Jared E. Knowles, Wisconsin Department of Public Instruction
Strategic Data Project Data Building Tasks
May 2012
Last updated: May 13, 2012

Contents

1 Overview

Housed at the Center for Education Policy Research at Harvard University, the Strategic Data Project (SDP) partners with school districts, school networks, and state agencies across the US. **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 significantly improve the quality of strategic policy and management decisions.

Core Strategies
To achieve our mission, SDP pursues three core strategies:

1. Placing top-notch analytic leaders as "Fellows" for two years with our partner agencies;

SDP supports more than 40 Data and Agency Fellows serving partner educational agencies–districts, states, and charter management organizations–across the nation. This number will grow to nearly 70 in 2012.

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

We have completed diagnostics in teacher effectiveness and / or college-going success in seven districts, with more diagnostics currently underway or planned in additional district and state partner agencies.

3. Disseminating our tools, methods, and lessons learned to many more education agencies.

Through the diagnostic analyses, we have developed a body of knowledge around effective data use. The release of this toolkit reflects SDP's third core strategy to spread knowledge and build capacity within educational agencies for effective data use.

SDP DIAGNOSTICS

Our 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. In an effort to make these analyses accessible and more widely used, this toolkit helps analysts collect data and produce analyses associated with the SDP College-Going and Human Capital diagnostics.

Notably, the diagnostic analyses in this release of our toolkit are specific to the College-Going diagnostic. The data collection (Identify), data cleaning (Clean), and best practices (Adopt) stages of the toolkit, however, are applicable to either diagnostic and convey general data use guidelines valuable to any analysts interested in increasing the quality and rigor of their analyses. Later releases will address the analyses in our Human Capital diagnostic.

2 Introduction

SDP Data Building Tasks

Congratulations on identifying the data elements that are essential for conducting rigorous analyses in your organization. **Clean** is the next stage in the SDP Toolkit for Effective Data Use. To successfully move through the **Clean** stage, you should review the **Iden-**

tify component of this toolkit. Upon completing this stage, you will have produced clean research files that will allow you to **Connect** and **Analyze** data related to college-going success in your agency.

THE TASKS

Clean consist of five tasks that share a similar structure. The tasks are geared toward analysts with at least moderately strong data background and comfort with statistics. Each task provides hands-on experience building specific components of the research file used for the SDP CollegeGoing Diagnostic Analyses.

The tasks are listed as follows:

Task 1 Student Attributes

Task 2 Student School Year

Task 3 Identifying the Ninth Grade Cohort

Task 4 Student School Enrollment

Task 5 Prior Achievement

Each task is accompanied by a practice file dataset upon which all data snapshots and output are based. These datasets consist of simulated data that have been fully de-identified.We strongly recommend that you use these datasets to work through the tasks and check your answers. The datasets are available for download at www.gse.harvard.edu/sdp/tools. Note that the tasks follow a logical sequence from Task 1 to Task 5, and some tasks require the output of previous tasks. However, because we provide all necessary practice files for each task, you may also choose to work on the tasks out of order. For instance, you may be first interested in identifying the ninth-grade cohort for students in your agency with Task 3.

To successfully complete all parts of this toolkit, however, you should work your way through all five tasks. The output of each task will be needed to successfully complete the Connect and Analyze stages of the toolkit. Lastly, it is important to note that the tasks do not show you how to develop every single component and detail of the files to be used in Connect and Analyze. Our goal is to equip you with an understanding for the core process of constructing robust, clean research files. We do, however, aim to explicitly indicate what additional elements are needed in the DATA DESCRIPTION section of each task to deliver a fully realized research file. Furthermore, we also provide a DECISION RULES GLOSSARY in the Appendix at the end of this document to provide guidance on how to approach the cleaning process for these additional elements.

For those who are less familiar with or who need to brush up on Stata use, we also include a STATA GLOSSARY of commonly used commands in the Appendix at the end of this document. Through this set of tasks, you will learn effective practices for: data transformations, new variable construction, and the implementation of key decision rules.

TASK STRUCTURE
The core of each task is a set of step-by-step instructions that guide you through the work. For each task you will find:

- Purpose — Clarifies the importance of the task.
- How to Start — Identifies the input file(s) you will need to complete the task and guidelines for apply the task to your own agency's data.
- Data Description — Lists the data elements you will need to complete the task and describes the uniqueness of key data elements.
- Instructions — Provides logical instructions on transforming the data with Stata code and fill-in-the-blank snapshots that help you visualize changes to your data.
- Solutions — Provides answers for the data snapshot exercises.

After completing these tasks, you will be well-positioned to use your own agencys data to construct similar clean research files needed in the Connect and Analyze stages.

Finally, if you find yourself in need of additional guidance, the friendly research team at SDP is available to help: sdp@gse.harvard.edu

3 Task 1: STUDENT ATTRIBUTES

3.1 PURPOSE

Through Task 1: Student Attributes, you will take the raw Student Attributes file and generate a cleaned Student Attributes output file that has only one observation per student. These data will allow you to examine college-going outcomes by race/ethnicity.

The core assignments of this task are to:

1. Resolve instances in which the same student appears with different values for race/ethnicity in different years. Our goal is to have only one race/ethnicity associated with each student.

2. Drop duplicate observations so the file is unique by student–that is, it contains only one observation per student. Upon completing this task, you will be have a clean Student_Attributes file that can then be used as to create the analysis file in Connect. From Task 1, Task 2 is a natural next step, in which you will clean the Student_School_Year file in preparation for Task 3 and Task 4.

3.2 HOW TO START

To begin, open the provided Student_Attributes practice file.

```r
# Read in Stata
library(foreign)  # required for .dta files
stuatt <- read.dta("data/Student_Attributes.dta")  # read data in the
data subdirectory
# We can also convert to .csv and import
```

The input file contains data for school years 2000-01 through 2006-07. Normally race is considered a time-invariant variable that is unique by student. In this instance, we deal with a case in which race is stored in a file unique by student and school year, which is instead time-variant. This task aims to take convert the dataset from being time-variant to being time-invariant.

If this is your first time going through the task, we recommend starting with the practice file, rather than your agency's own data file. Doing so will help you learn SDP's cleaning methodology and allow you to easily check your answers from a common dataset. You may then apply these methods to your agency's own Student_Attributes data with confidence. To learn more about the data you will need to collect in your agency, refer to Identify: Data Specification Guide and the DATA DESCRIPTION section of this document.

In addition to the practice file, you may also find it useful to complete the data snapshot exercises provided in the task. These exercises will allow you to visualize changes to the data occurring in each step of the task. Solutions for the exercises are provided at the end of the task.

3.3 DATA DESCRIPTION

In Identify: Data Specification Guide, we specify the data elements included in the Student_Attributes research file.[1]

In this task, we examine a partial version of the Student_Attributes file that includes only sid, school_year, and race_ethnicity. This partial version is presented to help you learn the Student_Attributes cleaning process to make a file unique by sid without having to worry about additional Student_Attributes variables such as male, hs_diploma , hs_diploma_type,

[1] You may be wondering how this specification compares to the version in Identify: Data Specification Guide. Here are the primary changes: First, the race_ethnicity variable is coded as a string rather than being numeric, as specified in the Data Specification Guide. You will correct this in the task as it will facilitate the process of making the file unique by sid.Second, we are examining a time-variant data set. In the Data Specification Guide, the Student_Attributes file is specified as being unique by sid. In this case, the data are time-variant and unique by sid and school_year. Note that some districts may actually store race_ethnicity in a time-variant form such as this, and it is our job through this task to make the data time-invariant, i.e. each student only has a single value for race_ethnicity across time. Third, we are examining a partial data set including only sid, school_year, and race_ethnicity. We do not include variables such as male, hs_diploma, or hs_diploma_type, or hs_diploma_date to simplify the task. These variables are essential for later analyses but are left for you to complete as a further exercise.For guidance on cleaning these additional variables, refer to the DECISION RULES GLOSSARY at the end of this document and use this task as a reference.

or hs_diploma_date. The relevant variables and definitions you will need to complete the task are illustrated below:

```
str(stuatt)

'data.frame': 59606 obs. of  3 variables:
 $ sid           : int  1 1 1 1 2 2 3 3 3 4 ...
 $ school_year   : int  2004 2005 2006 2007 2006 2007 2005 2006 2007 2005 ...
 $ race_ethnicity: chr  "B" "H" "H" "H" ...
 - attr(*, "datalabel")= chr ""
 - attr(*, "time.stamp")= chr "30 Jan 2012 15:37"
 - attr(*, "formats")= chr  "%10.0g" "%10.0g" "%9s"
 - attr(*, "types")= int  252 252 3
 - attr(*, "val.labels")= chr  "" "" ""
 - attr(*, "var.labels")= chr  "sdpsid" "schoolyear" "raceethnicity"
 - attr(*, "version")= int 12
```

Uniqueness: ideally, the data in its raw form would be unique by sid. However, this may not be the case as some agencies might record race_ethnicity in a time-variant manner, such as by school year. To address this, we explain how to take the raw research file from being unique by sid and school_year to being unique by sid alone. Once the file is unique by sid alone, it is ready to be incorporated into the analysis file in the Connect stage. Examine your Student_Attributes raw research file input dataset. According to the data specification, the file should be unique by sid. Examine the snapshot below to determine if it is unique as described.

```
head(stuatt)

  sid school_year race_ethnicity
1   1        2004              B
2   1        2005              H
3   1        2006              H
4   1        2007              H
5   2        2006              W
6   2        2007              B
```

Recode the raw race_ethnicity variable as numeric. Race_ethnicity is currently coded as a string variable, which is how some agencies may store this data . Replace the string values with numeric values as shown below. This numeric race variable will be easier to use in later stages of the task.

1= African American, not Hispanic
2= Asian American
3= Hispanic
4= American Indian
5= White, not Hispanic
6= Multiple / Other

```
stuatt$race_num <- NA  # Create variable race_num
# in data frame stuatt
unique(stuatt$race_ethnicity)  #check current values

[1] "B"   "H"   "W"   "A"   "NA"  "M/O"


# Generate numeric race code using conditional expressions in R (in brackets)

stuatt$race_num[stuatt$race_ethnicity == "B"] <- 1
stuatt$race_num[stuatt$race_ethnicity == "A"] <- 2
stuatt$race_num[stuatt$race_ethnicity == "H"] <- 3
stuatt$race_num[stuatt$race_ethnicity == "NA"] <- 4
stuatt$race_num[stuatt$race_ethnicity == "W"] <- 5
stuatt$race_num[stuatt$race_ethnicity == "M/O"] <- 6
unique(stuatt$race_num)

[1] 1 3 5 2 4 6
```

```
# In R categorical variables are best represented as factors Factors can have
# values, and labels Create a labeled factor for the new race_num variable

stuatt$race_num2 <- factor(stuatt$race_num, labels = c("Black", "Asian",
    "Hispanic", "Native American", "White", "MultipleOther"))

# Compare them to check using a cross-tabulation
table(stuatt$race_ethnicity, stuatt$race_num2)


      Black Asian Hispanic Native American White MultipleOther
  A       0  6588        0               0     0             0
  B   40220     0        0               0     0             0
  H       0     0     7798               0     0             0
  M/O     0     0        0               0     0           106
  NA      0     0        0             147     0             0
  W       0     0        0               0  4747             0


# Replace them
stuatt$race_num <- NULL
stuatt$race_ethnicity <- stuatt$race_num2
stuatt$race_num2 <- NULL

table(stuatt$race_ethnicity)  # counts


          Black           Asian        Hispanic Native American           White
          40220            6588            7798             147            4747
  MultipleOther
            106

prop.table(table(stuatt$race_ethnicity)) * 100  #percentages
```

```
          Black            Asian        Hispanic Native American            White
        67.4764          11.0526         13.0826          0.2466           7.9640
  MultipleOther
         0.1778
```

Check: What does the distribution of your race_ethnicity variable look like? Let's redraw the tables above in a more readable format.

```
library(xtable)  #beautify our output
print(xtable(prop.table(table(stuatt$race_ethnicity)) * 100), include.colnames =
FALSE,
    floating = FALSE, hline.after = NULL)
print(xtable(table(stuatt$race_ethnicity) * 100, digits = 0), include.colnames =
FALSE,
    floating = FALSE, hline.after = NULL)
```

Black	67.48
Asian	11.05
Hispanic	13.08
Native American	0.25
White	7.96
MultipleOther	0.18

Table 1: Proportions

Black	4022000
Asian	658800
Hispanic	779800
Native American	14700
White	474700
MultipleOther	10600

Table 2: Counts

Let's also draw a figure to show this distribution.

```
library(ggplot2)
qplot(stuatt$race_ethnicity, geom = "bar") + theme_bw() + xlab("Race/Ethnicity") +
    ylab("Count")
```

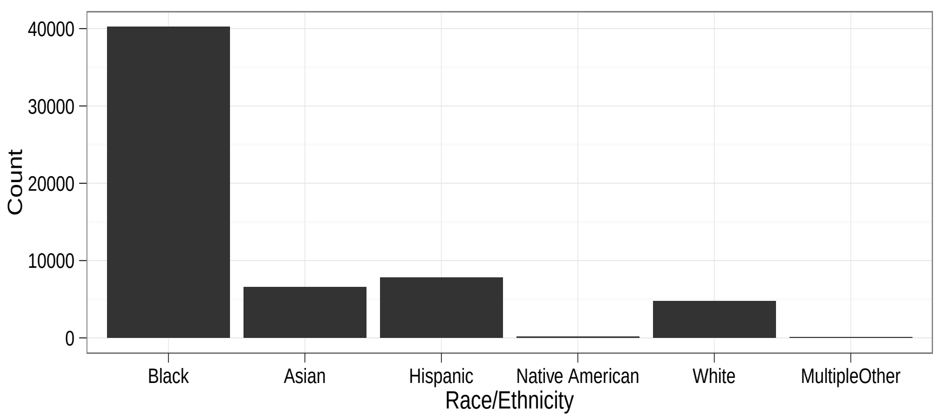

Figure 1: Distribution of Observation Race

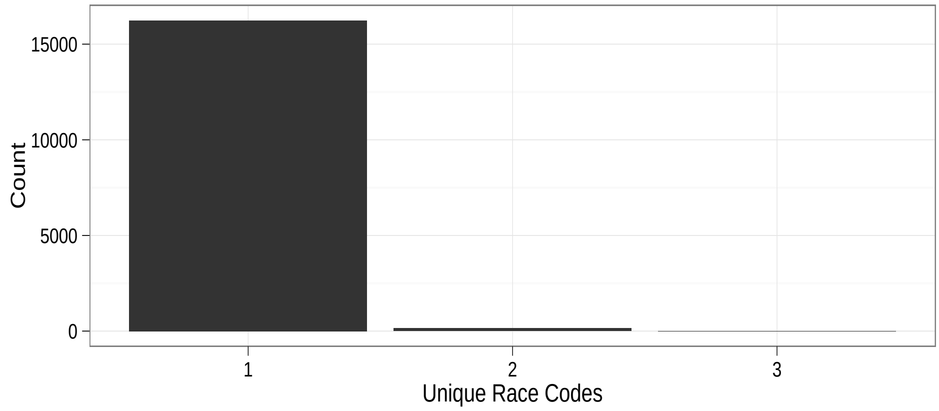

Figure 2: Number of Values

3.4 Consistent Value of race_ethnicity

First, create a variable indicating how many unique values race_ethnicity assumes for each student called nvals_race.

```
# Get number of unique values by sid
nvals <- tapply(stuatt$race_ethnicity, stuatt$sid, function(x) length(unique(x)))
table(nvals)

nvals
    1     2     3
16237   149     5
```

```
library(ggplot2)
qplot(as.factor(nvals), geom = "bar") + theme_bw() + xlab("Unique Race Codes") +
    ylab("Count")
```

Next, for students with more than one value for race_ethnicity, assign the modal value as the student's race.

```
# First we need to create a 'mode' function in R that mimics Stata statamode
# creates a list of the modal values and assigns '.'  If more than one mode
# exists
statamode <- function(x) {
    z <- table(as.vector(x))
    m <- names(z)[z == max(z)]
    if (length(m) == 1) {
        return(m)
    }
    return(".")
}

# Create new data frame for individual student Create nvals while we are at it
library(plyr)  # convenience functions for summarizing data in R
modes <- ddply(stuatt, .(sid), summarize, race_temp = statamode(race_ethnicity),
```

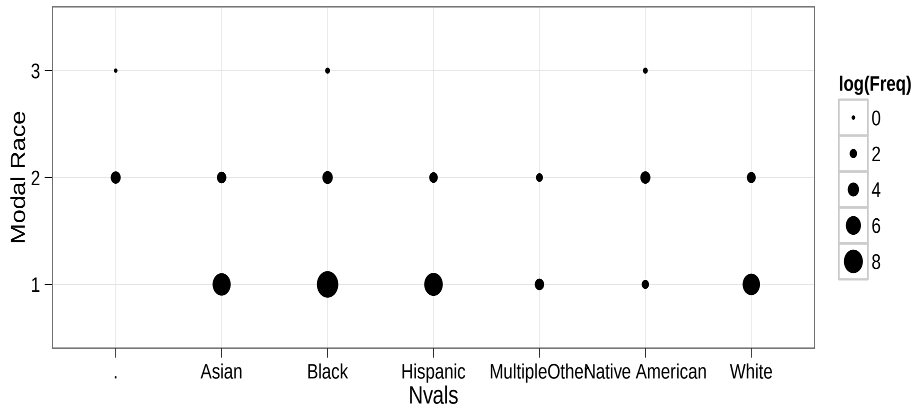

Figure 3: Distribution of Race and NVALS

```
    nvals = length(unique(race_ethnicity)))
tab1 <- table(modes$race_temp, modes$nvals)
addmargins(tab1, FUN = list(Total = sum), quiet = TRUE)



                       1     2     3 Total
  .                    0    28     1    29
  Asian             1924    21     0  1945
  Black            10576    35     2 10613
  Hispanic          2396    14     0  2410
  MultipleOther       20     6     0    26
  Native American      7    29     2    38
  White             1314    16     0  1330
  Total            16237   149     5 16391
```

Check: What does the distribution of the temporary race variable look like for students with only one unique race value and for students with more than one race value?

```
df <- as.data.frame(tab1)
qplot(Var1, Var2, geom = "point", size = log(Freq), data = df) + theme_bw() +
    xlab("Nvals") + ylab("Modal Race")
```

Note, thanks to the power of R, we can create a function to do this for us for other variables in the future and on other datasets:

```
source("functions.R")  # Read in the functions we have written
# All functions are available in the Appendix
a <- nvals(df = "stuatt", id = "sid", year = "school_year", var = "race_ethnicity")
# Here we pass R some characters to tell it which variables we care about This
# allows us to generalize beyond the race variable in the future df = data
# frame, id= student id, year= school year, and var= our variable of interest
head(a)

  sid var_temp nvals most_recent_year most_recent_var
1   1 Hispanic     2             2007        Hispanic
```

```
2   2       .     2             2007          Black
3   3    Black    2             2007          Black
4   4    Black    1             2007          Black
5   5    Black    1             2007          Black
6   7    Black    1             2001          Black

rm(a)
```

C. It appears that we now have 29 students with appended values for the race_temp variable. This occurs because these students' race_ethnicity has two or more modes, so none of the modes is selected as the variable mode in part B of this step. For these students, assign their most recently observed race value as their race_ethnicity.

```
# Create a variable indicating the latest school year
modes <- ddply(stuatt, .(sid), summarize, race_temp = statamode(race_ethnicity),
    nvals = length(unique(race_ethnicity)), most_recent_year = max(school_year),
    most_recent_race = tail(race_ethnicity, 1))
modes$race2[modes$race_temp != "."] <- modes$race_temp[modes$race_temp !=
    "."]
modes$race2[modes$race_temp == "."] <-
as.character(modes$most_recent_race[modes$race_temp ==
    "."])
head(modes)

  sid race_temp nvals most_recent_year most_recent_race    race2
1   1  Hispanic     2             2007         Hispanic Hispanic
2   2         .     2             2007            Black    Black
3   3     Black     2             2007            Black    Black
4   4     Black     1             2007            Black    Black
5   5     Black     1             2007            Black    Black
6   7     Black     1             2001            Black    Black

# Delete old vars on stuatt
stuatt <- subset(stuatt, select = c("sid", "school_year", "race_ethnicity"))
# Assign the value associated with the most recent year as the permanent
# race_ethnicity for the students with missing race
stuatt <- merge(stuatt, modes)
rm(modes)
stuatt$race_ethnicity <- stuatt$race2
stuatt <- subset(stuatt, select = c("sid", "school_year", "race_ethnicity"))
head(stuatt, n = 20)

   sid school_year race_ethnicity
1    1        2004       Hispanic
2    1        2005       Hispanic
3    1        2006       Hispanic
4    1        2007       Hispanic
5    2        2006          Black
6    2        2007          Black
7    3        2005          Black
8    3        2006          Black
9    3        2007          Black
10   4        2005          Black
```

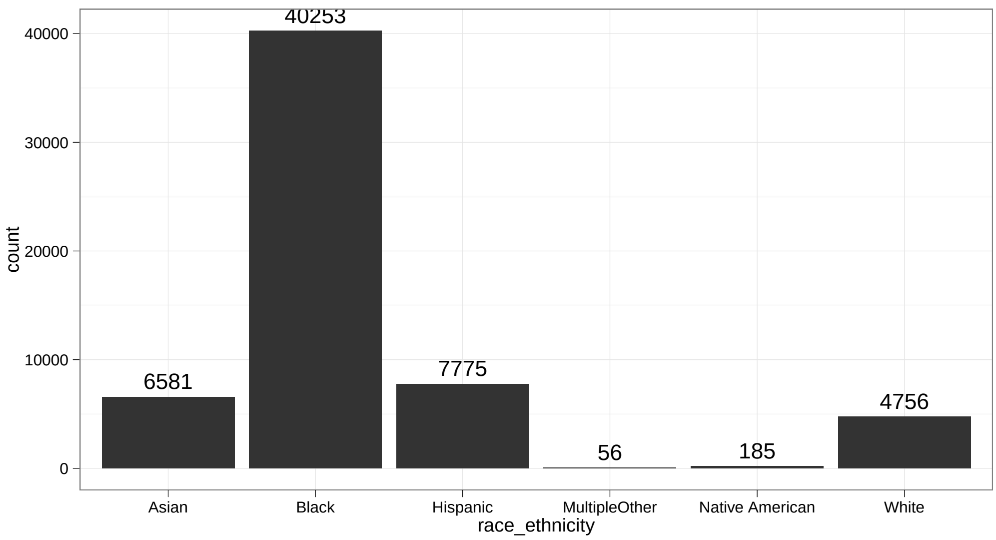

Figure 4: Student Race

```
11   4         2006          Black
12   4         2007          Black
13   5         2007          Black
14   7         2001          Black
15   8         2001          Black
16   9         2001          Black
17   9         2002          Black
18   11        2001          Black
19   11        2002          Black
20   11        2003          Black
```

Check: What is the distribution of the race_ethnicity in the final file? If all steps were completed correctly, the distributions should look exactly the same as in the Check steps for the final race_ethnicity variable in 3C.

```
qplot(race_ethnicity, data = stuatt, geom = "histogram") + theme_bw() +
    stat_bin(geom = "text", aes(label = ..count.., vjust = -0.5))

ymax not defined: adjusting position using y instead
```

Note we can automate all of Task 1 for the future, again using the power of functions:

```
# The Task 1 function starts with our raw data It performs all the tasks above
# And gives us back cleaned data that just needs variable renaming.
a <- task1(df = "stuatt", id = "sid", year = "school_year", var = "race_ethnicity")
head(a)

  sid school_year race_ethnicity var_temp nvals most_recent_year
1   1        2004       Hispanic Hispanic     1             2007
2   1        2005       Hispanic Hispanic     1             2007
3   1        2006       Hispanic Hispanic     1             2007
4   1        2007       Hispanic Hispanic     1             2007
```

```
5   2        2006          Black    Black      1              2007
6   2        2007          Black    Black      1              2007
  most_recent_var     var2
1        Hispanic Hispanic
2        Hispanic Hispanic
3        Hispanic Hispanic
4        Hispanic Hispanic
5           Black    Black
6           Black    Black
```

4 Task 2: STUDENT SCHOOL YEAR

4.1 PURPOSE

Through Task 2: Student School Year, you will take the raw Student School Year research file and generate a clean Student School Year output file that has only one observation per student and school year. First, we will create a time-invariant Free or Reduced Price Lunch (FRPL) variable as a proxy for students' poverty status. Then, we will ensure that only one grade level is assigned per student per school year.

The core assignments of this task is to:

1. Create a variable indicating whether each student was ever eligible for FRPL.
2. Resolve instances in which a student has more than one grade level in a single school year.
3. Drop duplicate observations so the file is unique by student and school year.

Upon completing this task, you will have a data set unique by student and school year, allowing you to assign students to the appropriate ninth grade cohort in Task 3.

4.2 HOW TO START

To begin, open the provided Student_School_Year_Preliminary practice file in R.

```
# This time read from a .csv file
stuyear <- read.csv("data/Student_School_Year.csv")
# Note that in RStudio we can click 'Import Dataset' in the Workspaces View
```

This practice file contains data on student grade level progression and FRPL eligibility through school years 2000-01 through 2006-07 for all grades. This file is unique by student, school_year, and grade_level. If this is your first time going through the task, we recommend starting with the practice file, rather than your agency's own data file. Doing so will help you learn SDP's cleaning methodology and allow you to easily check your answers from a common dataset. You may then apply these methods to your agencys own Student_School_Year data with confidence. To learn more about the data you will need to collect in your agency, refer to Identify: Data Specification Guide and the DATA DESCRIPTION section of this document. In addition to the practice file, you may also find it useful to complete the data snapshot exercises provided in the task. These exercises will allow you to visualize changes to the data occurring in each step of the task. Solutions for the exercises are provided at the end of the task.

4.3 DATA DESCRIPTION

In Identify: Data Specification Guide, we specify the data elements included in the Student_School_Year research file. In this task, however, we consider a partial version of the Student_School_Year file that includes only sid, school_year, grade_level, and frpl. This partial version is presented to help you learn the Student_School_Year cleaning process to make a file unique by sid and school_year without having to worry about additional Student_School_Year variables such as iep, ell, gifted, or days_enrolled. The relevant variables and definitions you will need to complete the task are illustrated below:

```
str(stuyear)

'data.frame': 56044 obs. of  4 variables:
 $ sid        : int  1 1 1 1 2 2 3 3 3 4 ...
 $ school_year: int  2004 2005 2006 2007 2006 2007 2006 2007 2007 2005 ...
 $ grade_level: int  9 9 10 11 10 11 10 8 9 11 ...
 $ frpl       : Factor w/ 3 levels "F","N","R": 2 2 3 3 1 1 1 1 1 2 ...
```

Uniqueness: ideally, this data set would be unique by sid + school_year. However, this may not be the case as some students may transfer grades mid-year and be reported with two separate grades within one school year. As they transfer, it is possible that their FRPL status might change as well. To address these circumstances, we explain how to take the raw research file from NOT being unique by sid and school_year to being unique by sid and school_year. Once the file is unique by sid and school_year, it is considered clean and ready for use in Task 3.

4.4 INSTRUCTIONS

4.4.1 Examine the Data Set

Examine your Student_Characteristics raw research input dataset. According to the data specification, the file should be unique by sid and school_year. Examine the snapshot below to determine if it is unique as described.[2]

```
head(stuyear, n = 12)

   sid school_year grade_level frpl
1    1        2004           9    N
2    1        2005           9    N
3    1        2006          10    R
4    1        2007          11    R
5    2        2006          10    F
6    2        2007          11    F
7    3        2006          10    F
8    3        2007           8    F
```

[2] Note that the student with sid of 3 has two different values for grade_level in 2007. The same is true of the student with sid of 5. Thus, the file is not unique by sid and school_year. The goal of the first half of this task is to create a time-invariant frpl binary variable that captures whether or not the student ever qualified for FRPL. The second half of the task will resolve issues of multiple grade_level observations within the same sid and school_year. This will make the file unique by sid and school_year.

```
9    3          2007            9     F
10   4          2005           11     N
11   4          2006           10     N
12   4          2007            9     N
```

4.4.2 Recode the Raw FRPL as Binary

Recode the raw frpl variable as binary. Frpl is currently coded as a string variable with separate designations for "reduced lunch" and "free lunch." Replace the string values with numeric values, combining the two designations into one, as shown below. This binary frpl variable will be useful to define a student as having ever been FRPL.

0= "N"
1= "R"
1= "F" (Do not distinguish between free and reduced price lunch)

```
stuyear$frpl_num <- 0  # create new variable
stuyear$frpl_num[stuyear$frpl == "R"] <- 1  #recode
stuyear$frpl_num[stuyear$frpl == "F"] <- 1
stuyear$frpl <- stuyear$frpl_num  # Replace frpl with new variable
stuyear$frpl_num <- NULL  # Drop
head(stuyear, n = 6)

  sid school_year grade_level frpl
1   1        2004           9    0
2   1        2005           9    0
3   1        2006          10    1
4   1        2007          11    1
5   2        2006          10    1
6   2        2007          11    1
```

Check: What does the distribution of the new numeric frpl variable look like?

```
addmargins(table(stuyear$frpl))


    0     1   Sum
14827 41217 56044
```

4.4.3 Create a binary indicator

Create a binary indicator equal to 1 if the student was ever eligible for FRPL, and 0 otherwise. In other words, if the student has a frpl value of 1 in any year, create a binary variable equal to 1 in all observations for that student; conversely, if a student never appears as being FRPL, the binary variable would be equal to 0 in all observations for that student.

```
stu <- ddply(stuyear, .(sid), summarize, ever_frpl = max(frpl))  # Create variable by
student
stuyear <- merge(stuyear, stu)  # merge back
```

Check: How many students have ever been FRPL?

```
addmargins(table(stu$ever_frpl))


    0     1   Sum
 2801 13090 15891
```

4.4.4 Resolve multiple grade-year entries

Resolve instances in which a student has more than one grade level listed for a given school year.

There are a total of 18 occasions in which a student has duplicate schoolyear observations.

```
stuyear$dupes <- dedupe(stuyear[, 1:2])  # Create indicator for all duplicated rows
table(stuyear$dupes)  # count them, TRUE is a duplicated element


FALSE  TRUE
56008    36
```

The SDP decision rule is to keep the highest grade_level when a student has multiple grade levels within the same year.

```
# In R it is faster to subset out the duplicated elements, fix them, and merge
# them back in
dupes <- subset(stuyear, dupes == TRUE)
for (i in dupes$sid) {
    dupes$grade_level[dupes$sid == i] <- max(dupes$grade_level[dupes$sid == i])
}
stuyear <- rbind(dupes, subset(stuyear, dupes == FALSE))
stuyear <- stuyear[with(stuyear, order(as.numeric(row.names(stuyear)))),
    ]
head(stuyear)

  sid school_year grade_level frpl ever_frpl dupes
1   1        2004           9    0         1 FALSE
2   1        2005           9    0         1 FALSE
3   1        2006          10    1         1 FALSE
4   1        2007          11    1         1 FALSE
5   2        2006          10    1         1 FALSE
6   2        2007          11    1         1 FALSE
```

4.4.5 Drop Duplicated Values

Drop any duplicate observations so the file is unique by student and school_year-that is, it contains only one observation for each student-school_year combination. We can confidently do that now, because the variables we need to keep are timeinvariant: their values are constant across years for every student.

```
stuyear$dupes <- duplicated(stuyear[, 1:2])  # Create new dupe indicator for
# one duplicate value

table(stuyear$dupes)


FALSE  TRUE
56026    18

stuyear <- subset(stuyear, dupes == FALSE)  # drop all duplicated terms
stuyear$dupes <- NULL  # Indicator not needed
head(stuyear, n = 10)

   sid school_year grade_level frpl ever_frpl
1    1        2004           9    0         1
2    1        2005           9    0         1
3    1        2006          10    1         1
4    1        2007          11    1         1
5    2        2006          10    1         1
6    2        2007          11    1         1
7    3        2006          10    1         1
8    3        2007           9    1         1
10   4        2005          11    0         0
11   4        2006          10    0         0
```

4.4.6 Export Data

```
# Not run write.csv(stuyear,file='data/Student_School_Year_Intermediate.csv') #
# CSV write.dta(stuyear,file='data/Student_School_Year_Intermediate.dta') #
# STATA
```