

restore
```

1. Merge

1.6 Merge the `student_school_year' tempfile to the working file you previously had in memory.

Perform an m:1 merge of the ``student_school_year'` tempfile to the working file using `sid` and `school_year` as unique identifiers.

```
merge m:1 sid school_year using `student_school_year', gen(_m_student_school_year)
```

Repeat the concept of renaming variables with a prefix and merging the resulting tempfile onto the previously existing working file via Steps 1.7 and 1.8 (see the next page for a visual summary of Steps 1.6–1.9).

1.7 Load Student_Attributes_Clean and rename all student variables with an 's_' prefix. Merge the `student_school_year' tempfile to the working file you previously had in memory.

```
preserve

  use "${clean}/Student_Attributes_Clean.dta", clear
  isid sid

  rename male s_male
  rename race_ethnicity s_race_ethnicity

  tempfile student_attributes
  save `student_attributes'

restore

merge m:1 sid using `student_attributes', gen(_m_student_attributes)
```

1.8 Perform a similar set of operations for the Staff files including: Staff_School_Year_Clean, Staff_Attributes_Clean and Staff_Certifications_Clean. Add a ìt_î prefix to identify these as teacher variables.

```
// Staff school year
preserve

  use "${clean}/Staff_School_
Year_Clean.dta", clear

  foreach var of varlist
school_code job_code
degree experience hire_date
termination_date {
    rename `var’ t_`var’
  }

  tempfile staff_school_year
  save `staff_school_year’

restore

merge m:1 tid school_year
using `staff_school_year’,
gen(_m_staff_school_year)
```

```
// Staff attributes
preserve

  use "${clean}/Staff_
Attributes_Clean.dta", clear

  // birth_date is not needed
  drop birth_date

  isid tid

  foreach var of varlist male
certification_pathway race_
ethnicity {
    rename `var' t_`var'
  }
  tempfile staff_attributes
  save `staff_attributes'

restore

merge m:1 tid using `staff_
attributes', gen(_m_staff_
attributes)
```

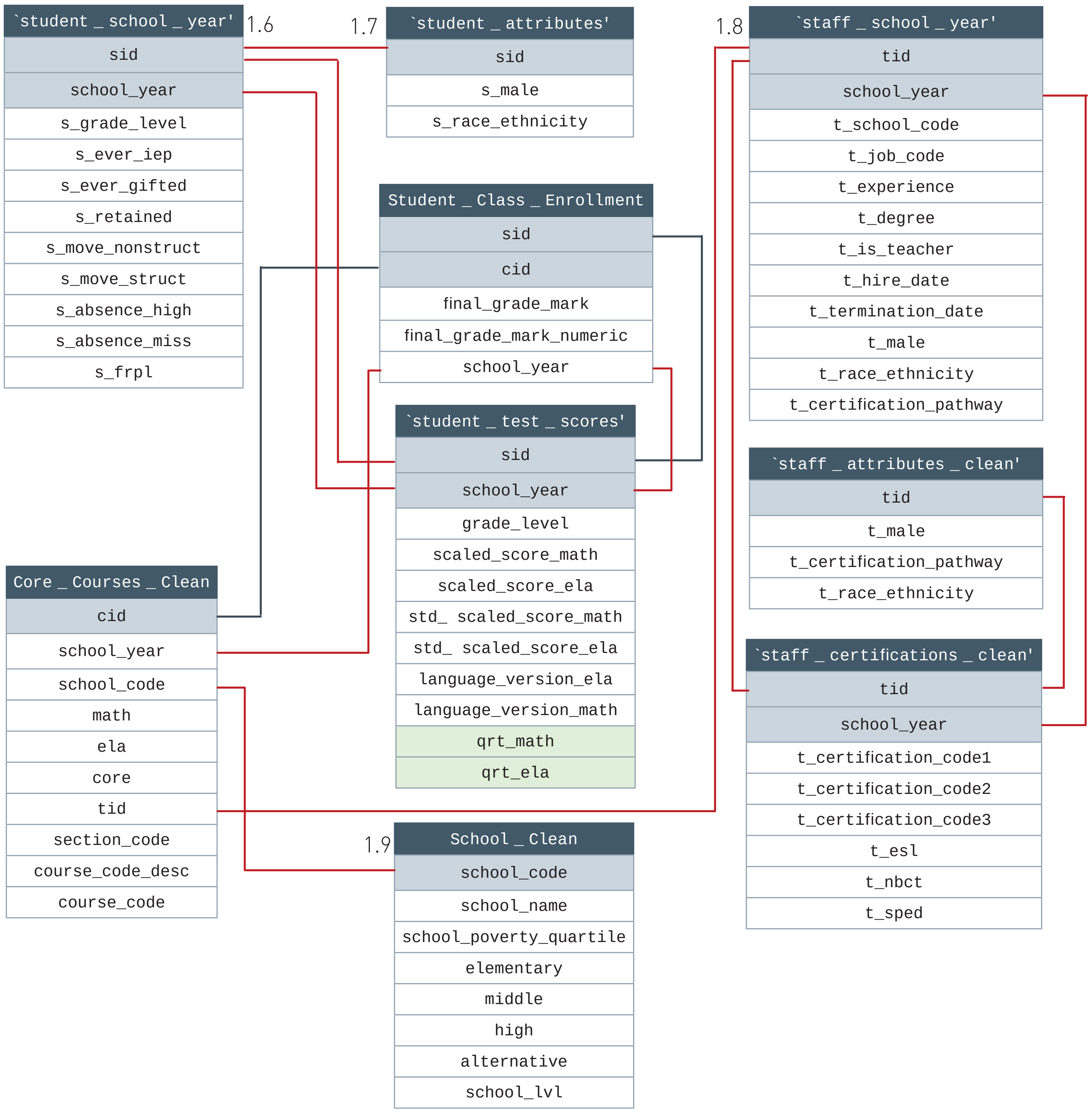
```
// Staff certifications
preserve

  use "${clean}/Staff_
Certifications_Clean.dta",
clear

  isid tid school_year

  foreach var of varlist
certification*{
    rename `var' t_`var'
  }
  tempfile staff_certifications
  save `staff_certifications'

restore

merge m:1 tid school_year
using `staff_certifications',
gen(_m_staff_certificatons)
```

1.9 Merge on the School file to obtain information on school names, types, and poverty level.

```
merge m:1 school_code using "${clean}/School_Clean.dta", gen(_m_school)
```

1. Merge

A visual summary of Steps 1.6–1.9.



1.10 Drop observations missing student ids, sort the file, and then save as Connect_Step1.dta.

```
        drop if mi(sid)
        sort sid cid school_year
        isid sid cid school_year
        save "${analysis}/Connect_Step1.dta", replace
}
```

2. Restrict

Purpose: Restrict the sample to tested grades and core subjects with students who have a single teacher and classes of plausible size. Restructure the data to be unique by student ID (`sid`) and `school_year`.

Restrict involves dropping certain subsets of observations under the assumption that the variables created earlier on **Clean** were implemented correctly for your particular dataset (e.g., you correctly defined the core courses indicator). Dropping observations should be done carefully: failing to do so may result in your inadvertently dropping data you should have kept.

Files needed: `Connect_Step1.dta`

After this step, you will have restricted the sample resulting from the Merge step to contain only tested grades, core subjects, and students with a single teacher in classes of plausible size. The result will be a file unique by `sid` and `school_year`.

```
if $step_2_restrict == 1 {
```

2.1 Load the Connect_Step1 file and drop the merge variables you will not need for Analyze.

The final analysis files do not need the results of all historical merges for the purposes of completing **Analyze**. You can also now drop any duplicates that may have resulted from **Connect** Step 1.

```
use "${analysis}/Connect_Step1.dta" , clear
drop _m_*
duplicates drop
```

2.2 Keep data only on core courses.

Your analysis files will be concerned only with core courses, most commonly math and ELA. Provided that your core courses have been well defined in **Clean** (i.e., you have not left out or misidentified any), drop any non-core courses that may not be well linked to standards on assessments. Also, make sure that these kept core courses are only in math and ELA and that the class identifier is not missing. Now that you have a file of only core courses, drop the core course indicator.

```
tab core, mi
keep if core == 1

assert math == 1 | ela == 1
assert cid != .

drop core
```

2.3 Drop courses that have more than one or no teacher.

The analysis files enable you to attribute student performance in a given subject to the student's teacher in that subject. To avoid dividing student performance among multiple teachers, find the number of teachers for a given class ID (`cid`) using the `nvals()` function of `egen`. First, make sure that most class IDs do not have this problem by tabulating the result. The vast majority of your courses should not have this problem. If this is the case, drop those courses that have more than one teacher or no teacher at all.

```
egen num_teach = nvals(tid), by(cid)
tab num_teach, mi

drop if num_teach != 1

drop num_teach
```

2. Restrict

2.4 Drop students who have more than one core course in a given subject in a given year.

For reasons similar to 2.3, find the number of class IDs for math and ELA using the `nvals()` function of `egen`. First, make sure that most class IDs do not have this problem (most students have only one core course in a given subject), and then drop those courses with more than one core course in a given subject.

```
foreach subj in math ela{

  egen num_core_`subj' = nvals(cid) if `subj' == 1, by(sid school_year)

  drop if num_core_`subj' > 1 & num_core_`subj' != .

  drop num_core_`subj'

}
```

2.5 Drop classes with a small number of students.

It is difficult to reliably attribute a teacher effectiveness measure (to be added in Step 4) to classes with a small number of students. As an arbitrary but reasonable cutoff, we impose a restriction of having at least five students in a class.

```
egen class_size = nvals(sid), by(cid)
drop if class_size <= 5
drop class_size
```

2.6 Check that each student has core courses in only one school in any year.

As an additional check, make sure that students have core courses for math and ELA in only one school in a school year.

```
egen num_schools = nvals(school_code), by(sid school_year)
assert num_schools == 1
drop num_schools
```

2.7 Restructure the data file to have math and ELA teacher IDs (tid) and math and ELA class IDs (cid) for each student and school year. This process will structure the data to be unique by student (sid) and school_year in the next substep.

Check that each `sid` and `school_year` contains at most one teacher and one class for math and ELA.

Fill in the newly created variables for each student–school year pair using the `max()` option of the `egen` command in preparation for making the data unique by `sid` and `school_year`.

```
foreach var in tid cid {

  foreach subj in math ela {
    gen `var'_`subj'_ = `var' if `subj' == 1
    egen check = nvals(`var'_`subj'_), by(sid school_year)

    assert check == 1 | check == .

    egen `var'_`subj' = max(`var'_`subj'_), by(sid school_year)
    drop check `var'_`subj'_

  }
}
```

2. Restrict

2.8 Drop duplicates and assert that the file is now unique by student and school_year.

Drop the variables no longer needed and drop duplicates. Ensure that the file is now unique by `sid` and `school_year`.

```
drop tid cid t_* math ela

duplicates drop

isid sid school_year
```

2.9 Generate dummy variables. Impute missing values to zero and define indicators for missing.

Create dummy variable versions of the categorical variables, such as FRPL and race/ethnicity.

```
gen s_black = (s_race_ethnicity == 1)
gen s_asian = (s_race_ethnicity == 2)
gen s_latino = (s_race_ethnicity == 3)
gen s_naam = (s_race_ethnicity == 4)
gen s_white = (s_race_ethnicity == 5)
gen s_mult = (s_race_ethnicity == 6)
gen s_racemiss = (s_race_ethnicity == 7)

gen s_fulllunch = (s_frpl == 0)
gen s_reducedlunch = (s_frpl == 1)
gen s_freelunch = (s_frpl == 2)
gen s_misslunch = (s_frpl == .)

replace s_absence_high = 0 if s_absence_miss == 1

foreach var of varlist s_iep s_ell s_retained s_gifted {
       gen `var'_miss = (`var' == .)
       replace `var' = 0 if `var'_miss == 1
}
```

2.10 Generate class- and cohort-level average variables.

Produce class- and cohort-level averages of various demographics for all students in a class or cohort.

Also, produce class- and cohort-level averages and standard deviations of prior year scores for all students in a class or cohort.

Class-level variables are denoted by the prefix `_CL`. Remember to generate class-level variables only for non-missing class IDs so you do not aggregate all missing class IDs together.

Cohort-level variables are denoted by the prefix `_CO`. Remember that a cohort is defined by the `grade_level`, `school_code`, and `school_year`.

```
foreach var of varlist s_retained s_iep-s_gifted s_absence_
high-s_male s_black-s_racemiss s_reducedlunch-s_gifted_miss
{

  foreach subj in math ela {
    egen _CLmean_`var'_`subj' = mean(`var') if
!mi(cid_`subj'), by(cid_`subj')
  }

  egen _COmean_`var' = mean(`var'), by(grade_level school_
code school_year)
}

foreach subj in math ela {

  egen _CLstd_scaled_score_`subj'_tm1 = mean(std_scaled_
score_`subj'_tm1) if !mi(cid_`subj'), by(cid_`subj')

  egen _CLstd_scaled_score_`subj'_tm1_sd = sd(std_scaled_
score_`subj'_tm1) if !mi(cid_`subj'), by(cid_`subj')

  egen _COstd_scaled_score_`subj'_tm1 = mean(std_scaled_
score_`subj'_tm1), by(grade_level school_code school_year)

  egen _COstd_scaled_score_`subj'_tm1_sd = sd(std_scaled_
score_`subj'_tm1), by(grade_level school_code school_year)
```

2. Restrict

2.11 Generate class and cohort size variables.

Use the `nvals()` function of the `egen` command to calculate the the number of students in a class and cohort.

```
foreach subj in math ela{
  egen _CLnumber_students_`subj' = nvals(sid) if
!mi(cid_`subj'), by(cid_`subj')
}

egen _COnumber_students = nvals(sid), by(grade_level
school_code school_year)
```

2.12 Generate variables that capture the proportion of students with a missing prior test score for both class and cohort.

Calculate the proportion of students with a missing prior test score for each class and cohort.

Use the class and cohort size results from the prior step as the denominator for this calculation.

```
foreach subj in math ela {
       egen temp_`subj' = nvals(sid) if mi(std_scaled_
score_`subj'_tm1) & !mi(cid_`subj'), by(cid_`subj')
       egen max_temp_`subj' = max(temp_`subj'), by(cid_`subj')
       bysort cid_`subj': gen _CLpct_missing_std_`subj'_tm1 =
max_temp_`subj'/_N

       drop max_temp_`subj' temp_`subj'

       egen temp_`subj' = nvals(sid) if mi(std_scaled_
score_`subj'_tm1), by(grade_level school_code school_year)
       egen max_temp_`subj' = max(temp_`subj'), by(grade_level
school_code school_year)

       bysort grade_level school_code school_year: gen _COpct_
missing_std_`subj'_tm1 = max_temp_`subj'/_N
       drop max_temp_`subj' temp_`subj'
}
```

2.13 Create school poverty measures at the year-school level using student FRPL indicators.

Create average FRPl status by collapsing the data by school code and year as a proxy for school poverty status. Then create quartiles using this measure. Save this measure in a tempfile `school_poverty_qrt`.

Then, merge `school_poverty_qrt` onto the data in memory.

```
preserve
       keep school_code school_year s_frpl
       drop if mi(school_code) | mi(school_year) | mi(s_frpl)
       collapse (mean) s_frpl, by(school_code school_year)
       isid school_code school_year
       gen school_poverty_quartile = .
       forval year = 2007/2011 {
              xtile temp_poverty_quartile = s_frpl if school_year
== `year', nq(4)

              replace school_poverty_quartile = temp_poverty_quar-
tile if school_year == `year'
              drop temp_poverty_quartile
       }
       assert !missing(school_poverty_quartile)
       label define pvt 1 "Lowest percentage of FRPL-eligible stu-
dents"
       label define pvt 2 "Second-lowest percentage of FRPL-eligi-
ble students" , add
       label define pvt 3 "Second-highest percentage of FRPL-eligi-
ble students" , add
       label define pvt 4 "Highest percentage of FRPL-eligible stu-
dents", add
       label values school_poverty_quartile pvt
       drop s_frpl

       tempfile school_poverty_qrt
       save `school_poverty_qrt'
restore
merge m:1 school_code school_year using `school_poverty_qrt',
keep(1 2 3) nogen
```

2. Restrict

2.14 Create average prior achievement measures at the year-school level using student test scores.

Create average prior achievement measures by school code and year. Save this measure in a tempfile `sch_achievement`.

Then, merge `sch_achievement` onto the data in memory.

```
preserve
      foreach var of varlist  std_scaled_score_ela std_scaled_score_math {
            egen school_avg_`var' = mean(`var'), by(school_code school_
year)
      }

      keep school_code school_year school_avg*
      duplicates drop
      tsset school_code school_year
      forvalues year = 2008/2011 {
            xtile temp_sch_math_qrt_tm1_`year' = L.school_avg_std_scaled_
score_math if school_year==`year', nq(4)
            xtile temp_sch_ela_qrt_tm1_`year' = L.school_avg_std_scaled_
score_ela if school_year==`year', nq(4)
      }

      gen sch_avg_prior_math_qrt = .
      gen sch_avg_prior_ela_qrt = .
      foreach subj in math ela {
            forvalues year = 2008/2011 {
                  replace sch_avg_prior_`subj'_qrt = temp_sch_`subj'_qrt_
tm1_`year' if  sch_avg_prior_`subj'_qrt ==.
            }
      }
      drop temp* school_avg_std_scaled_score*

      label define pmath 1 "Lowest average prior year math scores" 2 "Sec-
ond-lowest prior year math scores" 3 "Second-highest prior year math scores"
///
      4 "Highest prior year math scores"
      label values sch_avg_prior_math_qrt pmath

      label define pela 1 "Lowest average prior year ELA scores" 2 "Second-
lowest prior year ELA scores" 3 "Second-highest prior year ELA scores" ///
      4 "Highest prior year ELA scores"
      label values sch_avg_prior_ela_qrt pela

      label var sch_avg_prior_math_qrt "4 quartiles of school average prior
math scores"
      label var sch_avg_prior_ela_qrt "4 quartiles of school average prior
ELA scores"
      tempfile sch_achievement
      save `sch_achievement'

restore

merge m:1 school_code school_year using `sch_achievement', nogen
```

2.15 Sort the file, ensure uniqueness, and then save as Connect_Step2.dta.

Ensure that the file is unique by `sid` and `school_year`, sort the data, arrange the variables in a logical order, and save.

```
      isid sid school_year
      sort sid school_year
      order sid school_year school_code grade_level cid*
tid* scaled* std* qrt* language* s_* _CL* _CO*
      save ${analysis}/Connect_Step2.dta, replace

}
/
```

3. Generate

Purpose: Generate key variables required for the `Teacher_Year_Analysis` files, such as whether a teacher is a new hire, late hire, novice, or veteran.

Files needed: `Connect_Step1.dta`, `Connect_Step2.dta`

After this step, you will have an analysis file with a record of the teachers, `Teacher_Year_Analysis`.

```
if $step_3_generate == 1 {
```

3.1 Load the Connect_Step1 file and drop the student-level variables you do not need for the Teacher_Year_Analysis file.

Drop student-level and school-level variables. Student-level variables are not needed for the teacher-level analysis file. School-level variables will be re-appended after resolving discrepancies in the school code variable.

Drop any observations missing a teacher ID (`tid`).

```
use ${analysis}/Connect_Step1.dta, clear


drop sid math ela grade_level core *scaled*
language* s_* _m* /// qrt*
    school_name elementary middle high
    alternative school_lvl

drop if mi(tid)
```

3.2 Resolve any potential conflicts in the school_code variable by giving preference to the student data.

`Connect_Step1` has a school code variable from two different sources—the student data and teacher data. If these conflict, preference is given to the student data to reflect the true grouping of students by school. After this preference is accounted for, a teacher may still be associated with more than one school. In this case the school code will be chosen randomly.

First, create a variable, `n`, that counts the number of observations for each `tid school_year` combination. Then, mark the `tid school_year` observations that appear more than once in the dataset and have a school code from the student file different from the one from the teacher file.

```
bys tid school_year: gen n = _N

gen temp = 1
replace temp = 0 if n > 1 & school_code !=
t_school_code
```

Sort the observations so that in each `tid school_year` group the observations marked by a value of 1 for the variable `temp` appear on top (via the "`-`" prefix for descending order) and then randomly using the variable `rand`. Keep only the first observation within each `tid school_year`.

```
gen rand = runiform()
gsort tid school_year -temp rand
bys tid school_year: keep if _n == 1
```

If there is no `school_code` in the student data, the `school_code` from the teacher data should be used, if available.

Drop any temporary variables you created.

```
replace school_code = t_school_code if
school_code == .

drop t_school_code n temp rand
```

3. Generate

3.3 Merge on the school-level variables from the school file using the now-singular school code.

```
merge m:1 school_code using ${clean}/School_Clean.dta, keep(1 3) nogen
```

3.4 Create dummy variables for race/ethnicity, which will be used as controls or outcomes in Analyze.

```
gen t_black = (t_race_ethnicity == 1)
gen t_asian = (t_race_ethnicity == 2)
gen t_latino = (t_race_ethnicity == 3)
gen t_naam = (t_race_ethnicity == 4)
gen t_white = (t_race_ethnicity == 5)
gen t_mult = (t_race_ethnicity == 6)
gen t_racemiss = (t_race_ethnicity == 7) | (t_race_ethnicity == .)
```

3.5 Create an indicator for a teacher possessing either a Master's or Doctorate degree.

```
gen t_adv_degree = (t_degree == 2 | t_degree == 3)
replace t_adv_degree = . if t_degree == .
```

3.6 Create a new hire dummy variable.

To identify newly hired teachers, find the earliest year in which a staff member is observed as a teacher in the data, and set the new hire variable equal to 1 for that year.

Drop any temporary variables you created.

```
summ school_year
local min_school_year = r(min)
local max_school_year = r(max)
egen x = min(school_year) if t_is_teacher ==
1, by(tid)
egen first_observed_teacher = max(x), by(tid)
drop x
gen t_newhire = 0 if t_is_teacher == 1
replace t_newhire = 1 if school_year ==
first_observed_teacher & t_is_teacher == 1
replace t_newhire = . if school_year ==
`min_school_year'
drop first_observed_teacher
```

3. Generate

3.7 Create the t_novice, t_novice_ever, t_veteran, and t_veteran_newhire dummy variables.

A teacher is `t_novice` if his/her experience equals 1.

A teacher is `t_novice_ever` if s/he appears as a novice in any school year in the dataset.

A teacher is a t_`veteran_newhire` if s/he is a new hire and is not a novice.

```
gen t_novice = .
replace t_novice = 1 if t_experience == 1
/*& t_newhire == 1*/
replace t_novice = 0 if t_novice != 1 & t_
is_teacher == 1 & !missing(t_experience)

gen t_veteran_newhire = 0 if t_is_teacher ==
1 & !missing(t_experience)
replace t_veteran_newhire = 1 if t_newhire
== 1 & t_novice == 0 & !missing(t_experi-
ence)

egen t_novice_ever = max(t_novice), by(tid)
egen t_novice_ever_check = sum(t_novice),
by(tid)
assert t_novice_ever_check < 2 if t_is_
teacher==1
drop t_novice_ever_check
```

3.8 Create the retention dummy variables.

For a given year, the retention variables reflect the teacher's status for the following year.

- A teacher is defined as "staying" if s/he is employed as a teacher in the same school the following year.
- A teacher is defined as "transferring" if s/he is employed as a teacher at a different school the following year.
- A teacher is defined as "leaving" if s/he does not meet the criteria for the previous two variables.

Note that these variables should be missing for the last school year, in this case 2012, for which there is data, since no
information exists for the following year.

Assert that the variables are correctly defined so any teacher is assigned only one retention status in a `school_year`.

```
// Set the time series
tsset tid school_year

// Define next-year status variables
foreach var of varlist school_code t_is_
teacher {
      gen `var'_tp1 = F.`var'
}

assert !missing(school_code) if school_year
< `max_school_year'
gen t_stay = school_code == school_code_tp1
& t_is_teacher_tp1 == 1 ///
      if t_is_teacher == 1 & school_year <
`max_school_year'
gen t_transfer = school_code != school_code_
tp1 & t_is_teacher_tp1 == 1 ///
      if t_is_teacher == 1 & school_year <
`max_school_year'
gen t_leave = t_is_teacher_tp1 != 1 ///
      if t_is_teacher == 1 & school_year <
`max_school_year'
assert t_stay + t_transfer + t_leave == 1
if t_is_teacher == 1 & school_year < `max_
school_year'
```

3. Generate

3.9 Merge on the school-level poverty and average prior achievement variables created in Connect Step 2.

Preserve a version of the current file and open the `Connect_Step2` data file. Keep relevant variables and drop duplicate observations so that the file is unique by `school_code` and `school_year`.

Save a temporary file called `school_vars` and restore the main data file.

Then, merge the temporary file onto the main data file.

```
preserve
use ${analysis}/Connect_Step2.dta, clear
keep school_code school_year school_poverty_quartile
sch_avg_prior*
duplicates drop
isid school_code school_year
tempfile school_vars
save `school_vars'
restore

merge m:1 school_code school_year using `school_vars',
keep(1 3) nogen
```

3.10 Sort the file, ensure uniqueness, save a tempfile, and then save as Teacher_Year_Analysis.dta.

Sort the data, ensure that the file is unique by `tid` and `school_year`, and arrange the variables in a logical order.

Save a temporary file, `for_stu`, that you will use in Steps 3.10 and 3.11 to merge onto the student-level data using class ID (`cid`) to create the `Student_Teacher_Year_Analysis` file.

Save the final `Teacher_Year_Analysis` file. The `cid` is not needed in this file and can be dropped before saving.

```
       sort tid school_year
       isid tid school_year

       order tid school_year school_code school_name-
school_lvl school_poverty_quartile sch_avg_prior* ///

              t_black-t_racemiss t_is_teacher t_job_code
t_hire_date t_termination_date t_experience t_degree t_
adv_degree ///
              t_certification_pathway-t_certification_sped
t_newhire-t_leave cid

       tempfile for_stu
       save `for_stu'

       drop cid
       save ${analysis}/Teacher_Year_Analysis.dta,
replace

}

/
```

4. Understand and Merge

Purpose: Understand how to incorporate a measure of teacher effectiveness such as value-added estimates into your analysis and **merge** these estimates onto the `Teacher_Year_Analysis` file.

Files needed: `Teacher_Year_Analysis.dta`

After this step, you will have a `Teacher_Year_Analysis` file that includes a measure of teacher effectiveness to complete the entirety of **Analyze**: Recruitment, Placement, Development, Evaluation, and Retention.

```
if $step_4_understand==1 {
```

4.1 Understand one possible measure of teacher effectiveness, value-added estimates.

1. There are many valid ways in which agencies can measure teacher effectiveness:
   - student growth percentiles
   - student survey results
   - traditional classroom observations
   - classroom observation videos
   - value-added measures

   SDP uses value-added estimates as one way of analyzing patterns and trends in effectiveness across groups of teachers. This document will not walk you through the steps of value-added estimation, but aims to give you a general overview of the process. For the purpose of completing the toolkit, you are provided with a dataset with predetermined value-added estimates, but you should replace this dataset with the primary teacher effectiveness measures used by your agency.

2. Why is value-added used as a measure of teacher effectiveness?
   - Value-added aims to determine a teacher's contribution to student learning by taking into account his/her students' prior achievement and other factors outside the teacher's control. Value-added estimates are calculated using existing data.
   - **Value-added is one of many measures for understanding teacher effectiveness.** The **Technical Appendix** at the end of this document dives into additional detail, including limitations of value-added estimation that you should keep in mind.

3. At a high level, value-added estimation consists of four basic steps.
   - First, obtain the prior year performance of a student on a relevant assessment.
   - Second, obtain a set of "control" variables that include student characteristics such as free or reduced-price lunch (FRPL) status, race and ethnicity, gender, and so on. The first and second step together help account for factors outside a teacher's control when evaluating his/her performance.
   - Third, use a statistical model with the previously obtained "control" variables to determine the performance that a typical student with similar characteristics would be expected to have on the end-of-year assessment.
   - Fourth, compare the student's actual performance to the model's prediction for a typical student with these characteristics. Repeat this process for each student assigned to a given teacher. Then, average the results of these comparisons to generate a value-added score, or teacher effect, for that teacher. This results in a distribution of value-added scores for teachers in the agency.

For a more detailed look, see the **Technical Appendix**.

The process described above has been completed for you in the sample data. Use `Connect_TEM.dta` in 4.2 as a dataset containing the measure of teacher effectiveness needed to conduct the analyses in **Analyze**.

4. Understand and Merge

4.2 Load the Teacher_Year_Analysis file, merge on your chosen measure of teacher effectiveness, and save the final Teacher_Year_Analysis file.

Load the `Teacher_Year_Analysis` file.

```
use ${analysis}/Teacher_Year_Analysis.dta, clear
```

Merge on your chosen measure of teacher effectiveness using the teacher ID (`tid`) and `school_year`.

```
merge 1:1 tid school_year using ${analysis}/Connect_TEM.
dta, keep(1 3) nogen
```

Save the `Teacher_Year_Analysis` file, which now includes a measure of teacher effectiveness.

```
save ${analysis}/Teacher_Year_Analysis.dta, replace
```

4.3 Structure the student analysis file so that each student has two observations for each year: one that contains information relevant to math and the other to ELA.dta.

Looping over each subject, load the `Connect_Step2.dta` file, and keep only the information relevant to each subject. Then merge on the relevant teacher information from the recently saved temporary file, `for_stu`.

Merge the two subject files. This process also consolidates the subject-specific teacher ID variables into one teacher ID variable.

```
foreach subj in math ela {

  if "`subj'" == "math"{
    local alt_subj = "ela"
  }
  if "`subj'" == "ela"{
    local alt_subj = "math"
  }

  use ${analysis}/Connect_Step2.dta, clear
  drop *`alt_subj'*
  rename tid_`subj' tid

  merge m:1 tid school_year using ${analysis}/Connect_TEM.dta,
  keep(1 3) nogen
  merge m:1 tid school_year using `for_stu', keep(3) nogen
  gen subj = `subj';

  tempfile stu_tch_yr_`subj'
  save `stu_tch_yr_`subj''
}

use `stu_tch_yr_math', clear
merge 1:1 sid school_year using `stu_tch_yr_ela', nogen
```

4. Understand and Merge

4.4 Sort the file, ensure uniqueness, and then save as Student_Teacher_Year_Analysis.dta.

Sort the data, ensure that the file is unique by student ID, teacher, and school year, and arrange the variables in a logical order.

Save the final `Student_Teacher_Year_Analysis` file. The `cid` is not needed in this file and can be dropped before saving.

```
	sort sid school_year subj
	isid sid tid* school_year

	order sid school_year grade_level cid* school_code
	school_name-school_lvl tid*  ///

		t_race_ethnicity t_black-t_racemiss t_is_teach-
er t_job_code t_hire_date t_termination_date ///
		t_experience t_degree t_adv_degree t_certifica-
tion_pathway-t_certification_sped t_newhire-t_leave s_* ///
		scaled_score* std_scaled_score* qrt* language*
_CL* _CO*

	drop cid
	save ${analysis}/Student_Teacher_Year_Analysis.dta,
	replace

}

log close
```

Conclusion

You should now have the final `Student_Teacher_Year_Analysis` and `Teacher_Year_Analysis` files needed to complete the **Analyze** portion of this toolkit!

TECHNICAL APPENDIX: A BETTER UNDERSTANDING OF TEACHER EFFECTS ESTIMATES

Without providing Stata code that generates value-added measures, this section summarizes the technical components of SDP's value-added model. As stated earlier, value-added models are intuitively simple, but can be complicated in their execution. Building upon the conceptual description of value-added models in Step 4.1, we explain some of the technical details of value-added modeling.

First, we describe the components of the **model** that SDP uses to calculate value-added scores. Then we explain other elements of the process, such as **sample** selection and **estimation**. We conclude with a discussion of some of the **limitations** and **additional considerations** of value-added models.

Model

SDP calculates value-added scores using the following student-level equation:

(1) $a_{i,j,k,t} = A_{i,t-n}\alpha + S_{i,t}\beta + P_{j,k,t}\delta + T_{k,t}\gamma + E_{i,t}\rho + v_{i,j,k,t}$, where $v_{i,j,k,t} = \mu_k + \theta_{j,k,t} + \varepsilon_{i,j,k,t}$

The outcome, $a_{i,j,k,t}$, is the test score for student i, who is a member of class j taught by teacher k during school year t. In other words, our outcome is the current year's test score for a student assigned to a particular class and teacher. In most cases, $a_{i,j,k,t}$ represents a score on a state standardized test administered at the end of the school year.

That outcome test score is modeled as a function of the vectors $A_{i,t-1}$, $S_{i,t}$, $T_{k,t}$, and $P_{j,k,t}$. The composition of these vectors can be somewhat different from analysis to analysis, depending on data availability and the intended use of the results. In general, however, each is composed as follows:

$A_{i,t-1}$ represents information regarding student i's prior academic achievement, including:

- $a_{i,t-1}$ student i's test score in the same subject (e.g., math when predicting math) from the previous school year t-1,
- the square and cube of $a_{i,t-1}$,
- the interaction of $a_{i,t-1}$ and the grade level of that test,
- the interaction of $a_{i,t-1}$ and the language of that test,
- an indicator for switching language of testing between t-1 and t,
- the interaction of $a_{i,t-1}$, the grade level of that test, and an indicator for being a grade repeater, and
- $a'_{i,t-1}$ student i's test score in a different subject (e.g., reading when predicting math) from the previous school year t-1.

Sometimes a small number of students in a class will not have taken one or more exams the previous year, and thus do not have values for both. If a student is missing data for prior tests in subjects not being predicted, we impute the score as 0, and include an indicator variable in identifying such cases. If a student is missing data for prior tests in the subject predicted, that student is not included in the model.

All test scores, $a_{i,t-1}$ and $a'_{i,t-1}$ along with $a_{i,j,k,t}$, are standardized with a mean of zero and a standard deviation of one by subject, grade, year, and language of the test. This standardization is calculated based on the student's raw or scaled score compared to other students who took the same test within the system (e.g., Cambridge Public Schools students who took the Massachusetts Comprehensive Assessment System (MCAS) fourth-grade math test for 2010). This means that students' scores are standardized relative to those of their grade-level peers in the same agency or state. New tests based on the Common Core Standards (e.g., Smarter Balanced Assessment) could allow for interstate comparisons in the future.

$S_{i,t}$ represents other observable characteristics of student i during school year t, including variables for:

- gender,
- each racial or ethnic subgroup,
- each free or reduced price lunch program classification (as a proxy for family income),
- each English language learner classification, and an indicator for former English language learner classification,
- each IDEA (special education) classification,
- gifted education eligibility,
- whether the student was retained in grade (i.e., grade level for year t is the same as for year t-1),
- whether the student was new to the school (this includes structural transitions from elementary to middle schools and non-structural changes),
- an indicator for substantial absence (e.g., greater than 10% of enrolled days), and
- an indicator for calendar track in school.

$P_{j,k,t}$ is a vector of observable characteristics of student i's peers in class j and student i's peers in the same grade level within the same school. $P_{j,k,t}$ includes separately for class j and the school grade-level cohort:

- the means of the elements of $S_{i,t}$,
- the means and standard deviations of $a_{i,t-1}$ and $a'_{i,t-1}$,
- the proportion of peers who are missing test scores for $a_{i,t-1}$ and $a'_{i,t-1}$, and
- the number of students in class j and the number of students in the grade-level cohort.

$T_{k,t}$ is a vector of teacher k's characteristics or practices. It is defined differently from analysis to analysis. Unlike the other vectors, the estimated coefficients γ are often the focus of the analysis. Thus, $T_{k,t}$ might, as just a few examples, include:

- teacher k's scores on a classroom observation rubric,
- indicator variables for various experience groupings (e.g., novice, 1 year, 2 years, 3 years, 4–9 years, 10+ years) and teacher fixed effects to study the importance of experience on student achievement growth, and
- indicator variables for whether a teacher is National Board Certified or not.

For some analytic purposes we do not include $T_{k,t}$ at all; in these cases we are interested in capturing a teacher's total effect on student achievement growth compared to all other teachers regardless of teacher characteristics.

$v_{i,j,k,t}$ represents random effects at the teacher (μ_k) and class ($\theta_{j,k,t}$) levels and an error term ($\varepsilon_{i,j,k,t}$).

We do not generally include school fixed effects in Equation 1. Including school fixed effects would control for unobservable differences between schools (e.g., school culture). However, it would confine us to making comparisons of teachers only within the same school and relies on the unlikely assumption that teacher quality is distributed randomly across schools. We do check our inferences for robustness to the inclusion or exclusion of school fixed effects. When appropriate, we highlight the differences and potential implications.

Sample

Our analysis sample is constrained by the availability of required data, analytic choices to improve the estimation process, and other choices to aid interpretation of the results. As implied above, sample selection rules will depend on data availability and the intended use of the results. In general, however, we include student-by-year observations i (and the associated classes j and teachers k) when:

- $a_{i,j,k,t}$, $a_{i,t-1}$, and all of the variables included in $S_{i,t}$ are non missing,
- we can identify one specific class j (and one specific associated teacher k) in which the student i received instruction in the subject content (i.e., standards) measured on the outcome test $a_{i,j,k,t}$,
- the student did not transfer into the school mid-year (if the data allow this determination),
- the class j includes at least five students for whom the first two bullet points are true,
- the proportion of students in class j missing $a_{i,t-1}$ is less than 0.5, and

- the class j is not a self-contained class providing instruction to IDEA students exclusively, defined either through course title or by a class that is 50% or more IDEA students.

The second criterion, identifying one class, always requires assumptions about the courses themselves and the data available to link students and teachers to classes. Often those assumptions vary from system to system. Table 1 outlines the constructs we have in mind when defining classes with commonly available data. In short, we assign students only to classes that are in the typical course progression (e.g., Math 6 to Math 7 to Pre-algebra to Algebra) for the system (or school), and then only if the student had just one such class and teacher. If a student was enrolled in more than one normal progression class during a single school year (e.g., Mr. Smith's Math 6 first semester and Ms. Jones' Math 6 second semester, or Mr. Smith's Math 6 first term and Mr. Smith's Math 7 the remainder), we exclude that student from our analysis. We do not, however, exclude students who had one normal progression class but also had supplemental instruction in the same subject (e.g., Mr. Smith's Math 6 and Ms. Jones' Boost-Up Math concurrently). In this last case we generally include an indicator for supplemental instruction in Equation 1.

We believe this approach, outlined in Table 1, is appropriate for the analyses proposed in the SDP Toolkit for Effective Data Use, but not necessarily for all efforts to estimate teacher effects (a statement that is true of most all of this document).

Estimation

We use Hierarchal Linear Modeling (HLM) (or Linear Mixed Models) to estimate Equation 1 with nested random effects, μ_k and $\theta_{j,k,t}$, for each class j taught by teacher k in school year t. HLM provides empirical Bayes estimates of the teacher random effects, μ_k. Teachers who teach many classes and have many students are more likely to have reliable value-added estimates than teachers who have just a few students. This is because students provide unique evidence of their individual growth, which is then averaged among all the students in their class. In a small class, one student with extremely high (or low) growth can have a substantial impact on the class's average growth. That same student in a very large class or in one of a teacher's many classes will not change the average growth very much. The small class's teacher would have a less "reliable" (in a statistical sense) value-added score because there is not as much evidence to support it and outliers can have a large impact on the estimate. Empirical Bayes adjusts estimates to take into account their reliability (Raudenbush and Bryk, 2002). The model calculates the reliability of an estimate by comparing multiple class averages for classes taught by the same teacher and the variation of test scores among students within each class. Less reliable estimates are adjusted down (or up) to be closer to the mean of all teachers' estimates. This process is sometimes called "shrinking" the estimates. Shrinkage reduces random error that is associated with the class and student levels, including error due to small samples of students.

SDP reports value-added estimates in standard deviation units of student test scores. This metric is helpful both during the regression steps and when comparing the answers to different questions. To change each student's scaled test score to standard deviation units, we first subtract the overall mean scaled score, then divide by the overall mean standard deviation. Thus, the new average student test score will be zero and the standard deviation will be one.

If, for example, we report that National Board Certified teachers have an effect of 0.02 compared to all other teachers, that means that, after accounting for other factors, students are scoring 0.02 standard deviations higher when they are assigned to National Board Certified teachers. To provide a sense of magnitude, the black-white test score gap is generally estimated to be about 1.0 standard deviations, while three years of the whole-school reform model, Success for All, added about 0.2 standard deviations to student achievement in reading.

Limitations

Clearly, standardized tests are imperfect measures of student achievement. The tests do not nearly capture everything we value in what students learn or what teachers teach. Also, using different tests can change value-added estimates for individual teachers and, in some cases, the results from aggregate analyses based on them. Further, the statistical model for value-added rests on some assumptions that are too pristine for the real world. For example, the model assumes only a teacher within a tested subject is responsible for all gains in that subject (e.g., science teachers do not influence student performance in math). Current value-added models do not account for opportunities that students have to learn concepts on state tests outside of the primary teacher's class. Work is being done, however, to accommodate multiple teachers impacting student gains, so long as shared teaching occurs within a single tested subject and this phenomenon can be quantified.

Dynamic student grouping in which students are frequently regrouped with different peers and teachers is also problematic for value-added measures because the statistical models used to generate them require that we associate a group of students with a specific teacher for an entire year. For example, third grade teachers in the same school might decide to exchange classes so that one teacher teaches all students math and the other teacher teaches all students reading and ELA. Since we also account for the effect of a student's peers on the individual student's growth, we must determine which students are grouped together for which classes. Unless classroom rosters are validated by teachers and administrators when students (or teachers) move around to address specific learning objectives, this movement is not captured.

Not all value-added models of teacher effectiveness are created in the same way. Consequently, they do not produce the exact same results. Careful decisions need to be made about the control variables to include and which teachers, students, and classes go into the sample. The answers to those questions can sometimes change teacher value-added estimates.

Finally, measures of teacher value-added may be inaccurate due to information about students that is unaccounted for. For instance, if a classroom's performance on the state test was adversely affected by the noise at a nearby construction site, lower test scores would be attributed to the teacher, even though the teacher had no control over the noisy interruptions. These classroom-level shocks do, however, get averaged out with additional years of data. Additionally, there might be critical missing information about individual students. For example, students who receive outside tutoring could improve their outcomes above what the teacher contributed to, and students who are tired or ill on exam day could achieve lower scores in spite of very effective teaching. With a low enough score, this student could pull down the value-added measure for that teacher. Though all student scores are averaged for a teacher, it is conceivable that some individual instances could influence a teacher's value-added score.

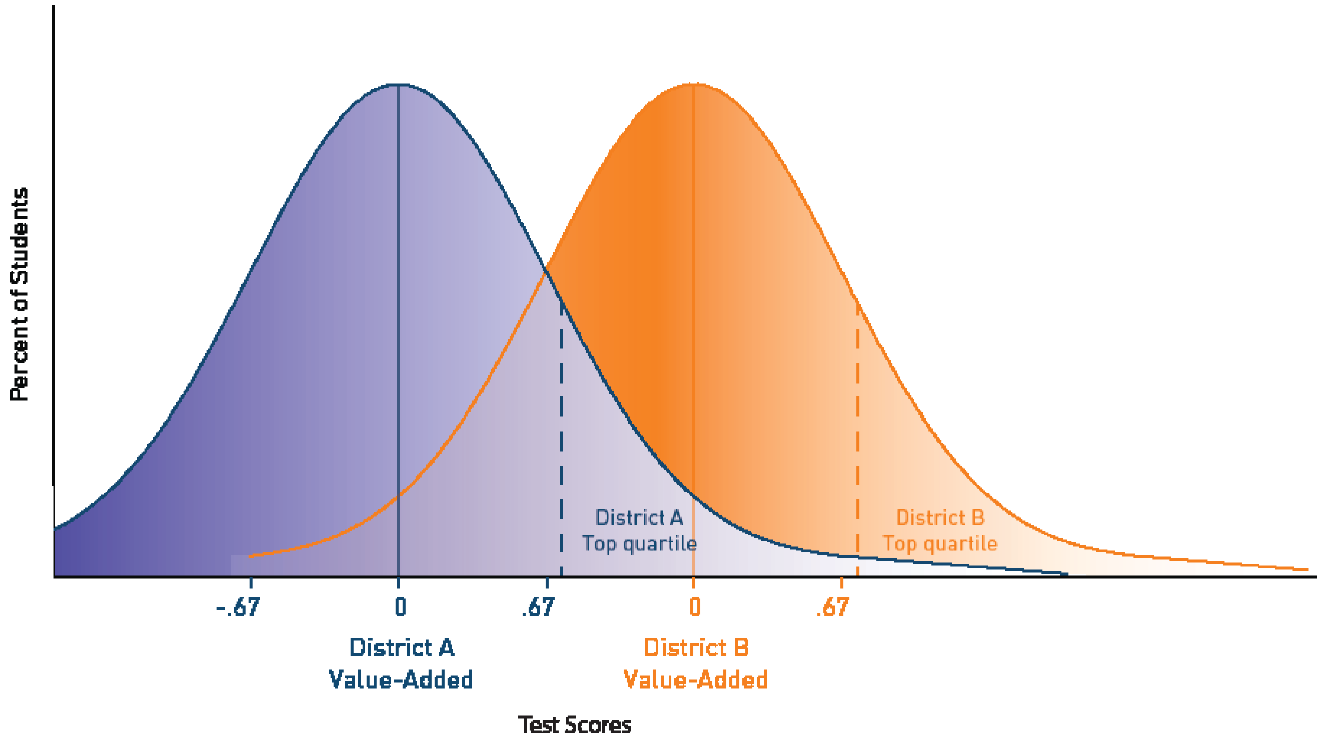
In some cases, even data that are usually recorded may be missing. For example, we omit students from the sample who do not have a prior test score. These students may be more likely to be transfer students or students who received test exemptions due to special needs. If we systematically omit students with slower expected growth trajectories because they lack prior test scores, we may distort teachers' value-added scores by not accounting for those students' growth.

Additional Considerations for Value-Added

As alluded to above, the data needed to construct value-added measures for large numbers of teachers are typically available and require no additional data collection. Ten years after the passage of No Child Left Behind, there is a terrific amount of untapped data about student achievement in the form of state assessment results. Because agencies already capture these data, analyzing them generates no additional data collection costs, but can generate great insight. In contrast, granular information from classroom observation and portfolio evaluations might not be readily available and could be costly to translate from paper copies to an analysis file on a large scale.

To be clear, we do not assert that state assessment scores are fully representative of a student's learning over the course of the year—the quality of state tests varies widely, and even the best tests do not comprehensively assess all that we want our students to know. Nonetheless, they are the measures for which schools are held accountable, and substantial evidence indicates that they are strongly predictive of educational attainment and future earnings.

For many reasons, including the limitations described above, our analyses focus on aggregates rather than individual teachers. Our mission is to educate and inform education agency leaders about patterns and trends of teacher effectiveness. In fact, we do not provide our partners with any "rankings" for individual teachers. Instead, we examine and share average findings for groups of teachers, often by dividing teachers into quartiles of effectiveness. We intend this information to inspire system-level management and policy decisions that improve teacher effectiveness throughout an entire agency, not to evaluate individual teachers. For example, one pattern we have found is that teachers in some of our partner agencies who are hired late are less effective, on average, than their peers who are hired during the regular recruiting season. With this knowledge, school leaders could establish policy and practices to hire teachers earlier and avoid appointing less effective latecomers. In addition, where fitting, we employ "pooled estimates" of value-added scores. That is, we average teachers' value-added estimates over multiple years. For teachers who have been in a district or agency for more than one year, these pooled estimates greatly reduce the influence of random measurement error. Importantly, value-added measures are relative. That is, we compare teachers to the average in their own district or agency, not to an external norm or benchmark. This means that a teacher who is in the top quartile in one district, such as District A below, may not be in the top quartile in another district, such as District B. Why is this? For one, agencies may have different levels of average effectiveness in their teaching force. Second, different agencies may use different tests that measure slightly different skills.

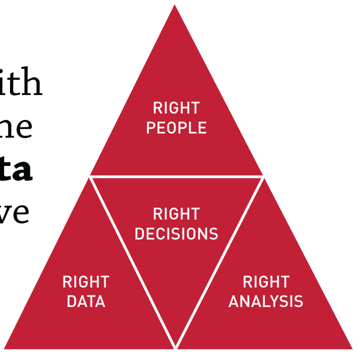


When used appropriately, value-added measures can produce diagnostic analyses with the potential to shape policies that increase students' access to highly effective teachers.

The Strategic Data Project

OVERVIEW

The Strategic Data Project (SDP), housed at the Center for Education Policy Research at Harvard University, partners with school districts, school networks, and state agencies across the United States. **Our mission is to transform the use of data in education to improve student achievement.** We believe that with the right people, the right data, and the right analyses, we can improve the quality of strategic policy and management decisions.



SDP AT A GLANCE

60	**AGENCY PARTNERS**
37	SCHOOL DISTRICTS
13	STATE EDUCATION DEPARTMENTS
7	NONPROFIT ORGANIZATIONS
3	CHARTER SCHOOL ORGANIZATIONS
108	**FELLOWS**
65	CURRENT
43	ALUMNI

CORE STRATEGIES

1. Building a network of top-notch data strategists who serve as fellows for two years with our partners

2. Conducting rigorous diagnostic analyses of teacher effectiveness and college-going success using existing agency data

3. Disseminating our tools, methods, and lessons learned to the education sector broadly

SDP DIAGNOSTICS

SDP's second core strategy, conducting rigorous diagnostic analyses using existing agency data, focuses on two core areas: (1) college-going success and attainment for students and (2) human capital (primarily examining teacher effectiveness).

The diagnostics are a set of analyses that frame actionable questions for education leaders. By asking questions such as "How well do students transition to postsecondary education?" or "How successfully is an agency recruiting effective teachers?" we support education leaders to develop a deep understanding of student achievement in their agency.

ABOUT THE SDP TOOLKIT FOR EFFECTIVE DATA USE

SDP's third core strategy is to disseminate our tools, methods, and lessons learned to education agencies broadly. This toolkit is meant to help analysts in all education agencies collect data and produce meaningful analyses in the areas of college-going success and teacher effectiveness. Notably, the analyses in this release of our toolkit primarily support questions related to college-going success. The data collection (Identify) and best practices (Adopt) stages of the toolkit, however, are applicable to any sort of diagnostic and convey general data use guidelines valuable to any analysts interested in increasing the quality and rigor of their analyses.

Center for Education Policy Research
HARVARD UNIVERSITY

CENTER FOR EDUCATION POLICY RESEARCH
STRATEGIC DATA PROJECT
50 CHURCH ST., 4TH FLOOR, CAMBRIDGE, MA 02138
VOX 617.496.1563
FAX 617.495.2614
WWW.GSE.HARVARD.EDU/SDP