**STATISTICAL CODE: A COMPARISON OF METHODS FOR ESTIMATING STATE SUBGROUP PERFORMANCE ON THE NATIONAL ASSESSMENT OF EDUCATIONAL PROGRESS**

This file provides the statistical code used in the dissertation, *A Comparison of Methods for Estimating State Subgroup Performance on the National Assessment of Educational Progress*, in the order in which corresponding analyses are presented in the dissertation. For each set of code, the computer program and file type used for running the analysis are named (e.g., R Studio/.RMD File). In addition, the section of the dissertation where the corresponding analysis appears is named (e.g., “Computing Direct Estimates”). Throughout this document, sample sizes are deliberately censored to comply with National Center for Education Statistics reporting policy.

**Contents**

[**Steps and Code for Calculating Median Standard Deviations of States for Subgroups of Interest (Chapter 3)**](#StepsandCodeforCalculatingMedianSt)**3**

[**Steps and Code used for Simulation Study (Chapter 3)**](#StepsandCodeusedorSimulationStudy)**5**

[**MICE check - Determining Range of Credible Values (Chapter 3)**](#MICECheck) **10**

[**The MICE Technique (Chapter 4)**](#TheMICETechnique) **11**

[**Determining Samples Sizes to Draw (Chapter 4)**](#DeterminingSamplesSizestoDraw) **57**

[**Computing Direct Estimates (Chapter 4)**](#ComputingDirectEstimates)**61**

[**Importing Direct Estimate Information from Stata into R and Pooling Parameter Estimates (Chapter 4)**](#ImportingDirectEstimate)**306**

[**The FH Technique (Chapter 4)**](#TheFHTechnique) **329**

[**The FLEX CS Technique (Chapter 4)**](#TheFLEXCSTechnique)**361**

[**Real World Application of FH for B subgroup (Chapter 4)**](#RealWorldApplicationofFHforB)**413**

**Steps and Code for Calculating Median Standard Deviations of States for Subgroups of Interest** **(Chapter 3)**

The text that follows in courier font represents R script. Lines of script that begin with a pound/hashtag symbol (i.e., #) are comments (not code). Note, the name of local filepaths (e.g., “C:/Foldername/Foldername/…/Filename”) are replaced with [filepath] throughout.

**Program/File Type:** R/.R File

**Dissertation Section:** Coverage

**#Import data**

pared <- read\_excel("[filepath]/NDE\_DataPull\_Mean\_SD\_042120.xls",

sheet = "Pared", range = "A9:E269")

race\_eth <- read\_excel("[filepath]/NDE\_DataPull\_Mean\_SD\_042120.xls",

sheet = "RaceEth\_Trend", range = "A9:E321")

ELs <- read\_excel("[filepath]/NDE\_DataPull\_Mean\_SD\_042120.xls",

sheet = "ELs", range = "A9:E113")

**#Data Cleaning**

library(tidyverse)

pared <- pared[, -1]

pared <- pared[, -3]

pared <- pared %>%

rename(subgroup = "Parental education level, from 2 questions")

race\_eth <- race\_eth[, -1]

race\_eth <- race\_eth[, -3]

race\_eth <- race\_eth %>%

rename(subgroup = "Race/ethnicity used to report trends, school-reported")

ELs <- ELs[, -1]

ELs <- ELs[, -3]

ELs <- ELs %>%

rename(subgroup = "Status as English Language Learner, 2 categories")

**#ROW BIND**

dat <- rbind(pared, race\_eth, ELs)

**#Filter out DC and DODEA**

dat <- dat %>%

filter(Jurisdiction != "District of Columbia") %>%

filter(Jurisdiction != "DoDEA") %>%

rename(sd = `Standard deviation`)

**#Change class of variables**

dat$sd <- as.numeric(dat$sd)

**#Compute medians for subgroups of interest**

NHS <- dat %>%

filter(subgroup == "Did not finish high school")

median(NHS$sd, na.rm = T)

HS <- dat %>%

filter(subgroup == "Graduated high school")

median(HS$sd, na.rm = T)

SBA <- dat %>%

filter(subgroup == "Some education after high school")

median(SBA$sd, na.rm = T)

BA <- dat %>%

filter(subgroup == "Graduated college")

median(BA$sd, na.rm = T)

B <- dat %>%

filter(subgroup == "Black")

median(B$sd, na.rm = T)

H <- dat %>%

filter(subgroup == "Hispanic")

median(H$sd, na.rm = T)

API <- dat %>%

filter(subgroup == "Asian/Pacific Islander")

median(API$sd, na.rm = T)

AINA <- dat %>%

filter(subgroup == "American Indian/Alaska Native")

median(AINA$sd, na.rm = T)

TP <- dat %>%

filter(subgroup == "Two or more races")

median(TP$sd, na.rm = T)

EL <- dat %>%

filter(subgroup == "ELL")

median(EL$sd, na.rm = T)

**Steps and Code used for Simulation Study (Chapter 3)**

The text that follows in courierfont represents R script. Lines of script that begin with a pound/hashtag symbol (#) are comments (not code).

**Program/File Type:** R/.R File

**Dissertation Section:** Criteria for Recommending a Technique

#1) Import example (artificial) dataset included in the EdSurvey package (the example dataset includes 16,915 students).

#Install and load EdSurvey package

install.packages("EdSurvey")

library(EdSurvey)

#import data and save to object names "df"

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

#reformat data into a data frame

df <- getData(df, c('composite', 'pared', 'sdracem', 'lep', 'origwt', 'jkunit', 'repgrp1'), omittedLevels = FALSE, addAttributes = TRUE)

#This creates a dataframe with 73 variables-- including plausible values, jackknife replicate weights, and origwt

#assign a unique identifier to df (this will facilitate a splitting/re-merging step later

df$ID <- seq.int(nrow(df))

#2) Determine a plausible number of students that NAEP might typically sample per state-subgroup pair.

# 2a. NAEP samples approximately 2500 students per state-subject-grade.

# (source: https://nces.ed.gov/nationsreportcard/about/statejoin.aspx)

#3) Draw a random sample of 2,500 students from the example dataset (of 16,915 students) to compute estimand means and standard deviations (make sure at least 62 student from subgroups of interest are sampled).

set.seed(2019)

target\_df <- df[sample(nrow(df), 2500, replace = FALSE), ]

#Are there at least 62 students subgroup of intrest in the random sample?

library(dplyr) #loads a package useful for data manipulation

target\_df %>% count(pared)

target\_df %>% count(sdracem) #tells us that there are not enough AINA students (just 39)

target\_df %>% count(lep)

#NB: Students who identify with two or more races are not represented in the example data. Thus, this subgroup is not included in this simulation study.

# Here I command R to randomly sample an extra 39 AINA students and then append those extra 39 AINA students to the "target\_df"

extra\_AINA <- df %>%

filter(sdracem=="Amer Ind/Alaska Natv") %>%

sample\_n(39, replace = FALSE)

target\_df <- rbind(target\_df, extra\_AINA)

#Did I resample any of the same AINA students?

tab <- target\_df %>%

filter(sdracem=="Amer Ind/Alaska Natv") %>%

count(ID) %>%

filter(n > 1)

print(tab$ID)

#ANSWER: Yes, I resampled ID#s 2413, 2753, 5258, 7298, 11786, 15158, 16194 (7 AINA students)

#Remove duplicates from "target\_df"

target\_df <- target\_df[ which(target\_df$ID!=2413),]

target\_df <- target\_df[ which(target\_df$ID!=2753),]

target\_df <- target\_df[ which(target\_df$ID!=5258),]

target\_df <- target\_df[ which(target\_df$ID!=7298),]

target\_df <- target\_df[ which(target\_df$ID!=11786),]

target\_df <- target\_df[ which(target\_df$ID!=15158),]

target\_df <- target\_df[ which(target\_df$ID!=16194),]

target\_df %>% count(sdracem)

#There are now 70 AINA students in "target\_df"

# 4) Calculate means and standard deviation of estimands

SD(subset(target\_df, pared == "Did not finish H.S."), variable = "composite")

SD(subset(target\_df, pared == "Graduated H.S."), variable = "composite")

SD(subset(target\_df, pared == "Some ed after H.S."), variable = "composite")

SD(subset(target\_df, pared == "Graduated college"), variable = "composite")

NHS\_target\_mean <- 263.2467

NHS\_SD <- 31.80236

HS\_target\_mean <- 266.4621

HS\_SD <- 30.60569

SBA\_target\_mean <- 278.4232

SBA\_SD <- 33.0456

BA\_target\_mean <- 288.9831

BA\_SD <- 35.62842

SD(subset(target\_df, sdracem == "Black"), variable = "composite")

SD(subset(target\_df, sdracem == "Hispanic"), variable = "composite")

SD(subset(target\_df, sdracem == "Asian/Pacific Island"), variable = "composite")

SD(subset(target\_df, sdracem == "Amer Ind/Alaska Natv"), variable = "composite")

B\_target\_mean <- 254.8383

B\_SD <- 32.55956

H\_target\_mean <- 260.9449

H\_SD <- 33.51711

API\_target\_mean <- 292.6967

API\_SD <- 36.84079

AINA\_target\_mean <- 272.8829

AINA\_SD <- 32.24216

SD(subset(target\_df, lep == "Yes"), variable = "composite")

EL\_target\_mean <- 240.929

EL\_SD <- 34.70914

# 5) Remove "target\_df" from "df" to create a "test\_df"

#testing

test\_df <- rbind(df, target\_df)

dup\_tab <- table(test\_df$ID)

dup\_tab <- as.data.frame(dup\_tab)

names(dup\_tab)[1] <- 'ID'

test\_df <- merge(test\_df, dup\_tab, by="ID")

test\_df <- subset(test\_df, Freq == 1)

# 6) Repeatedly draw samples of size equal to those used for corresponding target intervals per subgroup from the pseudo-population; example data from EdSurvey package (N=16,915).

NHS\_n <- 192

HS\_n <- 446

SBA\_n <- 445

BA\_n <- 1091

B\_n <- 491

H\_n <- 360

API\_n <- 110

AINA\_n <- 70

EL\_n <- 143

test\_df\_NHS <- subset(test\_df, pared == "Did not finish H.S.")

test\_df\_HS <- subset(test\_df, pared == "Graduated H.S.")

test\_df\_SBA <- subset(test\_df, pared == "Some ed after H.S.")

test\_df\_BA <- subset(test\_df, pared == "Graduated college")

test\_df\_B <- subset(test\_df, sdracem == "Black")

test\_df\_H <- subset(test\_df, sdracem == "Hispanic")

test\_df\_API <- subset(test\_df, sdracem == "Asian/Pacific Island")

test\_df\_AINA <- subset(test\_df, sdracem == "Amer Ind/Alaska Natv")

test\_df\_EL <- subset(test\_df, lep == "Yes")

NHS\_means <- replicate(1000, {

random\_sample <- test\_df\_NHS[sample(nrow(test\_df\_NHS), NHS\_n, replace=F), ]

results <- edsurveyTable(composite ~ 1, data = random\_sample)

results[["data"]][["MEAN"]]

})

HS\_means <- replicate(1000, {

random\_sample <- test\_df\_HS[sample(nrow(test\_df\_HS), HS\_n, replace=F), ]

results <- edsurveyTable(composite ~ 1, data = random\_sample)

results[["data"]][["MEAN"]]

})

SBA\_means <- replicate(1000, {

random\_sample <- test\_df\_SBA[sample(nrow(test\_df\_SBA), SBA\_n, replace=F), ]

results <- edsurveyTable(composite ~ 1, data = random\_sample)

results[["data"]][["MEAN"]]

})

BA\_means <- replicate(1000, {

random\_sample <- test\_df\_BA[sample(nrow(test\_df\_BA), BA\_n, replace=F), ]

results <- edsurveyTable(composite ~ 1, data = random\_sample)

results[["data"]][["MEAN"]]

})

B\_means <- replicate(1000, {

random\_sample <- test\_df\_B[sample(nrow(test\_df\_B), B\_n, replace=F), ]

results <- edsurveyTable(composite ~ 1, data = random\_sample)

results[["data"]][["MEAN"]]

})

H\_means <- replicate(1000, {

random\_sample <- test\_df\_H[sample(nrow(test\_df\_H), H\_n, replace=F), ]

results <- edsurveyTable(composite ~ 1, data = random\_sample)

results[["data"]][["MEAN"]]

})

API\_means <- replicate(1000, {

random\_sample <- test\_df\_API[sample(nrow(test\_df\_API), API\_n, replace=F), ]

results <- edsurveyTable(composite ~ 1, data = random\_sample)

results[["data"]][["MEAN"]]

})

AINA\_means <- replicate(1000, {

random\_sample <- test\_df\_AINA[sample(nrow(test\_df\_AINA), AINA\_n, replace=F), ]

results <- edsurveyTable(composite ~ 1, data = random\_sample)

results[["data"]][["MEAN"]]

})

EL\_means <- replicate(1000, {

random\_sample <- test\_df\_EL[sample(nrow(test\_df\_EL), EL\_n, replace=F), ]

results <- edsurveyTable(composite ~ 1, data = random\_sample)

results[["data"]][["MEAN"]]

})

#Bind means into a single dataframe

All\_means <- cbind(NHS\_means, HS\_means)

All\_means <- cbind(All\_means, SBA\_means)

All\_means <- cbind(All\_means, BA\_means)

All\_means <- cbind(All\_means, B\_means)

All\_means <- cbind(All\_means, H\_means)

All\_means <- cbind(All\_means, API\_means)

All\_means <- cbind(All\_means, AINA\_means)

All\_means <- cbind(All\_means, EL\_means)

All\_means <- as.data.frame(All\_means)

colnames(All\_means) <- c("NHS", "HS", "SBA", "BA", "B", "H", "API", "AINA", "EL")

#Save dataset so that I do not need to execute time-consuming code if log out.

write.csv(All\_means, "[filepath]/All\_test\_means.csv")

# 7) Create variables indicating absolute standardized mean difference between means from sets of 1000 and the

# corresponding target means (with sd associated with target sample used as denominator )

All\_means <- All\_means %>%

mutate(NHS\_MSD = abs((NHS - NHS\_target\_mean)/ NHS\_SD),

HS\_MSD = abs((HS - HS\_target\_mean)/ HS\_SD),

SBA\_MSD = abs((SBA - SBA\_target\_mean)/ SBA\_SD),

BA\_MSD = abs((BA - BA\_target\_mean)/ BA\_SD),

B\_MSD = abs((B - B\_target\_mean)/ B\_SD),

H\_MSD = abs((H - H\_target\_mean)/ H\_SD),

API\_MSD = abs((API - API\_target\_mean)/ API\_SD),

AINA\_MSD = abs((AINA - AINA\_target\_mean)/ AINA\_SD),

EL\_MSD = abs((EL - EL\_target\_mean)/ EL\_SD))

# 8) Compute "hit rates," proportion of times Braun's d is equal or less than 0.2

Hit\_rates <- All\_means %>% summarize(NHS\_hit\_rate = mean(NHS\_MSD <= 0.2),

HS\_hit\_rate = mean(HS\_MSD <= 0.2),

SBA\_hit\_rate = mean(SBA\_MSD <= 0.2),

BA\_hit\_rate = mean(BA\_MSD <= 0.2),

B\_hit\_rate = mean(B\_MSD <= 0.2),

H\_hit\_rate = mean(H\_MSD <= 0.2),

API\_hit\_rate = mean(API\_MSD <= 0.2),

AINA\_hit\_rate = mean(AINA\_MSD <= 0.2),

EL\_hit\_rate = mean(EL\_MSD <= 0.2))

# 9) Wrangle all informative statistics from this simulation study into a summary table

subgroup <- c("NHS", "HS", "SBA", "BA", "B", "H", "API", "AINA", "EL")

target\_x <- c(NHS\_target\_mean, HS\_target\_mean, SBA\_target\_mean, BA\_target\_mean, B\_target\_mean,

H\_target\_mean, API\_target\_mean, AINA\_target\_mean, EL\_target\_mean)

target\_sd <- c(NHS\_SD, HS\_SD, SBA\_SD, BA\_SD, B\_SD,

H\_SD, API\_SD, AINA\_SD, EL\_SD)

n <- c(NHS\_n, HS\_n, SBA\_n, BA\_n, B\_n,

H\_n, API\_n, AINA\_n, EL\_n)

Hit\_rates <- t(Hit\_rates)

summary\_table <- cbind(subgroup, target\_x)

summary\_table <- cbind(summary\_table, target\_sd)

summary\_table <- cbind(summary\_table, n)

summary\_table <- cbind(summary\_table, Hit\_rates)

summary\_table <- as.data.frame(summary\_table)

rownames(summary\_table) <- 1:nrow(summary\_table)

summary\_table <- summary\_table %>%

rename(target\_x = V2,

target\_sd = V3,

n = V4,

hit\_rate = V5,

subgroup = V1)

# 10) Export summary table

write.csv(summary\_table, "[filepath]/sim-study-MSD-summary-table.csv")

**MICE check - Determining Range of Credible Values (Chapter 3)**

The text that follows in courier font represents R script. Lines of script that begin with a pound/hashtag symbol (i.e., #) are comments (not code). Note, the name of local filepaths (e.g., “C:/Foldername/Foldername/…/Filename”) are replaced with [filepath] throughout.

**Program/File Type:** R/.R File

**Dissertation Section:** Verifying credibility of MICE-produced predicted values

#Import Test Sample Data

library(readr)

TestSampleAchievement <- read\_csv("[filepath]/TestSampleAchievement.csv")

#write function for computing non-outlier bounds

non\_outlier\_bounds <- function(x){

q1 <- quantile(x, .25, na.rm = TRUE)

q3 <- quantile(x, .75, na.rm = TRUE)

IQR <- q3 - q1

lowerbound <- q1 - (1.5\*IQR)

upperbound <- q3 + (1.5\*IQR)

print(c(lowerbound, upperbound))

}

#apply function to columns of test sample

bounds\_df <- apply(TestSampleAchievement[2:19], 2, non\_outlier\_bounds)

###Generating dotplot with hypothetical imputed values###

####For examining plausibility of values####

library(ggplot2)

library(dplyr)

library(extrafont)

loadfonts(device="win")

#set seed for reproducibility

set.seed(1)

#randomly draw 48 values from a normal distribution with a mean 265 and standard deviation 10.

hypothetical\_scores <- rnorm(48, 265, 10)

#build graph

as.data.frame(hypothetical\_scores) %>%

ggplot(., aes(x=1, y=hypothetical\_scores)) +

geom\_dotplot(binwidth = 1, binaxis = "y", stackdir = "center") +

theme(axis.title.x=element\_blank(),

axis.text.x=element\_blank(),

axis.ticks.x=element\_blank()) +

geom\_hline(yintercept = 278, color="red") +

geom\_hline(yintercept = 252, color="red") +

ylab("Score") +

theme(text=element\_text(size=14, family="serif"))

**The MICE Technique (Chapter 4)**

The text that follows in courier font represents R script. Lines of script that begin with a pound/hashtag symbol (i.e., #) are comments (not code). Note, the name of local filepaths (e.g., “C:/Foldername/Foldername/…/Filename”) are replaced with [filepath] throughout. Three consecive backticks (```) demarcate the start and end of a chunk (i.e., section) of code.

**Program/File Type:** R Studio/.RMD File

**Dissertation Section:** Estimates of Mean Math Achievement with the MICE technique

### Install/Load packages

```{r, message=FALSE, warning=FALSE}

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")

if(!require(mice)) install.packages("mice", repos = "http://cran.us.r-project.org")

if(!require(matrixStats)) install.packages("matrixStats", repos = "http://cran.us.r-project.org")

if(!require(extrafont)) install.packages("extrafont", repos = "http://cran.us.r-project.org")

loadfonts(device="win") #for loading fonts from "extrafont" package

```

### Import test sample (& some data manipulation)

```{r, message=FALSE, warning=FALSE}

df <- read\_csv("[filepath]/TestSampleAchievement.csv")

#Change variable name of "2+" for compatibility with mice function

df <- rename(df, "TP" = "2+")

#Remove state variable (column)

df\_mice <- df[, -1]

```

### Define Visiting Sequence

```{r, message=FALSE, warning=FALSE}

vis <- c("API", "TP", "BA", "SBA", "HS", "B", "NHS", "H", "AINA", "EL")

```

### Construct Predictor Matrix

```{r, message=FALSE, warning=FALSE}

for\_pred\_matrix <- mice(df\_mice, maxit = 0, print=F) #Enables creation a predictor matrix, which will be subsequently edited

pred\_matrix <- for\_pred\_matrix$pred #Creates an initial data matrix with 0s along the diagonal and 1s everywhere else

#Next sets of code remove 1s where I do not want corresponding columns to predict rows

pred\_matrix[1:2, ] <- 0

pred\_matrix[7, ] <- 0

pred\_matrix[14:18, ] <- 0

#these three lines assign 0s to columns for rows without missing data

pred\_matrix[3, 2] <- 0

pred\_matrix[3, 10] <- 0

pred\_matrix[3, 12:13] <- 0

pred\_matrix[3, 15:18] <- 0

#These four lines assign 0s to select columns in the NHS row

pred\_matrix[4, 7:13] <- 0

pred\_matrix[4, 15] <- 0

#These two lines assign 0s to select columns in the HS row

pred\_matrix[5, 7:13] <- 0

pred\_matrix[5, 15] <- 0

#These two lines assign 0s to select columns in the SBA row

pred\_matrix[6, 3] <- 0

pred\_matrix[6, 8:13] <- 0

pred\_matrix[6, 15] <- 0

#These three lines assign 0s to select columns from the BA row

pred\_matrix[8, 2] <- 0

pred\_matrix[8, 5:6] <- 0

pred\_matrix[8, 10:18] <- 0

#These three lines assign 0s to select cloumns from the B row

pred\_matrix[9, 1:2] <- 0

pred\_matrix[9, 4:7] <- 0

pred\_matrix[9, 10:12] <- 0

pred\_matrix[9, 14:18] <- 0

#These four lines assign 0s to select columns from the H row

pred\_matrix[10, 1] <- 0

pred\_matrix[10, 3:9] <- 0

pred\_matrix[10, 11:14] <- 0

pred\_matrix[10, 16:18] <- 0

#These three lines assign 0s to select columns from the API row

pred\_matrix[11, 1:9] <- 0

pred\_matrix[11, 12] <- 0

pred\_matrix[11, 14:18] <- 0

#These three lines assign 0s to select columns from the AINA row

pred\_matrix[12, 1:6] <- 0

pred\_matrix[12, 8:11] <- 0

pred\_matrix[12, 13] <- 0

pred\_matrix[12, 15:18] <- 0

#These three lines assign 0s to select columns fron the 2+ (TP) row

pred\_matrix[13, 1:8] <- 0

pred\_matrix[13, 10] <- 0

pred\_matrix[13, 12] <- 0

pred\_matrix[13, 14:18] <- 0

#These four lines assign 0s to select columns from the EL row

```

### Execute MICE procedure

#\*\*NHS\*\*

```{r, message=FALSE, warning=FALSE}

#First for the NHS variable:

mice\_procedure <- function(x){

df\_mice1 <- df\_mice

df\_mice1[x, 3] <- NA

mice(df\_mice1, m = 100, maxit = 15, visitSequence = vis, method = "norm", pred = pred\_matrix, seed = 2019)

}

NHS\_results <- lapply(1:length(df\_mice$NHS), mice\_procedure)

#A couple of things to note about the code above.

# First, 3 can also be set to "NHS" in the line that subsets df\_mice1.

# Second, setting m to 200 and maxit to 20 previously crashed my computer. #Thus, 100/15 is specified.

```

# \*\*Organize reults for NHS and compute mean and variance\*\*

```{r, message=FALSE, warning=FALSE}

NHS\_results\_df <- NHS\_results[[1]][["imp"]][["NHS"]] %>%

slice(1:2) %>%

rbind(NHS\_results[[3]][["imp"]][["NHS"]]) %>%

slice(-3, -5) %>%

rbind(NHS\_results[[4]][["imp"]][["NHS"]]) %>%

slice(-4, -6) %>%

rbind(NHS\_results[[5]][["imp"]][["NHS"]]) %>%

slice(-5, -7) %>%

rbind(NHS\_results[[6]][["imp"]][["NHS"]]) %>%

slice(-6, -8) %>%

rbind(NHS\_results[[7]][["imp"]][["NHS"]]) %>%

slice(-7, -9) %>%

rbind(NHS\_results[[8]][["imp"]][["NHS"]]) %>%

slice(-8, -10) %>%

rbind(NHS\_results[[9]][["imp"]][["NHS"]]) %>%

slice(-9, -11) %>%

rbind(NHS\_results[[10]][["imp"]][["NHS"]]) %>%

slice(-10, -12) %>%

rbind(NHS\_results[[11]][["imp"]][["NHS"]]) %>%

slice(-11, -13) %>%

rbind(NHS\_results[[12]][["imp"]][["NHS"]]) %>%

slice(-12, -14) %>%

rbind(NHS\_results[[13]][["imp"]][["NHS"]]) %>%

slice(-13, -15) %>%

rbind(NHS\_results[[14]][["imp"]][["NHS"]]) %>%

slice(-14, -16) %>%

rbind(NHS\_results[[15]][["imp"]][["NHS"]]) %>%

slice(-15, -17) %>%

rbind(NHS\_results[[16]][["imp"]][["NHS"]]) %>%

slice(-16, -18) %>%

rbind(NHS\_results[[17]][["imp"]][["NHS"]]) %>%

slice(-17, -19) %>%

rbind(NHS\_results[[18]][["imp"]][["NHS"]]) %>%

slice(-18, -20) %>%

rbind(NHS\_results[[19]][["imp"]][["NHS"]]) %>%

slice(-19, -21) %>%

rbind(NHS\_results[[20]][["imp"]][["NHS"]]) %>%

slice(-20, -22) %>%

rbind(NHS\_results[[21]][["imp"]][["NHS"]]) %>%

slice(-21, -23) %>%

rbind(NHS\_results[[22]][["imp"]][["NHS"]]) %>%

slice(-22, -24) %>%

rbind(NHS\_results[[23]][["imp"]][["NHS"]]) %>%

slice(-23, -25) %>%

rbind(NHS\_results[[24]][["imp"]][["NHS"]]) %>%

slice(-24, -26) %>%

rbind(NHS\_results[[25]][["imp"]][["NHS"]]) %>%

slice(-25, -27) %>%

rbind(NHS\_results[[26]][["imp"]][["NHS"]]) %>%

slice(-26, -28) %>%

rbind(NHS\_results[[27]][["imp"]][["NHS"]]) %>%

slice(-27, -29) %>%

rbind(NHS\_results[[28]][["imp"]][["NHS"]]) %>%

slice(-28, -30) %>%

rbind(NHS\_results[[29]][["imp"]][["NHS"]]) %>%

slice(-29, -31) %>%

rbind(NHS\_results[[30]][["imp"]][["NHS"]]) %>%

slice(-30, -32) %>%

rbind(NHS\_results[[31]][["imp"]][["NHS"]]) %>%

slice(-31, -33) %>%

rbind(NHS\_results[[32]][["imp"]][["NHS"]]) %>%

slice(-32, -34) %>%

rbind(NHS\_results[[33]][["imp"]][["NHS"]]) %>%

slice(-33, -35) %>%

rbind(NHS\_results[[34]][["imp"]][["NHS"]]) %>%

slice(-34, -36) %>%

rbind(NHS\_results[[35]][["imp"]][["NHS"]]) %>%

slice(-35, -37) %>%

rbind(NHS\_results[[36]][["imp"]][["NHS"]]) %>%

slice(-36, -38) %>%

rbind(NHS\_results[[37]][["imp"]][["NHS"]]) %>%

slice(-37, -39) %>%

rbind(NHS\_results[[38]][["imp"]][["NHS"]]) %>%

slice(-38, -40) %>%

rbind(NHS\_results[[39]][["imp"]][["NHS"]]) %>%

slice(-39, -41) %>%

rbind(NHS\_results[[40]][["imp"]][["NHS"]]) %>%

slice(-40, -42) %>%

rbind(NHS\_results[[41]][["imp"]][["NHS"]]) %>%

slice(-41, -43) %>%

rbind(NHS\_results[[42]][["imp"]][["NHS"]]) %>%

slice(-42, -44) %>%

rbind(NHS\_results[[43]][["imp"]][["NHS"]]) %>%

slice(-43, -45) %>%

rbind(NHS\_results[[45]][["imp"]][["NHS"]]) %>%

slice(-44) %>%

rbind(NHS\_results[[46]][["imp"]][["NHS"]]) %>%

slice(-46, -47) %>%

rbind(NHS\_results[[47]][["imp"]][["NHS"]]) %>%

slice(-47, -48) %>%

rbind(NHS\_results[[48]][["imp"]][["NHS"]]) %>%

slice(-48, -49) %>%

rbind(NHS\_results[[49]][["imp"]][["NHS"]]) %>%

slice(-49, -50) %>%

rbind(NHS\_results[[50]][["imp"]][["NHS"]]) %>%

slice(-50, -51)

NHS\_results\_df <- mutate(NHS\_results\_df, mean = rowMeans(NHS\_results\_df),

se = rowSds(as.matrix(NHS\_results\_df)))

#Save results to computer a .csv

write.csv(NHS\_results\_df, "[filepath]/NHS-MICE-Results.csv")

```

#\*\*HS\*\*

```{r, message=FALSE, warning=FALSE}

#Now for the HS variable:

mice\_procedure <- function(x){

df\_mice1 <- df\_mice

df\_mice1[x, 4] <- NA

mice(df\_mice1, m = 100, maxit = 15, visitSequence = vis, method = "norm", pred = pred\_matrix, seed = 2019)

}

HS\_results <- lapply(1:length(df\_mice$HS), mice\_procedure)

```

#\*\*Organize results for HS and compute mean and variance\*\*

```{r, message=FALSE, warning=FALSE}

HS\_results\_df <- HS\_results[[1]][["imp"]][["HS"]] %>%

slice(1:2) %>%

rbind(HS\_results[[3]][["imp"]][["HS"]]) %>%

slice(-3, -5) %>%

rbind(HS\_results[[4]][["imp"]][["HS"]]) %>%

slice(-4, -6) %>%

rbind(HS\_results[[5]][["imp"]][["HS"]]) %>%

slice(-5, -7) %>%

rbind(HS\_results[[6]][["imp"]][["HS"]]) %>%

slice(-6, -8) %>%

rbind(HS\_results[[7]][["imp"]][["HS"]]) %>%

slice(-7, -9) %>%

rbind(HS\_results[[8]][["imp"]][["HS"]]) %>%

slice(-8, -10) %>%

rbind(HS\_results[[9]][["imp"]][["HS"]]) %>%

slice(-9, -11) %>%

rbind(HS\_results[[10]][["imp"]][["HS"]]) %>%

slice(-10, -12) %>%

rbind(HS\_results[[11]][["imp"]][["HS"]]) %>%

slice(-11, -13) %>%

rbind(HS\_results[[12]][["imp"]][["HS"]]) %>%

slice(-12, -14) %>%

rbind(HS\_results[[13]][["imp"]][["HS"]]) %>%

slice(-13, -15) %>%

rbind(HS\_results[[14]][["imp"]][["HS"]]) %>%

slice(-14, -16) %>%

rbind(HS\_results[[15]][["imp"]][["HS"]]) %>%

slice(-15, -17) %>%

rbind(HS\_results[[16]][["imp"]][["HS"]]) %>%

slice(-16, -18) %>%

rbind(HS\_results[[17]][["imp"]][["HS"]]) %>%

slice(-17, -19) %>%

rbind(HS\_results[[18]][["imp"]][["HS"]]) %>%

slice(-18, -20) %>%

rbind(HS\_results[[19]][["imp"]][["HS"]]) %>%

slice(-19, -21) %>%

rbind(HS\_results[[20]][["imp"]][["HS"]]) %>%

slice(-20, -22) %>%

rbind(HS\_results[[21]][["imp"]][["HS"]]) %>%

slice(-21, -23) %>%

rbind(HS\_results[[22]][["imp"]][["HS"]]) %>%

slice(-22, -24) %>%

rbind(HS\_results[[23]][["imp"]][["HS"]]) %>%

slice(-23, -25) %>%

rbind(HS\_results[[24]][["imp"]][["HS"]]) %>%

slice(-24, -26) %>%

rbind(HS\_results[[25]][["imp"]][["HS"]]) %>%

slice(-25, -27) %>%

rbind(HS\_results[[26]][["imp"]][["HS"]]) %>%

slice(-26, -28) %>%

rbind(HS\_results[[27]][["imp"]][["HS"]]) %>%

slice(-27, -29) %>%

rbind(HS\_results[[28]][["imp"]][["HS"]]) %>%

slice(-28, -30) %>%

rbind(HS\_results[[29]][["imp"]][["HS"]]) %>%

slice(-29, -31) %>%

rbind(HS\_results[[30]][["imp"]][["HS"]]) %>%

slice(-30, -32) %>%

rbind(HS\_results[[31]][["imp"]][["HS"]]) %>%

slice(-31, -33) %>%

rbind(HS\_results[[32]][["imp"]][["HS"]]) %>%

slice(-32, -34) %>%

rbind(HS\_results[[33]][["imp"]][["HS"]]) %>%

slice(-33, -35) %>%

rbind(HS\_results[[34]][["imp"]][["HS"]]) %>%

slice(-34, -36) %>%

rbind(HS\_results[[35]][["imp"]][["HS"]]) %>%

slice(-35, -37) %>%

rbind(HS\_results[[36]][["imp"]][["HS"]]) %>%

slice(-36, -38) %>%

rbind(HS\_results[[37]][["imp"]][["HS"]]) %>%

slice(-37, -39) %>%

rbind(HS\_results[[38]][["imp"]][["HS"]]) %>%

slice(-38, -40) %>%

rbind(HS\_results[[39]][["imp"]][["HS"]]) %>%

slice(-39, -41) %>%

rbind(HS\_results[[40]][["imp"]][["HS"]]) %>%

slice(-40, -42) %>%

rbind(HS\_results[[41]][["imp"]][["HS"]]) %>%

slice(-41, -43) %>%

rbind(HS\_results[[42]][["imp"]][["HS"]]) %>%

slice(-42, -44) %>%

rbind(HS\_results[[43]][["imp"]][["HS"]]) %>%

slice(-43, -45) %>%

rbind(HS\_results[[45]][["imp"]][["HS"]]) %>%

slice(-44) %>%

rbind(HS\_results[[46]][["imp"]][["HS"]]) %>%

slice(-46, -47) %>%

rbind(HS\_results[[47]][["imp"]][["HS"]]) %>%

slice(-47, -48) %>%

rbind(HS\_results[[48]][["imp"]][["HS"]]) %>%

slice(-48, -49) %>%

rbind(HS\_results[[49]][["imp"]][["HS"]]) %>%

slice(-49, -50) %>%

rbind(HS\_results[[50]][["imp"]][["HS"]]) %>%

slice(-50, -51)

HS\_results\_df <- mutate(HS\_results\_df, mean = rowMeans(HS\_results\_df),

se = rowSds(as.matrix(HS\_results\_df)))

#Save results to computer a .csv

write.csv(HS\_results\_df, "[filepath]/HS-MICE-Results.csv")

```

#\*\*SBA\*\*

```{r, message=FALSE, warning=FALSE}

#Now for the SBA variable:

mice\_procedure <- function(x){

df\_mice1 <- df\_mice

df\_mice1[x, 5] <- NA

mice(df\_mice1, m = 100, maxit = 15, visitSequence = vis, method = "norm", pred = pred\_matrix, seed = 2019)

}

SBA\_results <- lapply(1:length(df\_mice$SBA), mice\_procedure)

```

#\*\*Organize results for SBA and compute mean and variance\*\*

```{r, message=FALSE, warning=FALSE}

SBA\_results\_df <- SBA\_results[[1]][["imp"]][["SBA"]] %>%

slice(1:2) %>%

rbind(SBA\_results[[3]][["imp"]][["SBA"]]) %>%

slice(-3, -5) %>%

rbind(SBA\_results[[4]][["imp"]][["SBA"]]) %>%

slice(-4, -6) %>%

rbind(SBA\_results[[5]][["imp"]][["SBA"]]) %>%

slice(-5, -7) %>%

rbind(SBA\_results[[6]][["imp"]][["SBA"]]) %>%

slice(-6, -8) %>%

rbind(SBA\_results[[7]][["imp"]][["SBA"]]) %>%

slice(-7, -9) %>%

rbind(SBA\_results[[8]][["imp"]][["SBA"]]) %>%

slice(-8, -10) %>%

rbind(SBA\_results[[9]][["imp"]][["SBA"]]) %>%

slice(-9, -11) %>%

rbind(SBA\_results[[10]][["imp"]][["SBA"]]) %>%

slice(-10, -12) %>%

rbind(SBA\_results[[11]][["imp"]][["SBA"]]) %>%

slice(-11, -13) %>%

rbind(SBA\_results[[12]][["imp"]][["SBA"]]) %>%

slice(-12, -14) %>%

rbind(SBA\_results[[13]][["imp"]][["SBA"]]) %>%

slice(-13, -15) %>%

rbind(SBA\_results[[14]][["imp"]][["SBA"]]) %>%

slice(-14, -16) %>%

rbind(SBA\_results[[15]][["imp"]][["SBA"]]) %>%

slice(-15, -17) %>%

rbind(SBA\_results[[16]][["imp"]][["SBA"]]) %>%

slice(-16, -18) %>%

rbind(SBA\_results[[17]][["imp"]][["SBA"]]) %>%

slice(-17, -19) %>%

rbind(SBA\_results[[18]][["imp"]][["SBA"]]) %>%

slice(-18, -20) %>%

rbind(SBA\_results[[19]][["imp"]][["SBA"]]) %>%

slice(-19, -21) %>%

rbind(SBA\_results[[20]][["imp"]][["SBA"]]) %>%

slice(-20, -22) %>%

rbind(SBA\_results[[21]][["imp"]][["SBA"]]) %>%

slice(-21, -23) %>%

rbind(SBA\_results[[22]][["imp"]][["SBA"]]) %>%

slice(-22, -24) %>%

rbind(SBA\_results[[23]][["imp"]][["SBA"]]) %>%

slice(-23, -25) %>%

rbind(SBA\_results[[24]][["imp"]][["SBA"]]) %>%

slice(-24, -26) %>%

rbind(SBA\_results[[25]][["imp"]][["SBA"]]) %>%

slice(-25, -27) %>%

rbind(SBA\_results[[26]][["imp"]][["SBA"]]) %>%

slice(-26, -28) %>%

rbind(SBA\_results[[27]][["imp"]][["SBA"]]) %>%

slice(-27, -29) %>%

rbind(SBA\_results[[28]][["imp"]][["SBA"]]) %>%

slice(-28, -30) %>%

rbind(SBA\_results[[29]][["imp"]][["SBA"]]) %>%

slice(-29, -31) %>%

rbind(SBA\_results[[30]][["imp"]][["SBA"]]) %>%

slice(-30, -32) %>%

rbind(SBA\_results[[31]][["imp"]][["SBA"]]) %>%

slice(-31, -33) %>%

rbind(SBA\_results[[32]][["imp"]][["SBA"]]) %>%

slice(-32, -34) %>%

rbind(SBA\_results[[33]][["imp"]][["SBA"]]) %>%

slice(-33, -35) %>%

rbind(SBA\_results[[34]][["imp"]][["SBA"]]) %>%

slice(-34, -36) %>%

rbind(SBA\_results[[35]][["imp"]][["SBA"]]) %>%

slice(-35, -37) %>%

rbind(SBA\_results[[36]][["imp"]][["SBA"]]) %>%

slice(-36, -38) %>%

rbind(SBA\_results[[37]][["imp"]][["SBA"]]) %>%

slice(-37, -39) %>%

rbind(SBA\_results[[38]][["imp"]][["SBA"]]) %>%

slice(-38, -40) %>%

rbind(SBA\_results[[39]][["imp"]][["SBA"]]) %>%

slice(-39, -41) %>%

rbind(SBA\_results[[40]][["imp"]][["SBA"]]) %>%

slice(-40, -42) %>%

rbind(SBA\_results[[41]][["imp"]][["SBA"]]) %>%

slice(-41, -43) %>%

rbind(SBA\_results[[42]][["imp"]][["SBA"]]) %>%

slice(-42, -44) %>%

rbind(SBA\_results[[43]][["imp"]][["SBA"]]) %>%

slice(-43, -45) %>%

rbind(SBA\_results[[45]][["imp"]][["SBA"]]) %>%

slice(-44) %>%

rbind(SBA\_results[[46]][["imp"]][["SBA"]]) %>%

slice(-46, -47) %>%

rbind(SBA\_results[[47]][["imp"]][["SBA"]]) %>%

slice(-47, -48) %>%

rbind(SBA\_results[[48]][["imp"]][["SBA"]]) %>%

slice(-48, -49) %>%

rbind(SBA\_results[[49]][["imp"]][["SBA"]]) %>%

slice(-49, -50) %>%

rbind(SBA\_results[[50]][["imp"]][["SBA"]]) %>%

slice(-50, -51)

SBA\_results\_df <- mutate(SBA\_results\_df, mean = rowMeans(SBA\_results\_df),

se = rowSds(as.matrix(SBA\_results\_df)))

#Save results to computer a .csv

write.csv(SBA\_results\_df, "[filepath]/SBA-MICE-Results.csv")

```

#\*\*BA\*\*

```{r, message=FALSE, warning=FALSE}

#Now for the BA variable:

mice\_procedure <- function(x){

df\_mice1 <- df\_mice

df\_mice1[x, 6] <- NA

mice(df\_mice1, m = 100, maxit = 15, visitSequence = vis, method = "norm", pred = pred\_matrix, seed = 2019)

}

BA\_results <- lapply(1:length(df\_mice$BA), mice\_procedure)

```

#\*\*Organize results for BA and compute mean and variance\*\*

```{r}

BA\_results\_df <- BA\_results[[1]][["imp"]][["BA"]] %>%

slice(1:2) %>%

rbind(BA\_results[[3]][["imp"]][["BA"]]) %>%

slice(-3, -5) %>%

rbind(BA\_results[[4]][["imp"]][["BA"]]) %>%

slice(-4, -6) %>%

rbind(BA\_results[[5]][["imp"]][["BA"]]) %>%

slice(-5, -7) %>%

rbind(BA\_results[[6]][["imp"]][["BA"]]) %>%

slice(-6, -8) %>%

rbind(BA\_results[[7]][["imp"]][["BA"]]) %>%

slice(-7, -9) %>%

rbind(BA\_results[[8]][["imp"]][["BA"]]) %>%

slice(-8, -10) %>%

rbind(BA\_results[[9]][["imp"]][["BA"]]) %>%

slice(-9, -11) %>%

rbind(BA\_results[[10]][["imp"]][["BA"]]) %>%

slice(-10, -12) %>%

rbind(BA\_results[[11]][["imp"]][["BA"]]) %>%

slice(-11, -13) %>%

rbind(BA\_results[[12]][["imp"]][["BA"]]) %>%

slice(-12, -14) %>%

rbind(BA\_results[[13]][["imp"]][["BA"]]) %>%

slice(-13, -15) %>%

rbind(BA\_results[[14]][["imp"]][["BA"]]) %>%

slice(-14, -16) %>%

rbind(BA\_results[[15]][["imp"]][["BA"]]) %>%

slice(-15, -17) %>%

rbind(BA\_results[[16]][["imp"]][["BA"]]) %>%

slice(-16, -18) %>%

rbind(BA\_results[[17]][["imp"]][["BA"]]) %>%

slice(-17, -19) %>%

rbind(BA\_results[[18]][["imp"]][["BA"]]) %>%

slice(-18, -20) %>%

rbind(BA\_results[[19]][["imp"]][["BA"]]) %>%

slice(-19, -21) %>%

rbind(BA\_results[[20]][["imp"]][["BA"]]) %>%

slice(-20, -22) %>%

rbind(BA\_results[[21]][["imp"]][["BA"]]) %>%

slice(-21, -23) %>%

rbind(BA\_results[[22]][["imp"]][["BA"]]) %>%

slice(-22, -24) %>%

rbind(BA\_results[[23]][["imp"]][["BA"]]) %>%

slice(-23, -25) %>%

rbind(BA\_results[[24]][["imp"]][["BA"]]) %>%

slice(-24, -26) %>%

rbind(BA\_results[[25]][["imp"]][["BA"]]) %>%

slice(-25, -27) %>%

rbind(BA\_results[[26]][["imp"]][["BA"]]) %>%

slice(-26, -28) %>%

rbind(BA\_results[[27]][["imp"]][["BA"]]) %>%

slice(-27, -29) %>%

rbind(BA\_results[[28]][["imp"]][["BA"]]) %>%

slice(-28, -30) %>%

rbind(BA\_results[[29]][["imp"]][["BA"]]) %>%

slice(-29, -31) %>%

rbind(BA\_results[[30]][["imp"]][["BA"]]) %>%

slice(-30, -32) %>%

rbind(BA\_results[[31]][["imp"]][["BA"]]) %>%

slice(-31, -33) %>%

rbind(BA\_results[[32]][["imp"]][["BA"]]) %>%

slice(-32, -34) %>%

rbind(BA\_results[[33]][["imp"]][["BA"]]) %>%

slice(-33, -35) %>%

rbind(BA\_results[[34]][["imp"]][["BA"]]) %>%

slice(-34, -36) %>%

rbind(BA\_results[[35]][["imp"]][["BA"]]) %>%

slice(-35, -37) %>%

rbind(BA\_results[[36]][["imp"]][["BA"]]) %>%

slice(-36, -38) %>%

rbind(BA\_results[[37]][["imp"]][["BA"]]) %>%

slice(-37, -39) %>%

rbind(BA\_results[[38]][["imp"]][["BA"]]) %>%

slice(-38, -40) %>%

rbind(BA\_results[[39]][["imp"]][["BA"]]) %>%

slice(-39, -41) %>%

rbind(BA\_results[[40]][["imp"]][["BA"]]) %>%

slice(-40, -42) %>%

rbind(BA\_results[[41]][["imp"]][["BA"]]) %>%

slice(-41, -43) %>%

rbind(BA\_results[[42]][["imp"]][["BA"]]) %>%

slice(-42, -44) %>%

rbind(BA\_results[[43]][["imp"]][["BA"]]) %>%

slice(-43, -45) %>%

rbind(BA\_results[[45]][["imp"]][["BA"]]) %>%

slice(-44) %>%

rbind(BA\_results[[46]][["imp"]][["BA"]]) %>%

slice(-46, -47) %>%

rbind(BA\_results[[47]][["imp"]][["BA"]]) %>%

slice(-47, -48) %>%

rbind(BA\_results[[48]][["imp"]][["BA"]]) %>%

slice(-48, -49) %>%

rbind(BA\_results[[49]][["imp"]][["BA"]]) %>%

slice(-49, -50) %>%

rbind(BA\_results[[50]][["imp"]][["BA"]]) %>%

slice(-50, -51)

BA\_results\_df <- mutate(BA\_results\_df, mean = rowMeans(BA\_results\_df),

se = rowSds(as.matrix(BA\_results\_df)))

#Save results to computer as a .csv

write.csv(BA\_results\_df, "[filepath]/BA-MICE-Results.csv")

```

#\*\*B\*\*

```{r, message=FALSE, warning=FALSE}

#Now for the B variable:

mice\_procedure <- function(x){

df\_mice1 <- df\_mice

df\_mice1[x, 8] <- NA

mice(df\_mice1, m = 100, maxit = 15, visitSequence = vis, method = "norm", pred = pred\_matrix, seed = 2019)

}

B\_results <- lapply(1:length(df\_mice$B), mice\_procedure)

```

#\*\*Organize results for B and compute mean and variance\*\*

```{r, message=FALSE, warning=FALSE}

B\_results\_df <- B\_results[[1]][["imp"]][["B"]] %>%

slice(1) %>%

rbind(B\_results[[2]][["imp"]][["B"]]) %>%

slice(1:2) %>%

rbind(B\_results[[3]][["imp"]][["B"]]) %>%

slice(1:3) %>%

rbind(B\_results[[4]][["imp"]][["B"]]) %>%

slice(1:4) %>%

rbind(B\_results[[5]][["imp"]][["B"]]) %>%

slice(1:5) %>%

rbind(B\_results[[6]][["imp"]][["B"]]) %>%

slice(1:6) %>%

rbind(B\_results[[7]][["imp"]][["B"]]) %>%

slice(1:7) %>%

rbind(B\_results[[8]][["imp"]][["B"]]) %>%

slice(1:8) %>%

rbind(B\_results[[9]][["imp"]][["B"]]) %>%

slice(1:9) %>%

rbind(B\_results[[10]][["imp"]][["B"]]) %>%

slice(1:10) %>%

rbind(B\_results[[13]][["imp"]][["B"]]) %>%

slice(1:13) %>%

rbind(B\_results[[14]][["imp"]][["B"]]) %>%

slice(1:13, 16) %>%

rbind(B\_results[[15]][["imp"]][["B"]]) %>%

slice(1:14, 17) %>%

rbind(B\_results[[16]][["imp"]][["B"]]) %>%

slice(1:15, 18) %>%

rbind(B\_results[[17]][["imp"]][["B"]]) %>%

slice(1:16, 19) %>%

rbind(B\_results[[18]][["imp"]][["B"]]) %>%

slice(1:17, 20) %>%

rbind(B\_results[[20]][["imp"]][["B"]]) %>%

slice(1:18, 20:21) %>%

rbind(B\_results[[21]][["imp"]][["B"]]) %>%

slice(1:20, 24) %>%

rbind(B\_results[[22]][["imp"]][["B"]]) %>%

slice(1:21, 25) %>%

rbind(B\_results[[23]][["imp"]][["B"]]) %>%

slice(1:22, 26) %>%

rbind(B\_results[[24]][["imp"]][["B"]]) %>%

slice(1:23, 27) %>%

rbind(B\_results[[25]][["imp"]][["B"]]) %>%

slice(1:24, 28) %>%

rbind(B\_results[[27]][["imp"]][["B"]]) %>%

slice(1:25, 29:30) %>%

rbind(B\_results[[28]][["imp"]][["B"]]) %>%

slice(1:27, 32) %>%

rbind(B\_results[[30]][["imp"]][["B"]]) %>%

slice(1:28, 33:34) %>%

rbind(B\_results[[32]][["imp"]][["B"]]) %>%

slice(1:30, 36:37) %>%

rbind(B\_results[[33]][["imp"]][["B"]]) %>%

slice(1:32, 39) %>%

rbind(B\_results[[34]][["imp"]][["B"]]) %>%

slice(1:33, 40) %>%

rbind(B\_results[[35]][["imp"]][["B"]]) %>%

slice(1:34, 41) %>%

rbind(B\_results[[36]][["imp"]][["B"]]) %>%

slice(1:35, 42) %>%

rbind(B\_results[[38]][["imp"]][["B"]]) %>%

slice(1:36, 43:44) %>%

rbind(B\_results[[39]][["imp"]][["B"]]) %>%

slice(1:38, 46) %>%

rbind(B\_results[[40]][["imp"]][["B"]]) %>%

slice(1:39, 47) %>%

rbind(B\_results[[42]][["imp"]][["B"]]) %>%

slice(1:40, 48:49) %>%

rbind(B\_results[[43]][["imp"]][["B"]]) %>%

slice(1:42, 51) %>%

rbind(B\_results[[46]][["imp"]][["B"]]) %>%

slice(1:43, 52:54) %>%

rbind(B\_results[[47]][["imp"]][["B"]]) %>%

slice(1:46, 57) %>%

rbind(B\_results[[48]][["imp"]][["B"]]) %>%

slice(1:47, 58) %>%

rbind(B\_results[[49]][["imp"]][["B"]]) %>%

slice(1:48, 59:60)

B\_results\_df <- mutate(B\_results\_df, mean = rowMeans(B\_results\_df),

se = rowSds(as.matrix(B\_results\_df)))

#Save results to computer as a .csv

write.csv(B\_results\_df, "[filepath]/B-MICE-Results.csv")

```

#\*\*H\*\*

```{r, message=FALSE, warning=FALSE}

#Now for the H variable:

mice\_procedure <- function(x){

df\_mice1 <- df\_mice

df\_mice1[x, 9] <- NA

mice(df\_mice1, m = 100, maxit = 15, visitSequence = vis, method = "norm", pred = pred\_matrix, seed = 2019)

}

H\_results <- lapply(1:length(df\_mice$H), mice\_procedure)

```

#\*\*Organize results for H and compute mean and variance\*\*

```{r, message=FALSE, warning=FALSE}

H\_results\_df <- H\_results[[1]][["imp"]][["H"]] %>%

slice(1) %>%

rbind(H\_results[[2]][["imp"]][["H"]]) %>%

slice(1:2) %>%

rbind(H\_results[[3]][["imp"]][["H"]]) %>%

slice(1:3) %>%

rbind(H\_results[[4]][["imp"]][["H"]]) %>%

slice(1:4) %>%

rbind(H\_results[[5]][["imp"]][["H"]]) %>%

slice(1:5) %>%

rbind(H\_results[[6]][["imp"]][["H"]]) %>%

slice(1:6) %>%

rbind(H\_results[[7]][["imp"]][["H"]]) %>%

slice(1:7) %>%

rbind(H\_results[[8]][["imp"]][["H"]]) %>%

slice(1:8) %>%

rbind(H\_results[[9]][["imp"]][["H"]]) %>%

slice(1:9) %>%

rbind(H\_results[[10]][["imp"]][["H"]]) %>%

slice(1:10) %>%

rbind(H\_results[[11]][["imp"]][["H"]]) %>%

slice(1:11) %>%

rbind(H\_results[[12]][["imp"]][["H"]]) %>%

slice(1:12) %>%

rbind(H\_results[[13]][["imp"]][["H"]]) %>%

slice(1:13) %>%

rbind(H\_results[[14]][["imp"]][["H"]]) %>%

slice(1:14) %>%

rbind(H\_results[[15]][["imp"]][["H"]]) %>%

slice(1:15) %>%

rbind(H\_results[[16]][["imp"]][["H"]]) %>%

slice(1:16) %>%

rbind(H\_results[[17]][["imp"]][["H"]]) %>%

slice(1:17) %>%

rbind(H\_results[[18]][["imp"]][["H"]]) %>%

slice(1:18) %>%

rbind(H\_results[[20]][["imp"]][["H"]]) %>%

slice(1:20) %>%

rbind(H\_results[[21]][["imp"]][["H"]]) %>%

slice(1:20, 22) %>%

rbind(H\_results[[22]][["imp"]][["H"]]) %>%

slice(1:21, 23) %>%

rbind(H\_results[[23]][["imp"]][["H"]]) %>%

slice(1:22, 24) %>%

rbind(H\_results[[24]][["imp"]][["H"]]) %>%

slice(1:23, 25) %>%

rbind(H\_results[[25]][["imp"]][["H"]]) %>%

slice(1:24, 26) %>%

rbind(H\_results[[26]][["imp"]][["H"]]) %>%

slice(1:25, 27) %>%

rbind(H\_results[[27]][["imp"]][["H"]]) %>%

slice(1:26, 28) %>%

rbind(H\_results[[28]][["imp"]][["H"]]) %>%

slice(1:27, 29) %>%

rbind(H\_results[[29]][["imp"]][["H"]]) %>%

slice(1:28, 30) %>%

rbind(H\_results[[30]][["imp"]][["H"]]) %>%

slice(1:29, 31) %>%

rbind(H\_results[[31]][["imp"]][["H"]]) %>%

slice(1:30, 32) %>%

rbind(H\_results[[32]][["imp"]][["H"]]) %>%

slice(1:31, 33) %>%

rbind(H\_results[[33]][["imp"]][["H"]]) %>%

slice(1:32, 34) %>%

rbind(H\_results[[34]][["imp"]][["H"]]) %>%

slice(1:33, 35) %>%

rbind(H\_results[[35]][["imp"]][["H"]]) %>%

slice(1:34, 36) %>%

rbind(H\_results[[36]][["imp"]][["H"]]) %>%

slice(1:35, 37) %>%

rbind(H\_results[[37]][["imp"]][["H"]]) %>%

slice(1:36, 38) %>%

rbind(H\_results[[38]][["imp"]][["H"]]) %>%

slice(1:37, 39) %>%

rbind(H\_results[[39]][["imp"]][["H"]]) %>%

slice(1:38, 40) %>%

rbind(H\_results[[40]][["imp"]][["H"]]) %>%

slice(1:39, 41) %>%

rbind(H\_results[[41]][["imp"]][["H"]]) %>%

slice(1:40, 42) %>%

rbind(H\_results[[42]][["imp"]][["H"]]) %>%

slice(1:41, 43) %>%

rbind(H\_results[[43]][["imp"]][["H"]]) %>%

slice(1:42, 44) %>%

rbind(H\_results[[44]][["imp"]][["H"]]) %>%

slice(1:43, 45) %>%

rbind(H\_results[[46]][["imp"]][["H"]]) %>%

slice(1:44, 46:47) %>%

rbind(H\_results[[47]][["imp"]][["H"]]) %>%

slice(1:46, 49) %>%

rbind(H\_results[[49]][["imp"]][["H"]]) %>%

slice(1:47, 50:51) %>%

rbind(H\_results[[50]][["imp"]][["H"]]) %>%

slice(1:49, 53)

H\_results\_df <- mutate(H\_results\_df, mean = rowMeans(H\_results\_df),

se = rowSds(as.matrix(H\_results\_df)))

#Save results to computer as a .csv

write.csv(H\_results\_df, "[filepath]/H-MICE-Results.csv")

```

#\*\*API\*\*

```{r, message=FALSE, warning=FALSE}

#Now for the API variable:

mice\_procedure <- function(x){

df\_mice1 <- df\_mice

df\_mice1[x, 10] <- NA

mice(df\_mice1, m = 100, maxit = 15, visitSequence = vis, method = "norm", pred = pred\_matrix, seed = 2019)

}

API\_results <- lapply(1:length(df\_mice$API), mice\_procedure)

```

#\*\*Organize results for API and compute mean and variance\*\*

```{r, message=FALSE, warning=FALSE}

API\_results\_df <- API\_results[[2]][["imp"]][["API"]] %>%

slice(1:2) %>%

rbind(API\_results[[3]][["imp"]][["API"]]) %>%

slice(1:2, 4) %>%

rbind(API\_results[[5]][["imp"]][["API"]]) %>%

slice(1:3, 5:6) %>%

rbind(API\_results[[6]][["imp"]][["API"]]) %>%

slice(1:5, 8) %>%

rbind(API\_results[[7]][["imp"]][["API"]]) %>%

slice(1:6, 9) %>%

rbind(API\_results[[8]][["imp"]][["API"]]) %>%

slice(1:7, 10) %>%

rbind(API\_results[[9]][["imp"]][["API"]]) %>%

slice(1:8, 11) %>%

rbind(API\_results[[10]][["imp"]][["API"]]) %>%

slice(1:9, 12) %>%

rbind(API\_results[[11]][["imp"]][["API"]]) %>%

slice(1:10, 13) %>%

rbind(API\_results[[13]][["imp"]][["API"]]) %>%

slice(1:11, 14:15) %>%

rbind(API\_results[[15]][["imp"]][["API"]]) %>%

slice(1:13, 17:18) %>%

rbind(API\_results[[16]][["imp"]][["API"]]) %>%

slice(1:15, 20) %>%

rbind(API\_results[[17]][["imp"]][["API"]]) %>%

slice(1:16, 21) %>%

rbind(API\_results[[20]][["imp"]][["API"]]) %>%

slice(1:17, 22:24) %>%

rbind(API\_results[[21]][["imp"]][["API"]]) %>%

slice(1:20, 27) %>%

rbind(API\_results[[22]][["imp"]][["API"]]) %>%

slice(1:21, 28) %>%

rbind(API\_results[[23]][["imp"]][["API"]]) %>%

slice(1:22, 29) %>%

rbind(API\_results[[28]][["imp"]][["API"]]) %>%

slice(1:23, 30:34) %>%

rbind(API\_results[[29]][["imp"]][["API"]]) %>%

slice(1:28, 39) %>%

rbind(API\_results[[30]][["imp"]][["API"]]) %>%

slice(1:29, 40) %>%

rbind(API\_results[[32]][["imp"]][["API"]]) %>%

slice(1:30, 41:42) %>%

rbind(API\_results[[33]][["imp"]][["API"]]) %>%

slice(1:32, 44) %>%

rbind(API\_results[[35]][["imp"]][["API"]]) %>%

slice(1:33, 45:46) %>%

rbind(API\_results[[37]][["imp"]][["API"]]) %>%

slice(1:35, 48:49) %>%

rbind(API\_results[[38]][["imp"]][["API"]]) %>%

slice(1:37, 51) %>%

rbind(API\_results[[39]][["imp"]][["API"]]) %>%

slice(1:38, 52) %>%

rbind(API\_results[[43]][["imp"]][["API"]]) %>%

slice(1:39, 53:56) %>%

rbind(API\_results[[46]][["imp"]][["API"]]) %>%

slice(1:43, 60:62) %>%

rbind(API\_results[[47]][["imp"]][["API"]]) %>%

slice(1:46, 65) %>%

rbind(API\_results[[49]][["imp"]][["API"]]) %>%

slice(1:47, 66:68)

API\_results\_df <- mutate(API\_results\_df, mean = rowMeans(API\_results\_df),

se = rowSds(as.matrix(API\_results\_df)))

#Save results to computer as a .csv

write.csv(API\_results\_df, "[filepath]/API-MICE-Results.csv")

```

#\*\*AINA\*\*

```{r, message=FALSE, warning=FALSE}

#Now for the AINA variable:

mice\_procedure <- function(x){

df\_mice1 <- df\_mice

df\_mice1[x, 11] <- NA

mice(df\_mice1, m = 100, maxit = 15, visitSequence = vis, method = "norm", pred = pred\_matrix, seed = 2019)

}

AINA\_results <- lapply(1:length(df\_mice$AINA), mice\_procedure)

```

#\*\*Organize results for API and compute mean and variance\*\*

```{r, message=FALSE, warning=FALSE}

AINA\_results\_df <- AINA\_results[[2]][["imp"]][["AINA"]] %>%

slice(1:2) %>%

rbind(AINA\_results[[3]][["imp"]][["AINA"]]) %>%

slice(1:2, 4) %>%

rbind(AINA\_results[[23]][["imp"]][["AINA"]]) %>%

slice(1:3, 5:24) %>%

rbind(AINA\_results[[26]][["imp"]][["AINA"]]) %>%

slice(1:23, 44:46) %>%

rbind(AINA\_results[[31]][["imp"]][["AINA"]]) %>%

slice(1:26, 49:53) %>%

rbind(AINA\_results[[33]][["imp"]][["AINA"]]) %>%

slice(1:31, 58:59) %>%

rbind(AINA\_results[[34]][["imp"]][["AINA"]]) %>%

slice(1:33, 61) %>%

rbind(AINA\_results[[36]][["imp"]][["AINA"]]) %>%

slice(1:34, 62:63) %>%

rbind(AINA\_results[[41]][["imp"]][["AINA"]]) %>%

slice(1:36, 65:69) %>%

rbind(AINA\_results[[44]][["imp"]][["AINA"]]) %>%

slice(1:41, 74:76) %>%

rbind(AINA\_results[[47]][["imp"]][["AINA"]]) %>%

slice(1:44, 79:81) %>%

rbind(AINA\_results[[49]][["imp"]][["AINA"]]) %>%

slice(1:47, 84:85) %>%

rbind(AINA\_results[[50]][["imp"]][["AINA"]]) %>%

slice(1:49, 87)

AINA\_results\_df <- mutate(AINA\_results\_df, mean = rowMeans(AINA\_results\_df),

se = rowSds(as.matrix(AINA\_results\_df)))

#Save results to computer as a .csv

write.csv(AINA\_results\_df, "[filepath]/AINA-MICE-Results.csv")

```

#\*\*TP\*\*

```{r, message=FALSE, warning=FALSE}

#Now for the TP variable:

mice\_procedure <- function(x){

df\_mice1 <- df\_mice

df\_mice1[x, 12] <- NA

mice(df\_mice1, m = 100, maxit = 15, visitSequence = vis, method = "norm", pred = pred\_matrix, seed = 2019)

}

TP\_results <- lapply(1:length(df\_mice$TP), mice\_procedure)

```

#\*\*Organize results for TP and compute mean and variance\*\*

```{r, message=FALSE, warning=FALSE}

TP\_results\_df <- TP\_results[[2]][["imp"]][["TP"]] %>%

slice(1:2) %>%

rbind(TP\_results[[5]][["imp"]][["TP"]]) %>%

slice(1:2, 4:6) %>%

rbind(TP\_results[[6]][["imp"]][["TP"]]) %>%

slice(1:5, 9) %>%

rbind(TP\_results[[9]][["imp"]][["TP"]]) %>%

slice(1:6, 10:12) %>%

rbind(TP\_results[[10]][["imp"]][["TP"]]) %>%

slice(1:9, 15) %>%

rbind(TP\_results[[11]][["imp"]][["TP"]]) %>%

slice(1:10, 16) %>%

rbind(TP\_results[[14]][["imp"]][["TP"]]) %>%

slice(1:11, 17:19) %>%

rbind(TP\_results[[15]][["imp"]][["TP"]]) %>%

slice(1:14, 22) %>%

rbind(TP\_results[[16]][["imp"]][["TP"]]) %>%

slice(1:15, 23) %>%

rbind(TP\_results[[17]][["imp"]][["TP"]]) %>%

slice(1:16, 24) %>%

rbind(TP\_results[[20]][["imp"]][["TP"]]) %>%

slice(1:17, 25:27) %>%

rbind(TP\_results[[23]][["imp"]][["TP"]]) %>%

slice(1:20, 30:32) %>%

rbind(TP\_results[[26]][["imp"]][["TP"]]) %>%

slice(1:23, 35:37) %>%

rbind(TP\_results[[27]][["imp"]][["TP"]]) %>%

slice(1:26, 40) %>%

rbind(TP\_results[[28]][["imp"]][["TP"]]) %>%

slice(1:27, 41) %>%

rbind(TP\_results[[33]][["imp"]][["TP"]]) %>%

slice(1:28, 42:46) %>%

rbind(TP\_results[[35]][["imp"]][["TP"]]) %>%

slice(1:33, 51:52) %>%

rbind(TP\_results[[36]][["imp"]][["TP"]]) %>%

slice(1:35, 54) %>%

rbind(TP\_results[[37]][["imp"]][["TP"]]) %>%

slice(1:36, 55) %>%

rbind(TP\_results[[38]][["imp"]][["TP"]]) %>%

slice(1:37, 56) %>%

rbind(TP\_results[[39]][["imp"]][["TP"]]) %>%

slice(1:38, 57) %>%

rbind(TP\_results[[43]][["imp"]][["TP"]]) %>%

slice(1:39, 58:61) %>%

rbind(TP\_results[[46]][["imp"]][["TP"]]) %>%

slice(1:43, 65:67) %>%

rbind(TP\_results[[47]][["imp"]][["TP"]]) %>%

slice(1:46, 70:73)

TP\_results\_df <- mutate(TP\_results\_df, mean = rowMeans(TP\_results\_df),

se = rowSds(as.matrix(TP\_results\_df)))

#Save results to computer as a .csv

write.csv(TP\_results\_df, "[filepath]/TP-MICE-Results.csv")

```

#\*\*EL\*\*

```{r, message=FALSE, warning=FALSE}

#Now for the EL variable:

mice\_procedure <- function(x){

df\_mice1 <- df\_mice

df\_mice1[x, 13] <- NA

mice(df\_mice1, m = 100, maxit = 15, visitSequence = vis, method = "norm", pred = pred\_matrix, seed = 2019)

}

EL\_results <- lapply(1:length(df\_mice$EL), mice\_procedure)

```

#\*\*Organize results for EL and compute mean and variance\*\*

```{r, message=FALSE, warning=FALSE}

EL\_results\_df <- EL\_results[[2]][["imp"]][["EL"]] %>%

slice(1:2) %>%

rbind(EL\_results[[3]][["imp"]][["EL"]]) %>%

slice(1:2, 4) %>%

rbind(EL\_results[[4]][["imp"]][["EL"]]) %>%

slice(1:3, 5) %>%

rbind(EL\_results[[5]][["imp"]][["EL"]]) %>%

slice(1:4, 6) %>%

rbind(EL\_results[[6]][["imp"]][["EL"]]) %>%

slice(1:5, 7) %>%

rbind(EL\_results[[7]][["imp"]][["EL"]]) %>%

slice(1:6, 8) %>%

rbind(EL\_results[[9]][["imp"]][["EL"]]) %>%

slice(1:7, 9:10) %>%

rbind(EL\_results[[10]][["imp"]][["EL"]]) %>%

slice(1:9, 12) %>%

rbind(EL\_results[[11]][["imp"]][["EL"]]) %>%

slice(1:10, 13) %>%

rbind(EL\_results[[13]][["imp"]][["EL"]]) %>%

slice(1:11, 14:15) %>%

rbind(EL\_results[[14]][["imp"]][["EL"]]) %>%

slice(1:13, 17) %>%

rbind(EL\_results[[15]][["imp"]][["EL"]]) %>%

slice(1:14, 18) %>%

rbind(EL\_results[[16]][["imp"]][["EL"]]) %>%

slice(1:15, 19) %>%

rbind(EL\_results[[20]][["imp"]][["EL"]]) %>%

slice(1:16, 20:23) %>%

rbind(EL\_results[[21]][["imp"]][["EL"]]) %>%

slice(1:20, 27) %>%

rbind(EL\_results[[22]][["imp"]][["EL"]]) %>%

slice(1:21, 28) %>%

rbind(EL\_results[[23]][["imp"]][["EL"]]) %>%

slice(1:22, 29) %>%

rbind(EL\_results[[28]][["imp"]][["EL"]]) %>%

slice(1:23, 30:34) %>%

rbind(EL\_results[[31]][["imp"]][["EL"]]) %>%

slice(1:28, 39:41) %>%

rbind(EL\_results[[32]][["imp"]][["EL"]]) %>%

slice(1:31, 44) %>%

rbind(EL\_results[[33]][["imp"]][["EL"]]) %>%

slice(1:32, 45) %>%

rbind(EL\_results[[35]][["imp"]][["EL"]]) %>%

slice(1:33, 46:47) %>%

rbind(EL\_results[[36]][["imp"]][["EL"]]) %>%

slice(1:35, 49) %>%

rbind(EL\_results[[38]][["imp"]][["EL"]]) %>%

slice(1:36, 50:51) %>%

rbind(EL\_results[[39]][["imp"]][["EL"]]) %>%

slice(1:38, 53) %>%

rbind(EL\_results[[40]][["imp"]][["EL"]]) %>%

slice(1:39, 54) %>%

rbind(EL\_results[[43]][["imp"]][["EL"]]) %>%

slice(1:40, 55:57) %>%

rbind(EL\_results[[44]][["imp"]][["EL"]]) %>%

slice(1:43, 60) %>%

rbind(EL\_results[[46]][["imp"]][["EL"]]) %>%

slice(1:44, 61:62) %>%

rbind(EL\_results[[47]][["imp"]][["EL"]]) %>%

slice(1:46, 64) %>%

rbind(EL\_results[[49]][["imp"]][["EL"]]) %>%

slice(1:47, 65:67)

EL\_results\_df <- mutate(EL\_results\_df, mean = rowMeans(EL\_results\_df),

se = rowSds(as.matrix(EL\_results\_df)))

#Save results to computer as a .csv

write.csv(EL\_results\_df, "[filepath]/EL-MICE-Results.csv")

```

### Free up memory

```{r, message=FALSE, warning=FALSE}

#Clear objects from environment to preserve memory

rm(list = ls())

```

### Import results from MICE procedure

```{r, message=FALSE, warning=FALSE}

NHS\_results\_df <- read\_csv("[filepath]/NHS-MICE-Results.csv")

HS\_results\_df <- read\_csv("[filepath]/HS-MICE-Results.csv")

SBA\_results\_df <- read\_csv("[filepath]/SBA-MICE-Results.csv")

BA\_results\_df <- read\_csv("[filepath]/BA-MICE-Results.csv")

B\_results\_df <- read\_csv("[filepath]/B-MICE-Results.csv")

H\_results\_df <- read\_csv("[filepath]/H-MICE-Results.csv")

API\_results\_df <- read\_csv("[filepath]/API-MICE-Results.csv")

AINA\_results\_df <- read\_csv("[filepath]/AINA-MICE-Results.csv")

TP\_results\_df <- read\_csv("[filepath]/TP-MICE-Results.csv")

EL\_results\_df <- read\_csv("[filepath]/EL-MICE-Results.csv")

```

### Wrangle all predicted values

```{r, message=FALSE, warning=FALSE}

#\*Save state abbreviations to an object called "State".\*

State <- read\_csv("[filepath]/Subgroups of Interest/2+.csv")

State <- State[,1]

```

#\*Column bind "State" to each of the subgroup results data frames\*

```{r, message=FALSE, warning=FALSE}

NHS\_results\_df <- cbind(State, NHS\_results\_df)

HS\_results\_df <- cbind(State, HS\_results\_df)

SBA\_results\_df <- cbind(State, SBA\_results\_df)

BA\_results\_df <- cbind(State, BA\_results\_df)

B\_results\_df <- cbind(State, B\_results\_df)

H\_results\_df <- cbind(State, H\_results\_df)

API\_results\_df <- cbind(State, API\_results\_df)

AINA\_results\_df <- cbind(State, AINA\_results\_df)

TP\_results\_df <- cbind(State, TP\_results\_df)

EL\_results\_df <- cbind(State, EL\_results\_df)

```

#Keep only State, mean and se variables

```{r, message=FALSE, warning=FALSE}

NHS\_results\_df <- NHS\_results\_df %>%

select("State", "mean", "se")

HS\_results\_df <- HS\_results\_df %>%

select("State", "mean", "se")

SBA\_results\_df <- SBA\_results\_df %>%

select("State", "mean", "se")

BA\_results\_df <- BA\_results\_df %>%

select("State", "mean", "se")

B\_results\_df <- B\_results\_df %>%

select("State", "mean", "se")

H\_results\_df <- H\_results\_df %>%

select("State", "mean", "se")

API\_results\_df <- API\_results\_df %>%

select("State", "mean", "se")

AINA\_results\_df <- AINA\_results\_df %>%

select("State", "mean", "se")

TP\_results\_df <- TP\_results\_df %>%

select("State", "mean", "se")

EL\_results\_df <- EL\_results\_df %>%

select("State", "mean", "se")

```

#Keep only target values

```{r, message=FALSE, warning=FALSE}

#NHS

NHS\_results\_df[2,2:3] <- NA

NHS\_results\_df[44,2:3] <- NA

#HS

HS\_results\_df[2,2:3] <- NA

HS\_results\_df[44,2:3] <- NA

#SBA

SBA\_results\_df[2,2:3] <- NA

SBA\_results\_df[44,2:3] <- NA

#BA

BA\_results\_df[2,2:3] <- NA

BA\_results\_df[44,2:3] <- NA

#B

B\_results\_df[11:12,2:3] <- NA

B\_results\_df[19,2:3] <- NA

B\_results\_df[26,2:3] <- NA

B\_results\_df[29,2:3] <- NA

B\_results\_df[31,2:3] <- NA

B\_results\_df[37,2:3] <- NA

B\_results\_df[41,2:3] <- NA

B\_results\_df[44:45,2:3] <- NA

B\_results\_df[50,2:3] <- NA

#H

H\_results\_df[19,2:3] <- NA

H\_results\_df[45,2:3] <- NA

H\_results\_df[48,2:3] <- NA

#API

API\_results\_df[1,2:3] <- NA

API\_results\_df[4,2:3] <- NA

API\_results\_df[12,2:3] <- NA

API\_results\_df[14,2:3] <- NA

API\_results\_df[18:19,2:3] <- NA

API\_results\_df[24:27,2:3] <- NA

API\_results\_df[31,2:3] <- NA

API\_results\_df[34,2:3] <- NA

API\_results\_df[36,2:3] <- NA

API\_results\_df[40:42,2:3] <- NA

API\_results\_df[44:45,2:3] <- NA

API\_results\_df[48,2:3] <- NA

API\_results\_df[50,2:3] <- NA

#AINA

AINA\_results\_df[1,2:3] <- NA

AINA\_results\_df[4:22,2:3] <- NA

AINA\_results\_df[24:25,2:3] <- NA

AINA\_results\_df[27:30,2:3] <- NA

AINA\_results\_df[32,2:3] <- NA

AINA\_results\_df[35,2:3] <- NA

AINA\_results\_df[37:40,2:3] <- NA

AINA\_results\_df[42:43,2:3] <- NA

AINA\_results\_df[45:46,2:3] <- NA

AINA\_results\_df[48,2:3] <- NA

#TP

TP\_results\_df[1,2:3] <- NA

TP\_results\_df[3:4,2:3] <- NA

TP\_results\_df[7:8,2:3] <- NA

TP\_results\_df[12:13,2:3] <- NA

TP\_results\_df[18:19,2:3] <- NA

TP\_results\_df[21:22,2:3] <- NA

TP\_results\_df[24:25,2:3] <- NA

TP\_results\_df[29:32,2:3] <- NA

TP\_results\_df[34,2:3] <- NA

TP\_results\_df[40:42,2:3] <- NA

TP\_results\_df[44:45,2:3] <- NA

TP\_results\_df[48:50,2:3] <- NA

#EL

EL\_results\_df[1,2:3] <- NA

EL\_results\_df[8,2:3] <- NA

EL\_results\_df[12,2:3] <- NA

EL\_results\_df[17:19,2:3] <- NA

EL\_results\_df[24:27,2:3] <- NA

EL\_results\_df[29:30,2:3] <- NA

EL\_results\_df[34,2:3] <- NA

EL\_results\_df[37,2:3] <- NA

EL\_results\_df[41:42,2:3] <- NA

EL\_results\_df[45,2:3] <- NA

EL\_results\_df[48,2:3] <- NA

EL\_results\_df[50,2:3] <- NA

```

#\*\*SPOT CHECKED?\*\*

#YES

### Check plausibility of MICE-based estimates of mean subgroup achievement

#\*\*NHS\*\*

```{r, message=FALSE, warning=FALSE}

NHS\_results\_df %>%

ggplot(., aes(x=1, y=mean)) +

geom\_dotplot(binwidth = 1, binaxis = "y", stackdir = "center") +

theme(axis.title.x=element\_blank(),

axis.text.x=element\_blank(),

axis.ticks.x=element\_blank()) +

geom\_hline(yintercept = 277.58, color="red") +

geom\_hline(yintercept = 251.62, color="red") +

ylab("Score") +

theme(text=element\_text(size=14, family="serif"))

```

#All of the predicted values are in bound. Though one (AL) is very close to falling under the lower bound.

#\*\*HS\*\*

```{r, message=FALSE, warning=FALSE}

HS\_results\_df %>%

ggplot(., aes(x=1, y=mean)) +

geom\_dotplot(binwidth = 1, binaxis = "y", stackdir = "center") +

theme(axis.title.x=element\_blank(),

axis.text.x=element\_blank(),

axis.ticks.x=element\_blank()) +

geom\_hline(yintercept = 282.8088, color="red") +

geom\_hline(yintercept = 253.2988, color="red") +

ylab("Score") +

theme(text=element\_text(size=14, family="serif"))

```

#All of the predicted values are in bound.

#\*\*SBA\*\*

```{r, message=FALSE, warning=FALSE}

SBA\_results\_df %>%

ggplot(., aes(x=1, y=mean)) +

geom\_dotplot(binwidth = 1, binaxis = "y", stackdir = "center") +

theme(axis.title.x=element\_blank(),

axis.text.x=element\_blank(),

axis.ticks.x=element\_blank()) +

geom\_hline(yintercept = 294.2487, color="red") +

geom\_hline(yintercept = 272.0388, color="red") +

ylab("Score") +

theme(text=element\_text(size=14, family="serif"))

```

#Two predicted values are out of bounds-- AL was lower and MA was higher. \*\*This variable (subgroup of interest) will need to be imputed with PMM.\*\*

#\*\*BA\*\*

```{r, message=FALSE, warning=FALSE}

BA\_results\_df %>%

ggplot(., aes(x=1, y=mean)) +

geom\_dotplot(binwidth = 1, binaxis = "y", stackdir = "center") +

theme(axis.title.x=element\_blank(),

axis.text.x=element\_blank(),

axis.ticks.x=element\_blank()) +

geom\_hline(yintercept = 309.5, color="red") +

geom\_hline(yintercept = 276.1, color="red") +

ylab("Score") +

theme(text=element\_text(size=14, family="serif"))

```

#Two predicted values are out of bounds-- AL was lower and MA was higher (again). #\*\*This variable (subgroup of interest) will need to be imputed with PMM.\*\*

#\*\*B\*\*

```{r, message=FALSE, warning=FALSE}

B\_results\_df %>%

ggplot(., aes(x=1, y=mean)) +

geom\_dotplot(binwidth = 1, binaxis = "y", stackdir = "center") +

theme(axis.title.x=element\_blank(),

axis.text.x=element\_blank(),

axis.ticks.x=element\_blank()) +

geom\_hline(yintercept = 273.9625, color="red") +

geom\_hline(yintercept = 245.0225, color="red") +

ylab("Score") +

theme(text=element\_text(size=14, family="serif"))

```

#All of the predicted values are in bound.

#\*\*H\*\*

```{r, message=FALSE, warning=FALSE}

H\_results\_df %>%

ggplot(., aes(x=1, y=mean)) +

geom\_dotplot(binwidth = 1, binaxis = "y", stackdir = "center") +

theme(axis.title.x=element\_blank(),

axis.text.x=element\_blank(),

axis.ticks.x=element\_blank()) +

geom\_hline(yintercept = 256.9075, color="red") +

geom\_hline(yintercept = 282.4075, color="red") +

ylab("Score") +

theme(text=element\_text(size=14, family="serif"))

```

#All predicted values for this subgroup are in bound.

#\*\*API\*\*

```{r, message=FALSE, warning=FALSE}

API\_results\_df %>%

ggplot(., aes(x=1, y=mean)) +

geom\_dotplot(binwidth = 1, binaxis = "y", stackdir = "center") +

theme(axis.title.x=element\_blank(),

axis.text.x=element\_blank(),

axis.ticks.x=element\_blank()) +

geom\_hline(yintercept = 334.6888, color="red") +

geom\_hline(yintercept = 276.1587, color="red") +

ylab("Score") +

theme(text=element\_text(size=14, family="serif"))

```

#All predicted values for this subgroup are in bound.

#\*\*AINA\*\*

```{r, message=FALSE, warning=FALSE}

AINA\_results\_df %>%

ggplot(., aes(x=1, y=mean)) +

geom\_dotplot(binwidth = 1, binaxis = "y", stackdir = "center") +

theme(axis.title.x=element\_blank(),

axis.text.x=element\_blank(),

axis.ticks.x=element\_blank()) +

geom\_hline(yintercept = 266.865, color="red") +

geom\_hline(yintercept = 251.305, color="red") +

ylab("Score") +

theme(text=element\_text(size=14, family="serif"))

```

#Two predicted values are out of bounds-- AZ was lower and MN was higher. \*\*This variable (subgroup of interest) will need to be imputed with PMM.\*\*

#\*\*2+\*\*

```{r, message=FALSE, warning=FALSE}

TP\_results\_df %>%

ggplot(., aes(x=1, y=mean)) +

geom\_dotplot(binwidth = 1, binaxis = "y", stackdir = "center") +

theme(axis.title.x=element\_blank(),

axis.text.x=element\_blank(),

axis.ticks.x=element\_blank()) +

geom\_hline(yintercept = 296.5063, color="red") +

geom\_hline(yintercept = 266.9763, color="red") +

ylab("Score") +

theme(text=element\_text(size=14, family="serif"))

```

#All predicted values for this subgroup are in bound.

#\*\*EL\*\*

```{r, message=FALSE, warning=FALSE}

EL\_results\_df %>%

ggplot(., aes(x=1, y=mean)) +

geom\_dotplot(binwidth = 1, binaxis = "y", stackdir = "center") +

theme(axis.title.x=element\_blank(),

axis.text.x=element\_blank(),

axis.ticks.x=element\_blank()) +

geom\_hline(yintercept = 275.975, color="red") +

geom\_hline(yintercept = 216.415, color="red") +

ylab("Score") +

theme(text=element\_text(size=14, family="serif"))

```

#All predicted values for this subgroup are in bound.

#\*\*Summary:\*\*

#Three subgroups of interest (incomplete variables) need to be imputed with PMM-- SBA, BA and AINA.

### Re-execute MICE procedure for SBA, BA and AINA

#\*\*First, define the method by which each missing variable willS be imputed\*\*

#In the previous set of imputations, "norm" was used for all of the incomplete variables and only needed to be specified once.

```{r, message=FALSE, warning=FALSE}

methods\_string <- c("", "", 'norm', 'norm', 'pmm', 'pmm', "", 'norm', 'norm', 'norm', 'pmm', 'norm', 'norm', "", "", "", "", "")

```

#\*\*SBA (PMM)\*\*

```{r, message=FALSE, warning=FALSE}

#Now for the SBA variable:

mice\_procedure <- function(x){

df\_mice1 <- df\_mice

df\_mice1[x, 5] <- NA

mice(df\_mice1, m = 100, maxit = 15, visitSequence = vis, method = methods\_string, pred = pred\_matrix, seed = 2019)

}

SBA\_results <- lapply(1:length(df\_mice$SBA), mice\_procedure)

```

#\*\*Organize results for SBA and compute mean and variance\*\*

```{r, message=FALSE, warning=FALSE}

SBA\_results\_df <- SBA\_results[[1]][["imp"]][["SBA"]] %>%

slice(1:2) %>%

rbind(SBA\_results[[3]][["imp"]][["SBA"]]) %>%

slice(-3, -5) %>%

rbind(SBA\_results[[4]][["imp"]][["SBA"]]) %>%

slice(-4, -6) %>%

rbind(SBA\_results[[5]][["imp"]][["SBA"]]) %>%

slice(-5, -7) %>%

rbind(SBA\_results[[6]][["imp"]][["SBA"]]) %>%

slice(-6, -8) %>%

rbind(SBA\_results[[7]][["imp"]][["SBA"]]) %>%

slice(-7, -9) %>%

rbind(SBA\_results[[8]][["imp"]][["SBA"]]) %>%

slice(-8, -10) %>%

rbind(SBA\_results[[9]][["imp"]][["SBA"]]) %>%

slice(-9, -11) %>%

rbind(SBA\_results[[10]][["imp"]][["SBA"]]) %>%

slice(-10, -12) %>%

rbind(SBA\_results[[11]][["imp"]][["SBA"]]) %>%

slice(-11, -13) %>%

rbind(SBA\_results[[12]][["imp"]][["SBA"]]) %>%

slice(-12, -14) %>%

rbind(SBA\_results[[13]][["imp"]][["SBA"]]) %>%

slice(-13, -15) %>%

rbind(SBA\_results[[14]][["imp"]][["SBA"]]) %>%

slice(-14, -16) %>%

rbind(SBA\_results[[15]][["imp"]][["SBA"]]) %>%

slice(-15, -17) %>%

rbind(SBA\_results[[16]][["imp"]][["SBA"]]) %>%

slice(-16, -18) %>%

rbind(SBA\_results[[17]][["imp"]][["SBA"]]) %>%

slice(-17, -19) %>%

rbind(SBA\_results[[18]][["imp"]][["SBA"]]) %>%

slice(-18, -20) %>%

rbind(SBA\_results[[19]][["imp"]][["SBA"]]) %>%

slice(-19, -21) %>%

rbind(SBA\_results[[20]][["imp"]][["SBA"]]) %>%

slice(-20, -22) %>%

rbind(SBA\_results[[21]][["imp"]][["SBA"]]) %>%

slice(-21, -23) %>%

rbind(SBA\_results[[22]][["imp"]][["SBA"]]) %>%

slice(-22, -24) %>%

rbind(SBA\_results[[23]][["imp"]][["SBA"]]) %>%

slice(-23, -25) %>%

rbind(SBA\_results[[24]][["imp"]][["SBA"]]) %>%

slice(-24, -26) %>%

rbind(SBA\_results[[25]][["imp"]][["SBA"]]) %>%

slice(-25, -27) %>%

rbind(SBA\_results[[26]][["imp"]][["SBA"]]) %>%

slice(-26, -28) %>%

rbind(SBA\_results[[27]][["imp"]][["SBA"]]) %>%

slice(-27, -29) %>%

rbind(SBA\_results[[28]][["imp"]][["SBA"]]) %>%

slice(-28, -30) %>%

rbind(SBA\_results[[29]][["imp"]][["SBA"]]) %>%

slice(-29, -31) %>%

rbind(SBA\_results[[30]][["imp"]][["SBA"]]) %>%

slice(-30, -32) %>%

rbind(SBA\_results[[31]][["imp"]][["SBA"]]) %>%

slice(-31, -33) %>%

rbind(SBA\_results[[32]][["imp"]][["SBA"]]) %>%

slice(-32, -34) %>%

rbind(SBA\_results[[33]][["imp"]][["SBA"]]) %>%

slice(-33, -35) %>%

rbind(SBA\_results[[34]][["imp"]][["SBA"]]) %>%

slice(-34, -36) %>%

rbind(SBA\_results[[35]][["imp"]][["SBA"]]) %>%

slice(-35, -37) %>%

rbind(SBA\_results[[36]][["imp"]][["SBA"]]) %>%

slice(-36, -38) %>%

rbind(SBA\_results[[37]][["imp"]][["SBA"]]) %>%

slice(-37, -39) %>%

rbind(SBA\_results[[38]][["imp"]][["SBA"]]) %>%

slice(-38, -40) %>%

rbind(SBA\_results[[39]][["imp"]][["SBA"]]) %>%

slice(-39, -41) %>%

rbind(SBA\_results[[40]][["imp"]][["SBA"]]) %>%

slice(-40, -42) %>%

rbind(SBA\_results[[41]][["imp"]][["SBA"]]) %>%

slice(-41, -43) %>%

rbind(SBA\_results[[42]][["imp"]][["SBA"]]) %>%

slice(-42, -44) %>%

rbind(SBA\_results[[43]][["imp"]][["SBA"]]) %>%

slice(-43, -45) %>%

rbind(SBA\_results[[45]][["imp"]][["SBA"]]) %>%

slice(-44) %>%

rbind(SBA\_results[[46]][["imp"]][["SBA"]]) %>%

slice(-46, -47) %>%

rbind(SBA\_results[[47]][["imp"]][["SBA"]]) %>%

slice(-47, -48) %>%

rbind(SBA\_results[[48]][["imp"]][["SBA"]]) %>%

slice(-48, -49) %>%

rbind(SBA\_results[[49]][["imp"]][["SBA"]]) %>%

slice(-49, -50) %>%

rbind(SBA\_results[[50]][["imp"]][["SBA"]]) %>%

slice(-50, -51)

SBA\_results\_df <- mutate(SBA\_results\_df, mean = rowMeans(SBA\_results\_df),

se = rowSds(as.matrix(SBA\_results\_df)))

#Save results to computer as .csv

write.csv(SBA\_results\_df, "[filepath]/SBA-MICE-Results.csv")

```

#\*\*BA (PMM)\*\*

```{r, message=FALSE, warning=FALSE}

#Now for the BA variable:

mice\_procedure <- function(x){

df\_mice1 <- df\_mice

df\_mice1[x, 6] <- NA

mice(df\_mice1, m = 100, maxit = 15, visitSequence = vis, method = methods\_string, pred = pred\_matrix, seed = 2019)

}

BA\_results <- lapply(1:length(df\_mice$BA), mice\_procedure)

```

#\*\*Organize results for BA and compute mean and variance\*\*

```{r, message=FALSE, warning=FALSE}

BA\_results\_df <- BA\_results[[1]][["imp"]][["BA"]] %>%

slice(1:2) %>%

rbind(BA\_results[[3]][["imp"]][["BA"]]) %>%

slice(-3, -5) %>%

rbind(BA\_results[[4]][["imp"]][["BA"]]) %>%

slice(-4, -6) %>%

rbind(BA\_results[[5]][["imp"]][["BA"]]) %>%

slice(-5, -7) %>%

rbind(BA\_results[[6]][["imp"]][["BA"]]) %>%

slice(-6, -8) %>%

rbind(BA\_results[[7]][["imp"]][["BA"]]) %>%

slice(-7, -9) %>%

rbind(BA\_results[[8]][["imp"]][["BA"]]) %>%

slice(-8, -10) %>%

rbind(BA\_results[[9]][["imp"]][["BA"]]) %>%

slice(-9, -11) %>%

rbind(BA\_results[[10]][["imp"]][["BA"]]) %>%

slice(-10, -12) %>%

rbind(BA\_results[[11]][["imp"]][["BA"]]) %>%

slice(-11, -13) %>%

rbind(BA\_results[[12]][["imp"]][["BA"]]) %>%

slice(-12, -14) %>%

rbind(BA\_results[[13]][["imp"]][["BA"]]) %>%

slice(-13, -15) %>%

rbind(BA\_results[[14]][["imp"]][["BA"]]) %>%

slice(-14, -16) %>%

rbind(BA\_results[[15]][["imp"]][["BA"]]) %>%

slice(-15, -17) %>%

rbind(BA\_results[[16]][["imp"]][["BA"]]) %>%

slice(-16, -18) %>%

rbind(BA\_results[[17]][["imp"]][["BA"]]) %>%

slice(-17, -19) %>%

rbind(BA\_results[[18]][["imp"]][["BA"]]) %>%

slice(-18, -20) %>%

rbind(BA\_results[[19]][["imp"]][["BA"]]) %>%

slice(-19, -21) %>%

rbind(BA\_results[[20]][["imp"]][["BA"]]) %>%

slice(-20, -22) %>%

rbind(BA\_results[[21]][["imp"]][["BA"]]) %>%

slice(-21, -23) %>%

rbind(BA\_results[[22]][["imp"]][["BA"]]) %>%

slice(-22, -24) %>%

rbind(BA\_results[[23]][["imp"]][["BA"]]) %>%

slice(-23, -25) %>%

rbind(BA\_results[[24]][["imp"]][["BA"]]) %>%

slice(-24, -26) %>%

rbind(BA\_results[[25]][["imp"]][["BA"]]) %>%

slice(-25, -27) %>%

rbind(BA\_results[[26]][["imp"]][["BA"]]) %>%

slice(-26, -28) %>%

rbind(BA\_results[[27]][["imp"]][["BA"]]) %>%

slice(-27, -29) %>%

rbind(BA\_results[[28]][["imp"]][["BA"]]) %>%

slice(-28, -30) %>%

rbind(BA\_results[[29]][["imp"]][["BA"]]) %>%

slice(-29, -31) %>%

rbind(BA\_results[[30]][["imp"]][["BA"]]) %>%

slice(-30, -32) %>%

rbind(BA\_results[[31]][["imp"]][["BA"]]) %>%

slice(-31, -33) %>%

rbind(BA\_results[[32]][["imp"]][["BA"]]) %>%

slice(-32, -34) %>%

rbind(BA\_results[[33]][["imp"]][["BA"]]) %>%

slice(-33, -35) %>%

rbind(BA\_results[[34]][["imp"]][["BA"]]) %>%

slice(-34, -36) %>%

rbind(BA\_results[[35]][["imp"]][["BA"]]) %>%

slice(-35, -37) %>%

rbind(BA\_results[[36]][["imp"]][["BA"]]) %>%

slice(-36, -38) %>%

rbind(BA\_results[[37]][["imp"]][["BA"]]) %>%

slice(-37, -39) %>%

rbind(BA\_results[[38]][["imp"]][["BA"]]) %>%

slice(-38, -40) %>%

rbind(BA\_results[[39]][["imp"]][["BA"]]) %>%

slice(-39, -41) %>%

rbind(BA\_results[[40]][["imp"]][["BA"]]) %>%

slice(-40, -42) %>%

rbind(BA\_results[[41]][["imp"]][["BA"]]) %>%

slice(-41, -43) %>%

rbind(BA\_results[[42]][["imp"]][["BA"]]) %>%

slice(-42, -44) %>%

rbind(BA\_results[[43]][["imp"]][["BA"]]) %>%

slice(-43, -45) %>%

rbind(BA\_results[[45]][["imp"]][["BA"]]) %>%

slice(-44) %>%

rbind(BA\_results[[46]][["imp"]][["BA"]]) %>%

slice(-46, -47) %>%

rbind(BA\_results[[47]][["imp"]][["BA"]]) %>%

slice(-47, -48) %>%

rbind(BA\_results[[48]][["imp"]][["BA"]]) %>%

slice(-48, -49) %>%

rbind(BA\_results[[49]][["imp"]][["BA"]]) %>%

slice(-49, -50) %>%

rbind(BA\_results[[50]][["imp"]][["BA"]]) %>%

slice(-50, -51)

BA\_results\_df <- mutate(BA\_results\_df, mean = rowMeans(BA\_results\_df),

se = rowSds(as.matrix(BA\_results\_df)))

#Save results to computer as a .csv

write.csv(BA\_results\_df, "[filepath]/BA-MICE-Results.csv")

```

#\*\*AINA\*\*

```{r, message=FALSE, warning=FALSE}

#Now for the AINA variable:

mice\_procedure <- function(x){

df\_mice1 <- df\_mice

df\_mice1[x, 11] <- NA

mice(df\_mice1, m = 100, maxit = 15, visitSequence = vis, method = methods\_string, pred = pred\_matrix, seed = 2019)

}

AINA\_results <- lapply(1:length(df\_mice$AINA), mice\_procedure)

```

#\*\*Organize results for API and compute mean and variance\*\*

```{r, message=FALSE, warning=FALSE}

AINA\_results\_df <- AINA\_results[[2]][["imp"]][["AINA"]] %>%

slice(1:2) %>%

rbind(AINA\_results[[3]][["imp"]][["AINA"]]) %>%

slice(1:2, 4) %>%

rbind(AINA\_results[[23]][["imp"]][["AINA"]]) %>%

slice(1:3, 5:24) %>%

rbind(AINA\_results[[26]][["imp"]][["AINA"]]) %>%

slice(1:23, 44:46) %>%

rbind(AINA\_results[[31]][["imp"]][["AINA"]]) %>%

slice(1:26, 49:53) %>%

rbind(AINA\_results[[33]][["imp"]][["AINA"]]) %>%

slice(1:31, 58:59) %>%

rbind(AINA\_results[[34]][["imp"]][["AINA"]]) %>%

slice(1:33, 61) %>%

rbind(AINA\_results[[36]][["imp"]][["AINA"]]) %>%

slice(1:34, 62:63) %>%

rbind(AINA\_results[[41]][["imp"]][["AINA"]]) %>%

slice(1:36, 65:69) %>%

rbind(AINA\_results[[44]][["imp"]][["AINA"]]) %>%

slice(1:41, 74:76) %>%

rbind(AINA\_results[[47]][["imp"]][["AINA"]]) %>%

slice(1:44, 79:81) %>%

rbind(AINA\_results[[49]][["imp"]][["AINA"]]) %>%

slice(1:47, 84:85) %>%

rbind(AINA\_results[[50]][["imp"]][["AINA"]]) %>%

slice(1:49, 87)

AINA\_results\_df <- mutate(AINA\_results\_df, mean = rowMeans(AINA\_results\_df),

se = rowSds(as.matrix(AINA\_results\_df)))

#Save results to computer as a .csv

write.csv(AINA\_results\_df, "[filepath]/AINA-MICE-Results.csv")

```

### Re-check plausibility of MICE-based estimates of mean subgroup achievement

#(this time just for SBA, BA and AINA)

```{r, message=FALSE, warning=FALSE}

#\*\*Again, free up memory\*\*

rm(list = ls())

```

#\*\*Import results for the three subgroups imputed with PMM\*\*

```{r, message=FALSE, warning=FALSE}

SBA\_results\_df <- read\_csv("[filepath]/SBA-MICE-Results.csv")

BA\_results\_df <- read\_csv("[filepath]/BA-MICE-Results.csv")

AINA\_results\_df <- read\_csv("[filepath]/AINA-MICE-Results.csv")

```

#\*\*Wrangle target values for the three subgroups imputed with PMM\*\*

#\*Save state abbreviations to an object called "State".\*

```{r, message=FALSE, warning=FALSE}

State <- read\_csv("[filepath]/Subgroups of Interest/2+.csv")

State <- State[,1]

```

#Column bind "State" to each of the subgroup results data frames

```{r, message=FALSE, warning=FALSE}

SBA\_results\_df <- cbind(State, SBA\_results\_df)

BA\_results\_df <- cbind(State, BA\_results\_df)

AINA\_results\_df <- cbind(State, AINA\_results\_df)

```

#Keep only State, mean and se variables

```{r, message=FALSE, warning=FALSE}

SBA\_results\_df <- SBA\_results\_df %>%

select("State", "mean", "se")

BA\_results\_df <- BA\_results\_df %>%

select("State", "mean", "se")

AINA\_results\_df <- AINA\_results\_df %>%

select("State", "mean", "se")

```

#Keep only target values

```{r, message=FALSE, warning=FALSE}

#SBA

SBA\_results\_df[2,2:3] <- NA

SBA\_results\_df[44,2:3] <- NA

#BA

BA\_results\_df[2,2:3] <- NA

BA\_results\_df[44,2:3] <- NA

#AINA

AINA\_results\_df[1,2:3] <- NA

AINA\_results\_df[4:22,2:3] <- NA

AINA\_results\_df[24:25,2:3] <- NA

AINA\_results\_df[27:30,2:3] <- NA

AINA\_results\_df[32,2:3] <- NA

AINA\_results\_df[35,2:3] <- NA

AINA\_results\_df[37:40,2:3] <- NA

AINA\_results\_df[42:43,2:3] <- NA

AINA\_results\_df[45:46,2:3] <- NA

AINA\_results\_df[48,2:3] <- NA

```

#\*\*SBA\*\*

```{r, message=FALSE, warning=FALSE}

SBA\_results\_df %>%

ggplot(., aes(x=1, y=mean)) +

geom\_dotplot(binwidth = 1, binaxis = "y", stackdir = "center") +

theme(axis.title.x=element\_blank(),

axis.text.x=element\_blank(),

axis.ticks.x=element\_blank()) +

geom\_hline(yintercept = 294.2487, color="red") +

geom\_hline(yintercept = 272.0388, color="red") +

ylab("Score") +

theme(text=element\_text(size=14, family="serif"))

```

#The predicted values for the SBA subgroup are now all in-bound.

#\*\*BA\*\*

```{r, message=FALSE, warning=FALSE}

BA\_results\_df %>%

ggplot(., aes(x=1, y=mean)) +

geom\_dotplot(binwidth = 1, binaxis = "y", stackdir = "center") +

theme(axis.title.x=element\_blank(),

axis.text.x=element\_blank(),

axis.ticks.x=element\_blank()) +

geom\_hline(yintercept = 309.5, color="red") +

geom\_hline(yintercept = 276.1, color="red") +

ylab("Score") +

theme(text=element\_text(size=14, family="serif"))

```

#The predicted values for the BA subgroup are now all in-bound.

#\*\*AINA\*\*

```{r, message=FALSE, warning=FALSE}

AINA\_results\_df %>%

ggplot(., aes(x=1, y=mean)) +

geom\_dotplot(binwidth = 1, binaxis = "y", stackdir = "center") +

theme(axis.title.x=element\_blank(),

axis.text.x=element\_blank(),

axis.ticks.x=element\_blank()) +

geom\_hline(yintercept = 266.865, color="red") +

geom\_hline(yintercept = 251.305, color="red") +

ylab("Score") +

theme(text=element\_text(size=14, family="serif"))

```

#The predicted values for the AINA subgroup are now all in-bound.

###Wrangle final predicted values and their variance estimate into one data file

```{r, message=FALSE, warning=FALSE}

#Clear environment

rm(list = ls())

```

#Import results from MICE procedure

```{r, message=FALSE, warning=FALSE}

NHS\_results\_df <- read\_csv("[filepath]/NHS-MICE-Results.csv")

HS\_results\_df <- read\_csv("[filepath]/HS-MICE-Results.csv")

SBA\_results\_df <- read\_csv("[filepath]/SBA-MICE-Results.csv")

BA\_results\_df <- read\_csv("[filepath]/BA-MICE-Results.csv")

B\_results\_df <- read\_csv("[filepath]/B-MICE-Results.csv")

H\_results\_df <- read\_csv("[filepath]/H-MICE-Results.csv")

API\_results\_df <- read\_csv("[filepath]/API-MICE-Results.csv")

AINA\_results\_df <- read\_csv("[filepath]/AINA-MICE-Results.csv")

TP\_results\_df <- read\_csv("[filepath]/TP-MICE-Results.csv")

EL\_results\_df <- read\_csv("[filepath]/EL-MICE-Results.csv")

```

#Save state abbreviations to an object called "State"

```{r, message=FALSE, warning=FALSE}

State <- read\_csv("[filepath]/Subgroups of Interest/2+.csv")

State <- State[,1]

```

#Column bind "State" to each of the subgroup results data frames

```{r, message=FALSE, warning=FALSE}

NHS\_results\_df <- cbind(State, NHS\_results\_df)

HS\_results\_df <- cbind(State, HS\_results\_df)

SBA\_results\_df <- cbind(State, SBA\_results\_df)

BA\_results\_df <- cbind(State, BA\_results\_df)

B\_results\_df <- cbind(State, B\_results\_df)

H\_results\_df <- cbind(State, H\_results\_df)

API\_results\_df <- cbind(State, API\_results\_df)

AINA\_results\_df <- cbind(State, AINA\_results\_df)

TP\_results\_df <- cbind(State, TP\_results\_df)

EL\_results\_df <- cbind(State, EL\_results\_df)

```

#Keep only State, mean and se variables

```{r, message=FALSE, warning=FALSE}

NHS\_results\_df <- NHS\_results\_df %>%

select("State", "mean", "se")

HS\_results\_df <- HS\_results\_df %>%

select("State", "mean", "se")

SBA\_results\_df <- SBA\_results\_df %>%

select("State", "mean", "se")

BA\_results\_df <- BA\_results\_df %>%

select("State", "mean", "se")

B\_results\_df <- B\_results\_df %>%

select("State", "mean", "se")

H\_results\_df <- H\_results\_df %>%

select("State", "mean", "se")

API\_results\_df <- API\_results\_df %>%

select("State", "mean", "se")

AINA\_results\_df <- AINA\_results\_df %>%

select("State", "mean", "se")

TP\_results\_df <- TP\_results\_df %>%

select("State", "mean", "se")

EL\_results\_df <- EL\_results\_df %>%

select("State", "mean", "se")

```

#Keep only target values

```{r, message=FALSE, warning=FALSE}

#NHS

NHS\_results\_df[2,2:3] <- NA

NHS\_results\_df[44,2:3] <- NA

#HS

HS\_results\_df[2,2:3] <- NA

HS\_results\_df[44,2:3] <- NA

#SBA

SBA\_results\_df[2,2:3] <- NA

SBA\_results\_df[44,2:3] <- NA

#BA

BA\_results\_df[2,2:3] <- NA

BA\_results\_df[44,2:3] <- NA

#B

B\_results\_df[11:12,2:3] <- NA

B\_results\_df[19,2:3] <- NA

B\_results\_df[26,2:3] <- NA

B\_results\_df[29,2:3] <- NA

B\_results\_df[31,2:3] <- NA

B\_results\_df[37,2:3] <- NA

B\_results\_df[41,2:3] <- NA

B\_results\_df[44:45,2:3] <- NA

B\_results\_df[50,2:3] <- NA

#H

H\_results\_df[19,2:3] <- NA

H\_results\_df[45,2:3] <- NA

H\_results\_df[48,2:3] <- NA

#API

API\_results\_df[1,2:3] <- NA

API\_results\_df[4,2:3] <- NA

API\_results\_df[12,2:3] <- NA

API\_results\_df[14,2:3] <- NA

API\_results\_df[18:19,2:3] <- NA

API\_results\_df[24:27,2:3] <- NA

API\_results\_df[31,2:3] <- NA

API\_results\_df[34,2:3] <- NA

API\_results\_df[36,2:3] <- NA

API\_results\_df[40:42,2:3] <- NA

API\_results\_df[44:45,2:3] <- NA

API\_results\_df[48,2:3] <- NA

API\_results\_df[50,2:3] <- NA

#AINA

AINA\_results\_df[1,2:3] <- NA

AINA\_results\_df[4:22,2:3] <- NA

AINA\_results\_df[24:25,2:3] <- NA

AINA\_results\_df[27:30,2:3] <- NA

AINA\_results\_df[32,2:3] <- NA

AINA\_results\_df[35,2:3] <- NA

AINA\_results\_df[37:40,2:3] <- NA

AINA\_results\_df[42:43,2:3] <- NA

AINA\_results\_df[45:46,2:3] <- NA

AINA\_results\_df[48,2:3] <- NA

#TP

TP\_results\_df[1,2:3] <- NA

TP\_results\_df[3:4,2:3] <- NA

TP\_results\_df[7:8,2:3] <- NA

TP\_results\_df[12:13,2:3] <- NA

TP\_results\_df[18:19,2:3] <- NA

TP\_results\_df[21:22,2:3] <- NA

TP\_results\_df[24:25,2:3] <- NA

TP\_results\_df[29:32,2:3] <- NA

TP\_results\_df[34,2:3] <- NA

TP\_results\_df[40:42,2:3] <- NA

TP\_results\_df[44:45,2:3] <- NA

TP\_results\_df[48:50,2:3] <- NA

#EL

EL\_results\_df[1,2:3] <- NA

EL\_results\_df[8,2:3] <- NA

EL\_results\_df[12,2:3] <- NA

EL\_results\_df[17:19,2:3] <- NA

EL\_results\_df[24:27,2:3] <- NA

EL\_results\_df[29:30,2:3] <- NA

EL\_results\_df[34,2:3] <- NA

EL\_results\_df[37,2:3] <- NA

EL\_results\_df[41:42,2:3] <- NA

EL\_results\_df[45,2:3] <- NA

EL\_results\_df[48,2:3] <- NA

EL\_results\_df[50,2:3] <- NA

```

#Save these predicted means and their standard errors

```{r, message=FALSE, warning=FALSE}

write.csv(NHS\_results\_df, "[filepath]/NHS-MICE-ESTIMATES.csv")

write.csv(HS\_results\_df, "[filepath]/HS-MICE-ESTIMATES.csv")

write.csv(SBA\_results\_df, "[filepath]/SBA-MICE-ESTIMATES.csv")

write.csv(BA\_results\_df, "[filepath]/BA-MICE-ESTIMATES.csv")

write.csv(B\_results\_df, "[filepath]/B-MICE-ESTIMATES.csv")

write.csv(H\_results\_df, "[filepath]/H-MICE-ESTIMATES.csv")

write.csv(API\_results\_df, "[filepath]/API-MICE-ESTIMATES.csv")

write.csv(AINA\_results\_df, "[filepath]/AINA-MICE-ESTIMATES.csv")

write.csv(TP\_results\_df, "[filepath]/TP-MICE-ESTIMATES.csv")

write.csv(EL\_results\_df, "[filepath]/EL-MICE-ESTIMATES.csv")

```

### Calculate wMAE for MICE technique

```{r, message=FALSE, warning=FALSE}

#\*\*Again, free up memory\*\*

rm(list = ls())

```

#\*\*Define a wMAE function\*\*

```{r, message=FALSE, warning=FALSE}

wmae <- function(x){

w\_abs\_errors <- abs(x[,2] - x[,4])/x[,3]

mean(w\_abs\_errors[,1])

}

```

#\*\*By subgroup\*\*

#NHS

```{r, message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

NHS\_predicted <- read\_csv("[filepath]/NHS-MICE-ESTIMATES.csv")

NHS\_predicted <- NHS\_predicted %>%

select(-"X1") %>%

rename("MICE\_mean" = mean) %>%

rename("MICE\_se" = se)

NHS\_observed <- read\_csv("[filepath]/Subgroups of Interest/NHS.csv")

NHS\_observed <- NHS\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

NHS <- inner\_join(NHS\_observed, NHS\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(NHS)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, message=FALSE, warning=FALSE}

#Make NHS tidy

source <- rep(c("NAEP"), 48)

source <- as.data.frame(source)

part1 <- cbind(NHS$State, source, NHS$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("MICE"), 48)

source <- as.data.frame(source)

part2 <- cbind(NHS$State, source, NHS$MICE\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

NHS\_tidy <- rbind(part1, part2)

NHS\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

NHS\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

NHS\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#HS

```{r, message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

HS\_predicted <- read\_csv("[filepath]/HS-MICE-ESTIMATES.csv")

HS\_predicted <- HS\_predicted %>%

select(-"X1") %>%

rename("MICE\_mean" = mean) %>%

rename("MICE\_se" = se)

HS\_observed <- read\_csv("[filepath]/Subgroups of Interest/HS.csv")

HS\_observed <- HS\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

HS <- inner\_join(HS\_observed, HS\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(HS)

```

#Create visuals and compute descriptive statistics to compare distributions

#Make HS tidy

```{r, message=FALSE, warning=FALSE}

source <- rep(c("NAEP"), 48)

source <- as.data.frame(source)

part1 <- cbind(HS$State, source, HS$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("MICE"), 48)

source <- as.data.frame(source)

part2 <- cbind(HS$State, source, HS$MICE\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

HS\_tidy <- rbind(part1, part2)

HS\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

HS\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

HS\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#SBA

```{r, message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

SBA\_predicted <- read\_csv("[filepath]/SBA-MICE-ESTIMATES.csv")

SBA\_predicted <- SBA\_predicted %>%

select(-"X1") %>%

rename("MICE\_mean" = mean) %>%

rename("MICE\_se" = se)

SBA\_observed <- read\_csv("[filepath]/Subgroups of Interest/SBA.csv")

SBA\_observed <- SBA\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

SBA <- inner\_join(SBA\_observed, SBA\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(SBA)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, message=FALSE, warning=FALSE}

#Make SBA tidy

source <- rep(c("NAEP"), 48)

source <- as.data.frame(source)

part1 <- cbind(SBA$State, source, SBA$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("MICE"), 48)

source <- as.data.frame(source)

part2 <- cbind(SBA$State, source, SBA$MICE\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

SBA\_tidy <- rbind(part1, part2)

SBA\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

SBA\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

SBA\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#BA

```{r, message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

BA\_predicted <- read\_csv("[filepath]/BA-MICE-ESTIMATES.csv")

BA\_predicted <- BA\_predicted %>%

select(-"X1") %>%

rename("MICE\_mean" = mean) %>%

rename("MICE\_se" = se)

BA\_observed <- read\_csv("[filepath]/Subgroups of Interest/BA.csv")

BA\_observed <- BA\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

BA <- inner\_join(BA\_observed, BA\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(BA)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, message=FALSE, warning=FALSE}

#Make BA tidy

source <- rep(c("NAEP"), 48)

source <- as.data.frame(source)

part1 <- cbind(BA$State, source, BA$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("MICE"), 48)

source <- as.data.frame(source)

part2 <- cbind(BA$State, source, BA$MICE\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

BA\_tidy <- rbind(part1, part2)

BA\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

BA\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

BA\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#B

```{r, message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

B\_predicted <- read\_csv("[filepath]/B-MICE-ESTIMATES.csv")

B\_predicted <- B\_predicted %>%

select(-"X1") %>%

rename("MICE\_mean" = mean) %>%

rename("MICE\_se" = se)

B\_observed <- read\_csv("[filepath]/Subgroups of Interest/B.csv")

B\_observed <- B\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

B <- inner\_join(B\_observed, B\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(B)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, message=FALSE, warning=FALSE}

#Make B tidy

source <- rep(c("NAEP"), 39)

source <- as.data.frame(source)

part1 <- cbind(B$State, source, B$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("MICE"), 39)

source <- as.data.frame(source)

part2 <- cbind(B$State, source, B$MICE\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

B\_tidy <- rbind(part1, part2)

B\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

B\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

B\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#H

```{r, message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

H\_predicted <- read\_csv("[filepath]/H-MICE-ESTIMATES.csv")

H\_predicted <- H\_predicted %>%

select(-"X1") %>%

rename("MICE\_mean" = mean) %>%

rename("MICE\_se" = se)

H\_observed <- read\_csv("[filepath]/Subgroups of Interest/H.csv")

H\_observed <- H\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

H <- inner\_join(H\_observed, H\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(H)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, message=FALSE, warning=FALSE}

#Make H tidy

source <- rep(c("NAEP"), 47)

source <- as.data.frame(source)

part1 <- cbind(H$State, source, H$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("MICE"), 47)

source <- as.data.frame(source)

part2 <- cbind(H$State, source, H$MICE\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

H\_tidy <- rbind(part1, part2)

H\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

H\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

H\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#API

```{r, message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

API\_predicted <- read\_csv("[filepath]/API-MICE-ESTIMATES.csv")

API\_predicted <- API\_predicted %>%

select(-"X1") %>%

rename("MICE\_mean" = mean) %>%

rename("MICE\_se" = se)

API\_observed <- read\_csv("[filepath]/Subgroups of Interest/API.csv")

API\_observed <- API\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

API <- inner\_join(API\_observed, API\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(API)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, message=FALSE, warning=FALSE}

#Make API tidy

source <- rep(c("NAEP"), 30)

source <- as.data.frame(source)

part1 <- cbind(API$State, source, API$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("MICE"), 30)

source <- as.data.frame(source)

part2 <- cbind(API$State, source, API$MICE\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

API\_tidy <- rbind(part1, part2)

API\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

API\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

API\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#AINA

```{r, message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

AINA\_predicted <- read\_csv("[filepath]/AINA-MICE-ESTIMATES.csv")

AINA\_predicted <- AINA\_predicted %>%

select(-"X1") %>%

rename("MICE\_mean" = mean) %>%

rename("MICE\_se" = se)

AINA\_observed <- read\_csv("[filepath]/Subgroups of Interest/AINA.csv")

AINA\_observed <- AINA\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

AINA <- inner\_join(AINA\_observed, AINA\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(AINA)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, message=FALSE, warning=FALSE}

#Make AINA tidy

source <- rep(c("NAEP"), 13)

source <- as.data.frame(source)

part1 <- cbind(AINA$State, source, AINA$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("MICE"), 13)

source <- as.data.frame(source)

part2 <- cbind(AINA$State, source, AINA$MICE\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

AINA\_tidy <- rbind(part1, part2)

AINA\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

AINA\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

AINA\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#TP

#Import/Wrangle predicted and observed data

TP\_predicted <- read\_csv("[filepath]/TP-MICE-ESTIMATES.csv")

TP\_predicted <- TP\_predicted %>%

select(-"X1") %>%

rename("MICE\_mean" = mean) %>%

rename("MICE\_se" = se)

TP\_observed <- read\_csv("[filepath]/Subgroups of Interest/2+.csv")

TP\_observed <- TP\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

TP <- inner\_join(TP\_observed, TP\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(TP)

#Create visuals and compute descriptive statistics to compare distributions

```{r, message=FALSE, warning=FALSE}

#Make TP tidy

source <- rep(c("NAEP"), 24)

source <- as.data.frame(source)

part1 <- cbind(TP$State, source, TP$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("MICE"), 24)

source <- as.data.frame(source)

part2 <- cbind(TP$State, source, TP$MICE\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

TP\_tidy <- rbind(part1, part2)

TP\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

TP\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

TP\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#EL

```{r, message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

EL\_predicted <- read\_csv("[filepath]/EL-MICE-ESTIMATES.csv")

EL\_predicted <- EL\_predicted %>%

select(-"X1") %>%

rename("MICE\_mean" = mean) %>%

rename("MICE\_se" = se)

EL\_observed <- read\_csv("[filepath]/Subgroups of Interest/EL.csv")

EL\_observed <- EL\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

EL <- inner\_join(EL\_observed, EL\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(EL)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, message=FALSE, warning=FALSE}

#Make EL tidy

source <- rep(c("NAEP"), 31)

source <- as.data.frame(source)

part1 <- cbind(EL$State, source, EL$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("MICE"), 31)

source <- as.data.frame(source)

part2 <- cbind(EL$State, source, EL$MICE\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

EL\_tidy <- rbind(part1, part2)

EL\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

EL\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

EL\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

```{r, message=FALSE, warning=FALSE}

#\*\*Overall\*\*

#Row bind subgroup datasets into an "overall dataset"

overall <- rbind(NHS, HS, SBA, BA, B, H, API, AINA, TP, EL)

#Calculate wMAE for MICE across subgroups

wmae(overall)

```

### Calculate Coverage for MICE

```{r, message=FALSE, warning=FALSE}

#First, add the median of the NAEP-reported state-level standard deviations to each respective subgroup's data set (this permits calculation of the b-statistic).

NHS$median\_sd <- 31.5

HS$median\_sd <- 32.6

SBA$median\_sd <- 30.6

BA$median\_sd <- 34.4

B$median\_sd <- 33.4

H$median\_sd <- 34.0

API$median\_sd <- 38.1

AINA$median\_sd <- 35.4

TP$median\_sd <- 35.2

EL$median\_sd <- 33.3

```

#\*\*By subgroup\*\*

```{r, message=FALSE, warning=FALSE}

#NHS

NHS$b\_statisic <- abs(NHS$NAEP\_mean-NHS$MICE\_mean)/NHS$median\_sd

mean(NHS$b\_statisic <= 0.20)

```

#NHS-visual

```{r, message=FALSE, warning=FALSE}

NHS$lowerbound <- NHS$NAEP\_mean - 0.2\*NHS$median\_sd

NHS$upperbound <- NHS$NAEP\_mean + 0.2\*NHS$median\_sd

NHS <- NHS %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = NHS, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = NHS, aes(MICE\_mean, State, col = "MICE prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#HS

```{r, message=FALSE, warning=FALSE}

HS$b\_statisic <- abs(HS$NAEP\_mean-HS$MICE\_mean)/HS$median\_sd

mean(HS$b\_statisic <= 0.20)

```

#HS-visual

```{r, message=FALSE, warning=FALSE}

HS$lowerbound <- HS$NAEP\_mean - 0.2\*HS$median\_sd

HS$upperbound <- HS$NAEP\_mean + 0.2\*HS$median\_sd

HS <- HS %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = HS, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = HS, aes(MICE\_mean, State, col = "MICE prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#SBA

```{r, message=FALSE, warning=FALSE}

SBA$b\_statisic <- abs(SBA$NAEP\_mean-SBA$MICE\_mean)/SBA$median\_sd

mean(SBA$b\_statisic <= 0.20)

```

#SBA-visual

```{r, message=FALSE, warning=FALSE}

SBA$lowerbound <- SBA$NAEP\_mean - 0.2\*SBA$median\_sd

SBA$upperbound <- SBA$NAEP\_mean + 0.2\*SBA$median\_sd

SBA <- SBA %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = SBA, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = SBA, aes(MICE\_mean, State, col = "MICE prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#BA

```{r, message=FALSE, warning=FALSE}

BA$b\_statisic <- abs(BA$NAEP\_mean-BA$MICE\_mean)/BA$median\_sd

mean(BA$b\_statisic <= 0.20)

```

#BA-visual

```{r, message=FALSE, warning=FALSE}

BA$lowerbound <- BA$NAEP\_mean - 0.2\*BA$median\_sd

BA$upperbound <- BA$NAEP\_mean + 0.2\*BA$median\_sd

BA <- BA %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = BA, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = BA, aes(MICE\_mean, State, col = "MICE prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#B

```{r, message=FALSE, warning=FALSE}

B$b\_statisic <- abs(B$NAEP\_mean-B$MICE\_mean)/B$median\_sd

mean(B$b\_statisic <= 0.20)

```

#B-visual

```{r, message=FALSE, warning=FALSE}

B$lowerbound <- B$NAEP\_mean - 0.2\*B$median\_sd

B$upperbound <- B$NAEP\_mean + 0.2\*B$median\_sd

B <- B %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = B, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = B, aes(MICE\_mean, State, col = "MICE prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#H

```{r, message=FALSE, warning=FALSE}

H$b\_statisic <- abs(H$NAEP\_mean-H$MICE\_mean)/H$median\_sd

mean(H$b\_statisic <= 0.20)

```

#H-visual

```{r, message=FALSE, warning=FALSE}

H$lowerbound <- H$NAEP\_mean - 0.2\*H$median\_sd

H$upperbound <- H$NAEP\_mean + 0.2\*H$median\_sd

H <- H %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = H, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = H, aes(MICE\_mean, State, col = "MICE prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#API

```{r, message=FALSE, warning=FALSE}

API$b\_statisic <- abs(API$NAEP\_mean-API$MICE\_mean)/API$median\_sd

mean(API$b\_statisic <= 0.20)

```

#API-visual

```{r, message=FALSE, warning=FALSE}

API$lowerbound <- API$NAEP\_mean - 0.2\*API$median\_sd

API$upperbound <- API$NAEP\_mean + 0.2\*API$median\_sd

API <- API %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = API, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = API, aes(MICE\_mean, State, col = "MICE prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#AINA

```{r, message=FALSE, warning=FALSE}

AINA$b\_statisic <- abs(AINA$NAEP\_mean-AINA$MICE\_mean)/AINA$median\_sd

mean(AINA$b\_statisic <= 0.20)

```

#AINA-visual

```{r, message=FALSE, warning=FALSE}

AINA$lowerbound <- AINA$NAEP\_mean - 0.2\*AINA$median\_sd

AINA$upperbound <- AINA$NAEP\_mean + 0.2\*AINA$median\_sd

AINA <- AINA %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = AINA, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = AINA, aes(MICE\_mean, State, col = "MICE prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#TP (2+)

```{r, message=FALSE, warning=FALSE}

TP$b\_statisic <- abs(TP$NAEP\_mean-TP$MICE\_mean)/TP$median\_sd

mean(TP$b\_statisic <= 0.20)

```

#TP-visual

```{r, message=FALSE, warning=FALSE}

TP$lowerbound <- TP$NAEP\_mean - 0.2\*TP$median\_sd

TP$upperbound <- TP$NAEP\_mean + 0.2\*TP$median\_sd

TP <- TP %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = TP, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = TP, aes(MICE\_mean, State, col = "MICE prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#EL

```{r, message=FALSE, warning=FALSE}

EL$b\_statisic <- abs(EL$NAEP\_mean-EL$MICE\_mean)/EL$median\_sd

mean(EL$b\_statisic <= 0.20)

```

#EL-visual

```{r, message=FALSE, warning=FALSE}

EL$lowerbound <- EL$NAEP\_mean - 0.2\*EL$median\_sd

EL$upperbound <- EL$NAEP\_mean + 0.2\*EL$median\_sd

EL <- EL %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = EL, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = EL, aes(MICE\_mean, State, col = "MICE prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#\*\*Overall\*\*

```{r, message=FALSE, warning=FALSE}

#Row bind datasets

#Row bind subgroup datasets into an "overall dataset"

overall <- rbind(NHS, HS, SBA, BA, B, H, API, AINA, TP, EL)

#Calculate coverage across groups

mean(overall$b\_statisic <= 0.20)

```

**Determining Samples Sizes to Draw (Chapter 4)**

The text that follows in courier font represents *STATA* script. Lines of script that begin with a slash (i.e., /) are comments (not code). Note, the name of filepaths (e.g., “C:/Foldername/Foldername/…/Filename”) are replaced with [filepath] throughout. In addition, places in text that would reveal count sizes are replaced with [censored] to comply with NAEP reporting requirements and protect student privacy.

**Program/File Type:** Stata/.DTA File

**Dissertation Section:** Estimates of Mean Math Achievement with the FH technique

//Import data

use "[filepath]/Data-in-stata.dta"

//Keep only public school students

keep if PUBPRIV == 1 // Drops total samples from [censored]cases to [censored]

//First, for NHS subgroup, keep only AK (2) and UT (49)

keep if FIPS15 == 2 | FIPS15 == 49

//Examine number of cases (students) per "PARED" group

tab PARED

////////////////////////RESULTS for NHS, HS, SBA, & BA//////////////////////////

//For some reason, AK and UT students did not indicate their parents level

// of education. Over 99% of values for this variable are coded as

// omitted for these two states. Instead I will use the maximum sample size

// used for the remaining subgroups of interest. The rationale here is that

// we can typically expect at least 62 students to be be drawn for the PARED

// subgroups during a typical year. This maximum is [censored] (same as B

// subgroup).

//

// NB: I can draw this conclusion from the limited amount of code because

// AK and UT are the only states for which mean estimates are not reported

// for all of the parental level of education subgroups.

////////////////////////////////////////////////////////////////////////////////

clear

//Re-Import data

use "[filepath]/Data-in-stata.dta"

//Keep only public school students

keep if PUBPRIV == 1 // Drops total sample from [censored] cases to [censored]

//Now, for B subgroup keep only HI (15), ID (16), ME (23), MT (30), NH (33),

// NM (35), OR (41), SD (46), UT (49), VT, (50), WY (56)

keep if FIPS15 == 15 | FIPS15 == 16 | FIPS15 == 23 | FIPS15 == 30 ///

| FIPS15 == 33 | FIPS15 == 35 | FIPS15 == 41 | FIPS15 == 46 | FIPS15 == 49 ///

| FIPS15 == 50 | FIPS15 == 56

//Examine number of cases (students) per SDRACEM group

tab SDRACEM FIPS15

//The number of B cases (students) per state are [censored]

//////////////////////////////////RESULTS for B/////////////////////////////////

//The sample to size to draw for the B subgroup is [censored].

////////////////////////////////////////////////////////////////////////////////

clear

//Re-Import data

use "[filepath]/Data-in-stata.dta"

//Keep only public school students

keep if PUBPRIV == 1 // Drops total sample from [censored] cases to [censored]

//Now, for H subgroup keep only ME (23), VT (50) and WV (54).

keep if FIPS15 == 23 | FIPS15 == 50 | FIPS15 == 54

//Examine number of cases (students) per "SDRACEM" group

tab SDRACEM FIPS15

//The number of H cases (students) per state are [censored]

//////////////////////////////////RESULTS for H/////////////////////////////////

//The sample to size to draw for the H subgroup is [censored].

////////////////////////////////////////////////////////////////////////////////

clear

//Re-Import data

use "[filepath]/Data-in-stata.dta"

//Keep only public school students

keep if PUBPRIV == 1 // Drops total sample from [censored] cases to [censored]

//Now, for API subgroup keep only AL(1), AR(5), ID(16), IN(18), LA(22), ME(23),

// MS(28), MO(29), MT(30), NE(31), NM(35), ND(38), OK(40), SC(45), SD(46),

// TN(47), UT(49), VT(50), WV(54), WY(56)

keep if FIPS15 == 1 | FIPS15 == 5 | FIPS15 == 16 | FIPS15 == 18 ///

| FIPS15 == 22 | FIPS15 == 23 | FIPS15 == 28 | FIPS15 == 29 | FIPS15 == 30 ///

| FIPS15 == 31 | FIPS15 == 35 | FIPS15 == 38 | FIPS15 == 40 | FIPS15 == 45 ///

| FIPS15 == 46 | FIPS15 == 47 | FIPS15 == 49 | FIPS15 == 50 | FIPS15 == 54 ///

| FIPS15 == 56

//Examine number of cases (students) per "SDRACEM" group

tab SDRACEM FIPS15

//The number of API cases (students) per state are [censored]

//////////////////////////////////RESULTS for API///////////////////////////////

//The sample to size to draw for the API subgroup is [censored].

////////////////////////////////////////////////////////////////////////////////

// NEED to use "DRACEM" for 2+ group

//Re-Import data

use "[filepath]/Data-in-stata.dta"

//Keep only public school students

keep if PUBPRIV == 1 // Drops total sample from [censored] cases to [censored]

//Now, for AINA subgroup keep only AL(1), AR(5), CA(6), CO(8), CT(9), DE(10), FL(12), GA(13), HI(15), ID(16), IL(17), IN(18), IA(19), KS(20), KY(21), LA(22), ME(23), MD(24), MA(25), MI(26), MS(28), MO(29), NE(31), NV(32), NH(33), NJ(34), NY(36), OH(39), OR(41), PA(42), RI(44), SC(45), TN (47), TX(48), VT(50), VA(51), WV(54)

keep if FIPS15 == 1 | FIPS15 == 5 | FIPS15 == 6 | FIPS15 == 8 ///

| FIPS15 == 9 | FIPS15 == 10 | FIPS15 == 12 | FIPS15 == 13 | FIPS15 == 15 ///

| FIPS15 == 16 | FIPS15 == 17 | FIPS15 == 18 | FIPS15 == 19 | FIPS15 == 20 ///

| FIPS15 == 21 | FIPS15 == 22 | FIPS15 == 23 | FIPS15 == 24 | FIPS15 == 25 ///

| FIPS15 == 26 | FIPS15 == 28 | FIPS15 == 29 | FIPS15 == 31 | FIPS15 == 32 ///

| FIPS15 == 33 | FIPS15 == 34 | FIPS15 == 36 | FIPS15 == 39 | FIPS15 == 41 ///

| FIPS15 == 42 | FIPS15 == 44 | FIPS15 == 45 | FIPS15 == 47 | FIPS15 == 48 ///

| FIPS15 == 50 | FIPS15 == 51 | FIPS15 == 54

tab SDRACEM FIPS15

//The number of AINA cases (students) per state are [censored]

//////////////////////////////////RESULTS for AINA///////////////////////////////

//The sample to size to draw for the AINA subgroup is [censored].

////////////////////////////////////////////////////////////////////////////////

clear

//Re-Import data

use "[filepath]/Data-in-stata.dta"

//Keep only public school students

keep if PUBPRIV == 1 // Drops total sample from [censored] cases to [censored]

//Now, for 2+ subgroup keep only AL(1), AZ(4), AR(5), CT(9), DE(10), ID(16),

// IL(17), LA(22), ME(23), MA(25), MI(26), MS(28), MO(29), NH(33), NJ(34),

// NM(35), NY(36), ND(38), SC(45), SD(46), TN(47), UT(49), VT(50), WV(54),

// WI(55), WY(56)

keep if FIPS15 == 1 | FIPS15 == 4 | FIPS15 == 5 | FIPS15 == 9 ///

| FIPS15 == 10 | FIPS15 == 16 | FIPS15 == 17 | FIPS15 == 22 | FIPS15 == 23 ///

| FIPS15 == 25 | FIPS15 == 26 | FIPS15 == 28 | FIPS15 == 29 | FIPS15 == 33 ///

| FIPS15 == 34 | FIPS15 == 35 | FIPS15 == 36 | FIPS15 == 38 | FIPS15 == 45 ///

| FIPS15 == 46 | FIPS15 == 47 | FIPS15 == 49 | FIPS15 == 50 | FIPS15 == 54 ///

| FIPS15 == 55 | FIPS15 == 56

tab SDRACEM FIPS15

//The number of 2+ cases (students) per state are

// [censored]

//////////////////////////////////RESULTS for 2+ (TP)///////////////////////////

//The sample to size to draw for the 2+ subgroup is [censored].

////////////////////////////////////////////////////////////////////////////////

clear

//Re-Import data

use "[filepath]/Data-in-stata.dta"

//Keep only public school students

keep if PUBPRIV == 1 // Drops total sample from [censored] cases to [censored]

//Now, for EL subgroup keep only AL(1), DE (10), ID (16), KY(21), LA(22),

// ME(23), MS(28), MO(29), MT(30), NE(31), NH(33), NJ(34), ND(38), OR(41),

// SD(46), TN(47), VT(50), WV(54), WY(56)

keep if FIPS15 == 1 | FIPS15 == 10 | FIPS15 == 16 | FIPS15 == 21 ///

| FIPS15 == 22 | FIPS15 == 23 | FIPS15 == 28 | FIPS15 == 29 | FIPS15 == 30 ///

| FIPS15 == 31 | FIPS15 == 33 | FIPS15 == 34 | FIPS15 == 38 | FIPS15 == 41 ///

| FIPS15 == 46 | FIPS15 == 47 | FIPS15 == 50 | FIPS15 == 54 | FIPS15 == 56 ///

tab LEP FIPS15

//The number of EL cases (students) per state are

// [censored]

//////////////////////////////////RESULTS for EL////////////////////////////////

//The sample to size to draw for the EL subgroup is [censored].

// Curiously, five separate states, for which mean achievement of the EL subgroup // was not reported, had more than 62 students.

////////////////////////////////////////////////////////////////////////////////

**Computing Direct Estimates (Chapter 4)**

The text that follows in courier font represents *STATA* script. Lines of script that begin with a slash (i.e., /) are comments (not code). Note, the name of filepaths (e.g., “C:/Foldername/Foldername/…/Filename”) are replaced with [filepath] throughout. In addition, places in text that would reveal count sizes are replaced with [censored] to comply with NAEP reporting requirements and protect student privacy.

**Program/File Type:** Stata/.DTA File

**Dissertation Section:** Description of FH estimates by subgroup

//////Computing Direct Estimates for NHS Subgroup//////

//Target values are for all states except for AK and UT//

/////State label reference///

//Alabama(1) Alaska(2) Arizona(4) Arkansas(5) California(6) Colorado(8)

//Connecticut(9) Delaware(10) Florida(12) Georgia(13) Hawaii(15) Idaho(16)

//Illinois(17) Indiana(18) Iowa(19) Kansas(20) Kentucky(21) Louisiana(22)

//Maine(23) Maryland(24) Massachusetts(25) Michigan(26) Minnesota(27)

//Mississippi(28) Missouri(29) Montana(30) Nebraska(31) Nevada(32)

//New Hampshire(33) New Jersey(34) New Mexico(35) New York(36) North Carolina(37)

//North Dakota(38) Ohio(39) Oklahoma(40) Oregon(41) Pennsylvania(42)

//Rhode Island(44) South Carolina(45) South Dakota(46) Tennessee(47)

//Texas(48) Utah (49) Vermont(50) Virginia(51) Washington(53)

//West Virginia(54) Wisconsin(55) Wyoming(56)

/////////////////////Alabama(1)//////////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 1

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-AL.dta", replace

//^variable

clear

/////////////////////Arizona(4)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 4

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-AZ.dta", replace

//^variable

clear

/////////////////////Arkansas(5)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 5

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-AR.dta", replace

//^variable

clear

/////////////////////California(6)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 6

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-CA.dta", replace

//^variable

clear

/////////////////////Colorado(8)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 8

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-CO.dta", replace

//^variable

clear

/////////////////////Connecticut(9)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 9

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-CT.dta", replace

//^variable

clear

/////////////////////Delaware(10)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 10

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-DE.dta", replace

//^variable

clear

/////////////////////Florida(12)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 12

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-FL.dta", replace

//^variable

clear

/////////////////////Georgia(13)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 13

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-GA.dta", replace

//^variable

clear

/////////////////////Hawaii(15)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 15

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-HI.dta", replace

//^variable

clear

/////////////////////Idaho(16)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 16

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-ID.dta", replace

//^variable

clear

/////////////////////Illinois(17)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 17

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-IL.dta", replace

//^variable

clear

/////////////////////Indiana(18)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 18

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-IN.dta", replace

//^variable

clear

/////////////////////Iowa(19)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 19

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-IA.dta", replace

//^variable

clear

/////////////////////Kansas(20)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 20

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-KS.dta", replace

//^variable

clear

/////////////////////Kentucky(21)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 21

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-KY.dta", replace

//^variable

clear

/////////////////////Louisiana(22)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 22

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-LA.dta", replace

//^variable

clear

/////////////////////Maine(23)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 23

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-ME.dta", replace

//^variable

clear

/////////////////////Maryland(24)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 24

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-MD.dta", replace

//^variable

clear

/////////////////////Massachusetts(25)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 25

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-MA.dta", replace

//^variable

clear

/////////////////////Michigan(26)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 26

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-MI.dta", replace

//^variable

clear

/////////////////////Minnesota(27)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 27

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-MN.dta", replace

//^variable

clear

/////////////////////Mississippi(28)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 28

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-MS.dta", replace

//^variable

clear

/////////////////////Missouri(29)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 29

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-MO.dta", replace

//^variable

clear

/////////////////////Montana(30)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 30

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-MT.dta", replace

//^variable

clear

/////////////////////Nebraska(31)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 31

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-NE.dta", replace

//^variable

clear

/////////////////////Nevada(32)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 32

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-NV.dta", replace

//^variable

clear

///////////////////////New Hampshire(33)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 33

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-NH.dta", replace

//^variable

clear

/////////////////////New Jersey(34)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 34

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-NJ.dta", replace

//^variable

clear

/////////////////////New Mexico(35)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 35

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-NM.dta", replace

//^variable

clear

/////////////////////New York(36)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 36

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-NY.dta", replace

//^variable

clear

/////////////////////North Carolina(37)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 37

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-NC.dta", replace

//^variable

clear

/////////////////////North Dakota(38)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 38

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-ND.dta", replace

//^variable

clear

/////////////////////Ohio(39)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 39

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-OH.dta", replace

//^variable

clear

/////////////////////Oklahoma(40)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 40

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-OK.dta", replace

//^variable

clear

/////////////////////Oregon(41)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 41

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-OR.dta", replace

//^variable

clear

/////////////////////Pennsylvania(42)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 42

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-PA.dta", replace

//^variable

clear

/////////////////////Rhode Island(44)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 44

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-RI.dta", replace

//^variable

clear

/////////////////////South Carolina(45)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 45

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-SC.dta", replace

//^variable

clear

/////////////////////South Dakota(46)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 46

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-SD.dta", replace

//^variable

clear

/////////////////////Tennessee(47)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 47

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-TN.dta", replace

//^variable

clear

/////////////////////Texas(48)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 48

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-TX.dta", replace

//^variable

clear

/////////////////////Vermont(50)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 50

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-VT.dta", replace

//^variable

clear

/////////////////////Virginia(51)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 51

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-VA.dta", replace

//^variable

clear

/////////////////////Washington(53)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 53

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-WA.dta", replace

//^variable

clear

/////////////////////West Virginia(54)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 54

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-WV.dta", replace

//^variable

clear

/////////////////////Wisconsin(55)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 55

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-WI.dta", replace

//^variable

clear

/////////////////////Wyoming(56)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 56

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/NHS-WY.dta", replace

//^variable

clear

//////Computing Direct Estimates for HS Subgroup//////

//Target values are for all states except for AK and UT//

/////State label reference///

//Alabama(1) Alaska(2) Arizona(4) Arkansas(5) California(6) Colorado(8)

//Connecticut(9) Delaware(10) Florida(12) Georgia(13) Hawaii(15) Idaho(16)

//Illinois(17) Indiana(18) Iowa(19) Kansas(20) Kentucky(21) Louisiana(22)

//Maine(23) Maryland(24) Massachusetts(25) Michigan(26) Minnesota(27)

//Mississippi(28) Missouri(29) Montana(30) Nebraska(31) Nevada(32)

//New Hampshire(33) New Jersey(34) New Mexico(35) New York(36) North Carolina(37)

//North Dakota(38) Ohio(39) Oklahoma(40) Oregon(41) Pennsylvania(42)

//Rhode Island(44) South Carolina(45) South Dakota(46) Tennessee(47)

//Texas(48) Utah (49) Vermont(50) Virginia(51) Washington(53)

//West Virginia(54) Wisconsin(55) Wyoming(56)

/////////////////////Alabama(1)//////////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 1

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-AL.dta", replace

//^variable

clear

/////////////////////Arizona(4)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 4

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-AZ.dta", replace

//^variable

clear

/////////////////////Arkansas(5)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 5

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-AR.dta", replace

//^variable

clear

/////////////////////California(6)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 6

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-CA.dta", replace

//^variable

clear

/////////////////////Colorado(8)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 8

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-CO.dta", replace

//^variable

clear

/////////////////////Connecticut(9)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 9

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-CT.dta", replace

//^variable

clear

/////////////////////Delaware(10)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 10

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-DE.dta", replace

//^variable

clear

/////////////////////Florida(12)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 12

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-FL.dta", replace

//^variable

clear

/////////////////////Georgia(13)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 13

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-GA.dta", replace

//^variable

clear

/////////////////////Hawaii(15)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 15

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-HI.dta", replace

//^variable

clear

/////////////////////Idaho(16)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 16

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-ID.dta", replace

//^variable

clear

/////////////////////Illinois(17)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 17

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-IL.dta", replace

//^variable

clear

/////////////////////Indiana(18)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 18

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-IN.dta", replace

//^variable

clear

/////////////////////Iowa(19)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 19

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-IA.dta", replace

//^variable

clear

/////////////////////Kansas(20)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 20

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-KS.dta", replace

//^variable

clear

/////////////////////Kentucky(21)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 21

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-KY.dta", replace

//^variable

clear

/////////////////////Louisiana(22)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 22

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-LA.dta", replace

//^variable

clear

/////////////////////Maine(23)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 23

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-ME.dta", replace

//^variable

clear

/////////////////////Maryland(24)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 24

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-MD.dta", replace

//^variable

clear

/////////////////////Massachusetts(25)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 25

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-MA.dta", replace

//^variable

clear

/////////////////////Michigan(26)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 26

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-MI.dta", replace

//^variable

clear

/////////////////////Minnesota(27)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 27

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-MN.dta", replace

//^variable

clear

/////////////////////Mississippi(28)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 28

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-MS.dta", replace

//^variable

clear

/////////////////////Missouri(29)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 29

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-MO.dta", replace

//^variable

clear

/////////////////////Montana(30)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 30

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-MT.dta", replace

//^variable

clear

/////////////////////Nebraska(31)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 31

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-NE.dta", replace

//^variable

clear

/////////////////////Nevada(32)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 32

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-NV.dta", replace

//^variable

clear

///////////////////////New Hampshire(33)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 33

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-NH.dta", replace

//^variable

clear

/////////////////////New Jersey(34)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 34

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-NJ.dta", replace

//^variable

clear

/////////////////////New Mexico(35)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 35

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-NM.dta", replace

//^variable

clear

/////////////////////New York(36)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 36

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-NY.dta", replace

//^variable

clear

/////////////////////North Carolina(37)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 37

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-NC.dta", replace

//^variable

clear

/////////////////////North Dakota(38)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 38

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-ND.dta", replace

//^variable

clear

/////////////////////Ohio(39)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 39

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-OH.dta", replace

//^variable

clear

/////////////////////Oklahoma(40)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 40

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-OK.dta", replace

//^variable

clear

/////////////////////Oregon(41)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 41

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-OR.dta", replace

//^variable

clear

/////////////////////Pennsylvania(42)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 42

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-PA.dta", replace

//^variable

clear

/////////////////////Rhode Island(44)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 44

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-RI.dta", replace

//^variable

clear

/////////////////////South Carolina(45)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 45

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-SC.dta", replace

//^variable

clear

/////////////////////South Dakota(46)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 46

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-SD.dta", replace

//^variable

clear

/////////////////////Tennessee(47)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 47

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-TN.dta", replace

//^variable

clear

/////////////////////Texas(48)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 48

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-TX.dta", replace

//^variable

clear

/////////////////////Vermont(50)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 50

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-VT.dta", replace

//^variable

clear

/////////////////////Virginia(51)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 51

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-VA.dta", replace

//^variable

clear

/////////////////////Washington(53)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 53

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-WA.dta", replace

//^variable

clear

/////////////////////West Virginia(54)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 54

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-WV.dta", replace

//^variable

clear

/////////////////////Wisconsin(55)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 55

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-WI.dta", replace

//^variable

clear

/////////////////////Wyoming(56)/////////////////////

use "[pathfile]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[pathfile]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 56

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[pathfile]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[pathfile]/HS-WY.dta", replace

//^variable

clear

//////Computing Direct Estimates for SBA Subgroup//////

//Target values are for all states except for AK and UT//

/////State label reference///

//Alabama(1) Alaska(2) Arizona(4) Arkansas(5) California(6) Colorado(8)

//Connecticut(9) Delaware(10) Florida(12) Georgia(13) Hawaii(15) Idaho(16)

//Illinois(17) Indiana(18) Iowa(19) Kansas(20) Kentucky(21) Louisiana(22)

//Maine(23) Maryland(24) Massachusetts(25) Michigan(26) Minnesota(27)

//Mississippi(28) Missouri(29) Montana(30) Nebraska(31) Nevada(32)

//New Hampshire(33) New Jersey(34) New Mexico(35) New York(36) North Carolina(37)

//North Dakota(38) Ohio(39) Oklahoma(40) Oregon(41) Pennsylvania(42)

//Rhode Island(44) South Carolina(45) South Dakota(46) Tennessee(47)

//Texas(48) Utah (49) Vermont(50) Virginia(51) Washington(53)

//West Virginia(54) Wisconsin(55) Wyoming(56)

/////////////////////Alabama(1)//////////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 1

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-AL.dta", replace

//^variable

clear

/////////////////////Arizona(4)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 4

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-AZ.dta", replace

//^variable

clear

/////////////////////Arkansas(5)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 5

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-AR.dta", replace

//^variable

clear

/////////////////////California(6)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 6

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-CA.dta", replace

//^variable

clear

/////////////////////Colorado(8)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 8

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-CO.dta", replace

//^variable

clear

/////////////////////Connecticut(9)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 9

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-CT.dta", replace

//^variable

clear

/////////////////////Delaware(10)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 10

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-DE.dta", replace

//^variable

clear

/////////////////////Florida(12)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 12

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-FL.dta", replace

//^variable

clear

/////////////////////Georgia(13)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 13

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-GA.dta", replace

//^variable

clear

/////////////////////Hawaii(15)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 15

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-HI.dta", replace

//^variable

clear

/////////////////////Idaho(16)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 16

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-ID.dta", replace

//^variable

clear

/////////////////////Illinois(17)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 17

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-IL.dta", replace

//^variable

clear

/////////////////////Indiana(18)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 18

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-IN.dta", replace

//^variable

clear

/////////////////////Iowa(19)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 19

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-IA.dta", replace

//^variable

clear

/////////////////////Kansas(20)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 20

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-KS.dta", replace

//^variable

clear

/////////////////////Kentucky(21)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 21

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-KY.dta", replace

//^variable

clear

/////////////////////Louisiana(22)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 22

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-LA.dta", replace

//^variable

clear

/////////////////////Maine(23)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 23

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-ME.dta", replace

//^variable

clear

/////////////////////Maryland(24)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 24

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-MD.dta", replace

//^variable

clear

/////////////////////Massachusetts(25)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 25

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-MA.dta", replace

//^variable

clear

/////////////////////Michigan(26)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 26

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-MI.dta", replace

//^variable

clear

/////////////////////Minnesota(27)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 27

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-MN.dta", replace

//^variable

clear

/////////////////////Mississippi(28)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 28

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-MS.dta", replace

//^variable

clear

/////////////////////Missouri(29)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 29

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-MO.dta", replace

//^variable

clear

/////////////////////Montana(30)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 30

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-MT.dta", replace

//^variable

clear

/////////////////////Nebraska(31)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 31

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-NE.dta", replace

//^variable

clear

/////////////////////Nevada(32)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 32

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-NV.dta", replace

//^variable

clear

///////////////////////New Hampshire(33)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 33

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-NH.dta", replace

//^variable

clear

/////////////////////New Jersey(34)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 34

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-NJ.dta", replace

//^variable

clear

/////////////////////New Mexico(35)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 35

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-NM.dta", replace

//^variable

clear

/////////////////////New York(36)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 36

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-NY.dta", replace

//^variable

clear

/////////////////////North Carolina(37)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 37

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-NC.dta", replace

//^variable

clear

/////////////////////North Dakota(38)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 38

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-ND.dta", replace

//^variable

clear

/////////////////////Ohio(39)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 39

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-OH.dta", replace

//^variable

clear

/////////////////////Oklahoma(40)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 40

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-OK.dta", replace

//^variable

clear

/////////////////////Oregon(41)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 41

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-OR.dta", replace

//^variable

clear

/////////////////////Pennsylvania(42)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 42

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-PA.dta", replace

//^variable

clear

/////////////////////Rhode Island(44)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 44

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-RI.dta", replace

//^variable

clear

/////////////////////South Carolina(45)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 45

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-SC.dta", replace

//^variable

clear

/////////////////////South Dakota(46)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 46

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-SD.dta", replace

//^variable

clear

/////////////////////Tennessee(47)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 47

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-TN.dta", replace

//^variable

clear

/////////////////////Texas(48)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 48

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-TX.dta", replace

//^variable

clear

/////////////////////Vermont(50)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 50

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-VT.dta", replace

//^variable

clear

/////////////////////Virginia(51)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 51

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-VA.dta", replace

//^variable

clear

/////////////////////Washington(53)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 53

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-WA.dta", replace

//^variable

clear

/////////////////////West Virginia(54)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 54

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-WV.dta", replace

//^variable

clear

/////////////////////Wisconsin(55)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 55

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-WI.dta", replace

//^variable

clear

/////////////////////Wyoming(56)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 56

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/SBA-WY.dta", replace

//^variable

clear

//////Computing Direct Estimates for BA Subgroup//////

//Target values are for all states except for AK and UT//

/////State label reference///

//Alabama(1) Alaska(2) Arizona(4) Arkansas(5) California(6) Colorado(8)

//Connecticut(9) Delaware(10) Florida(12) Georgia(13) Hawaii(15) Idaho(16)

//Illinois(17) Indiana(18) Iowa(19) Kansas(20) Kentucky(21) Louisiana(22)

//Maine(23) Maryland(24) Massachusetts(25) Michigan(26) Minnesota(27)

//Mississippi(28) Missouri(29) Montana(30) Nebraska(31) Nevada(32)

//New Hampshire(33) New Jersey(34) New Mexico(35) New York(36) North Carolina(37)

//North Dakota(38) Ohio(39) Oklahoma(40) Oregon(41) Pennsylvania(42)

//Rhode Island(44) South Carolina(45) South Dakota(46) Tennessee(47)

//Texas(48) Utah (49) Vermont(50) Virginia(51) Washington(53)

//West Virginia(54) Wisconsin(55) Wyoming(56)

/////////////////////Alabama(1)//////////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 1

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-AL.dta", replace

//^variable

clear

/////////////////////Arizona(4)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 4

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-AZ.dta", replace

//^variable

clear

/////////////////////Arkansas(5)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 5

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-AR.dta", replace

//^variable

clear

/////////////////////California(6)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 6

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-CA.dta", replace

//^variable

clear

/////////////////////Colorado(8)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 8

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-CO.dta", replace

//^variable

clear

/////////////////////Connecticut(9)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 9

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-CT.dta", replace

//^variable

clear

/////////////////////Delaware(10)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 10

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-DE.dta", replace

//^variable

clear

/////////////////////Florida(12)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 12

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-FL.dta", replace

//^variable

clear

/////////////////////Georgia(13)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 13

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-GA.dta", replace

//^variable

clear

/////////////////////Hawaii(15)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 15

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-HI.dta", replace

//^variable

clear

/////////////////////Idaho(16)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 16

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-ID.dta", replace

//^variable

clear

/////////////////////Illinois(17)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 17

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-IL.dta", replace

//^variable

clear

/////////////////////Indiana(18)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 18

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-IN.dta", replace

//^variable

clear

/////////////////////Iowa(19)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 19

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-IA.dta", replace

//^variable

clear

/////////////////////Kansas(20)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 20

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-KS.dta", replace

//^variable

clear

/////////////////////Kentucky(21)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 21

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-KY.dta", replace

//^variable

clear

/////////////////////Louisiana(22)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 22

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-LA.dta", replace

//^variable

clear

/////////////////////Maine(23)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 23

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-ME.dta", replace

//^variable

clear

/////////////////////Maryland(24)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 24

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-MD.dta", replace

//^variable

clear

/////////////////////Massachusetts(25)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 25

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-MA.dta", replace

//^variable

clear

/////////////////////Michigan(26)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 26

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-MI.dta", replace

//^variable

clear

/////////////////////Minnesota(27)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 27

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-MN.dta", replace

//^variable

clear

/////////////////////Mississippi(28)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 28

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-MS.dta", replace

//^variable

clear

/////////////////////Missouri(29)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 29

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-MO.dta", replace

//^variable

clear

/////////////////////Montana(30)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 30

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-MT.dta", replace

//^variable

clear

/////////////////////Nebraska(31)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 31

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-NE.dta", replace

//^variable

clear

/////////////////////Nevada(32)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 32

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-NV.dta", replace

//^variable

clear

///////////////////////New Hampshire(33)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 33

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-NH.dta", replace

//^variable

clear

/////////////////////New Jersey(34)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 34

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-NJ.dta", replace

//^variable

clear

/////////////////////New Mexico(35)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 35

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-NM.dta", replace

//^variable

clear

/////////////////////New York(36)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 36

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-NY.dta", replace

//^variable

clear

/////////////////////North Carolina(37)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 37

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-NC.dta", replace

//^variable

clear

/////////////////////North Dakota(38)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 38

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-ND.dta", replace

//^variable

clear

/////////////////////Ohio(39)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 39

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-OH.dta", replace

//^variable

clear

/////////////////////Oklahoma(40)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 40

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-OK.dta", replace

//^variable

clear

/////////////////////Oregon(41)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 41

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-OR.dta", replace

//^variable

clear

/////////////////////Pennsylvania(42)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 42

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-PA.dta", replace

//^variable

clear

/////////////////////Rhode Island(44)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 44

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-RI.dta", replace

//^variable

clear

/////////////////////South Carolina(45)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 45

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-SC.dta", replace

//^variable

clear

/////////////////////South Dakota(46)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 46

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-SD.dta", replace

//^variable

clear

/////////////////////Tennessee(47)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 47

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-TN.dta", replace

//^variable

clear

/////////////////////Texas(48)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 48

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-TX.dta", replace

//^variable

clear

/////////////////////Vermont(50)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 50

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-VT.dta", replace

//^variable

clear

/////////////////////Virginia(51)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 51

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-VA.dta", replace

//^variable

clear

/////////////////////Washington(53)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 53

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-WA.dta", replace

//^variable

clear

/////////////////////West Virginia(54)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 54

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-WV.dta", replace

//^variable

clear

/////////////////////Wisconsin(55)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 55

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-WI.dta", replace

//^variable

clear

/////////////////////Wyoming(56)/////////////////////

use “[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & PARED == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using “[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 56

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use “[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save “[filepath]/BA-WY.dta", replace

//^variable

clear

//////Computing Direct Estimates for B Subgroup//////

// Target values are for all states except for...

// HI, ID, ME, MT, NH, NM, OR, SD, UT, VT, WY

/////State label reference///

//Alabama(1) Alaska(2) Arizona(4) Arkansas(5) California(6) Colorado(8)

//Connecticut(9) Delaware(10) Florida(12) Georgia(13) Hawaii(15) Idaho(16)

//Illinois(17) Indiana(18) Iowa(19) Kansas(20) Kentucky(21) Louisiana(22)

//Maine(23) Maryland(24) Massachusetts(25) Michigan(26) Minnesota(27)

//Mississippi(28) Missouri(29) Montana(30) Nebraska(31) Nevada(32)

//New Hampshire(33) New Jersey(34) New Mexico(35) New York(36) North Carolina(37)

//North Dakota(38) Ohio(39) Oklahoma(40) Oregon(41) Pennsylvania(42)

//Rhode Island(44) South Carolina(45) South Dakota(46) Tennessee(47)

//Texas(48) Utah (49) Vermont(50) Virginia(51) Washington(53)

//West Virginia(54) Wisconsin(55) Wyoming(56)

/////////////////////Alabama(1)//////////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 1

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-AL.dta", replace

//^variable

clear

/////////////////////Alaska(2)//////////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 2

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-AK.dta", replace

//^variable

clear

/////////////////////Arizona(4)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 4

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-AZ.dta", replace

//^variable

clear

/////////////////////Arkansas(5)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 5

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-AR.dta", replace

//^variable

clear

/////////////////////California(6)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 6

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-CA.dta", replace

//^variable

clear

/////////////////////Colorado(8)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 8

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-CO.dta", replace

//^variable

clear

/////////////////////Connecticut(9)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 9

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-CT.dta", replace

//^variable

clear

/////////////////////Delaware(10)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 10

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-DE.dta", replace

//^variable

clear

/////////////////////Florida(12)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 12

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-FL.dta", replace

//^variable

clear

/////////////////////Georgia(13)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 13

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-GA.dta", replace

//^variable

clear

/////////////////////Illinois(17)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 17

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-IL.dta", replace

//^variable

clear

/////////////////////Indiana(18)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 18

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-IN.dta", replace

//^variable

clear

/////////////////////Iowa(19)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 19

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-IA.dta", replace

//^variable

clear

/////////////////////Kansas(20)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 20

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-KS.dta", replace

//^variable

clear

/////////////////////Kentucky(21)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 21

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-KY.dta", replace

//^variable

clear

/////////////////////Louisiana(22)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 22

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-LA.dta", replace

//^variable

clear

/////////////////////Maryland(24)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 24

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-MD.dta", replace

//^variable

clear

/////////////////////Massachusetts(25)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 25

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-MA.dta", replace

//^variable

clear

/////////////////////Michigan(26)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 26

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-MI.dta", replace

//^variable

clear

/////////////////////Minnesota(27)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 27

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-MN.dta", replace

//^variable

clear

/////////////////////Mississippi(28)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 28

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-MS.dta", replace

//^variable

clear

/////////////////////Missouri(29)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 29

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-MO.dta", replace

//^variable

clear

/////////////////////Nebraska(31)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 31

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-NE.dta", replace

//^variable

clear

/////////////////////Nevada(32)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 32

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-NV.dta", replace

//^variable

clear

/////////////////////New Jersey(34)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 34

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-NJ.dta", replace

//^variable

clear

/////////////////////New York(36)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 36

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-NY.dta", replace

//^variable

clear

/////////////////////North Carolina(37)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 37

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-NC.dta", replace

//^variable

clear

/////////////////////North Dakota(38)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 38

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-ND.dta", replace

//^variable

clear

/////////////////////Ohio(39)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 39

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-OH.dta", replace

//^variable

clear

/////////////////////Oklahoma(40)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 40

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-OK.dta", replace

//^variable

clear

/////////////////////Pennsylvania(42)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 42

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-PA.dta", replace

//^variable

clear

/////////////////////Rhode Island(44)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 44

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-RI.dta", replace

//^variable

clear

/////////////////////South Carolina(45)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 45

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-SC.dta", replace

//^variable

clear

/////////////////////Tennessee(47)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 47

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-TN.dta", replace

//^variable

clear

/////////////////////Texas(48)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 48

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-TX.dta", replace

//^variable

clear

/////////////////////Virginia(51)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 51

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-VA.dta", replace

//^variable

clear

/////////////////////Washington(53)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 53

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-WA.dta", replace

//^variable

clear

/////////////////////West Virginia(54)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 54

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-WV.dta", replace

//^variable

clear

/////////////////////Wisconsin(55)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 2 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 55

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/B-WI.dta", replace

//^variable

clear

/////Computing Direct Estimates for H Subgroup//////

//Target values are for all states except for ME, VT and WV

/////State label reference///

//Alabama(1) Alaska(2) Arizona(4) Arkansas(5) California(6) Colorado(8)

//Connecticut(9) Delaware(10) Florida(12) Georgia(13) Hawaii(15) Idaho(16)

//Illinois(17) Indiana(18) Iowa(19) Kansas(20) Kentucky(21) Louisiana(22)

//Maine(23) Maryland(24) Massachusetts(25) Michigan(26) Minnesota(27)

//Mississippi(28) Missouri(29) Montana(30) Nebraska(31) Nevada(32)

//New Hampshire(33) New Jersey(34) New Mexico(35) New York(36) North Carolina(37)

//North Dakota(38) Ohio(39) Oklahoma(40) Oregon(41) Pennsylvania(42)

//Rhode Island(44) South Carolina(45) South Dakota(46) Tennessee(47)

//Texas(48) Utah (49) Vermont(50) Virginia(51) Washington(53)

//West Virginia(54) Wisconsin(55) Wyoming(56)

/////////////////////Alabama(1)//////////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 1

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-AL.dta", replace

//^variable

clear

/////////////////////Alaska(2)//////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 2

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-AK.dta", replace

//^variable

clear

/////////////////////Arizona(4)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 4

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-AZ.dta", replace

//^variable

clear

/////////////////////Arkansas(5)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 5

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-AR.dta", replace

//^variable

clear

/////////////////////California(6)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 6

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-CA.dta", replace

//^variable

clear

/////////////////////Colorado(8)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 8

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-CO.dta", replace

//^variable

clear

/////////////////////Connecticut(9)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 9

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-CT.dta", replace

//^variable

clear

/////////////////////Delaware(10)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 10

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-DE.dta", replace

//^variable

clear

/////////////////////Florida(12)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 12

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-FL.dta", replace

//^variable

clear

/////////////////////Georgia(13)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 13

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-GA.dta", replace

//^variable

clear

/////////////////////Hawaii(15)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 15

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-HI.dta", replace

//^variable

clear

/////////////////////Idaho(16)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 16

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-ID.dta", replace

//^variable

clear

/////////////////////Illinois(17)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 17

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-IL.dta", replace

//^variable

clear

/////////////////////Indiana(18)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 18

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-IN.dta", replace

//^variable

clear

/////////////////////Iowa(19)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 19

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-IA.dta", replace

//^variable

clear

/////////////////////Kansas(20)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 20

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-KS.dta", replace

//^variable

clear

/////////////////////Kentucky(21)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 21

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-KY.dta", replace

//^variable

clear

/////////////////////Louisiana(22)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 22

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-LA.dta", replace

//^variable

clear

/////////////////////Maryland(24)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 24

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-MD.dta", replace

//^variable

clear

/////////////////////Massachusetts(25)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 25

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-MA.dta", replace

//^variable

clear

/////////////////////Michigan(26)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 26

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-MI.dta", replace

//^variable

clear

/////////////////////Minnesota(27)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 27

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-MN.dta", replace

//^variable

clear

/////////////////////Mississippi(28)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 28

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-MS.dta", replace

//^variable

clear

/////////////////////Missouri(29)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 29

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-MO.dta", replace

//^variable

clear

/////////////////////Montana(30)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 30

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-MT.dta", replace

//^variable

clear

/////////////////////Nebraska(31)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 31

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-NE.dta", replace

//^variable

clear

/////////////////////Nevada(32)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 32

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-NV.dta", replace

//^variable

clear

///////////////////////New Hampshire(33)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 33

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-NH.dta", replace

//^variable

clear

/////////////////////New Jersey(34)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 34

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-NJ.dta", replace

//^variable

clear

/////////////////////New Mexico(35)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 35

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-NM.dta", replace

//^variable

clear

/////////////////////New York(36)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 36

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-NY.dta", replace

//^variable

clear

/////////////////////North Carolina(37)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 37

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-NC.dta", replace

//^variable

clear

/////////////////////North Dakota(38)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 38

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-ND.dta", replace

//^variable

clear

/////////////////////Ohio(39)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 39

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-OH.dta", replace

//^variable

clear

/////////////////////Oklahoma(40)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 40

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-OK.dta", replace

//^variable

clear

/////////////////////Oregon(41)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 41

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-OR.dta", replace

//^variable

clear

/////////////////////Pennsylvania(42)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 42

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-PA.dta", replace

//^variable

clear

/////////////////////Rhode Island(44)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 44

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-RI.dta", replace

//^variable

clear

/////////////////////South Carolina(45)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 45

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-SC.dta", replace

//^variable

clear

/////////////////////South Dakota(46)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 46

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-SD.dta", replace

//^variable

clear

/////////////////////Tennessee(47)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 47

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-TN.dta", replace

//^variable

clear

/////////////////////Texas(48)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 48

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-TX.dta", replace

//^variable

clear

/////////////////////Utah (49)///////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 49

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-UT.dta", replace

//^variable

clear

/////////////////////Virginia(51)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 51

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-VA.dta", replace

//^variable

clear

/////////////////////Washington(53)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 53

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-WA.dta", replace

//^variable

clear

/////////////////////Wisconsin(55)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 55

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-WI.dta", replace

//^variable

clear

/////////////////////Wyoming(56)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 3 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 56

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/H-WY.dta", replace

//^variable

clear

//////Computing Direct Estimates for API Subgroup//////

//Target values are for all states except for AL, AR, ID, IN, LA, ME,

// MS, MO, MT, NE, NM, ND, OK, SC, SD, TN, UT, VT, WV, WY

/////State label reference///

//Alabama(1) Alaska(2) Arizona(4) Arkansas(5) California(6) Colorado(8)

//Connecticut(9) Delaware(10) Florida(12) Georgia(13) Hawaii(15) Idaho(16)

//Illinois(17) Indiana(18) Iowa(19) Kansas(20) Kentucky(21) Louisiana(22)

//Maine(23) Maryland(24) Massachusetts(25) Michigan(26) Minnesota(27)

//Mississippi(28) Missouri(29) Montana(30) Nebraska(31) Nevada(32)

//New Hampshire(33) New Jersey(34) New Mexico(35) New York(36) North Carolina(37)

//North Dakota(38) Ohio(39) Oklahoma(40) Oregon(41) Pennsylvania(42)

//Rhode Island(44) South Carolina(45) South Dakota(46) Tennessee(47)

//Texas(48) Utah (49) Vermont(50) Virginia(51) Washington(53)

//West Virginia(54) Wisconsin(55) Wyoming(56)

/////////////////////Alaska(2)//////////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 2

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-AK.dta", replace

//^variable

clear

/////////////////////Arizona(4)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 4

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-AZ.dta", replace

//^variable

clear

/////////////////////California(6)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 6

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-CA.dta", replace

//^variable

clear

/////////////////////Colorado(8)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 8

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-CO.dta", replace

//^variable

clear

/////////////////////Connecticut(9)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 9

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-CT.dta", replace

//^variable

clear

/////////////////////Delaware(10)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 10

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-DE.dta", replace

//^variable

clear

/////////////////////Florida(12)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 12

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-FL.dta", replace

//^variable

clear

/////////////////////Georgia(13)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 13

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-GA.dta", replace

//^variable

clear

/////////////////////Hawaii(15)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 15

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-HI.dta", replace

//^variable

clear

/////////////////////Illinois(17)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 17

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-IL.dta", replace

//^variable

clear

/////////////////////Iowa(19)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 19

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-IA.dta", replace

//^variable

clear

/////////////////////Kansas(20)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 20

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-KS.dta", replace

//^variable

clear

/////////////////////Kentucky(21)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 21

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-KY.dta", replace

//^variable

clear

/////////////////////Maryland(24)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 24

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-MD.dta", replace

//^variable

clear

/////////////////////Massachusetts(25)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 25

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-MA.dta", replace

//^variable

clear

/////////////////////Michigan(26)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 26

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-MI.dta", replace

//^variable

clear

/////////////////////Minnesota(27)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 27

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-MN.dta", replace

//^variable

clear

/////////////////////Nevada(32)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 32

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-NV.dta", replace

//^variable

clear

///////////////////////New Hampshire(33)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 33

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-NH.dta", replace

//^variable

clear

/////////////////////New Jersey(34)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 34

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-NJ.dta", replace

//^variable

clear

/////////////////////New York(36)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 36

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-NY.dta", replace

//^variable

clear

/////////////////////North Carolina(37)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 37

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-NC.dta", replace

//^variable

clear

/////////////////////Ohio(39)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 39

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-OH.dta", replace

//^variable

clear

/////////////////////Oregon(41)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 41

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-OR.dta", replace

//^variable

clear

/////////////////////Pennsylvania(42)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 42

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-PA.dta", replace

//^variable

clear

/////////////////////Rhode Island(44)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 44

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-RI.dta", replace

//^variable

clear

/////////////////////Texas(48)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 48

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-TX.dta", replace

//^variable

clear

/////////////////////Virginia(51)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 51

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-VA.dta", replace

//^variable

clear

/////////////////////Washington(53)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 53

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-WA.dta", replace

//^variable

clear

/////////////////////Wisconsin(55)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 4 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 55

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/API-WI.dta", replace

//^variable

clear

//////Computing Direct Estimates for AINA Subgroup//////

//Target values are for all states except for AL, AR, CA, CO, CT, DE, FL, GA,

// HI, ID, IL, IN, IA, KS, KY, LA, ME, MD, MA, MI, MS, MO, NE, NV, NH, NJ, NY,

// OH, OR, PA, RI, SC, TN, TX, VT, VA, WV

/////State label reference///

//Alabama(1) Alaska(2) Arizona(4) Arkansas(5) California(6) Colorado(8)

//Connecticut(9) Delaware(10) Florida(12) Georgia(13) Hawaii(15) Idaho(16)

//Illinois(17) Indiana(18) Iowa(19) Kansas(20) Kentucky(21) Louisiana(22)

//Maine(23) Maryland(24) Massachusetts(25) Michigan(26) Minnesota(27)

//Mississippi(28) Missouri(29) Montana(30) Nebraska(31) Nevada(32)

//New Hampshire(33) New Jersey(34) New Mexico(35) New York(36) North Carolina(37)

//North Dakota(38) Ohio(39) Oklahoma(40) Oregon(41) Pennsylvania(42)

//Rhode Island(44) South Carolina(45) South Dakota(46) Tennessee(47)

//Texas(48) Utah (49) Vermont(50) Virginia(51) Washington(53)

//West Virginia(54) Wisconsin(55) Wyoming(56)

/////////////////////Alaska(2)//////////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 5 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 2

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/AINA-AK.dta", replace

//^variable

clear

/////////////////////Arizona(4)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 5 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 4

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/AINA-AZ.dta", replace

//^variable

clear

/////////////////////Minnesota(27)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 5 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 27

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/AINA-MN.dta", replace

//^variable

clear

////////////////////Montana(30)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 5 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 30

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/AINA-MT.dta", replace

//^variable

clear

/////////////////////New Mexico(35)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 5 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 35

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/AINA-NM.dta", replace

//^variable

clear

/////////////////////North Carolina(37)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 5 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 37

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/AINA-NC.dta", replace

//^variable

clear

/////////////////////North Dakota(38)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 5 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 38

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/AINA-ND.dta", replace

//^variable

clear

/////////////////////Oklahoma(40)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 5 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 40

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/AINA-OK.dta", replace

//^variable

clear

/////////////////////South Dakota(46)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 5 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 46

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/AINA-SD.dta", replace

//^variable

clear

////////////////////Utah(49)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 5 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 49

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/AINA-UT.dta", replace

//^variable

clear

/////////////////////Washington(53)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 5 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 53

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/AINA-WA.dta", replace

//^variable

clear

/////////////////////Wisconsin(55)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 5 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 55

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/AINA-WI.dta", replace

//^variable

clear

/////////////////////Wyoming(56)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 5 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 56

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/AINA-WY.dta", replace

//^variable

clear

//////Computing Direct Estimates for 2+ (TP) Subgroup//////

//Target values are for all states except for AL, AZ, AR, CT, DE, ID, IL,

// LA, ME, MA, MI, MS, MO, NH, NJ, NM, NY, ND, SC, SD, TN, UT, VT, WV, WI, WY//

/////State label reference///

//Alabama(1) Alaska(2) Arizona(4) Arkansas(5) California(6) Colorado(8)

//Connecticut(9) Delaware(10) Florida(12) Georgia(13) Hawaii(15) Idaho(16)

//Illinois(17) Indiana(18) Iowa(19) Kansas(20) Kentucky(21) Louisiana(22)

//Maine(23) Maryland(24) Massachusetts(25) Michigan(26) Minnesota(27)

//Mississippi(28) Missouri(29) Montana(30) Nebraska(31) Nevada(32)

//New Hampshire(33) New Jersey(34) New Mexico(35) New York(36) North Carolina(37)

//North Dakota(38) Ohio(39) Oklahoma(40) Oregon(41) Pennsylvania(42)

//Rhode Island(44) South Carolina(45) South Dakota(46) Tennessee(47)

//Texas(48) Utah (49) Vermont(50) Virginia(51) Washington(53)

//West Virginia(54) Wisconsin(55) Wyoming(56)

/////////////////////Alaska(2)//////////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 2

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-AK.dta", replace

//^variable

clear

/////////////////////California(6)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 6

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-CA.dta", replace

//^variable

clear

/////////////////////Colorado(8)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 8

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-CO.dta", replace

//^variable

clear

/////////////////////Florida(12)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 12

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-FL.dta", replace

//^variable

clear

/////////////////////Georgia(13)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 13

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-GA.dta", replace

//^variable

clear

/////////////////////Hawaii(15)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 15

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-HI.dta", replace

//^variable

clear

/////////////////////Indiana(18)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 18

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-IN.dta", replace

//^variable

clear

/////////////////////Iowa(19)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 19

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-IA.dta", replace

//^variable

clear

/////////////////////Kansas(20)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 20

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-KS.dta", replace

//^variable

clear

/////////////////////Kentucky(21)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 21

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-KY.dta", replace

//^variable

clear

/////////////////////Maryland(24)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 24

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-MD.dta", replace

//^variable

clear

/////////////////////Minnesota(27)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 27

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-MN.dta", replace

//^variable

clear

/////////////////////Montana(30)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 30

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-MT.dta", replace

//^variable

clear

/////////////////////Nebraska(31)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 31

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-NE.dta", replace

//^variable

clear

/////////////////////Nevada(32)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 32

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-NV.dta", replace

//^variable

clear

/////////////////////North Carolina(37)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 37

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-NC.dta", replace

//^variable

clear

/////////////////////Ohio(39)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 39

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-OH.dta", replace

//^variable

clear

/////////////////////Oklahoma(40)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 40

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-OK.dta", replace

//^variable

clear

/////////////////////Oregon(41)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 41

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-OR.dta", replace

//^variable

clear

/////////////////////Pennsylvania(42)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 42

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-PA.dta", replace

//^variable

clear

/////////////////////Rhode Island(44)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 44

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-RI.dta", replace

//^variable

clear

/////////////////////Texas(48)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 48

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-TX.dta", replace

//^variable

clear

/////////////////////Virginia(51)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 51

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-VA.dta", replace

//^variable

clear

/////////////////////Washington(53)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & SDRACEM == 6 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 53

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/TP-WA.dta", replace

//^variable

clear

//////Computing Direct Estimates for EL Subgroup//////

//Target values are for all states except for AL, DE, ID, KY, LA,

// ME, MS, MO, MT, NE, NH, NJ, ND, OR, SD, TN, VT, WV, WY

/////State label reference///

//Alabama(1) Alaska(2) Arizona(4) Arkansas(5) California(6) Colorado(8)

//Connecticut(9) Delaware(10) Florida(12) Georgia(13) Hawaii(15) Idaho(16)

//Illinois(17) Indiana(18) Iowa(19) Kansas(20) Kentucky(21) Louisiana(22)

//Maine(23) Maryland(24) Massachusetts(25) Michigan(26) Minnesota(27)

//Mississippi(28) Missouri(29) Montana(30) Nebraska(31) Nevada(32)

//New Hampshire(33) New Jersey(34) New Mexico(35) New York(36) North Carolina(37)

//North Dakota(38) Ohio(39) Oklahoma(40) Oregon(41) Pennsylvania(42)

//Rhode Island(44) South Carolina(45) South Dakota(46) Tennessee(47)

//Texas(48) Utah (49) Vermont(50) Virginia(51) Washington(53)

//West Virginia(54) Wisconsin(55) Wyoming(56)

/////////////////////Alaska(2)//////////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 2

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-AK.dta", replace

//^variable

clear

/////////////////////Arizona(4)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 4

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-AZ.dta", replace

//^variable

clear

/////////////////////Arkansas(5)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 5

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-AR.dta", replace

//^variable

clear

/////////////////////California(6)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 6

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-CA.dta", replace

//^variable

clear

/////////////////////Colorado(8)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 8

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-CO.dta", replace

//^variable

clear

/////////////////////Connecticut(9)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 9

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-CT.dta", replace

//^variable

clear

/////////////////////Florida(12)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 12

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-FL.dta", replace

//^variable

clear

/////////////////////Georgia(13)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 13

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-GA.dta", replace

//^variable

clear

/////////////////////Hawaii(15)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 15

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-HI.dta", replace

//^variable

clear

/////////////////////Illinois(17)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 17

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-IL.dta", replace

//^variable

clear

/////////////////////Indiana(18)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 18

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-IN.dta", replace

//^variable

clear

/////////////////////Iowa(19)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 19

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-IA.dta", replace

//^variable

clear

/////////////////////Kansas(20)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 20

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-KS.dta", replace

//^variable

clear

/////////////////////Maryland(24)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 24

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-MD.dta", replace

//^variable

clear

/////////////////////Massachusetts(25)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 25

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-MA.dta", replace

//^variable

clear

/////////////////////Michigan(26)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 26

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-MI.dta", replace

//^variable

clear

/////////////////////Minnesota(27)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 27

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-MN.dta", replace

//^variable

clear

/////////////////////Nevada(32)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 32

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-NV.dta", replace

//^variable

clear

/////////////////////New Mexico(35)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 35

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-NM.dta", replace

//^variable

clear

/////////////////////New York(36)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 36

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-NY.dta", replace

//^variable

clear

/////////////////////North Carolina(37)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 37

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-NC.dta", replace

//^variable

clear

/////////////////////Ohio(39)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 39

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-OH.dta", replace

//^variable

clear

/////////////////////Oklahoma(40)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 40

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-OK.dta", replace

//^variable

clear

/////////////////////Pennsylvania(42)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 42

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-PA.dta", replace

//^variable

clear

/////////////////////Rhode Island(44)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 44

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-RI.dta", replace

//^variable

clear

/////////////////////South Carolina(45)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 45

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-SC.dta", replace

//^variable

clear

/////////////////////Texas(48)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 48

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-TX.dta", replace

//^variable

clear

/////////////////////Utah(49)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 49

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-UT.dta", replace

//^variable

clear

/////////////////////Virginia(51)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 51

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-VA.dta", replace

//^variable

clear

/////////////////////Washington(53)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 53

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-WA.dta", replace

//^variable

clear

/////////////////////Wisconsin(55)/////////////////////

use "[filepath]/Data-in-stata.dta"

keep if PUBPRIV == 1 & LEP == 1 // < variable for subgroup

//take a random sample of [censored].

set seed 2019

sort FIPS15

by FIPS15: sample [censored], count

svyset [pweight = ORIGWT] , jkrweight(SRWT\*) vce(jackknife) mse

postfile buffer mean\_ach stderr using "[filepath]/results.dta", replace

forvalues i=1(1)20 {

svy: mean MRPCM`i' if FIPS15 == 55

mat results = r(table)

local mean\_ach = results[1,1]

local stderr = results[2,1]

post buffer (`mean\_ach') (`stderr')

}

postclose buffer

clear

use "[filepath]/results.dta"

gen mean\_var = stderr^2

drop stderr

save "[filepath]/EL-WI.dta", replace

//^variable

clear

**Importing Direct Estimate Information from STATA into R and Pooling Parameter Estimates (Chapter 4)**

This file takes the "[subgroup-state]".dta (stata) data files, which each contain 20 mean estimates and 20 mean variance estimates (one for each set of plausible values from the restricted-use data), and then pools these mean and mean variance estimates according to Rubin's Rule's (1987) to form pooled mean and mean variance estimates. The text that follows in courier font represents R script. Lines of script that begin with a pound/hashtag symbol (i.e., #) are comments (not code). Note, the name of local filepaths (e.g., “C:/Foldername/Foldername/…/Filename”) are replaced with [filepath] throughout.

**Program/File Type**: R/.R File

**Dissertation Section:** Description of FH estimates by subgroup

library(haven)

#####NHS#####

#write function that pools variance and than takes the sqrt (the se)

pooled\_se <- function(x){

within\_var <- mean(x)

between\_var <- var(x)

sampling\_var <- between\_var/20

sqrt(sum(within\_var, between\_var, sampling\_var))

}

#write function that reads .dta file with mean plausible values and

# associated mean variance estimates, and returns a mean and se.

mean\_and\_se <- function(x) {

mean\_de <- apply(x[,1], 2, mean)

se\_de <- apply(x[,2], 2, pooled\_se)

print(c("x", mean\_de, se\_de))

}

#Import Data for subgroup

NHS\_AL <- read\_dta("[filepath]/NHS-AL.dta")

NHS\_AR <- read\_dta("[filepath]/NHS-AR.dta")

NHS\_AZ <- read\_dta("[filepath]/NHS-AZ.dta")

NHS\_CA <- read\_dta("[filepath]/NHS-CA.dta")

NHS\_CO <- read\_dta("[filepath]/NHS-CO.dta")

NHS\_CT <- read\_dta("[filepath]/NHS-CT.dta")

NHS\_DE <- read\_dta("[filepath]/NHS-DE.dta")

NHS\_FL <- read\_dta("[filepath]/NHS-FL.dta")

NHS\_GA <- read\_dta("[filepath]/NHS-GA.dta")

NHS\_HI <- read\_dta("[filepath]/NHS-HI.dta")

NHS\_IA <- read\_dta("[filepath]/NHS-IA.dta")

NHS\_ID <- read\_dta("[filepath]/NHS-ID.dta")

NHS\_IL <- read\_dta("[filepath]/NHS-IL.dta")

NHS\_IN <- read\_dta("[filepath]/NHS-IN.dta")

NHS\_KS <- read\_dta("[filepath]/NHS-KS.dta")

NHS\_KY <- read\_dta("[filepath]/NHS-KY.dta")

NHS\_LA <- read\_dta("[filepath]/NHS-LA.dta")

NHS\_MA <- read\_dta("[filepath]/NHS-MA.dta")

NHS\_MD <- read\_dta("[filepath]/NHS-MD.dta")

NHS\_ME <- read\_dta("[filepath]/NHS-ME.dta")

NHS\_MI <- read\_dta("[filepath]/NHS-MI.dta")

NHS\_MN <- read\_dta("[filepath]/NHS-MN.dta")

NHS\_MO <- read\_dta("[filepath]/NHS-MO.dta")

NHS\_MS <- read\_dta("[filepath]/NHS-MS.dta")

NHS\_MT <- read\_dta("[filepath]/NHS-MT.dta")

NHS\_NC <- read\_dta("[filepath]/NHS-NC.dta")

NHS\_ND <- read\_dta("[filepath]/NHS-ND.dta")

NHS\_NE <- read\_dta("[filepath]/NHS-NE.dta")

NHS\_NH <- read\_dta("[filepath]/NHS-NH.dta")

NHS\_NJ <- read\_dta("[filepath]/NHS-NJ.dta")

NHS\_NM <- read\_dta("[filepath]/NHS-NM.dta")

NHS\_NV <- read\_dta("[filepath]/NHS-NV.dta")

NHS\_NY <- read\_dta("[filepath]/NHS-NY.dta")

NHS\_OH <- read\_dta("[filepath]/NHS-OH.dta")

NHS\_OK <- read\_dta("[filepath]/NHS-OK.dta")

NHS\_OR <- read\_dta("[filepath]/NHS-OR.dta")

NHS\_PA <- read\_dta("[filepath]/NHS-PA.dta")

NHS\_RI <- read\_dta("[filepath]/NHS-RI.dta")

NHS\_SC <- read\_dta("[filepath]/NHS-SC.dta")

NHS\_SD <- read\_dta("[filepath]/NHS-SD.dta")

NHS\_TN <- read\_dta("[filepath]/NHS-TN.dta")

NHS\_TX <- read\_dta("[filepath]/NHS-TX.dta")

NHS\_VA <- read\_dta("[filepath]/NHS-VA.dta")

NHS\_VT <- read\_dta("[filepath]/NHS-VT.dta")

NHS\_WA <- read\_dta("[filepath]/NHS-WA.dta")

NHS\_WI <- read\_dta("[filepath]/NHS-WI.dta")

NHS\_WV <- read\_dta("[filepath]/NHS-WV.dta")

NHS\_WY <- read\_dta("[filepath]/NHS-WY.dta")

#apply mean\_and\_se() function to the imported data sets

AL <- mean\_and\_se(NHS\_AL)

AR <- mean\_and\_se(NHS\_AR)

AZ <- mean\_and\_se(NHS\_AZ)

CA <- mean\_and\_se(NHS\_CA)

CO <- mean\_and\_se(NHS\_CO)

CT <- mean\_and\_se(NHS\_CT)

DE <- mean\_and\_se(NHS\_DE)

FL <- mean\_and\_se(NHS\_FL)

GA <- mean\_and\_se(NHS\_GA)

HI <- mean\_and\_se(NHS\_HI)

IA <- mean\_and\_se(NHS\_IA)

ID <- mean\_and\_se(NHS\_ID)

IL <- mean\_and\_se(NHS\_IL)

IN <- mean\_and\_se(NHS\_IN)

KS <- mean\_and\_se(NHS\_KS)

KY <- mean\_and\_se(NHS\_KY)

LA <- mean\_and\_se(NHS\_LA)

MA <- mean\_and\_se(NHS\_MA)

MD <- mean\_and\_se(NHS\_MD)

ME <- mean\_and\_se(NHS\_ME)

MI <- mean\_and\_se(NHS\_MI)

MN <- mean\_and\_se(NHS\_MN)

MO <- mean\_and\_se(NHS\_MO)

MS <- mean\_and\_se(NHS\_MS)

MT <- mean\_and\_se(NHS\_MT)

NC <- mean\_and\_se(NHS\_NC)

ND <- mean\_and\_se(NHS\_ND)

NE <- mean\_and\_se(NHS\_NE)

NH <- mean\_and\_se(NHS\_NH)

NJ <- mean\_and\_se(NHS\_NJ)

NM <- mean\_and\_se(NHS\_NM)

NV <- mean\_and\_se(NHS\_NV)

NY <- mean\_and\_se(NHS\_NY)

OH <- mean\_and\_se(NHS\_OH)

OK <- mean\_and\_se(NHS\_OK)

OR <- mean\_and\_se(NHS\_OR)

PA <- mean\_and\_se(NHS\_PA)

RI <- mean\_and\_se(NHS\_RI)

SC <- mean\_and\_se(NHS\_SC)

SD <- mean\_and\_se(NHS\_SD)

TN <- mean\_and\_se(NHS\_TN)

TX <- mean\_and\_se(NHS\_TX)

VA <- mean\_and\_se(NHS\_VA)

VT <- mean\_and\_se(NHS\_VT)

WA <- mean\_and\_se(NHS\_WA)

WI <- mean\_and\_se(NHS\_WI)

WV <- mean\_and\_se(NHS\_WV)

WY <- mean\_and\_se(NHS\_WY)

#Row bind results

NHS\_tv\_df\_de\_results <- rbind(AL, AR, AZ, CA, CO, CT, DE, FL, GA, HI,

IA, ID, IL, IN, KS, KY, LA, MA, MD, ME,

MI, MN, MO, MS, MT, NC, ND, NE, NH, NJ,

NM, NV, NY, OH, OK, OR, PA, RI, SC, SD,

TN, TX, VA, VT, WA, WI, WV, WY)

#Define object as a data frame

NHS\_tv\_df\_de\_results <- as.data.frame(NHS\_tv\_df\_de\_results)

#Change variable names

names(NHS\_tv\_df\_de\_results) <- c("State", "NHS\_direct\_est", "NHS\_se")

#Write state abbreviations into the "State" variable

NHS\_tv\_df\_de\_results$State <- c("AL", "AR", "AZ", "CA", "CO", "CT", "DE",

"FL", "GA", "HI", "IA", "ID", "IL", "IN",

"KS", "KY", "LA", "MA", "MD", "ME", "MI",

"MN", "MO", "MS", "MT", "NC", "ND", "NE",

"NH", "NJ", "NM", "NV", "NY", "OH", "OK",

"OR", "PA", "RI", "SC", "SD", "TN", "TX",

"VA", "VT", "WA", "WI", "WV", "WY")

#Export (save) data set

write.csv(NHS\_tv\_df\_de\_results,

"[filepath]/NHS\_tv\_df\_de\_results.csv")

#Free up memory

rm(list = ls())

#####HS#####

#write function that pools variance and than takes the sqrt (the se)

pooled\_se <- function(x){

within\_var <- mean(x)

between\_var <- var(x)

sampling\_var <- between\_var/20

sqrt(sum(within\_var, between\_var, sampling\_var))

}

#write function that reads .dta file with mean plausible values and

# associated mean variance estimates, and returns a mean and se.

mean\_and\_se <- function(x) {

mean\_de <- apply(x[,1], 2, mean)

se\_de <- apply(x[,2], 2, pooled\_se)

print(c("x", mean\_de, se\_de))

}

#Import Data for subgroup

HS\_AL <- read\_dta("[filepath]/HS-AL.dta")

HS\_AR <- read\_dta("[filepath]/HS-AR.dta")

HS\_AZ <- read\_dta("[filepath]/HS-AZ.dta")

HS\_CA <- read\_dta("[filepath]/HS-CA.dta")

HS\_CO <- read\_dta("[filepath]/HS-CO.dta")

HS\_CT <- read\_dta("[filepath]/HS-CT.dta")

HS\_DE <- read\_dta("[filepath]/HS-DE.dta")

HS\_FL <- read\_dta("[filepath]/HS-FL.dta")

HS\_GA <- read\_dta("[filepath]/HS-GA.dta")

HS\_HI <- read\_dta("[filepath]/HS-HI.dta")

HS\_IA <- read\_dta("[filepath]/HS-IA.dta")

HS\_ID <- read\_dta("[filepath]/HS-ID.dta")

HS\_IL <- read\_dta("[filepath]/HS-IL.dta")

HS\_IN <- read\_dta("[filepath]/HS-IN.dta")

HS\_KS <- read\_dta("[filepath]/HS-KS.dta")

HS\_KY <- read\_dta("[filepath]/HS-KY.dta")

HS\_LA <- read\_dta("[filepath]/HS-LA.dta")

HS\_MA <- read\_dta("[filepath]/HS-MA.dta")

HS\_MD <- read\_dta("[filepath]/HS-MD.dta")

HS\_ME <- read\_dta("[filepath]/HS-ME.dta")

HS\_MI <- read\_dta("[filepath]/HS-MI.dta")

HS\_MN <- read\_dta("[filepath]/HS-MN.dta")

HS\_MO <- read\_dta("[filepath]/HS-MO.dta")

HS\_MS <- read\_dta("[filepath]/HS-MS.dta")

HS\_MT <- read\_dta("[filepath]/HS-MT.dta")

HS\_NC <- read\_dta("[filepath]/HS-NC.dta")

HS\_ND <- read\_dta("[filepath]/HS-ND.dta")

HS\_NE <- read\_dta("[filepath]/HS-NE.dta")

HS\_NH <- read\_dta("[filepath]/HS-NH.dta")

HS\_NJ <- read\_dta("[filepath]/HS-NJ.dta")

HS\_NM <- read\_dta("[filepath]/HS-NM.dta")

HS\_NV <- read\_dta("[filepath]/HS-NV.dta")

HS\_NY <- read\_dta("[filepath]/HS-NY.dta")

HS\_OH <- read\_dta("[filepath]/HS-OH.dta")

HS\_OK <- read\_dta("[filepath]/HS-OK.dta")

HS\_OR <- read\_dta("[filepath]/HS-OR.dta")

HS\_PA <- read\_dta("[filepath]/HS-PA.dta")

HS\_RI <- read\_dta("[filepath]/HS-RI.dta")

HS\_SC <- read\_dta("[filepath]/HS-SC.dta")

HS\_SD <- read\_dta("[filepath]/HS-SD.dta")

HS\_TN <- read\_dta("[filepath]/HS-TN.dta")

HS\_TX <- read\_dta("[filepath]/HS-TX.dta")

HS\_VA <- read\_dta("[filepath]/HS-VA.dta")

HS\_VT <- read\_dta("[filepath]/HS-VT.dta")

HS\_WA <- read\_dta("[filepath]/HS-WA.dta")

HS\_WI <- read\_dta("[filepath]/HS-WI.dta")

HS\_WV <- read\_dta("[filepath]/HS-WV.dta")

HS\_WY <- read\_dta("[filepath]/HS-WY.dta")

#apply mean\_and\_se() function to the imported data sets

AL <- mean\_and\_se(HS\_AL)

AR <- mean\_and\_se(HS\_AR)

AZ <- mean\_and\_se(HS\_AZ)

CA <- mean\_and\_se(HS\_CA)

CO <- mean\_and\_se(HS\_CO)

CT <- mean\_and\_se(HS\_CT)

DE <- mean\_and\_se(HS\_DE)

FL <- mean\_and\_se(HS\_FL)

GA <- mean\_and\_se(HS\_GA)

HI <- mean\_and\_se(HS\_HI)

IA <- mean\_and\_se(HS\_IA)

ID <- mean\_and\_se(HS\_ID)

IL <- mean\_and\_se(HS\_IL)

IN <- mean\_and\_se(HS\_IN)

KS <- mean\_and\_se(HS\_KS)

KY <- mean\_and\_se(HS\_KY)

LA <- mean\_and\_se(HS\_LA)

MA <- mean\_and\_se(HS\_MA)

MD <- mean\_and\_se(HS\_MD)

ME <- mean\_and\_se(HS\_ME)

MI <- mean\_and\_se(HS\_MI)

MN <- mean\_and\_se(HS\_MN)

MO <- mean\_and\_se(HS\_MO)

MS <- mean\_and\_se(HS\_MS)

MT <- mean\_and\_se(HS\_MT)

NC <- mean\_and\_se(HS\_NC)

ND <- mean\_and\_se(HS\_ND)

NE <- mean\_and\_se(HS\_NE)

NH <- mean\_and\_se(HS\_NH)

NJ <- mean\_and\_se(HS\_NJ)

NM <- mean\_and\_se(HS\_NM)

NV <- mean\_and\_se(HS\_NV)

NY <- mean\_and\_se(HS\_NY)

OH <- mean\_and\_se(HS\_OH)

OK <- mean\_and\_se(HS\_OK)

OR <- mean\_and\_se(HS\_OR)

PA <- mean\_and\_se(HS\_PA)

RI <- mean\_and\_se(HS\_RI)

SC <- mean\_and\_se(HS\_SC)

SD <- mean\_and\_se(HS\_SD)

TN <- mean\_and\_se(HS\_TN)

TX <- mean\_and\_se(HS\_TX)

VA <- mean\_and\_se(HS\_VA)

VT <- mean\_and\_se(HS\_VT)

WA <- mean\_and\_se(HS\_WA)

WI <- mean\_and\_se(HS\_WI)

WV <- mean\_and\_se(HS\_WV)

WY <- mean\_and\_se(HS\_WY)

#Row bind results

HS\_tv\_df\_de\_results <- rbind(AL, AR, AZ, CA, CO, CT, DE, FL, GA, HI,

IA, ID, IL, IN, KS, KY, LA, MA, MD, ME,

MI, MN, MO, MS, MT, NC, ND, NE, NH, NJ,

NM, NV, NY, OH, OK, OR, PA, RI, SC, SD,

TN, TX, VA, VT, WA, WI, WV, WY)

#Define object as a data frame

HS\_tv\_df\_de\_results <- as.data.frame(HS\_tv\_df\_de\_results)

#Change variable names

names(HS\_tv\_df\_de\_results) <- c("State", "HS\_direct\_est", "HS\_se")

#will need to change variable names

#Write state abbreviations into the "State" variable

HS\_tv\_df\_de\_results$State <- c("AL", "AR", "AZ", "CA", "CO", "CT", "DE",

"FL", "GA", "HI", "IA", "ID", "IL", "IN",

"KS", "KY", "LA", "MA", "MD", "ME", "MI",

"MN", "MO", "MS", "MT", "NC", "ND", "NE",

"NH", "NJ", "NM", "NV", "NY", "OH", "OK",

"OR", "PA", "RI", "SC", "SD", "TN", "TX",

"VA", "VT", "WA", "WI", "WV", "WY")

#Export (save) data set

write.csv(HS\_tv\_df\_de\_results,

"[filepath]/HS\_tv\_df\_de\_results.csv")

#Free up memory

rm(list = ls())

#####SBA#####

#write function that pools variance and than takes the sqrt (the se)

pooled\_se <- function(x){

within\_var <- mean(x)

between\_var <- var(x)

sampling\_var <- between\_var/20

sqrt(sum(within\_var, between\_var, sampling\_var))

}

#write function that reads .dta file with mean plausible values and

# associated mean variance estimates, and returns a mean and se.

mean\_and\_se <- function(x) {

mean\_de <- apply(x[,1], 2, mean)

se\_de <- apply(x[,2], 2, pooled\_se)

print(c("x", mean\_de, se\_de))

}

#Import Data for subgroup

SBA\_AL <- read\_dta("[filepath]/SBA-AL.dta")

SBA\_AR <- read\_dta("[filepath]/SBA-AR.dta")

SBA\_AZ <- read\_dta("[filepath]/SBA-AZ.dta")

SBA\_CA <- read\_dta("[filepath]/SBA-CA.dta")

SBA\_CO <- read\_dta("[filepath]/SBA-CO.dta")

SBA\_CT <- read\_dta("[filepath]/SBA-CT.dta")

SBA\_DE <- read\_dta("[filepath]/SBA-DE.dta")

SBA\_FL <- read\_dta("[filepath]/SBA-FL.dta")

SBA\_GA <- read\_dta("[filepath]/SBA-GA.dta")

SBA\_HI <- read\_dta("[filepath]/SBA-HI.dta")

SBA\_IA <- read\_dta("[filepath]/SBA-IA.dta")

SBA\_ID <- read\_dta("[filepath]/SBA-ID.dta")

SBA\_IL <- read\_dta("[filepath]/SBA-IL.dta")

SBA\_IN <- read\_dta("[filepath]/SBA-IN.dta")

SBA\_KS <- read\_dta("[filepath]/SBA-KS.dta")

SBA\_KY <- read\_dta("[filepath]/SBA-KY.dta")

SBA\_LA <- read\_dta("[filepath]/SBA-LA.dta")

SBA\_MA <- read\_dta("[filepath]/SBA-MA.dta")

SBA\_MD <- read\_dta("[filepath]/SBA-MD.dta")

SBA\_ME <- read\_dta("[filepath]/SBA-ME.dta")

SBA\_MI <- read\_dta("[filepath]/SBA-MI.dta")

SBA\_MN <- read\_dta("[filepath]/SBA-MN.dta")

SBA\_MO <- read\_dta("[filepath]/SBA-MO.dta")

SBA\_MS <- read\_dta("[filepath]/SBA-MS.dta")

SBA\_MT <- read\_dta("[filepath]/SBA-MT.dta")

SBA\_NC <- read\_dta("[filepath]/SBA-NC.dta")

SBA\_ND <- read\_dta("[filepath]/SBA-ND.dta")

SBA\_NE <- read\_dta("[filepath]/SBA-NE.dta")

SBA\_NH <- read\_dta("[filepath]/SBA-NH.dta")

SBA\_NJ <- read\_dta("[filepath]/SBA-NJ.dta")

SBA\_NM <- read\_dta("[filepath]/SBA-NM.dta")

SBA\_NV <- read\_dta("[filepath]/SBA-NV.dta")

SBA\_NY <- read\_dta("[filepath]/SBA-NY.dta")

SBA\_OH <- read\_dta("[filepath]/SBA-OH.dta")

SBA\_OK <- read\_dta("[filepath]/SBA-OK.dta")

SBA\_OR <- read\_dta("[filepath]/SBA-OR.dta")

SBA\_PA <- read\_dta("[filepath]/SBA-PA.dta")

SBA\_RI <- read\_dta("[filepath]/SBA-RI.dta")

SBA\_SC <- read\_dta("[filepath]/SBA-SC.dta")

SBA\_SD <- read\_dta("[filepath]/SBA-SD.dta")

SBA\_TN <- read\_dta("[filepath]/SBA-TN.dta")

SBA\_TX <- read\_dta("[filepath]/SBA-TX.dta")

SBA\_VA <- read\_dta("[filepath]/SBA-VA.dta")

SBA\_VT <- read\_dta("[filepath]/SBA-VT.dta")

SBA\_WA <- read\_dta("[filepath]/SBA-WA.dta")

SBA\_WI <- read\_dta("[filepath]/SBA-WI.dta")

SBA\_WV <- read\_dta("[filepath]/SBA-WV.dta")

SBA\_WY <- read\_dta("[filepath]/SBA-WY.dta")

#apply mean\_and\_se() function to the imported data sets

AL <- mean\_and\_se(SBA\_AL)

AR <- mean\_and\_se(SBA\_AR)

AZ <- mean\_and\_se(SBA\_AZ)

CA <- mean\_and\_se(SBA\_CA)

CO <- mean\_and\_se(SBA\_CO)

CT <- mean\_and\_se(SBA\_CT)

DE <- mean\_and\_se(SBA\_DE)

FL <- mean\_and\_se(SBA\_FL)

GA <- mean\_and\_se(SBA\_GA)

HI <- mean\_and\_se(SBA\_HI)

IA <- mean\_and\_se(SBA\_IA)

ID <- mean\_and\_se(SBA\_ID)

IL <- mean\_and\_se(SBA\_IL)

IN <- mean\_and\_se(SBA\_IN)

KS <- mean\_and\_se(SBA\_KS)

KY <- mean\_and\_se(SBA\_KY)

LA <- mean\_and\_se(SBA\_LA)

MA <- mean\_and\_se(SBA\_MA)

MD <- mean\_and\_se(SBA\_MD)

ME <- mean\_and\_se(SBA\_ME)

MI <- mean\_and\_se(SBA\_MI)

MN <- mean\_and\_se(SBA\_MN)

MO <- mean\_and\_se(SBA\_MO)

MS <- mean\_and\_se(SBA\_MS)

MT <- mean\_and\_se(SBA\_MT)

NC <- mean\_and\_se(SBA\_NC)

ND <- mean\_and\_se(SBA\_ND)

NE <- mean\_and\_se(SBA\_NE)

NH <- mean\_and\_se(SBA\_NH)

NJ <- mean\_and\_se(SBA\_NJ)

NM <- mean\_and\_se(SBA\_NM)

NV <- mean\_and\_se(SBA\_NV)

NY <- mean\_and\_se(SBA\_NY)

OH <- mean\_and\_se(SBA\_OH)

OK <- mean\_and\_se(SBA\_OK)

OR <- mean\_and\_se(SBA\_OR)

PA <- mean\_and\_se(SBA\_PA)

RI <- mean\_and\_se(SBA\_RI)

SC <- mean\_and\_se(SBA\_SC)

SD <- mean\_and\_se(SBA\_SD)

TN <- mean\_and\_se(SBA\_TN)

TX <- mean\_and\_se(SBA\_TX)

VA <- mean\_and\_se(SBA\_VA)

VT <- mean\_and\_se(SBA\_VT)

WA <- mean\_and\_se(SBA\_WA)

WI <- mean\_and\_se(SBA\_WI)

WV <- mean\_and\_se(SBA\_WV)

WY <- mean\_and\_se(SBA\_WY)

#Row bind results

SBA\_tv\_df\_de\_results <- rbind(AL, AR, AZ, CA, CO, CT, DE, FL, GA, HI,

IA, ID, IL, IN, KS, KY, LA, MA, MD, ME,

MI, MN, MO, MS, MT, NC, ND, NE, NH, NJ,

NM, NV, NY, OH, OK, OR, PA, RI, SC, SD,

TN, TX, VA, VT, WA, WI, WV, WY)

#Define object as a data frame

SBA\_tv\_df\_de\_results <- as.data.frame(SBA\_tv\_df\_de\_results)

#Change variable names

names(SBA\_tv\_df\_de\_results) <- c("State", "SBA\_direct\_est", "SBA\_se")

#will need to change variable names

#Write state abbreviations into the "State" variable

SBA\_tv\_df\_de\_results$State <- c("AL", "AR", "AZ", "CA", "CO", "CT", "DE",

"FL", "GA", "HI", "IA", "ID", "IL", "IN",

"KS", "KY", "LA", "MA", "MD", "ME", "MI",

"MN", "MO", "MS", "MT", "NC", "ND", "NE",

"NH", "NJ", "NM", "NV", "NY", "OH", "OK",

"OR", "PA", "RI", "SC", "SD", "TN", "TX",

"VA", "VT", "WA", "WI", "WV", "WY")

#Export (save) data set

write.csv(SBA\_tv\_df\_de\_results,

"[filepath]/SBA\_tv\_df\_de\_results.csv")

#Free up memory

rm(list = ls())

#####BA#####

#write function that pools variance and than takes the sqrt (the se)

pooled\_se <- function(x){

within\_var <- mean(x)

between\_var <- var(x)

sampling\_var <- between\_var/20

sqrt(sum(within\_var, between\_var, sampling\_var))

}

#write function that reads .dta file with mean plausible values and

# associated mean variance estimates, and returns a mean and se.

mean\_and\_se <- function(x) {

mean\_de <- apply(x[,1], 2, mean)

se\_de <- apply(x[,2], 2, pooled\_se)

print(c("x", mean\_de, se\_de))

}

#Import Data for subgroup

BA\_AL <- read\_dta("[filepath]/BA-AL.dta")

BA\_AR <- read\_dta("[filepath]/BA-AR.dta")

BA\_AZ <- read\_dta("[filepath]/BA-AZ.dta")

BA\_CA <- read\_dta("[filepath]/BA-CA.dta")

BA\_CO <- read\_dta("[filepath]/BA-CO.dta")

BA\_CT <- read\_dta("[filepath]/BA-CT.dta")

BA\_DE <- read\_dta("[filepath]/BA-DE.dta")

BA\_FL <- read\_dta("[filepath]/BA-FL.dta")

BA\_GA <- read\_dta("[filepath]/BA-GA.dta")

BA\_HI <- read\_dta("[filepath]/BA-HI.dta")

BA\_IA <- read\_dta("[filepath]/BA-IA.dta")

BA\_ID <- read\_dta("[filepath]/BA-ID.dta")

BA\_IL <- read\_dta("[filepath]/BA-IL.dta")

BA\_IN <- read\_dta("[filepath]/BA-IN.dta")

BA\_KS <- read\_dta("[filepath]/BA-KS.dta")

BA\_KY <- read\_dta("[filepath]/BA-KY.dta")

BA\_LA <- read\_dta("[filepath]/BA-LA.dta")

BA\_MA <- read\_dta("[filepath]/BA-MA.dta")

BA\_MD <- read\_dta("[filepath]/BA-MD.dta")

BA\_ME <- read\_dta("[filepath]/BA-ME.dta")

BA\_MI <- read\_dta("[filepath]/BA-MI.dta")

BA\_MN <- read\_dta("[filepath]/BA-MN.dta")

BA\_MO <- read\_dta("[filepath]/BA-MO.dta")

BA\_MS <- read\_dta("[filepath]/BA-MS.dta")

BA\_MT <- read\_dta("[filepath]/BA-MT.dta")

BA\_NC <- read\_dta("[filepath]/BA-NC.dta")

BA\_ND <- read\_dta("[filepath]/BA-ND.dta")

BA\_NE <- read\_dta("[filepath]/BA-NE.dta")

BA\_NH <- read\_dta("[filepath]/BA-NH.dta")

BA\_NJ <- read\_dta("[filepath]/BA-NJ.dta")

BA\_NM <- read\_dta("[filepath]/BA-NM.dta")

BA\_NV <- read\_dta("[filepath]/BA-NV.dta")

BA\_NY <- read\_dta("[filepath]/BA-NY.dta")

BA\_OH <- read\_dta("[filepath]/BA-OH.dta")

BA\_OK <- read\_dta("[filepath]/BA-OK.dta")

BA\_OR <- read\_dta("[filepath]/BA-OR.dta")

BA\_PA <- read\_dta("[filepath]/BA-PA.dta")

BA\_RI <- read\_dta("[filepath]/BA-RI.dta")

BA\_SC <- read\_dta("[filepath]/BA-SC.dta")

BA\_SD <- read\_dta("[filepath]/BA-SD.dta")

BA\_TN <- read\_dta("[filepath]/BA-TN.dta")

BA\_TX <- read\_dta("[filepath]/BA-TX.dta")

BA\_VA <- read\_dta("[filepath]/BA-VA.dta")

BA\_VT <- read\_dta("[filepath]/BA-VT.dta")

BA\_WA <- read\_dta("[filepath]/BA-WA.dta")

BA\_WI <- read\_dta("[filepath]/BA-WI.dta")

BA\_WV <- read\_dta("[filepath]/BA-WV.dta")

BA\_WY <- read\_dta("[filepath]/BA-WY.dta")

#apply mean\_and\_se() function to the imported data sets

AL <- mean\_and\_se(BA\_AL)

AR <- mean\_and\_se(BA\_AR)

AZ <- mean\_and\_se(BA\_AZ)

CA <- mean\_and\_se(BA\_CA)

CO <- mean\_and\_se(BA\_CO)

CT <- mean\_and\_se(BA\_CT)

DE <- mean\_and\_se(BA\_DE)

FL <- mean\_and\_se(BA\_FL)

GA <- mean\_and\_se(BA\_GA)

HI <- mean\_and\_se(BA\_HI)

IA <- mean\_and\_se(BA\_IA)

ID <- mean\_and\_se(BA\_ID)

IL <- mean\_and\_se(BA\_IL)

IN <- mean\_and\_se(BA\_IN)

KS <- mean\_and\_se(BA\_KS)

KY <- mean\_and\_se(BA\_KY)

LA <- mean\_and\_se(BA\_LA)

MA <- mean\_and\_se(BA\_MA)

MD <- mean\_and\_se(BA\_MD)

ME <- mean\_and\_se(BA\_ME)

MI <- mean\_and\_se(BA\_MI)

MN <- mean\_and\_se(BA\_MN)

MO <- mean\_and\_se(BA\_MO)

MS <- mean\_and\_se(BA\_MS)

MT <- mean\_and\_se(BA\_MT)

NC <- mean\_and\_se(BA\_NC)

ND <- mean\_and\_se(BA\_ND)

NE <- mean\_and\_se(BA\_NE)

NH <- mean\_and\_se(BA\_NH)

NJ <- mean\_and\_se(BA\_NJ)

NM <- mean\_and\_se(BA\_NM)

NV <- mean\_and\_se(BA\_NV)

NY <- mean\_and\_se(BA\_NY)

OH <- mean\_and\_se(BA\_OH)

OK <- mean\_and\_se(BA\_OK)

OR <- mean\_and\_se(BA\_OR)

PA <- mean\_and\_se(BA\_PA)

RI <- mean\_and\_se(BA\_RI)

SC <- mean\_and\_se(BA\_SC)

SD <- mean\_and\_se(BA\_SD)

TN <- mean\_and\_se(BA\_TN)

TX <- mean\_and\_se(BA\_TX)

VA <- mean\_and\_se(BA\_VA)

VT <- mean\_and\_se(BA\_VT)

WA <- mean\_and\_se(BA\_WA)

WI <- mean\_and\_se(BA\_WI)

WV <- mean\_and\_se(BA\_WV)

WY <- mean\_and\_se(BA\_WY)

#Row bind results

BA\_tv\_df\_de\_results <- rbind(AL, AR, AZ, CA, CO, CT, DE, FL, GA, HI,

IA, ID, IL, IN, KS, KY, LA, MA, MD, ME,

MI, MN, MO, MS, MT, NC, ND, NE, NH, NJ,

NM, NV, NY, OH, OK, OR, PA, RI, SC, SD,

TN, TX, VA, VT, WA, WI, WV, WY)

#Define object as a data frame

BA\_tv\_df\_de\_results <- as.data.frame(BA\_tv\_df\_de\_results)

#Change variable names

names(BA\_tv\_df\_de\_results) <- c("State", "BA\_direct\_est", "BA\_se")

#will need to change variable names

#Write state abbreviations into the "State" variable

BA\_tv\_df\_de\_results$State <- c("AL", "AR", "AZ", "CA", "CO", "CT", "DE",

"FL", "GA", "HI", "IA", "ID", "IL", "IN",

"KS", "KY", "LA", "MA", "MD", "ME", "MI",

"MN", "MO", "MS", "MT", "NC", "ND", "NE",

"NH", "NJ", "NM", "NV", "NY", "OH", "OK",

"OR", "PA", "RI", "SC", "SD", "TN", "TX",

"VA", "VT", "WA", "WI", "WV", "WY")

#Export (save) data set

write.csv(BA\_tv\_df\_de\_results,

"[filepath]/BA\_tv\_df\_de\_results.csv")

#Free up memory

rm(list = ls())

#####B#####

#write function that pools variance and than takes the sqrt (the se)

pooled\_se <- function(x){

within\_var <- mean(x)

between\_var <- var(x)

sampling\_var <- between\_var/20

sqrt(sum(within\_var, between\_var, sampling\_var))

}

#write function that reads .dta file with mean plausible values and

# associated mean variance estimates, and returns a mean and se.

mean\_and\_se <- function(x) {

mean\_de <- apply(x[,1], 2, mean)

se\_de <- apply(x[,2], 2, pooled\_se)

print(c("x", mean\_de, se\_de))

}

#Import Data for subgroup

B\_AL <- read\_dta("[filepath]/B-AL.dta")

B\_AK <- read\_dta("[filepath]/B-AK.dta")

B\_AR <- read\_dta("[filepath]/B-AR.dta")

B\_AZ <- read\_dta("[filepath]/B-AZ.dta")

B\_CA <- read\_dta("[filepath]/B-CA.dta")

B\_CO <- read\_dta("[filepath]/B-CO.dta")

B\_CT <- read\_dta("[filepath]/B-CT.dta")

B\_DE <- read\_dta("[filepath]/B-DE.dta")

B\_FL <- read\_dta("[filepath]/B-FL.dta")

B\_GA <- read\_dta("[filepath]/B-GA.dta")

B\_IA <- read\_dta("[filepath]/B-IA.dta")

B\_IL <- read\_dta("[filepath]/B-IL.dta")

B\_IN <- read\_dta("[filepath]/B-IN.dta")

B\_KS <- read\_dta("[filepath]/B-KS.dta")

B\_KY <- read\_dta("[filepath]/B-KY.dta")

B\_LA <- read\_dta("[filepath]/B-LA.dta")

B\_MA <- read\_dta("[filepath]/B-MA.dta")

B\_MD <- read\_dta("[filepath]/B-MD.dta")

B\_MI <- read\_dta("[filepath]/B-MI.dta")

B\_MN <- read\_dta("[filepath]/B-MN.dta")

B\_MO <- read\_dta("[filepath]/B-MO.dta")

B\_MS <- read\_dta("[filepath]/B-MS.dta")

B\_NC <- read\_dta("[filepath]/B-NC.dta")

B\_ND <- read\_dta("[filepath]/B-ND.dta")

B\_NE <- read\_dta("[filepath]/B-NE.dta")

B\_NJ <- read\_dta("[filepath]/B-NJ.dta")

B\_NV <- read\_dta("[filepath]/B-NV.dta")

B\_NY <- read\_dta("[filepath]/B-NY.dta")

B\_OH <- read\_dta("[filepath]/B-OH.dta")

B\_OK <- read\_dta("[filepath]/B-OK.dta")

B\_PA <- read\_dta("[filepath]/B-PA.dta")

B\_RI <- read\_dta("[filepath]/B-RI.dta")

B\_SC <- read\_dta("[filepath]/B-SC.dta")

B\_TN <- read\_dta("[filepath]/B-TN.dta")

B\_TX <- read\_dta("[filepath]/B-TX.dta")

B\_VA <- read\_dta("[filepath]/B-VA.dta")

B\_WA <- read\_dta("[filepath]/B-WA.dta")

B\_WI <- read\_dta("[filepath]/B-WI.dta")

B\_WV <- read\_dta("[filepath]/B-WV.dta")

#apply mean\_and\_se() function to the imported data sets

AL <- mean\_and\_se(B\_AL)

AK <- mean\_and\_se(B\_AK)

AR <- mean\_and\_se(B\_AR)

AZ <- mean\_and\_se(B\_AZ)

CA <- mean\_and\_se(B\_CA)

CO <- mean\_and\_se(B\_CO)

CT <- mean\_and\_se(B\_CT)

DE <- mean\_and\_se(B\_DE)

FL <- mean\_and\_se(B\_FL)

GA <- mean\_and\_se(B\_GA)

IA <- mean\_and\_se(B\_IA)

IL <- mean\_and\_se(B\_IL)

IN <- mean\_and\_se(B\_IN)

KS <- mean\_and\_se(B\_KS)

KY <- mean\_and\_se(B\_KY)

LA <- mean\_and\_se(B\_LA)

MA <- mean\_and\_se(B\_MA)

MD <- mean\_and\_se(B\_MD)

MI <- mean\_and\_se(B\_MI)

MN <- mean\_and\_se(B\_MN)

MO <- mean\_and\_se(B\_MO)

MS <- mean\_and\_se(B\_MS)

NC <- mean\_and\_se(B\_NC)

ND <- mean\_and\_se(B\_ND)

NE <- mean\_and\_se(B\_NE)

NJ <- mean\_and\_se(B\_NJ)

NV <- mean\_and\_se(B\_NV)

NY <- mean\_and\_se(B\_NY)

OH <- mean\_and\_se(B\_OH)

OK <- mean\_and\_se(B\_OK)

PA <- mean\_and\_se(B\_PA)

RI <- mean\_and\_se(B\_RI)

SC <- mean\_and\_se(B\_SC)

TN <- mean\_and\_se(B\_TN)

TX <- mean\_and\_se(B\_TX)

VA <- mean\_and\_se(B\_VA)

WA <- mean\_and\_se(B\_WA)

WI <- mean\_and\_se(B\_WI)

WV <- mean\_and\_se(B\_WV)

#Row bind results

B\_tv\_df\_de\_results <- rbind(AL, AK, AR, AZ, CA, CO, CT, DE, FL, GA,

IA, IL, IN, KS, KY, LA, MA, MD,

MI, MN, MO, MS, NC, ND, NE,

NJ, NV, NY, OH, OK, PA, RI, SC,

TN, TX, VA, WA, WI, WV)

#Define object as a data frame

B\_tv\_df\_de\_results <- as.data.frame(B\_tv\_df\_de\_results)

#Change variable names

names(B\_tv\_df\_de\_results) <- c("State", "B\_direct\_est", "B\_se")

#Write state abbreviations into the "State" variable

B\_tv\_df\_de\_results$State <- c("AL", "AK", "AR", "AZ", "CA", "CO", "CT",

"DE", "FL", "GA", "IA", "IL",

"IN", "KS", "KY", "LA", "MA", "MD",

"MI", "MN", "MO", "MS", "NC", "ND",

"NE", "NJ", "NV", "NY", "OH",

"OK", "PA", "RI", "SC", "TN",

"TX", "VA", "WA", "WI", "WV")

#Export (save) data set

write.csv(B\_tv\_df\_de\_results,

"[filepath]/B\_tv\_df\_de\_results.csv")

#Free up memory

rm(list = ls())

#####H#####

#write function that pools variance and than takes the sqrt (the se)

pooled\_se <- function(x){

within\_var <- mean(x)

between\_var <- var(x)

sampling\_var <- between\_var/20

sqrt(sum(within\_var, between\_var, sampling\_var))

}

#write function that reads .dta file with mean plausible values and

# associated mean variance estimates, and returns a mean and se.

mean\_and\_se <- function(x) {

mean\_de <- apply(x[,1], 2, mean)

se\_de <- apply(x[,2], 2, pooled\_se)

print(c("x", mean\_de, se\_de))

}

#Import Data for subgroup

H\_AL <- read\_dta("[filepath]/H-AL.dta")

H\_AK <- read\_dta("[filepath]/H-AK.dta")

H\_AR <- read\_dta("[filepath]/H-AR.dta")

H\_AZ <- read\_dta("[filepath]/H-AZ.dta")

H\_CA <- read\_dta("[filepath]/H-CA.dta")

H\_CO <- read\_dta("[filepath]/H-CO.dta")

H\_CT <- read\_dta("[filepath]/H-CT.dta")

H\_DE <- read\_dta("[filepath]/H-DE.dta")

H\_FL <- read\_dta("[filepath]/H-FL.dta")

H\_GA <- read\_dta("[filepath]/H-GA.dta")

H\_HI <- read\_dta("[filepath]/H-HI.dta")

H\_IA <- read\_dta("[filepath]/H-IA.dta")

H\_ID <- read\_dta("[filepath]/H-ID.dta")

H\_IL <- read\_dta("[filepath]/H-IL.dta")

H\_IN <- read\_dta("[filepath]/H-IN.dta")

H\_KS <- read\_dta("[filepath]/H-KS.dta")

H\_KY <- read\_dta("[filepath]/H-KY.dta")

H\_LA <- read\_dta("[filepath]/H-LA.dta")

H\_MA <- read\_dta("[filepath]/H-MA.dta")

H\_MD <- read\_dta("[filepath]/H-MD.dta")

H\_MI <- read\_dta("[filepath]/H-MI.dta")

H\_MN <- read\_dta("[filepath]/H-MN.dta")

H\_MO <- read\_dta("[filepath]/H-MO.dta")

H\_MS <- read\_dta("[filepath]/H-MS.dta")

H\_MT <- read\_dta("[filepath]/H-MT.dta")

H\_NC <- read\_dta("[filepath]/H-NC.dta")

H\_ND <- read\_dta("[filepath]/H-ND.dta")

H\_NE <- read\_dta("[filepath]/H-NE.dta")

H\_NH <- read\_dta("[filepath]/H-NH.dta")

H\_NJ <- read\_dta("[filepath]/H-NJ.dta")

H\_NM <- read\_dta("[filepath]/H-NM.dta")

H\_NV <- read\_dta("[filepath]/H-NV.dta")

H\_NY <- read\_dta("[filepath]/H-NY.dta")

H\_OH <- read\_dta("[filepath]/H-OH.dta")

H\_OK <- read\_dta("[filepath]/H-OK.dta")

H\_OR <- read\_dta("[filepath]/H-OR.dta")

H\_PA <- read\_dta("[filepath]/H-PA.dta")

H\_RI <- read\_dta("[filepath]/H-RI.dta")

H\_SC <- read\_dta("[filepath]/H-SC.dta")

H\_SD <- read\_dta("[filepath]/H-SD.dta")

H\_TN <- read\_dta("[filepath]/H-TN.dta")

H\_TX <- read\_dta("[filepath]/H-TX.dta")

H\_UT <- read\_dta("[filepath]/H-UT.dta")

H\_VA <- read\_dta("[filepath]/H-VA.dta")

H\_WA <- read\_dta("[filepath]/H-WA.dta")

H\_WI <- read\_dta("[filepath]/H-WI.dta")

H\_WY <- read\_dta("[filepath]/H-WY.dta")

#apply mean\_and\_se() function to the imported data sets

AL <- mean\_and\_se(H\_AL)

AK <- mean\_and\_se(H\_AK)

AR <- mean\_and\_se(H\_AR)

AZ <- mean\_and\_se(H\_AZ)

CA <- mean\_and\_se(H\_CA)

CO <- mean\_and\_se(H\_CO)

CT <- mean\_and\_se(H\_CT)

DE <- mean\_and\_se(H\_DE)

FL <- mean\_and\_se(H\_FL)

GA <- mean\_and\_se(H\_GA)

HI <- mean\_and\_se(H\_HI)

IA <- mean\_and\_se(H\_IA)

ID <- mean\_and\_se(H\_ID)

IL <- mean\_and\_se(H\_IL)

IN <- mean\_and\_se(H\_IN)

KS <- mean\_and\_se(H\_KS)

KY <- mean\_and\_se(H\_KY)

LA <- mean\_and\_se(H\_LA)

MA <- mean\_and\_se(H\_MA)

MD <- mean\_and\_se(H\_MD)

MI <- mean\_and\_se(H\_MI)

MN <- mean\_and\_se(H\_MN)

MO <- mean\_and\_se(H\_MO)

MS <- mean\_and\_se(H\_MS)

MT <- mean\_and\_se(H\_MT)

NC <- mean\_and\_se(H\_NC)

ND <- mean\_and\_se(H\_ND)

NE <- mean\_and\_se(H\_NE)

NH <- mean\_and\_se(H\_NH)

NJ <- mean\_and\_se(H\_NJ)

NM <- mean\_and\_se(H\_NM)

NV <- mean\_and\_se(H\_NV)

NY <- mean\_and\_se(H\_NY)

OH <- mean\_and\_se(H\_OH)

OK <- mean\_and\_se(H\_OK)

OR <- mean\_and\_se(H\_OR)

PA <- mean\_and\_se(H\_PA)

RI <- mean\_and\_se(H\_RI)

SC <- mean\_and\_se(H\_SC)

SD <- mean\_and\_se(H\_SD)

TN <- mean\_and\_se(H\_TN)

TX <- mean\_and\_se(H\_TX)

UT <- mean\_and\_se(H\_UT)

VA <- mean\_and\_se(H\_VA)

WA <- mean\_and\_se(H\_WA)

WI <- mean\_and\_se(H\_WI)

WY <- mean\_and\_se(H\_WY)

#Row bind results

H\_tv\_df\_de\_results <- rbind(AL, AK, AR, AZ, CA, CO, CT, DE, FL, GA,

HI, IA, ID, IL, IN, KS, KY, LA, MA, MD,

MI, MN, MO, MS, MT, NC, ND, NE, NH,

NJ, NM, NV, NY, OH, OK, OR, PA, RI, SC,

SD, TN, TX, UT, VA, WA, WI, WY)

#Define object as a data frame

H\_tv\_df\_de\_results <- as.data.frame(H\_tv\_df\_de\_results)

#Change variable names

names(H\_tv\_df\_de\_results) <- c("State", "H\_direct\_est", "H\_se")

#will need to change variable names

#Write state abbreviations into the "State" variable

H\_tv\_df\_de\_results$State <- c("AL", "AK", "AR", "AZ", "CA", "CO", "CT",

"DE", "FL", "GA", "HI", "IA", "ID", "IL",

"IN", "KS", "KY", "LA", "MA", "MD",

"MI", "MN", "MO", "MS", "MT", "NC", "ND",

"NE", "NH", "NJ", "NM", "NV", "NY", "OH",

"OK", "OR", "PA", "RI", "SC", "SD", "TN",

"TX", "UT", "VA", "WA", "WI",

"WY")

#Export (save) data set

write.csv(H\_tv\_df\_de\_results,

"[filepath]/H\_tv\_df\_de\_results.csv")

#Free up memory

rm(list = ls())

#####API#####

#write function that pools variance and than takes the sqrt (the se)

pooled\_se <- function(x){

within\_var <- mean(x)

between\_var <- var(x)

sampling\_var <- between\_var/20

sqrt(sum(within\_var, between\_var, sampling\_var))

}

#write function that reads .dta file with mean plausible values and

# associated mean variance estimates, and returns a mean and se.

mean\_and\_se <- function(x) {

mean\_de <- apply(x[,1], 2, mean)

se\_de <- apply(x[,2], 2, pooled\_se)

print(c("x", mean\_de, se\_de))

}

#Import Data for subgroup

API\_AK <- read\_dta("[filepath]/API-AK.dta")

API\_AZ <- read\_dta("[filepath]/API-AZ.dta")

API\_CA <- read\_dta("[filepath]/API-CA.dta")

API\_CO <- read\_dta("[filepath]/API-CO.dta")

API\_CT <- read\_dta("[filepath]/API-CT.dta")

API\_DE <- read\_dta("[filepath]/API-DE.dta")

API\_FL <- read\_dta("[filepath]/API-FL.dta")

API\_GA <- read\_dta("[filepath]/API-GA.dta")

API\_HI <- read\_dta("[filepath]/API-HI.dta")

API\_IA <- read\_dta("[filepath]/API-IA.dta")

API\_IL <- read\_dta("[filepath]/API-IL.dta")

API\_KS <- read\_dta("[filepath]/API-KS.dta")

API\_KY <- read\_dta("[filepath]/API-KY.dta")

API\_MA <- read\_dta("[filepath]/API-MA.dta")

API\_MD <- read\_dta("[filepath]/API-MD.dta")

API\_MI <- read\_dta("[filepath]/API-MI.dta")

API\_MN <- read\_dta("[filepath]/API-MN.dta")

API\_NC <- read\_dta("[filepath]/API-NC.dta")

API\_NH <- read\_dta("[filepath]/API-NH.dta")

API\_NJ <- read\_dta("[filepath]/API-NJ.dta")

API\_NV <- read\_dta("[filepath]/API-NV.dta")

API\_NY <- read\_dta("[filepath]/API-NY.dta")

API\_OH <- read\_dta("[filepath]/API-OH.dta")

API\_OR <- read\_dta("[filepath]/API-OR.dta")

API\_PA <- read\_dta("[filepath]/API-PA.dta")

API\_RI <- read\_dta("[filepath]/API-RI.dta")

API\_TX <- read\_dta("[filepath]/API-TX.dta")

API\_VA <- read\_dta("[filepath]/API-VA.dta")

API\_WA <- read\_dta("[filepath]/API-WA.dta")

API\_WI <- read\_dta("[filepath]/API-WI.dta")

#apply mean\_and\_se() function to the imported data sets

AK <- mean\_and\_se(API\_AK)

AZ <- mean\_and\_se(API\_AZ)

CA <- mean\_and\_se(API\_CA)

CO <- mean\_and\_se(API\_CO)

CT <- mean\_and\_se(API\_CT)

DE <- mean\_and\_se(API\_DE)

FL <- mean\_and\_se(API\_FL)

GA <- mean\_and\_se(API\_GA)

HI <- mean\_and\_se(API\_HI)

IA <- mean\_and\_se(API\_IA)

IL <- mean\_and\_se(API\_IL)

KS <- mean\_and\_se(API\_KS)

KY <- mean\_and\_se(API\_KY)

MA <- mean\_and\_se(API\_MA)

MD <- mean\_and\_se(API\_MD)

MI <- mean\_and\_se(API\_MI)

MN <- mean\_and\_se(API\_MN)

NC <- mean\_and\_se(API\_NC)

NH <- mean\_and\_se(API\_NH)

NJ <- mean\_and\_se(API\_NJ)

NV <- mean\_and\_se(API\_NV)

NY <- mean\_and\_se(API\_NY)

OH <- mean\_and\_se(API\_OH)

OR <- mean\_and\_se(API\_OR)

PA <- mean\_and\_se(API\_PA)

RI <- mean\_and\_se(API\_RI)

TX <- mean\_and\_se(API\_TX)

VA <- mean\_and\_se(API\_VA)

WA <- mean\_and\_se(API\_WA)

WI <- mean\_and\_se(API\_WI)

WV <- mean\_and\_se(API\_WV)

#Row bind results

API\_tv\_df\_de\_results <- rbind(AK, AZ, CA, CO, CT, DE, FL, GA,

HI, IA, IL, KS, KY, MA, MD,

MI, MN, NC, NH,

NJ, NV, NY, OH, OR, PA, RI,

TX, VA, WA, WI)

#Define object as a data frame

API\_tv\_df\_de\_results <- as.data.frame(API\_tv\_df\_de\_results)

#Change variable names

names(API\_tv\_df\_de\_results) <- c("State", "API\_direct\_est", "API\_se")

#will need to change variable names

#Write state abbreviations into the "State" variable

API\_tv\_df\_de\_results$State <- c("AK", "AZ", "CA", "CO", "CT", "DE",

"FL", "GA", "HI", "IA", "IL", "KS",

"KY", "MA", "MD", "MI", "MN", "NC",

"NH", "NJ", "NV", "NY", "OH", "OR",

"PA", "RI", "TX", "VA", "WA", "WI")

#Export (save) data set

write.csv(API\_tv\_df\_de\_results,

"[filepath]/API\_tv\_df\_de\_results.csv")

#Free up memory

rm(list = ls())

#####AINA#####

#write function that pools variance and than takes the sqrt (the se)

pooled\_se <- function(x){

within\_var <- mean(x)

between\_var <- var(x)

sampling\_var <- between\_var/20

sqrt(sum(within\_var, between\_var, sampling\_var))

}

#write function that reads .dta file with mean plausible values and

# associated mean variance estimates, and returns a mean and se.

mean\_and\_se <- function(x) {

mean\_de <- apply(x[,1], 2, mean)

se\_de <- apply(x[,2], 2, pooled\_se)

print(c("x", mean\_de, se\_de))

}

#Import Data for subgroup

AINA\_AK <- read\_dta("[filepath]/AINA-AK.dta")

AINA\_AZ <- read\_dta("[filepath]/AINA-AZ.dta")

AINA\_MN <- read\_dta("[filepath]/AINA-MN.dta")

AINA\_MT <- read\_dta("[filepath]/AINA-MT.dta")

AINA\_NC <- read\_dta("[filepath]/AINA-NC.dta")

AINA\_ND <- read\_dta("[filepath]/AINA-ND.dta")

AINA\_NM <- read\_dta("[filepath]/AINA-NM.dta")

AINA\_OK <- read\_dta("[filepath]/AINA-OK.dta")

AINA\_SD <- read\_dta("[filepath]/AINA-SD.dta")

AINA\_UT <- read\_dta("[filepath]/AINA-UT.dta")

AINA\_WA <- read\_dta("[filepath]/AINA-WA.dta")

AINA\_WI <- read\_dta("[filepath]/AINA-WI.dta")

AINA\_WY <- read\_dta("[filepath]/AINA-WY.dta")

#apply mean\_and\_se() function to the imported data sets

AK <- mean\_and\_se(AINA\_AK)

AZ <- mean\_and\_se(AINA\_AZ)

MN <- mean\_and\_se(AINA\_MN)

MT <- mean\_and\_se(AINA\_MT)

NC <- mean\_and\_se(AINA\_NC)

ND <- mean\_and\_se(AINA\_ND)

NM <- mean\_and\_se(AINA\_NM)

OK <- mean\_and\_se(AINA\_OK)

SD <- mean\_and\_se(AINA\_SD)

UT <- mean\_and\_se(AINA\_UT)

WA <- mean\_and\_se(AINA\_WA)

WI <- mean\_and\_se(AINA\_WI)

WY <- mean\_and\_se(AINA\_WY)

#Row bind results

AINA\_tv\_df\_de\_results <- rbind(AK, AZ, MN, MT, NC, ND, NM, OK, SD, UT,

WA, WI, WY)

#Define object as a data frame

AINA\_tv\_df\_de\_results <- as.data.frame(AINA\_tv\_df\_de\_results)

#Change variable names

names(AINA\_tv\_df\_de\_results) <- c("State", "AINA\_direct\_est", "AINA\_se")

#will need to change variable names

#Write state abbreviations into the "State" variable

AINA\_tv\_df\_de\_results$State <- c("AK", "AZ", "MN", "MT", "NC", "ND", "NM",

"OK", "SD", "UT", "WA", "WI", "WY")

#Export (save) data set

write.csv(AINA\_tv\_df\_de\_results,

"[filepath]/AINA\_tv\_df\_de\_results.csv")

#Free up memory

rm(list = ls())

#####2+/TP#####

#write function that pools variance and than takes the sqrt (the se)

pooled\_se <- function(x){

within\_var <- mean(x)

between\_var <- var(x)

sampling\_var <- between\_var/20

sqrt(sum(within\_var, between\_var, sampling\_var))

}

#write function that reads .dta file with mean plausible values and

# associated mean variance estimates, and returns a mean and se.

mean\_and\_se <- function(x) {

mean\_de <- apply(x[,1], 2, mean)

se\_de <- apply(x[,2], 2, pooled\_se)

print(c("x", mean\_de, se\_de))

}

#Import Data for subgroup

TP\_AK <- read\_dta("[filepath]/TP-AK.dta")

TP\_CA <- read\_dta("[filepath]/TP-CA.dta")

TP\_CO <- read\_dta("[filepath]/TP-CO.dta")

TP\_FL <- read\_dta("[filepath]/TP-FL.dta")

TP\_GA <- read\_dta("[filepath]/TP-GA.dta")

TP\_HI <- read\_dta("[filepath]/TP-HI.dta")

TP\_IA <- read\_dta("[filepath]/TP-IA.dta")

TP\_IN <- read\_dta("[filepath]/TP-IN.dta")

TP\_KS <- read\_dta("[filepath]/TP-KS.dta")

TP\_KY <- read\_dta("[filepath]/TP-KY.dta")

TP\_MD <- read\_dta("[filepath]/TP-MD.dta")

TP\_MN <- read\_dta("[filepath]/TP-MN.dta")

TP\_MT <- read\_dta("[filepath]/TP-MT.dta")

TP\_NC <- read\_dta("[filepath]/TP-NC.dta")

TP\_NE <- read\_dta("[filepath]/TP-NE.dta")

TP\_NV <- read\_dta("[filepath]/TP-NV.dta")

TP\_OH <- read\_dta("[filepath]/TP-OH.dta")

TP\_OK <- read\_dta("[filepath]/TP-OK.dta")

TP\_OR <- read\_dta("[filepath]/TP-OR.dta")

TP\_PA <- read\_dta("[filepath]/TP-PA.dta")

TP\_RI <- read\_dta("[filepath]/TP-RI.dta")

TP\_TX <- read\_dta("[filepath]/TP-TX.dta")

TP\_VA <- read\_dta("[filepath]/TP-VA.dta")

TP\_WA <- read\_dta("[filepath]/TP-WA.dta")

#apply mean\_and\_se() function to the imported data sets

AK <- mean\_and\_se(TP\_AK)

CA <- mean\_and\_se(TP\_CA)

CO <- mean\_and\_se(TP\_CO)

FL <- mean\_and\_se(TP\_FL)

GA <- mean\_and\_se(TP\_GA)

HI <- mean\_and\_se(TP\_HI)

IA <- mean\_and\_se(TP\_IA)

IN <- mean\_and\_se(TP\_IN)

KS <- mean\_and\_se(TP\_KS)

KY <- mean\_and\_se(TP\_KY)

MD <- mean\_and\_se(TP\_MD)

MN <- mean\_and\_se(TP\_MN)

MT <- mean\_and\_se(TP\_MT)

NC <- mean\_and\_se(TP\_NC)

NE <- mean\_and\_se(TP\_NE)

NV <- mean\_and\_se(TP\_NV)

OH <- mean\_and\_se(TP\_OH)

OK <- mean\_and\_se(TP\_OK)

OR <- mean\_and\_se(TP\_OR)

PA <- mean\_and\_se(TP\_PA)

RI <- mean\_and\_se(TP\_RI)

TX <- mean\_and\_se(TP\_TX)

VA <- mean\_and\_se(TP\_VA)

WA <- mean\_and\_se(TP\_WA)

#Row bind results

TP\_tv\_df\_de\_results <- rbind(AK, CA, CO, FL, GA,

HI, IA, IN, KS, KY, MD,

MN, MT, NC, NE,

NV, OH, OK, OR, PA, RI, TX,

VA, WA)

#Define object as a data frame

TP\_tv\_df\_de\_results <- as.data.frame(TP\_tv\_df\_de\_results)

#Change variable names

names(TP\_tv\_df\_de\_results) <- c("State", "TP\_direct\_est", "TP\_se")

#will need to change variable names

#Write state abbreviations into the "State" variable

TP\_tv\_df\_de\_results$State <- c("AK", "CA", "CO", "FL", "GA",

"HI", "IA", "IN", "KS", "KY", "MD",

"MN", "MT", "NC", "NE",

"NV", "OH", "OK", "OR", "PA", "RI",

"TX", "VA", "WA")

#Export (save) data set

write.csv(TP\_tv\_df\_de\_results,

"[filepath]/TP\_tv\_df\_de\_results.csv")

#Free up memory

rm(list = ls())

#####EL#####

#write function that pools variance and than takes the sqrt (the se)

pooled\_se <- function(x){

within\_var <- mean(x)

between\_var <- var(x)

sampling\_var <- between\_var/20

sqrt(sum(within\_var, between\_var, sampling\_var))

}

#write function that reads .dta file with mean plausible values and

# associated mean variance estimates, and returns a mean and se.

mean\_and\_se <- function(x) {

mean\_de <- apply(x[,1], 2, mean)

se\_de <- apply(x[,2], 2, pooled\_se)

print(c("x", mean\_de, se\_de))

}

#Import Data for subgroup

EL\_AK <- read\_dta("[filepath]/EL-AK.dta")

EL\_AR <- read\_dta("[filepath]/EL-AR.dta")

EL\_AZ <- read\_dta("[filepath]/EL-AZ.dta")

EL\_CA <- read\_dta("[filepath]/EL-CA.dta")

EL\_CO <- read\_dta("[filepath]/EL-CO.dta")

EL\_CT <- read\_dta("[filepath]/EL-CT.dta")

EL\_FL <- read\_dta("[filepath]/EL-FL.dta")

EL\_GA <- read\_dta("[filepath]/EL-GA.dta")

EL\_HI <- read\_dta("[filepath]/EL-HI.dta")

EL\_IA <- read\_dta("[filepath]/EL-IA.dta")

EL\_IL <- read\_dta("[filepath]/EL-IL.dta")

EL\_IN <- read\_dta("[filepath]/EL-IN.dta")

EL\_KS <- read\_dta("[filepath]/EL-KS.dta")

EL\_MA <- read\_dta("[filepath]/EL-MA.dta")

EL\_MD <- read\_dta("[filepath]/EL-MD.dta")

EL\_MI <- read\_dta("[filepath]/EL-MI.dta")

EL\_MN <- read\_dta("[filepath]/EL-MN.dta")

EL\_NC <- read\_dta("[filepath]/EL-NC.dta")

EL\_NM <- read\_dta("[filepath]/EL-NM.dta")

EL\_NV <- read\_dta("[filepath]/EL-NV.dta")

EL\_NY <- read\_dta("[filepath]/EL-NY.dta")

EL\_OH <- read\_dta("[filepath]/EL-OH.dta")

EL\_OK <- read\_dta("[filepath]/EL-OK.dta")

EL\_PA <- read\_dta("[filepath]/EL-PA.dta")

EL\_RI <- read\_dta("[filepath]/EL-RI.dta")

EL\_SC <- read\_dta("[filepath]/EL-SC.dta")

EL\_TX <- read\_dta("[filepath]/EL-TX.dta")

EL\_UT <- read\_dta("[filepath]/EL-UT.dta")

EL\_VA <- read\_dta("[filepath]/EL-VA.dta")

EL\_WA <- read\_dta("[filepath]/EL-WA.dta")

EL\_WI <- read\_dta("[filepath]/EL-WI.dta")

#apply mean\_and\_se() function to the imported data sets

AK <- mean\_and\_se(EL\_AK)

AR <- mean\_and\_se(EL\_AR)

AZ <- mean\_and\_se(EL\_AZ)

CA <- mean\_and\_se(EL\_CA)

CO <- mean\_and\_se(EL\_CO)

CT <- mean\_and\_se(EL\_CT)

FL <- mean\_and\_se(EL\_FL)

GA <- mean\_and\_se(EL\_GA)

HI <- mean\_and\_se(EL\_HI)

IA <- mean\_and\_se(EL\_IA)

IL <- mean\_and\_se(EL\_IL)

IN <- mean\_and\_se(EL\_IN)

KS <- mean\_and\_se(EL\_KS)

MA <- mean\_and\_se(EL\_MA)

MD <- mean\_and\_se(EL\_MD)

MI <- mean\_and\_se(EL\_MI)

MN <- mean\_and\_se(EL\_MN)

NC <- mean\_and\_se(EL\_NC)

NM <- mean\_and\_se(EL\_NM)

NV <- mean\_and\_se(EL\_NV)

NY <- mean\_and\_se(EL\_NY)

OH <- mean\_and\_se(EL\_OH)

OK <- mean\_and\_se(EL\_OK)

PA <- mean\_and\_se(EL\_PA)

RI <- mean\_and\_se(EL\_RI)

SC <- mean\_and\_se(EL\_SC)

TX <- mean\_and\_se(EL\_TX)

UT <- mean\_and\_se(EL\_UT)

VA <- mean\_and\_se(EL\_VA)

WA <- mean\_and\_se(EL\_WA)

WI <- mean\_and\_se(EL\_WI)

#Row bind results

EL\_tv\_df\_de\_results <- rbind(AK, AR, AZ, CA, CO, CT, FL, GA,

HI, IA, IL, IN, KS, MA, MD,

MI, MN, NC,

NM, NV, NY, OH, OK, PA, RI, SC,

TX, UT, VA, WA, WI)

#Define object as a data frame

EL\_tv\_df\_de\_results <- as.data.frame(EL\_tv\_df\_de\_results)

#Change variable names

names(EL\_tv\_df\_de\_results) <- c("State", "EL\_direct\_est", "EL\_se")

#will need to change variable names

#Write state abbreviations into the "State" variable

EL\_tv\_df\_de\_results$State <- c("AK", "AR", "AZ", "CA", "CO", "CT",

"FL", "GA", "HI", "IA", "IL", "IN",

"KS", "MA", "MD", "MI", "MN", "NC",

"NM", "NV", "NY", "OH", "OK", "PA",

"RI", "SC", "TX", "UT", "VA", "WA",

"WI")

#Export (save) data set

write.csv(EL\_tv\_df\_de\_results,

"[filepath]/EL\_tv\_df\_de\_results.csv")

#Free up memory

rm(list = ls())

###Bind datasets to...

State <- c("AL", "AK", "AZ", "AR", "CA", "CO", "CT",

"DE", "FL", "GA", "HI", "ID", "IL", "IN",

"IA", "KS", "KY", "LA", "ME", "MD", "MA",

"MI", "MN", "MS", "MO", "MT", "NE", "NV",

"NH", "NJ", "NM", "NY", "NC", "ND", "OH",

"OK", "OR", "PA", "RI", "SC", "SD", "TN",

"TX", "UT", "VT", "VA", "WA", "WV", "WI",

"WY")

de\_dfs <- data.frame(State)

NHS\_tv\_df\_de\_results <-

read.csv("[filepath]/NHS\_tv\_df\_de\_results.csv")

NHS\_tv\_df\_de\_results <- NHS\_tv\_df\_de\_results[,-1]

HS\_tv\_df\_de\_results <-

read.csv("[filepath]/HS\_tv\_df\_de\_results.csv")

HS\_tv\_df\_de\_results <- HS\_tv\_df\_de\_results[, -1]

SBA\_tv\_df\_de\_results <-

read.csv("[filepath]/SBA\_tv\_df\_de\_results.csv")

SBA\_tv\_df\_de\_results <- SBA\_tv\_df\_de\_results[,-1]

BA\_tv\_df\_de\_results <-

read.csv("[filepath]/BA\_tv\_df\_de\_results.csv")

BA\_tv\_df\_de\_results <- BA\_tv\_df\_de\_results[,-1]

B\_tv\_df\_de\_results <-

read.csv("[filepath]/B\_tv\_df\_de\_results.csv")

B\_tv\_df\_de\_results <- B\_tv\_df\_de\_results[,-1]

H\_tv\_df\_de\_results <-

read.csv("[filepath]/H\_tv\_df\_de\_results.csv")

H\_tv\_df\_de\_results <- H\_tv\_df\_de\_results[,-1]

API\_tv\_df\_de\_results <-

read.csv("[filepath]/API\_tv\_df\_de\_results.csv")

API\_tv\_df\_de\_results <- API\_tv\_df\_de\_results[,-1]

AINA\_tv\_df\_de\_results <-

read.csv("[filepath]/AINA\_tv\_df\_de\_results.csv")

AINA\_tv\_df\_de\_results <- AINA\_tv\_df\_de\_results[,-1]

TP\_tv\_df\_de\_results <-

read.csv("[filepath]/TP\_tv\_df\_de\_results.csv")

TP\_tv\_df\_de\_results <- TP\_tv\_df\_de\_results[,-1]

EL\_tv\_df\_de\_results <-

read.csv("[filepath]/EL\_tv\_df\_de\_results.csv")

EL\_tv\_df\_de\_results <- EL\_tv\_df\_de\_results[,-1]

library(dplyr)

de\_dfs <- full\_join(de\_dfs, NHS\_tv\_df\_de\_results, by = "State") %>%

full\_join(HS\_tv\_df\_de\_results, by = "State") %>%

full\_join(SBA\_tv\_df\_de\_results, by = "State") %>%

full\_join(BA\_tv\_df\_de\_results, by = "State") %>%

full\_join(B\_tv\_df\_de\_results, by = "State") %>%

full\_join(H\_tv\_df\_de\_results, by = "State") %>%

full\_join(API\_tv\_df\_de\_results, by = "State") %>%

full\_join(AINA\_tv\_df\_de\_results, by = "State") %>%

full\_join(TP\_tv\_df\_de\_results, by = "State") %>%

full\_join(EL\_tv\_df\_de\_results, by = "State")

#Export (save) direct estimates file

write.csv(de\_dfs, "[filepath]/de\_dfs.csv")

**The FH Technique (Chapter 4)**

The text that follows in courier font represents R script. Lines of script that begin with a pound/hashtag symbol (i.e., #) are comments (not code). Note, the name of local filepaths (e.g., “C:/Foldername/Foldername/…/Filename”) are replaced with [filepath] throughout. Three consecive backticks (```) demarcate the start and end of a chunk (i.e., section) of code).

**Program/File Type:** R Studio/.RMD File

**Dissertation Section:** Description of FH estimates by subgroup

### Install/Load packages

```{r, message=FALSE, warning=FALSE}

if(!require(tidyverse)) install.packages("tidyverse",

repos = "http://cran.us.r-project.org")

if(!require(EdSurvey)) install.packages("EdSurvey",

repos = "http://cran.us.r-project.org")

if(!require(sae)) install.packages("sae",

repos = "http://cran.us.r-project.org")

```

### Import & Wrangle Administrative Data

#\*\*%B-H-AINA variable\*\*

#These data are gathered through a publicly-available National Center for Education Statistics (NCES) website which hosts the Common Core of Data (CCD). These particular data were downloaded from http://nces.ed.gov/ccd/elsi/ through the Elementary / Secondary Information System "tableGenerator".

#The downloaded data include the following state-level enrollment data for the 2014-15 academic year and grade 8 students.

#- American Indian/Alaska Native - male

#- American Indian/Alaska Native - female

#- Asian or Asian/Pacific Islander - male

#- Asian or Asian/Pacific Islander - female

#- Hispanic - male

#- Hispanic - female

#- Black - male

#- Black - female

#- White - male

#- White - female

#- Two or more races - male

#- Two or more races - female

```{r, message=FALSE, warning=FALSE}

#Import

demographic\_df <- read\_csv("[filepath]/Demographic Data for FH Approach.csv") %>%

dplyr::slice(6:57)

#Change variable names

var\_names <- demographic\_df[1,]

names(demographic\_df) <- var\_names

#One more slice

demographic\_df <- dplyr::slice(demographic\_df, 2:52)

#Remove DC

demographic\_df <- filter(demographic\_df, `State Name` != "DISTRICT OF COLUMBIA")

#Change class (type) of variable to permit arithmetic

demographic\_df <- apply(demographic\_df[,2:13], 2, as.numeric)

#restore State abbreviation variable

State <- read\_csv("[filepath]/2+.csv")

State <- State[,1]

demographic\_df <- cbind(State, demographic\_df)

rm(State, var\_names)

#Combine male and female variables

demographic\_df$AINA <- demographic\_df[,2] + demographic\_df[,3]

demographic\_df$API <- demographic\_df[,4] + demographic\_df[,5]

demographic\_df$H <- demographic\_df[,6] + demographic\_df[,7]

demographic\_df$B <- demographic\_df[,8] + demographic\_df[,9]

demographic\_df$W <- demographic\_df[,10] + demographic\_df[,11]

demographic\_df$TP <- demographic\_df[,12] + demographic\_df[,13]

demographic\_df <- dplyr::select(demographic\_df, "State", "AINA", "API", "H", "B", "W", "TP")

#create "total" variable

demographic\_df$total <- apply(demographic\_df[,2:7], 1, sum)

#Create B\_H\_AINA variable

demographic\_df$BHAINA\_total <- demographic\_df$AINA + demographic\_df$H + demographic\_df$B

demographic\_df$B\_H\_AINA <- demographic\_df$BHAINA\_total/demographic\_df$total

#Change scale of B\_H\_AINA from proportion to percent

demographic\_df$B\_H\_AINA <- demographic\_df$B\_H\_AINA\*100

#remove redundancies

demographic\_df <- dplyr::select(demographic\_df, "State", "B\_H\_AINA")

```

#\*\*FER variable\*\*

#Composite measure of a state’s median household income and wealth (Family Economic Resources; “FER”)

#The income data come from page 3 of the following document on the US Census website:

https://www.census.gov/content/dam/Census/library/publications/2017/acs/acsbr16-02.pdf

#The wealth data come from page 9 of the following document on the US Census website:

https://www.census.gov/content/dam/Census/library/working-papers/2017/demo/FY2016-129.pdf

```{r, warning=FALSE, message=FALSE}

med\_income <- c(45182, 74165, 52062, 42530, 65087, 64598, 72121, 61882, 49852, 51753, 74451, 48728, 60094, 50896, 55172, 54520, 45541, 46106, 52111, 76596, 71146, 51584, 64188, 40910, 50642, 49924, 55474, 53320, 70813, 73242, 45710, 61311, 48420, 60944, 51610, 49062, 54748, 56207, 58826, 47790, 53746, 47818, 56139,

63794, 57565, 66916, 64764, 42620, 56115, 60570)

#bind

demographic\_df <- cbind(demographic\_df, med\_income)

med\_wealth <- c(83349, 120365, 79785, 78554, 96190, 118180, 147278, 126219, 88938, 78710, 153570, 95389,

102768, 90247, 108512, 96608, 87998, 86574, 115971, 136853, 148838, 87983, 133224, 75772, 91123, 112580,

96347, 63224, 148468, 143831, 88135, 100543, 93956, 103615, 87717, 82256, 93621, 113131, 108967, 93925,

99726, 87508, 78825, 104950, 141716, 119459, 106626, 92262, 111986, 119763)

#bind

demographic\_df <- cbind(demographic\_df, med\_wealth)

#correct names of 3rd and 4th variable

names(demographic\_df) <- c("State", "B\_H\_AINA", "med\_income", "med\_wealth")

#create composite variable (FER)

demographic\_df$FER <- (demographic\_df$med\_income + demographic\_df$med\_wealth)/2

#remove redundancies

demographic\_df <- dplyr::select(demographic\_df, "State", "B\_H\_AINA", "FER")

#cleanup environment

rm(med\_income, med\_wealth)

```

#\*\*%EL variable\*\*

```{r message=FALSE, warning=FALSE}

#These data are from the National Center for Education Statistics (NCES). The data are retrieved from the following website:

https://nces.ed.gov/programs/digest/d19/tables/dt19\_204.20.asp

#Import

EL\_df <- read\_csv("[filepath]/EL-Data-For-FH-Approach.csv")

#Clean

EL\_df <- cbind(EL\_df[,1], EL\_df[,19])

names(EL\_df) <- c("State", "p\_EL")

EL\_df <- EL\_df %>%

slice(5:69) %>%

filter(State != "NA") %>%

slice(-9) %>%

slice(1:50) %>%

dplyr::select(p\_EL)

#bind

demographic\_df <- cbind(demographic\_df, EL\_df)

#cleanup environment

rm(EL\_df)

```

#\*\*SQI variable\*\*

#The variable, SQI, is measured on a continuous scale with scores ranging 0.0 to 100.0, and reflects the average of states’ “Chance for Success” and “School Finance” ratings. The Chance for Success rating is meant to capture lifelong learning opportunities for students—beginning with early childhood, and progressing through K-12 education into adulthood. The School Finance rating is based on school spending patterns as well as how education dollars are distributed across each state (Education Week Research Center, 2015).

#The data are gathered from the following website:

https://www.edweek.org/ew/qc/2015/2015-state-report-cards-map.html?intc=EW-QC15-LFTNAV

#NB: Hawaii is a single-district jurisdiction. As a result, it is not possible to calculate measures of financial equity, which capture the distribution of funding across districts within a state. Thus Hawaii does not receive a \*composite\* grade for school finance. Instead, its "Spending" score takes the place of school finance, which is one of the sub-components of the overall school finance measure in other states.

```{r message=FALSE, warning=FALSE}

chance\_for\_success <- c(71.2, 76.2, 71.2, 71.0, 72.8, 83.4, 87.5, 78.5, 75.5, 74.3, 80.0, 74.2, 80.1, 76.8, 84.1, 82.7, 75.3, 69.9, 80.0, 85.8, 91.9, 76.1, 87.3, 68.5, 77.4, 77.0, 83.7, 65.6, 89.6, 87.6, 67.2, 80.9, 76.2, 87.5, 76.6, 72.0, 74.8, 82.6, 79.7, 73.3, 78.9, 73.1, 73.4, 80.5, 85.7, 84.9, 80.0, 70.3, 82.6, 82.5)

school\_finance <- c(69.7, 81.7, 65.1, 73.1, 67.0, 68.8, 87.1, 83.2, 68.9, 71.2, 81.9, 59.1, 78.7, 73.3, 73.9, 75.5, 73.5, 75.8, 78.7, 86.8, 83.0, 74.6, 75.7, 66.9, 70.7, 71.7, 75.4, 62.8, 78.6, 86.8, 69.0, 88.8, 66.5, 74.3, 76.7, 66.6, 68.9, 82.2, 86.5, 70.8, 66.6, 68.4, 65.5, 61.7, 86.1, 76.9, 71.8, 89.2, 82.0, 89.3)

#bind

predictors\_df <- cbind(demographic\_df, chance\_for\_success, school\_finance)

#correct names of last two variables

names(predictors\_df) <- c("State", "B\_H\_AINA", "FER", "EL", "chance\_for\_success", "school\_finance")

#Create SQI variable

predictors\_df$SQI <- (predictors\_df$chance\_for\_success + predictors\_df$school\_finance)/2

#remove irrelevant variables from df

predictors\_df <- dplyr::select(predictors\_df, State, B\_H\_AINA, FER, EL, SQI)

#cleanup environment

rm(demographic\_df, chance\_for\_success, school\_finance)

```

#\*\*%BA variable\*\*

#This variable is operationalized as the percent of adults (25 and older) by state that have earned a bachelor’s or more advanced degree (%BA).

#Data are collected from the American Community Survey. Specifically, the following website:

#https://data.census.gov/cedsci/table?d=ACS%205-Year%20Estimates%20Data%20Profiles&table=DP02&tid=ACSDP5Y2015.DP02&y=2015&g=0400000US39,32,31,78,34,33,36,35,38,37,72,30,29,28,21,20,23,22,66,69,25,24,27,26,60,18,17,19,54,10,53,56,12,55,13,16,15,50,51,06,09,08,42,45,44,47,46,05,49,04,48,41,40,02\_0100000US&hidePreview=true&t=Educational%20Attainment&tp=true&moe=true

```{r, warning=FALSE, message=FALSE}

#Import

Percent\_BAplus\_df <- read\_csv("[filepath]/Percent\_BAplus\_df.csv")

#Merge in %BA variable

predictors\_df <- inner\_join(predictors\_df, Percent\_BAplus\_df, by = "State")

#cleanup environment

rm(Percent\_BAplus\_df)

```

#\*\*%AA variable\*\*

#Gathered from the American Community Survey (ACS), a program of the US Census. Specifically, from this website--

https://data.census.gov/cedsci/table?t=004%20-%20Black%20or%20African%20American%20alone&tid=ACSSPP1Y2015.S0201&hidePreview=true&g=0400000US02,04,05,06,08,09,10,12,13,15,16,17,18,19,20,21,22,23,24,26,25,27,28,29,31,30,32,33,34,35,36,37,38,39,40,41,42,44,46,01,45,47,48,49,51,50,53,54,55,56&moe=false&tp=true

#The data are filtered for the "Black of African American" population across across states.

#NB: This metric for 2015 (1-year estimate), is not available for 12/50 states. The data are simply missing in the ACS data set.

These states are the following: AK, HI, ID, ME, MT, NH, NM, ND, SD, UT, VT, & WY. These are states in which the prevalence of Black residents is relatively low.

#Instead, "5-year" estimates are used for these states. Estimates based on pooling data for these jurisdictions across multiple years. Specifically, from this website--

https://data.census.gov/cedsci/table?tid=ACSDT5Y2015.B05003B&t=Race%20and%20Ethnicity&vintage=2015&g=0400000US02,15,16,23,30,33,35,38,46,49,50,56&hidePreview=true&tp=false

#NB: This variable represents an estimate of the proportion of the Black population in each state born in the US.

```{r, warning=FALSE, message=FALSE}

#Import data

Black\_df\_from\_Census <- read\_csv("[filepath]/Black\_df\_from\_Census.csv")

#Change from propotion to percent

Black\_df\_from\_Census$AA <- Black\_df\_from\_Census$AA\*100

#Bind AA variable

predictors\_df <- cbind(predictors\_df, Black\_df\_from\_Census$AA)

#correct variable names

names(predictors\_df) <- c("State", "B\_H\_AINA", "FER", "EL", "SQI", "p\_BA", "AA")

#Clean up environment

rm(Black\_df\_from\_Census)

```

#\*\*%MX variable\*\*

#The data that form this variable come from the following two Census websites:

#https://data.census.gov/cedsci/table?q=Hispanic&g=0400000US39,32,31,78,34,33,36,35,38,37,72,30,29,28,21,20,23,22,66,69,25,24,27,26,60,18,17,19,54,10,53,56,12,55,13,16,15,50,51,06,09,08,42,45,44,47,46,05,49,04,48,41,40,02,01\_0100000US&tid=ACSDT1Y2015.B03001&t=Hispanic%20or%20Latino&vintage=2018&hidePreview=true&tp=true

#https://data.census.gov/cedsci/table?q=Hispanic&g=0400000US39,32,31,78,34,33,36,35,38,37,72,30,29,28,21,20,23,22,66,69,25,24,27,26,60,18,17,19,54,10,53,56,12,55,13,16,15,50,51,06,09,08,42,45,44,47,46,05,49,04,48,41,40,02,01\_0100000US&tid=ACSDT5Y2015.B03001&t=Hispanic%20or%20Latino&vintage=2018&hidePreview=true&tp=true

#The former provides one-year estimates and the latter 5-year estimates. There are 7 states (AK, ME, MT, ND, SD, VT, WY) for which 5-year estimates are used. These represent states with relatively small proportions of Hispanics.

```{r, warning=FALSE, message=FALSE}

#Import data

Hispanic\_df\_from\_Census <- read\_csv("[filepath]/Hispanic\_df\_from\_Census.csv")

#Bind AA variable

predictors\_df <- cbind(predictors\_df, Hispanic\_df\_from\_Census$MX)

#cOrrect variable names

names(predictors\_df) <- c("State", "B\_H\_AINA", "FER", "EL", "SQI", "P\_BA", "AA", "MX")

#Clean up environment

rm(Hispanic\_df\_from\_Census)

```

#\*\*%A variable\*\*

#This last predictor variable is the percent of Asian or Pacific Islander grade 8 students by state who identify as Asian but not Pacific Islander (%A) in 2014-15.

#One limitation: PI data not available for NEw YORK in 2014-15 (nor in adjacent years; 2013-14 or 2015-16). The data for NY represent grade 10 students in 2016-17 (so still the same target cohort.)

```{r, warning=FALSE, message=FALSE}

#Import data

API\_df <- read\_csv("[filepath]/API\_df\_from\_CCD.csv") %>%

slice(6:57)

#Remove DC

API\_df <- API\_df[-10,]

#Change class of certain variables

API\_df <- apply(API\_df[,2:5], 2, as.numeric)

#Remove first (left over) line

API\_df <- API\_df[-1,]

#Define API\_df as a data frame

API\_df <- as.data.frame(API\_df)

#Create "API" and "PI" variables

API\_df$API <- API\_df$X2 + API\_df$X3

API\_df$PI <- API\_df$X4 + API\_df$X5

#Create %A variable

API\_df$A <- 1-(API\_df$PI/API\_df$API)

options(digits = 3)

API\_df$A <- API\_df$A\*100

#Bind A variable

predictors\_df <- cbind(predictors\_df, API\_df$A)

#cOrrect variable names

names(predictors\_df) <- c("State", "B\_H\_AINA", "FER", "p\_EL", "SQI", "p\_BA", "AA", "MX", "A")

#Clean up environment

rm(API\_df)

```

#\*\*Tidy up predictors dataset\*\*

```{r}

#Make sure all values are rounded to the nearest tenth (one decimal)

predictors\_df$B\_H\_AINA<- format(round(predictors\_df$B\_H\_AINA), nsmall=1)

predictors\_df$AA <- format(round(predictors\_df$AA), nsmall=1)

predictors\_df$MX <- format(round(predictors\_df$MX), nsmall=1)

#make sure all predictor variables are numeric

predictors\_df[, 2:9] <- apply(predictors\_df[, 2:9], 2, as.numeric)

```

### Combine direct estimate data and state-level predictors

```{r warning=FALSE, message=FALSE}

#import de\_dfs

de\_dfs <- read\_csv("[filepath]/de\_dfs.csv")

de\_dfs <- de\_dfs[,-1]

FH\_df <- full\_join(de\_dfs, predictors\_df)

#make sure merged dataset is 50 rows long

```

### Merge NAEP reported estimates of mean math achievent into the data set

```{r warning=FALSE, message=FALSE}

#Import the NAEP-reported means and standard errors for target values

NR\_TV\_Mean\_SE <- read\_csv("[filepath]/NR\_TV\_Mean\_SE.csv")

#use full join

FH\_df <- full\_join(FH\_df, NR\_TV\_Mean\_SE, by = "State")

#verify that are still only 50 rows

#Export (save) FH\_df

write.csv(FH\_df, "[filepath]/FH-DF.csv")

```

### Execute FH

#NHS

```{r warning=FALSE, message=FALSE}

options(digits = 5)

FH\_procedure <- function(x){

FH\_df1 <- FH\_df

FH\_df1[x, 30] <- FH\_df1[x, 2] #30 corresponds with NR\_NHS\_Mean, 2 with NHS\_direct\_est

FH\_df1[x, 31] <- FH\_df1[x, 3] #31 corresponds with NR\_NHS\_SE, 3 with NHS\_se

FH\_df1 <- filter(FH\_df1, NR\_NHS\_Mean != "NA") #drops non target value rows

attach(FH\_df1)

mseFH(NR\_NHS\_Mean ~ B\_H\_AINA + FER + p\_EL + SQI, NR\_NHS\_SE^2) #line changes per subgroup

}

NHS\_FH\_results <- lapply(1:length(FH\_df$NR\_NHS\_Mean), FH\_procedure)

###########################################################################

# Extract the data of interest from NHS\_FH\_results (saved as a list object)

###########################################################################

#Set up function to automate extraction of eblups

get\_eblup <- function(x){

NHS\_FH\_results[[x]]$est$eblup[x]

}

#Automate with lapply

NHS\_eblups <- lapply(1:48, get\_eblup) #line changes per subgroup

#Wrangle eblups into data frame and long format

NHS\_eblups <- t(as.data.frame(NHS\_eblups))

rownames(NHS\_eblups) <- c()

NHS\_eblups <- as.data.frame(NHS\_eblups)

names(NHS\_eblups) <- c("NHS eblups")

########################################################################################

#Set up function to automate extraction of mse values for eblups

get\_mse <- function(x){

NHS\_FH\_results[[x]]$mse[x]

}

#Automate with lapply

NHS\_eblup\_se <- lapply(1:48, get\_mse) #line changes per subgroup

#Wrangle eblup MSEs into data frame and long format

NHS\_eblup\_se <- t(as.data.frame(NHS\_eblup\_se))

rownames(NHS\_eblup\_se) <- c()

NHS\_eblup\_se <- as.data.frame(NHS\_eblup\_se)

names(NHS\_eblup\_se) <- c("NHS\_eblup\_se")

#Bind eblup and mse values, along with their corresponding state acronyms (in first column)

NHS\_NR <-

read\_csv("[filepath]/NHS.csv") %>%

filter(Mean != "NA")

NHS\_FH\_results <- cbind(NHS\_NR$State, NHS\_eblups, NHS\_eblup\_se)

names(NHS\_FH\_results) <- c("State", "NHS\_eblups", "NHS\_eblup\_se")

#One last step to make sure the \_se variable actually includes standard errors (not mean squared errors)

NHS\_FH\_results$NHS\_eblup\_se <- sqrt(NHS\_FH\_results$NHS\_eblup\_se)

```

#HS

```{r warning=FALSE, message=FALSE}

FH\_procedure <- function(x){

FH\_df1 <- FH\_df

FH\_df1[x, 32] <- FH\_df1[x, 4] #32 corresponds with NR\_HS\_Mean, 4 with HS\_direct\_est

FH\_df1[x, 33] <- FH\_df1[x, 5] #33 corresponds with NR\_HS\_SE, 5 with HS\_se

FH\_df1 <- filter(FH\_df1, NR\_HS\_Mean != "NA") #drops non target value rows

attach(FH\_df1)

mseFH(NR\_HS\_Mean ~ B\_H\_AINA + FER + p\_EL + SQI, NR\_HS\_SE^2) #line changes per subgroup

}

HS\_FH\_results <- lapply(1:length(FH\_df$NR\_HS\_Mean), FH\_procedure)

###########################################################################

# Extract the data of interest from HS\_FH\_results (saved as a list object)

###########################################################################

#Set up function to automate extraction of eblups

get\_eblup <- function(x){

HS\_FH\_results[[x]]$est$eblup[x]

}

#Automate with lapply

HS\_eblups <- lapply(1:48, get\_eblup) #line changes per subgroup

#Wrangle eblups into data frame and long format

HS\_eblups <- t(as.data.frame(HS\_eblups))

rownames(HS\_eblups) <- c()

HS\_eblups <- as.data.frame(HS\_eblups)

names(HS\_eblups) <- c("HS eblups")

########################################################################################

#Set up function to automate extraction of mse values for eblups

get\_mse <- function(x){

HS\_FH\_results[[x]]$mse[x]

}

#Automate with lapply

HS\_eblup\_se <- lapply(1:48, get\_mse) #line changes per subgroup

#Wrangle eblup MSEs into data frame and long format

HS\_eblup\_se <- t(as.data.frame(HS\_eblup\_se))

rownames(HS\_eblup\_se) <- c()

HS\_eblup\_se <- as.data.frame(HS\_eblup\_se)

names(HS\_eblup\_se) <- c("HS\_eblup\_se")

#Bind eblup and mse values, along with their corresponding state acronyms (in first column)

HS\_NR <-

read\_csv("[filepath]/HS.csv") %>%

filter(Mean != "NA")

HS\_FH\_results <- cbind(HS\_NR$State, HS\_eblups, HS\_eblup\_se)

names(HS\_FH\_results) <- c("State", "HS\_eblups", "HS\_eblup\_se")

#One last step to make sure the \_se variable actually includes standard errors (not mean squared errors)

HS\_FH\_results$HS\_eblup\_se <- sqrt(HS\_FH\_results$HS\_eblup\_se)

```

#SBA

```{r warning=FALSE, message=FALSE}

FH\_procedure <- function(x){

FH\_df1 <- FH\_df

FH\_df1[x, 34] <- FH\_df1[x, 6] #34 corresponds with NR\_SBA\_Mean, 6 with SBA\_direct\_est

FH\_df1[x, 35] <- FH\_df1[x, 7] #35 corresponds with NR\_SBA\_SE, 7 with SBA\_se

FH\_df1 <- filter(FH\_df1, NR\_SBA\_Mean != "NA") #drops non target value rows

attach(FH\_df1)

mseFH(NR\_SBA\_Mean ~ B\_H\_AINA + FER + p\_EL + SQI, NR\_SBA\_SE^2) #line changes per subgroup

}

SBA\_FH\_results <- lapply(1:length(FH\_df$NR\_SBA\_Mean), FH\_procedure)

###########################################################################

# Extract the data of interest from SBA\_FH\_results (saved as a list object)

###########################################################################

#Set up function to automate extraction of eblups

get\_eblup <- function(x){

SBA\_FH\_results[[x]]$est$eblup[x]

}

#Automate with lapply

SBA\_eblups <- lapply(1:48, get\_eblup) #line changes per subgroup

#Wrangle eblups into data frame and long format

SBA\_eblups <- t(as.data.frame(SBA\_eblups))

rownames(SBA\_eblups) <- c()

SBA\_eblups <- as.data.frame(SBA\_eblups)

names(SBA\_eblups) <- c("SBA eblups")

########################################################################################

#Set up function to automate extraction of mse values for eblups

get\_mse <- function(x){

SBA\_FH\_results[[x]]$mse[x]

}

#Automate with lapply

SBA\_eblup\_se <- lapply(1:48, get\_mse) #line changes per subgroup

#Wrangle eblup MSEs into data frame and long format

SBA\_eblup\_se <- t(as.data.frame(SBA\_eblup\_se))

rownames(SBA\_eblup\_se) <- c()

SBA\_eblup\_se <- as.data.frame(SBA\_eblup\_se)

names(SBA\_eblup\_se) <- c("SBA\_eblup\_se")

#Bind eblup and mse values, along with their corresponding state acronyms (in first column)

SBA\_NR <-

read\_csv("[filepath]/SBA.csv") %>%

filter(Mean != "NA")

SBA\_FH\_results <- cbind(SBA\_NR$State, SBA\_eblups, SBA\_eblup\_se)

names(SBA\_FH\_results) <- c("State", "SBA\_eblups", "SBA\_eblup\_se")

#One last step to make sure the \_se variable actually includes standard errors (not mean squared errors)

SBA\_FH\_results$SBA\_eblup\_se <- sqrt(SBA\_FH\_results$SBA\_eblup\_se)

```

#BA

```{r warning=FALSE, message=FALSE}

FH\_procedure <- function(x){

FH\_df1 <- FH\_df

FH\_df1[x, 36] <- FH\_df1[x, 8] #36 corresponds with NR\_BA\_Mean, 8 with BA\_direct\_est

FH\_df1[x, 37] <- FH\_df1[x, 9] #37 corresponds with NR\_BA\_SE, 9 with BA\_se

FH\_df1 <- filter(FH\_df1, NR\_BA\_Mean != "NA") #drops non target value rows

attach(FH\_df1)

mseFH(NR\_BA\_Mean ~ B\_H\_AINA + FER + p\_EL + SQI, NR\_BA\_SE^2) #line changes per subgroup

}

BA\_FH\_results <- lapply(1:length(FH\_df$NR\_BA\_Mean), FH\_procedure)

###########################################################################

# Extract the data of interest from BA\_FH\_results (saved as a list object)

###########################################################################

#Set up function to automate extraction of eblups

get\_eblup <- function(x){

BA\_FH\_results[[x]]$est$eblup[x]

}

#Automate with lapply

BA\_eblups <- lapply(1:48, get\_eblup) #line changes per subgroup

#Wrangle eblups into data frame and long format

BA\_eblups <- t(as.data.frame(BA\_eblups))

rownames(BA\_eblups) <- c()

BA\_eblups <- as.data.frame(BA\_eblups)

names(BA\_eblups) <- c("BA eblups")

########################################################################################

#Set up function to automate extraction of mse values for eblups

get\_mse <- function(x){

BA\_FH\_results[[x]]$mse[x]

}

#Automate with lapply

BA\_eblup\_se <- lapply(1:48, get\_mse) #line changes per subgroup

#Wrangle eblup MSEs into data frame and long format

BA\_eblup\_se <- t(as.data.frame(BA\_eblup\_se))

rownames(BA\_eblup\_se) <- c()

BA\_eblup\_se <- as.data.frame(BA\_eblup\_se)

names(BA\_eblup\_se) <- c("BA\_eblup\_se")

#Bind eblup and mse values, along with their corresponding state acronyms (in first column)

BA\_NR <-

read\_csv("[filepath]/BA.csv") %>%

filter(Mean != "NA")

BA\_FH\_results <- cbind(BA\_NR$State, BA\_eblups, BA\_eblup\_se)

names(BA\_FH\_results) <- c("State", "BA\_eblups", "BA\_eblup\_se")

#One last step to make sure the \_se variable actually includes standard errors (not mean squared errors)

BA\_FH\_results$BA\_eblup\_se <- sqrt(BA\_FH\_results$BA\_eblup\_se)

```

#B

```{r warning=FALSE, message=FALSE}

FH\_procedure <- function(x){

FH\_df1 <- FH\_df

FH\_df1[x, 38] <- FH\_df1[x, 10] #38 corresponds with NR\_B\_Mean, 10 with B\_direct\_est

FH\_df1[x, 39] <- FH\_df1[x, 11] #39 corresponds with NR\_B\_SE, 11 with B\_se

FH\_df1 <- filter(FH\_df1, NR\_B\_Mean != "NA") #drops non target value rows

attach(FH\_df1)

mseFH(NR\_B\_Mean ~ p\_BA + FER + SQI + AA, NR\_B\_SE^2) #line changes per subgroup

}

B\_FH\_results <- lapply(1:length(FH\_df$NR\_B\_Mean), FH\_procedure)

###########################################################################

# Extract the data of interest from B\_FH\_results (saved as a list object)

###########################################################################

#Set up function to automate extraction of eblups

get\_eblup <- function(x){

B\_FH\_results[[x]]$est$eblup[x]

}

#Automate with lapply

B\_eblups <- lapply(1:39, get\_eblup) #line changes per subgroup

#Wrangle eblups into data frame and long format

B\_eblups <- t(as.data.frame(B\_eblups))

rownames(B\_eblups) <- c()

B\_eblups <- as.data.frame(B\_eblups)

names(B\_eblups) <- c("B eblups")

########################################################################################

#Set up function to automate extraction of mse values for eblups

get\_mse <- function(x){

B\_FH\_results[[x]]$mse[x]

}

#Automate with lapply

B\_eblup\_se <- lapply(1:39, get\_mse) #line changes per subgroup

#Wrangle eblup MSEs into data frame and long format

B\_eblup\_se <- t(as.data.frame(B\_eblup\_se))

rownames(B\_eblup\_se) <- c()

B\_eblup\_se <- as.data.frame(B\_eblup\_se)

names(B\_eblup\_se) <- c("B\_eblup\_se")

#Bind eblup and mse values, along with their corresponding state acronyms (in first column)

B\_NR <-

read\_csv("[filepath]/B.csv") %>%

filter(Mean != "NA")

B\_FH\_results <- cbind(B\_NR$State, B\_eblups, B\_eblup\_se)

names(B\_FH\_results) <- c("State", "B\_eblups", "B\_eblup\_se")

#One last step to make sure the \_se variable actually includes standard errors (not mean squared errors)

B\_FH\_results$B\_eblup\_se <- sqrt(B\_FH\_results$B\_eblup\_se)

```

#H

```{r warning=FALSE, message=FALSE}

FH\_procedure <- function(x){

FH\_df1 <- FH\_df

FH\_df1[x, 40] <- FH\_df1[x, 12] #40 corresponds with NR\_H\_Mean, 12 with H\_direct\_est

FH\_df1[x, 41] <- FH\_df1[x, 13] #41 corresponds with NR\_H\_SE, 13 with H\_se

FH\_df1 <- filter(FH\_df1, NR\_H\_Mean != "NA") #drops non target value rows

attach(FH\_df1)

mseFH(NR\_H\_Mean ~ p\_BA + FER + p\_EL + SQI + MX, NR\_H\_SE^2) #line changes per subgroup

}

H\_FH\_results <- lapply(1:length(FH\_df$NR\_H\_Mean), FH\_procedure)

###########################################################################

# Extract the data of interest from H\_FH\_results (saved as a list object)

###########################################################################

#Set up function to automate extraction of eblups

get\_eblup <- function(x){

H\_FH\_results[[x]]$est$eblup[x]

}

#Automate with lapply

H\_eblups <- lapply(1:47, get\_eblup) #line changes per subgroup

#Wrangle eblups into data frame and long format

H\_eblups <- t(as.data.frame(H\_eblups))

rownames(H\_eblups) <- c()

H\_eblups <- as.data.frame(H\_eblups)

names(H\_eblups) <- c("H eblups")

########################################################################################

#Set up function to automate extraction of mse values for eblups

get\_mse <- function(x){

H\_FH\_results[[x]]$mse[x]

}

#Automate with lapply

H\_eblup\_se <- lapply(1:47, get\_mse) #line changes per subgroup

#Wrangle eblup MSEs into data frame and long format

H\_eblup\_se <- t(as.data.frame(H\_eblup\_se))

rownames(H\_eblup\_se) <- c()

H\_eblup\_se <- as.data.frame(H\_eblup\_se)

names(H\_eblup\_se) <- c("H\_eblup\_se")

#Bind eblup and mse values, along with their corresponding state acronyms (in first column)

H\_NR <-

read\_csv("[filepath]/H.csv") %>%

filter(Mean != "NA")

H\_FH\_results <- cbind(H\_NR$State, H\_eblups, H\_eblup\_se)

names(H\_FH\_results) <- c("State", "H\_eblups", "H\_eblup\_se")

#One last step to make sure the \_se variable actually includes standard errors (not mean squared errors)

H\_FH\_results$H\_eblup\_se <- sqrt(H\_FH\_results$H\_eblup\_se)

```

#API

```{r warning=FALSE, message=FALSE}

FH\_procedure <- function(x){

FH\_df1 <- FH\_df

FH\_df1[x, 42] <- FH\_df1[x, 14] #42 corresponds with NR\_API\_Mean, 14 with API\_direct\_est

FH\_df1[x, 43] <- FH\_df1[x, 15] #43 corresponds with NR\_API\_SE, 15 with API\_se

FH\_df1 <- filter(FH\_df1, NR\_API\_Mean != "NA") #drops non target value rows

attach(FH\_df1)

mseFH(NR\_API\_Mean ~ p\_BA + FER + p\_EL + SQI + A, NR\_API\_SE^2) #line changes per subgroup

}

API\_FH\_results <- lapply(1:length(FH\_df$NR\_API\_Mean), FH\_procedure)

###########################################################################

# Extract the data of interest from API\_FH\_results (saved as a list object)

###########################################################################

#Set up function to automate extraction of eblups

get\_eblup <- function(x){

API\_FH\_results[[x]]$est$eblup[x]

}

#Automate with lapply

API\_eblups <- lapply(1:30, get\_eblup) #line changes per subgroup

#Wrangle eblups into data frame and long format

API\_eblups <- t(as.data.frame(API\_eblups))

rownames(API\_eblups) <- c()

API\_eblups <- as.data.frame(API\_eblups)

names(API\_eblups) <- c("API eblups")

########################################################################################

#Set up function to automate extraction of mse values for eblups

get\_mse <- function(x){

API\_FH\_results[[x]]$mse[x]

}

#Automate with lapply

API\_eblup\_se <- lapply(1:30, get\_mse) #line changes per subgroup

#Wrangle eblup MSEs into data frame and long format

API\_eblup\_se <- t(as.data.frame(API\_eblup\_se))

rownames(API\_eblup\_se) <- c()

API\_eblup\_se <- as.data.frame(API\_eblup\_se)

names(API\_eblup\_se) <- c("API\_eblup\_se")

#Bind eblup and mse values, along with their corresponding state acronyms (in first column)

API\_NR <-

read\_csv("[filepath]/API.csv") %>%

filter(Mean != "NA")

API\_FH\_results <- cbind(API\_NR$State, API\_eblups, API\_eblup\_se)

names(API\_FH\_results) <- c("State", "API\_eblups", "API\_eblup\_se")

#One last step to make sure the \_se variable actually includes standard errors (not mean squared errors)

API\_FH\_results$API\_eblup\_se <- sqrt(API\_FH\_results$API\_eblup\_se)

```

#AINA

```{r warning=FALSE, message=FALSE}

FH\_procedure <- function(x){

FH\_df1 <- FH\_df

FH\_df1[x, 44] <- FH\_df1[x, 16] #44 corresponds with NR\_AINA\_Mean, 16 with AINA\_direct\_est

FH\_df1[x, 45] <- FH\_df1[x, 17] #45 corresponds with NR\_AINA\_SE, 17 with AINA\_se

FH\_df1 <- filter(FH\_df1, NR\_AINA\_Mean != "NA") #drops non target value rows

attach(FH\_df1)

mseFH(NR\_AINA\_Mean ~ p\_BA + FER, NR\_AINA\_SE^2) #line changes per subgroup

}

AINA\_FH\_results <- lapply(1:length(FH\_df$NR\_AINA\_Mean), FH\_procedure)

###########################################################################

# Extract the data of interest from AINA\_FH\_results (saved as a list object)

###########################################################################

#Set up function to automate extraction of eblups

get\_eblup <- function(x){

AINA\_FH\_results[[x]]$est$eblup[x]

}

#Automate with lapply

AINA\_eblups <- lapply(1:13, get\_eblup) #line changes per subgroup

#Wrangle eblups into data frame and long format

AINA\_eblups <- t(as.data.frame(AINA\_eblups))

rownames(AINA\_eblups) <- c()

AINA\_eblups <- as.data.frame(AINA\_eblups)

names(AINA\_eblups) <- c("AINA eblups")

########################################################################################

#Set up function to automate extraction of mse values for eblups

get\_mse <- function(x){

AINA\_FH\_results[[x]]$mse[x]

}

#Automate with lapply

AINA\_eblup\_se <- lapply(1:13, get\_mse) #line changes per subgroup

#Wrangle eblup MSEs into data frame and long format

AINA\_eblup\_se <- t(as.data.frame(AINA\_eblup\_se))

rownames(AINA\_eblup\_se) <- c()

AINA\_eblup\_se <- as.data.frame(AINA\_eblup\_se)

names(AINA\_eblup\_se) <- c("AINA\_eblup\_se")

#Bind eblup and mse values, along with their corresponding state acronyms (in first column)

AINA\_NR <-

read\_csv("[filepath]/AINA.csv") %>%

filter(Mean != "NA")

AINA\_FH\_results <- cbind(AINA\_NR$State, AINA\_eblups, AINA\_eblup\_se)

names(AINA\_FH\_results) <- c("State", "AINA\_eblups", "AINA\_eblup\_se")

#One last step to make sure the \_se variable actually includes standard errors (not mean squared errors)

AINA\_FH\_results$AINA\_eblup\_se <- sqrt(AINA\_FH\_results$AINA\_eblup\_se)

```

#TP

```{r warning=FALSE, message=FALSE}

FH\_procedure <- function(x){

FH\_df1 <- FH\_df

FH\_df1[x, 46] <- FH\_df1[x, 18] #46 corresponds with NR\_TP\_Mean, 18 with TP\_direct\_est

FH\_df1[x, 47] <- FH\_df1[x, 19] #47 corresponds with NR\_TP\_SE, 19 with TP\_se

FH\_df1 <- filter(FH\_df1, NR\_TP\_Mean != "NA") #drops non target value rows

attach(FH\_df1)

mseFH(NR\_TP\_Mean ~ p\_BA + B\_H\_AINA + FER + SQI, NR\_TP\_SE^2) #line changes per subgroup

}

TP\_FH\_results <- lapply(1:length(FH\_df$NR\_TP\_Mean), FH\_procedure)

###########################################################################

# Extract the data of interest from TP\_FH\_results (saved as a list object)

###########################################################################

#Set up function to automate extraction of eblups

get\_eblup <- function(x){

TP\_FH\_results[[x]]$est$eblup[x]

}

#Automate with lapply

TP\_eblups <- lapply(1:24, get\_eblup) #line changes per subgroup

#Wrangle eblups into data frame and long format

TP\_eblups <- t(as.data.frame(TP\_eblups))

rownames(TP\_eblups) <- c()

TP\_eblups <- as.data.frame(TP\_eblups)

names(TP\_eblups) <- c("TP eblups")

########################################################################################

#Set up function to automate extraction of mse values for eblups

get\_mse <- function(x){

TP\_FH\_results[[x]]$mse[x]

}

#Automate with lapply

TP\_eblup\_se <- lapply(1:24, get\_mse) #line changes per subgroup

#Wrangle eblup MSEs into data frame and long format

TP\_eblup\_se <- t(as.data.frame(TP\_eblup\_se))

rownames(TP\_eblup\_se) <- c()

TP\_eblup\_se <- as.data.frame(TP\_eblup\_se)

names(TP\_eblup\_se) <- c("TP\_eblup\_se")

#Bind eblup and mse values, along with their corresponding state acronyms (in first column)

TP\_NR <-

read\_csv("[filepath]/2+.csv") %>%

filter(Mean != "NA")

TP\_FH\_results <- cbind(TP\_NR$State, TP\_eblups, TP\_eblup\_se)

names(TP\_FH\_results) <- c("State", "TP\_eblups", "TP\_eblup\_se")

#One last step to make sure the \_se variable actually includes standard errors (not mean squared errors)

TP\_FH\_results$TP\_eblup\_se <- sqrt(TP\_FH\_results$TP\_eblup\_se)

```

#EL

```{r warning=FALSE, message=FALSE}

FH\_procedure <- function(x){

FH\_df1 <- FH\_df

FH\_df1[x, 48] <- FH\_df1[x, 20] #48 corresponds with NR\_EL\_Mean, 20 with EL\_direct\_est

FH\_df1[x, 49] <- FH\_df1[x, 21] #49 corresponds with NR\_EL\_SE, 21 with EL\_se

FH\_df1 <- filter(FH\_df1, NR\_EL\_Mean != "NA") #drops non target value rows

attach(FH\_df1)

mseFH(NR\_EL\_Mean ~ B\_H\_AINA + FER + SQI, NR\_EL\_SE^2) #line changes per subgroup

}

EL\_FH\_results <- lapply(1:length(FH\_df$NR\_EL\_Mean), FH\_procedure)

###########################################################################

# Extract the data of interest from EL\_FH\_results (saved as a list object)

###########################################################################

#Set up function to automate extraction of eblups

get\_eblup <- function(x){

EL\_FH\_results[[x]]$est$eblup[x]

}

#Automate with lapply

EL\_eblups <- lapply(1:31, get\_eblup) #line changes per subgroup

#Wrangle eblups into data frame and long format

EL\_eblups <- t(as.data.frame(EL\_eblups))

rownames(EL\_eblups) <- c()

EL\_eblups <- as.data.frame(EL\_eblups)

names(EL\_eblups) <- c("EL eblups")

########################################################################################

#Set up function to automate extraction of mse values for eblups

get\_mse <- function(x){

EL\_FH\_results[[x]]$mse[x]

}

#Automate with lapply

EL\_eblup\_se <- lapply(1:31, get\_mse) #line changes per subgroup

#Wrangle eblup MSEs into data frame and long format

EL\_eblup\_se <- t(as.data.frame(EL\_eblup\_se))

rownames(EL\_eblup\_se) <- c()

EL\_eblup\_se <- as.data.frame(EL\_eblup\_se)

names(EL\_eblup\_se) <- c("EL\_eblup\_se")

#Bind eblup and mse values, along with their corresponding state acronyms (in first column)

EL\_NR <-

read\_csv("[filepath]/EL.csv") %>%

filter(Mean != "NA")

EL\_FH\_results <- cbind(EL\_NR$State, EL\_eblups, EL\_eblup\_se)

names(EL\_FH\_results) <- c("State", "EL\_eblups", "EL\_eblup\_se")

#One last step to make sure the \_se variable actually includes standard errors (not mean squared errors)

EL\_FH\_results$EL\_eblup\_se <- sqrt(EL\_FH\_results$EL\_eblup\_se)

```

### Export and prepare estimates for calculating accuracy statistics

```{r warning=FALSE, message=FALSE}

#NHS

write.csv(NHS\_FH\_results, "[filepath]/NHS-FH-ESTIMATES.csv")

#HS

write.csv(HS\_FH\_results, "[filepath]/HS-FH-ESTIMATES.csv")

#SBA

write.csv(SBA\_FH\_results, "[filepath]/SBA-FH-ESTIMATES.csv")

#BA

write.csv(BA\_FH\_results, "[filepath]/BA-FH-ESTIMATES.csv")

#B

write.csv(B\_FH\_results, "[filepath]/B-FH-ESTIMATES.csv")

#H

write.csv(H\_FH\_results, "[filepath]/H-FH-ESTIMATES.csv")

#API

write.csv(API\_FH\_results, "[filepath]/API-FH-ESTIMATES.csv")

#AINA

write.csv(AINA\_FH\_results, "[filepath]/AINA-FH-ESTIMATES.csv")

#TP

write.csv(TP\_FH\_results, "[filepath]/TP-FH-ESTIMATES.csv")

#EL

write.csv(EL\_FH\_results, "[filepath]/EL-FH-ESTIMATES.csv")

```

### Calculate wMAE for FH technique

```{r warning=FALSE, message=FALSE}

\*\*Free up memory\*\*

rm(list = ls())

```

#\*\*Define a wMAE function\*\*

```{r warning=FALSE, message=FALSE}

wmae <- function(x){

w\_abs\_errors <- abs(x[,2] - x[,4])/x[,3]

mean(w\_abs\_errors[,1])

}

```

#NHS

```{r warning=FALSE, message=FALSE}

#Import/Wrangle predicted and observed data

NHS\_predicted <- read\_csv("[filepath]/NHS-FH-ESTIMATES.csv")

NHS\_predicted <- NHS\_predicted[,-1]

names(NHS\_predicted) <- c("State", "FH\_mean", "FH\_se")

NHS\_observed <- read\_csv("[filepath]/NHS.csv")

NHS\_observed <- NHS\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

NHS <- inner\_join(NHS\_observed, NHS\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(NHS)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, warning=FALSE, message=FALSE}

#Make NHS tidy

source <- rep(c("NAEP"), 48)

source <- as.data.frame(source)

part1 <- cbind(NHS$State, source, NHS$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("FH"), 48)

source <- as.data.frame(source)

part2 <- cbind(NHS$State, source, NHS$FH\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

NHS\_tidy <- rbind(part1, part2)

NHS\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

NHS\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

NHS\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#HS

```{r message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

HS\_predicted <- read\_csv("[filepath]/HS-FH-ESTIMATES.csv")

HS\_predicted <- HS\_predicted[,-1]

names(HS\_predicted) <- c("State", "FH\_mean", "FH\_se")

HS\_observed <- read\_csv("[filepath]/HS.csv")

HS\_observed <- HS\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

HS <- inner\_join(HS\_observed, HS\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(HS)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, warning=FALSE, message=FALSE}

#Make HS tidy

source <- rep(c("NAEP"), 48)

source <- as.data.frame(source)

part1 <- cbind(HS$State, source, HS$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("FH"), 48)

source <- as.data.frame(source)

part2 <- cbind(HS$State, source, HS$FH\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

HS\_tidy <- rbind(part1, part2)

HS\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

HS\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

HS\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#SBA

```{r message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

SBA\_predicted <- read\_csv("[filepath]/SBA-FH-ESTIMATES.csv")

SBA\_predicted <- SBA\_predicted[,-1]

names(SBA\_predicted) <- c("State", "FH\_mean", "FH\_se")

SBA\_observed <- read\_csv("[filepath]/SBA.csv")

SBA\_observed <- SBA\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

SBA <- inner\_join(SBA\_observed, SBA\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(SBA)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, warning=FALSE, message=FALSE}

#Make SBA tidy

source <- rep(c("NAEP"), 48)

source <- as.data.frame(source)

part1 <- cbind(SBA$State, source, SBA$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("FH"), 48)

source <- as.data.frame(source)

part2 <- cbind(SBA$State, source, SBA$FH\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

SBA\_tidy <- rbind(part1, part2)

SBA\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

SBA\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

SBA\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#BA

```{r message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

BA\_predicted <- read\_csv("[filepath]/BA-FH-ESTIMATES.csv")

BA\_predicted <- BA\_predicted[,-1]

names(BA\_predicted) <- c("State", "FH\_mean", "FH\_se")

BA\_observed <- read\_csv("[filepath]/BA.csv")

BA\_observed <- BA\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

BA <- inner\_join(BA\_observed, BA\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(BA)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, warning=FALSE, message=FALSE}

#Make BA tidy

source <- rep(c("NAEP"), 48)

source <- as.data.frame(source)

part1 <- cbind(BA$State, source, BA$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("FH"), 48)

source <- as.data.frame(source)

part2 <- cbind(BA$State, source, BA$FH\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

BA\_tidy <- rbind(part1, part2)

BA\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

BA\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

BA\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#B

```{r message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

B\_predicted <- read\_csv("[filepath]/B-FH-ESTIMATES.csv")

B\_predicted <- B\_predicted[,-1]

names(B\_predicted) <- c("State", "FH\_mean", "FH\_se")

B\_observed <- read\_csv("[filepath]/B.csv")

B\_observed <- B\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

B <- inner\_join(B\_observed, B\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(B)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r message=FALSE, warning=FALSE}

#Make B tidy

source <- rep(c("NAEP"), 39)

source <- as.data.frame(source)

part1 <- cbind(B$State, source, B$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("FH"), 39)

source <- as.data.frame(source)

part2 <- cbind(B$State, source, B$FH\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

B\_tidy <- rbind(part1, part2)

B\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

B\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

B\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#H

```{r message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

H\_predicted <- read\_csv("[filepath]/H-FH-ESTIMATES.csv")

H\_predicted <- H\_predicted[,-1]

names(H\_predicted) <- c("State", "FH\_mean", "FH\_se")

H\_observed <- read\_csv("[filepath]/H.csv")

H\_observed <- H\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

H <- inner\_join(H\_observed, H\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(H)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, warning=FALSE, message=FALSE}

#Make H tidy

source <- rep(c("NAEP"), 47)

source <- as.data.frame(source)

part1 <- cbind(H$State, source, H$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("FH"), 47)

source <- as.data.frame(source)

part2 <- cbind(H$State, source, H$FH\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

H\_tidy <- rbind(part1, part2)

H\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

H\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

H\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#API

```{r message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

API\_predicted <- read\_csv("[filepath]/API-FH-ESTIMATES.csv")

API\_predicted <- API\_predicted[,-1]

names(API\_predicted) <- c("State", "FH\_mean", "FH\_se")

API\_observed <- read\_csv("[filepath]/API.csv")

API\_observed <- API\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

API <- inner\_join(API\_observed, API\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(API)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, warning=FALSE, message=FALSE}

#Make API tidy

source <- rep(c("NAEP"), 30)

source <- as.data.frame(source)

part1 <- cbind(API$State, source, API$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("FH"), 30)

source <- as.data.frame(source)

part2 <- cbind(API$State, source, API$FH\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

API\_tidy <- rbind(part1, part2)

API\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

API\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

API\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#AINA

```{r message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

AINA\_predicted <- read\_csv("[filepath]/AINA-FH-ESTIMATES.csv")

AINA\_predicted <- AINA\_predicted[,-1]

names(AINA\_predicted) <- c("State", "FH\_mean", "FH\_se")

AINA\_observed <- read\_csv("[filepath]/AINA.csv")

AINA\_observed <- AINA\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

AINA <- inner\_join(AINA\_observed, AINA\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(AINA)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, warning=FALSE, message=FALSE}

#Make AINA tidy

source <- rep(c("NAEP"), 13)

source <- as.data.frame(source)

part1 <- cbind(AINA$State, source, AINA$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("FH"), 13)

source <- as.data.frame(source)

part2 <- cbind(AINA$State, source, AINA$FH\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

AINA\_tidy <- rbind(part1, part2)

AINA\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

AINA\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

AINA\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#TP

```{r message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

TP\_predicted <- read\_csv("[filepath]/TP-FH-ESTIMATES.csv")

TP\_predicted <- TP\_predicted[,-1]

names(TP\_predicted) <- c("State", "FH\_mean", "FH\_se")

TP\_observed <- read\_csv("[filepath]/2+.csv")

TP\_observed <- TP\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

TP <- inner\_join(TP\_observed, TP\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(TP)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, warning=FALSE, message=FALSE}

#Make TP tidy

source <- rep(c("NAEP"), 24)

source <- as.data.frame(source)

part1 <- cbind(TP$State, source, TP$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("FH"), 24)

source <- as.data.frame(source)

part2 <- cbind(TP$State, source, TP$FH\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

TP\_tidy <- rbind(part1, part2)

TP\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

TP\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

TP\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#EL

```{r message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

EL\_predicted <- read\_csv("[filepath]/EL-FH-ESTIMATES.csv")

EL\_predicted <- EL\_predicted[,-1]

names(EL\_predicted) <- c("State", "FH\_mean", "FH\_se")

EL\_observed <- read\_csv("[filepath]/EL.csv")

EL\_observed <- EL\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

EL <- inner\_join(EL\_observed, EL\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(EL)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, warning=FALSE, message=FALSE}

#Make EL tidy

source <- rep(c("NAEP"), 31)

source <- as.data.frame(source)

part1 <- cbind(EL$State, source, EL$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("FH"), 31)

source <- as.data.frame(source)

part2 <- cbind(EL$State, source, EL$FH\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

EL\_tidy <- rbind(part1, part2)

EL\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

EL\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

EL\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#\*\*Overall\*\*

```{r message=FALSE, warning=FALSE}

#Row bind subgroup datasets into an "overall dataset"

overall <- rbind(NHS, HS, SBA, BA, B, H, API, AINA, TP, EL)

#Calculate wMAE for FH across subgroups

wmae(overall)

```

### Calculate Coverage for FH Technique

#First, add the median of the NAEP-reported state-level standard deviations to each respective subgroup's data set (this permits calculation of the b-statistic).

```{r message=FALSE, warning=FALSE}

NHS$median\_sd <- 31.5

HS$median\_sd <- 32.6

SBA$median\_sd <- 30.6

BA$median\_sd <- 34.4

B$median\_sd <- 33.4

H$median\_sd <- 34.0

API$median\_sd <- 38.1

AINA$median\_sd <- 35.4

TP$median\_sd <- 35.2

EL$median\_sd <- 33.3

```

#\*\*By subgroup\*\*

#NHS

```{r message=FALSE, warning=FALSE}

NHS$b\_statisic <- abs(NHS$NAEP\_mean-NHS$FH\_mean)/NHS$median\_sd

mean(NHS$b\_statisic <= 0.20)

```

#NHS-visual

```{r message=FALSE, warning=FALSE}

NHS$lowerbound <- NHS$NAEP\_mean - 0.2\*NHS$median\_sd

NHS$upperbound <- NHS$NAEP\_mean + 0.2\*NHS$median\_sd

NHS <- NHS %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = NHS, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = NHS, aes(FH\_mean, State, col = "FH prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#HS

```{r message=FALSE, warning=FALSE}

HS$b\_statisic <- abs(HS$NAEP\_mean-HS$FH\_mean)/HS$median\_sd

mean(HS$b\_statisic <= 0.20)

```

#HS-visual

```{r message=FALSE, warning=FALSE}

HS$lowerbound <- HS$NAEP\_mean - 0.2\*HS$median\_sd

HS$upperbound <- HS$NAEP\_mean + 0.2\*HS$median\_sd

HS <- HS %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = HS, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = HS, aes(FH\_mean, State, col = "FH prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#SBA

```{r message=FALSE, warning=FALSE}

SBA$b\_statisic <- abs(SBA$NAEP\_mean-SBA$FH\_mean)/SBA$median\_sd

mean(SBA$b\_statisic <= 0.20)

```

#SBA-visual

```{r message=FALSE, warning=FALSE}

SBA$lowerbound <- SBA$NAEP\_mean - 0.2\*SBA$median\_sd

SBA$upperbound <- SBA$NAEP\_mean + 0.2\*SBA$median\_sd

SBA <- SBA %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = SBA, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = SBA, aes(FH\_mean, State, col = "FH prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#BA

```{r message=FALSE, warning=FALSE}

BA$b\_statisic <- abs(BA$NAEP\_mean-BA$FH\_mean)/BA$median\_sd

mean(BA$b\_statisic <= 0.20)

```

#BA-visual

```{r message=FALSE, warning=FALSE}

BA$lowerbound <- BA$NAEP\_mean - 0.2\*BA$median\_sd

BA$upperbound <- BA$NAEP\_mean + 0.2\*BA$median\_sd

BA <- BA %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = BA, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = BA, aes(FH\_mean, State, col = "FH prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#B

```{r message=FALSE, warning=FALSE}

B$b\_statisic <- abs(B$NAEP\_mean-B$FH\_mean)/B$median\_sd

mean(B$b\_statisic <= 0.20)

```

#B-visual

```{r message=FALSE, warning=FALSE}

B$lowerbound <- B$NAEP\_mean - 0.2\*B$median\_sd

B$upperbound <- B$NAEP\_mean + 0.2\*B$median\_sd

B <- B %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = B, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = B, aes(FH\_mean, State, col = "FH prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#H

```{r message=FALSE, warning=FALSE}

H$b\_statisic <- abs(H$NAEP\_mean-H$FH\_mean)/H$median\_sd

mean(H$b\_statisic <= 0.20)

```

#H-visual

```{r message=FALSE, warning=FALSE}

H$lowerbound <- H$NAEP\_mean - 0.2\*H$median\_sd

H$upperbound <- H$NAEP\_mean + 0.2\*H$median\_sd

H <- H %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = H, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = H, aes(FH\_mean, State, col = "FH prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#API

```{r message=FALSE, warning=FALSE}

API$b\_statisic <- abs(API$NAEP\_mean-API$FH\_mean)/API$median\_sd

mean(API$b\_statisic <= 0.20)

```

#API-visual

```{r message=FALSE, warning=FALSE}

API$lowerbound <- API$NAEP\_mean - 0.2\*API$median\_sd

API$upperbound <- API$NAEP\_mean + 0.2\*API$median\_sd

API <- API %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = API, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = API, aes(FH\_mean, State, col = "FH prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#AINA

```{r message=FALSE, warning=FALSE}

AINA$b\_statisic <- abs(AINA$NAEP\_mean-AINA$FH\_mean)/AINA$median\_sd

mean(AINA$b\_statisic <= 0.20)

```

#AINA-visual

```{r message=FALSE, warning=FALSE}

AINA$lowerbound <- AINA$NAEP\_mean - 0.2\*AINA$median\_sd

AINA$upperbound <- AINA$NAEP\_mean + 0.2\*AINA$median\_sd

AINA <- AINA %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = AINA, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = AINA, aes(FH\_mean, State, col = "FH prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#TP (2+)

```{r message=FALSE, warning=FALSE}

TP$b\_statisic <- abs(TP$NAEP\_mean-TP$FH\_mean)/TP$median\_sd

mean(TP$b\_statisic <= 0.20)

```

#TP-visual

```{r message=FALSE, warning=FALSE}

TP$lowerbound <- TP$NAEP\_mean - 0.2\*TP$median\_sd

TP$upperbound <- TP$NAEP\_mean + 0.2\*TP$median\_sd

TP <- TP %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = TP, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = TP, aes(FH\_mean, State, col = "FH prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#EL

```{r message=FALSE, warning=FALSE}

EL$b\_statisic <- abs(EL$NAEP\_mean-EL$FH\_mean)/EL$median\_sd

mean(EL$b\_statisic <= 0.20)

```

#EL-visual

```{r message=FALSE, warning=FALSE}

EL$lowerbound <- EL$NAEP\_mean - 0.2\*EL$median\_sd

EL$upperbound <- EL$NAEP\_mean + 0.2\*EL$median\_sd

EL <- EL %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = EL, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = EL, aes(FH\_mean, State, col = "FH prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#\*\*Overall\*\*

#Row bind datasets

```{r message=FALSE, warning=FALSE}

#Row bind subgroup datasets into an "overall dataset"

overall <- rbind(NHS, HS, SBA, BA, B, H, API, AINA, TP, EL)

#Calculate coverage across groups

mean(overall$b\_statisic <= 0.20)

```

**The FLEX CS Technique (Chapter 4)**

The text that follows in courier font represents R script. Lines of script that begin with a pound/hashtag symbol (i.e., #) are comments (not code). Note, the name of local filepaths (e.g., “C:/Foldername/Foldername/…/Filename”) are replaced with [filepath] throughout. Three consecive backticks (```) demarcate the start and end of a chunk (i.e., section) of code).

**Program/File Type:** R Studio/.RMD File

**Dissertation Section:** Estimates of Mean Math Achievement with the FLEX CS technique

### Install/Load packages

```{r, message=FALSE, warning=FALSE}

if(!require(tidyverse)) install.packages("tidyverse",

repos = "http://cran.us.r-project.org")

if(!require(mice)) install.packages("mice",

repos = "http://cran.us.r-project.org")

if(!require(matrixStats)) install.packages("matrixStats",

repos = "http://cran.us.r-project.org")

if(!require(olsrr)) install.packages("olsrr",

repos = "http://cran.us.r-project.org")

if(!require(EdSurvey)) install.packages("EdSurvey",

repos = "http://cran.us.r-project.org")

if(!require(sae)) install.packages("sae",

repos = "http://cran.us.r-project.org")

```

#Import test sample (& some data manipulation)

```{r, message=FALSE, warning=FALSE, eval=FALSE}

df <- read\_csv("[filepath]/TestSampleAchievement.csv")

#Change variable name of "2+" for compatibility with mice function

df <- rename(df, "TP" = "2+")

#Remove state variable (column)

df\_mice <- df[, -1]

```

#\*\*Define (NEW) Visiting Sequence (based on the 3 reduced imputation models below)\*\*

#BA ~ I + HS + SBA + W + NEL + M + F

#SBA ~ E + HS + BA + NEL + NSWD + M + F

#HS ~ E + SBA + BA + NEL + NSWD + M + F

#Note, these are the only subgroups of interest that meet the specified criteria for computing MICE subestimates.

```{r, message=FALSE, warning=FALSE, eval=FALSE}

vis <- c("BA", "SBA", "HS")

```

#\*\*Construct Predictor Matrix\*\*

```{r, message=FALSE, warning=FALSE, eval=FALSE}

for\_pred\_matrix <- mice(df\_mice, maxit = 0, print=F) #Enables creation a predictor matrix, which will be subsequently edited

pred\_matrix <- for\_pred\_matrix$pred #Creates an initial data matrix with 0s along the diagonal and 1s everywhere else

#Next sets of code remove 1s where I do not want corresponding columns to predict rows

pred\_matrix[1:3, ] <- 0

pred\_matrix[7:18, ] <- 0

#these lines assign 0s to columns for rows (subgroups) that won't be imputed (all but HS, SBA, BA)

pred\_matrix[4, 2:3] <- 0

pred\_matrix[4, 7:13] <- 0

pred\_matrix[4, 15] <- 0

#These lines assign 0s to select columns in the HS row (all but E + SBA + BA + NEL + NSWD + M + F)

pred\_matrix[5, 2:3] <- 0

pred\_matrix[5, 7:13] <- 0

pred\_matrix[5, 15] <- 0

#These lines assign 0s to select columns in the SBA row (all but E + HS + BA + NEL + NSWD + M + F)

pred\_matrix[6, 1] <- 0

pred\_matrix[6, 3] <- 0

pred\_matrix[6, 8:13] <- 0

pred\_matrix[6, 15:16] <- 0

#These lines assign 0s to select columns in the BA row (all but I + HS + SBA + W + NEL + M + F)

```

#\*\*HS\*\*

```{r, message=FALSE, warning=FALSE, eval=FALSE}

#First for the HS variable:

mice\_procedure <- function(x){

df\_mice1 <- df\_mice

df\_mice1[x, 4] <- NA

mice(df\_mice1, m = 100, maxit = 15, visitSequence = vis, method = "norm", pred = pred\_matrix, seed = 2019)

}

HS\_results <- lapply(1:length(df\_mice$HS), mice\_procedure)

```

#\*\*Organize results for HS and compute mean and variance\*\*

```{r, message=FALSE, warning=FALSE, eval=FALSE}

HS\_results\_df <- HS\_results[[1]][["imp"]][["HS"]] %>%

slice(1:2) %>%

rbind(HS\_results[[3]][["imp"]][["HS"]]) %>%

slice(-3, -5) %>%

rbind(HS\_results[[4]][["imp"]][["HS"]]) %>%

slice(-4, -6) %>%

rbind(HS\_results[[5]][["imp"]][["HS"]]) %>%

slice(-5, -7) %>%

rbind(HS\_results[[6]][["imp"]][["HS"]]) %>%

slice(-6, -8) %>%

rbind(HS\_results[[7]][["imp"]][["HS"]]) %>%

slice(-7, -9) %>%

rbind(HS\_results[[8]][["imp"]][["HS"]]) %>%

slice(-8, -10) %>%

rbind(HS\_results[[9]][["imp"]][["HS"]]) %>%

slice(-9, -11) %>%

rbind(HS\_results[[10]][["imp"]][["HS"]]) %>%

slice(-10, -12) %>%

rbind(HS\_results[[11]][["imp"]][["HS"]]) %>%

slice(-11, -13) %>%

rbind(HS\_results[[12]][["imp"]][["HS"]]) %>%

slice(-12, -14) %>%

rbind(HS\_results[[13]][["imp"]][["HS"]]) %>%

slice(-13, -15) %>%

rbind(HS\_results[[14]][["imp"]][["HS"]]) %>%

slice(-14, -16) %>%

rbind(HS\_results[[15]][["imp"]][["HS"]]) %>%

slice(-15, -17) %>%

rbind(HS\_results[[16]][["imp"]][["HS"]]) %>%

slice(-16, -18) %>%

rbind(HS\_results[[17]][["imp"]][["HS"]]) %>%

slice(-17, -19) %>%

rbind(HS\_results[[18]][["imp"]][["HS"]]) %>%

slice(-18, -20) %>%

rbind(HS\_results[[19]][["imp"]][["HS"]]) %>%

slice(-19, -21) %>%

rbind(HS\_results[[20]][["imp"]][["HS"]]) %>%

slice(-20, -22) %>%

rbind(HS\_results[[21]][["imp"]][["HS"]]) %>%

slice(-21, -23) %>%

rbind(HS\_results[[22]][["imp"]][["HS"]]) %>%

slice(-22, -24) %>%

rbind(HS\_results[[23]][["imp"]][["HS"]]) %>%

slice(-23, -25) %>%

rbind(HS\_results[[24]][["imp"]][["HS"]]) %>%

slice(-24, -26) %>%

rbind(HS\_results[[25]][["imp"]][["HS"]]) %>%

slice(-25, -27) %>%

rbind(HS\_results[[26]][["imp"]][["HS"]]) %>%

slice(-26, -28) %>%

rbind(HS\_results[[27]][["imp"]][["HS"]]) %>%

slice(-27, -29) %>%

rbind(HS\_results[[28]][["imp"]][["HS"]]) %>%

slice(-28, -30) %>%

rbind(HS\_results[[29]][["imp"]][["HS"]]) %>%

slice(-29, -31) %>%

rbind(HS\_results[[30]][["imp"]][["HS"]]) %>%

slice(-30, -32) %>%

rbind(HS\_results[[31]][["imp"]][["HS"]]) %>%

slice(-31, -33) %>%

rbind(HS\_results[[32]][["imp"]][["HS"]]) %>%

slice(-32, -34) %>%

rbind(HS\_results[[33]][["imp"]][["HS"]]) %>%

slice(-33, -35) %>%

rbind(HS\_results[[34]][["imp"]][["HS"]]) %>%

slice(-34, -36) %>%

rbind(HS\_results[[35]][["imp"]][["HS"]]) %>%

slice(-35, -37) %>%

rbind(HS\_results[[36]][["imp"]][["HS"]]) %>%

slice(-36, -38) %>%

rbind(HS\_results[[37]][["imp"]][["HS"]]) %>%

slice(-37, -39) %>%

rbind(HS\_results[[38]][["imp"]][["HS"]]) %>%

slice(-38, -40) %>%

rbind(HS\_results[[39]][["imp"]][["HS"]]) %>%

slice(-39, -41) %>%

rbind(HS\_results[[40]][["imp"]][["HS"]]) %>%

slice(-40, -42) %>%

rbind(HS\_results[[41]][["imp"]][["HS"]]) %>%

slice(-41, -43) %>%

rbind(HS\_results[[42]][["imp"]][["HS"]]) %>%

slice(-42, -44) %>%

rbind(HS\_results[[43]][["imp"]][["HS"]]) %>%

slice(-43, -45) %>%

rbind(HS\_results[[45]][["imp"]][["HS"]]) %>%

slice(-44) %>%

rbind(HS\_results[[46]][["imp"]][["HS"]]) %>%

slice(-46, -47) %>%

rbind(HS\_results[[47]][["imp"]][["HS"]]) %>%

slice(-47, -48) %>%

rbind(HS\_results[[48]][["imp"]][["HS"]]) %>%

slice(-48, -49) %>%

rbind(HS\_results[[49]][["imp"]][["HS"]]) %>%

slice(-49, -50) %>%

rbind(HS\_results[[50]][["imp"]][["HS"]]) %>%

slice(-50, -51)

HS\_results\_df <- mutate(HS\_results\_df, mean = rowMeans(HS\_results\_df),

se = rowSds(as.matrix(HS\_results\_df)))

#Save results to computer a .csv

write.csv(HS\_results\_df, "[filepath]/HS-MICE-SUBESTIMATE-Results.csv")

```

#\*\*SBA\*\*

```{r, message=FALSE, warning=FALSE, eval=FALSE}

#Now for the SBA variable:

mice\_procedure <- function(x){

df\_mice1 <- df\_mice

df\_mice1[x, 5] <- NA

mice(df\_mice1, m = 100, maxit = 15, visitSequence = vis, method = "norm", pred = pred\_matrix, seed = 2019)

}

SBA\_results <- lapply(1:length(df\_mice$SBA), mice\_procedure)

```

#\*\*Organize results for SBA and compute mean and variance\*\*

```{r, message=FALSE, warning=FALSE, eval=FALSE}

SBA\_results\_df <- SBA\_results[[1]][["imp"]][["SBA"]] %>%

slice(1:2) %>%

rbind(SBA\_results[[3]][["imp"]][["SBA"]]) %>%

slice(-3, -5) %>%

rbind(SBA\_results[[4]][["imp"]][["SBA"]]) %>%

slice(-4, -6) %>%

rbind(SBA\_results[[5]][["imp"]][["SBA"]]) %>%

slice(-5, -7) %>%

rbind(SBA\_results[[6]][["imp"]][["SBA"]]) %>%

slice(-6, -8) %>%

rbind(SBA\_results[[7]][["imp"]][["SBA"]]) %>%

slice(-7, -9) %>%

rbind(SBA\_results[[8]][["imp"]][["SBA"]]) %>%

slice(-8, -10) %>%

rbind(SBA\_results[[9]][["imp"]][["SBA"]]) %>%

slice(-9, -11) %>%

rbind(SBA\_results[[10]][["imp"]][["SBA"]]) %>%

slice(-10, -12) %>%

rbind(SBA\_results[[11]][["imp"]][["SBA"]]) %>%

slice(-11, -13) %>%

rbind(SBA\_results[[12]][["imp"]][["SBA"]]) %>%

slice(-12, -14) %>%

rbind(SBA\_results[[13]][["imp"]][["SBA"]]) %>%

slice(-13, -15) %>%

rbind(SBA\_results[[14]][["imp"]][["SBA"]]) %>%

slice(-14, -16) %>%

rbind(SBA\_results[[15]][["imp"]][["SBA"]]) %>%

slice(-15, -17) %>%

rbind(SBA\_results[[16]][["imp"]][["SBA"]]) %>%

slice(-16, -18) %>%

rbind(SBA\_results[[17]][["imp"]][["SBA"]]) %>%

slice(-17, -19) %>%

rbind(SBA\_results[[18]][["imp"]][["SBA"]]) %>%

slice(-18, -20) %>%

rbind(SBA\_results[[19]][["imp"]][["SBA"]]) %>%

slice(-19, -21) %>%

rbind(SBA\_results[[20]][["imp"]][["SBA"]]) %>%

slice(-20, -22) %>%

rbind(SBA\_results[[21]][["imp"]][["SBA"]]) %>%

slice(-21, -23) %>%

rbind(SBA\_results[[22]][["imp"]][["SBA"]]) %>%

slice(-22, -24) %>%

rbind(SBA\_results[[23]][["imp"]][["SBA"]]) %>%

slice(-23, -25) %>%

rbind(SBA\_results[[24]][["imp"]][["SBA"]]) %>%

slice(-24, -26) %>%

rbind(SBA\_results[[25]][["imp"]][["SBA"]]) %>%

slice(-25, -27) %>%

rbind(SBA\_results[[26]][["imp"]][["SBA"]]) %>%

slice(-26, -28) %>%

rbind(SBA\_results[[27]][["imp"]][["SBA"]]) %>%

slice(-27, -29) %>%

rbind(SBA\_results[[28]][["imp"]][["SBA"]]) %>%

slice(-28, -30) %>%

rbind(SBA\_results[[29]][["imp"]][["SBA"]]) %>%

slice(-29, -31) %>%

rbind(SBA\_results[[30]][["imp"]][["SBA"]]) %>%

slice(-30, -32) %>%

rbind(SBA\_results[[31]][["imp"]][["SBA"]]) %>%

slice(-31, -33) %>%

rbind(SBA\_results[[32]][["imp"]][["SBA"]]) %>%

slice(-32, -34) %>%

rbind(SBA\_results[[33]][["imp"]][["SBA"]]) %>%

slice(-33, -35) %>%

rbind(SBA\_results[[34]][["imp"]][["SBA"]]) %>%

slice(-34, -36) %>%

rbind(SBA\_results[[35]][["imp"]][["SBA"]]) %>%

slice(-35, -37) %>%

rbind(SBA\_results[[36]][["imp"]][["SBA"]]) %>%

slice(-36, -38) %>%

rbind(SBA\_results[[37]][["imp"]][["SBA"]]) %>%

slice(-37, -39) %>%

rbind(SBA\_results[[38]][["imp"]][["SBA"]]) %>%

slice(-38, -40) %>%

rbind(SBA\_results[[39]][["imp"]][["SBA"]]) %>%

slice(-39, -41) %>%

rbind(SBA\_results[[40]][["imp"]][["SBA"]]) %>%

slice(-40, -42) %>%

rbind(SBA\_results[[41]][["imp"]][["SBA"]]) %>%

slice(-41, -43) %>%

rbind(SBA\_results[[42]][["imp"]][["SBA"]]) %>%

slice(-42, -44) %>%

rbind(SBA\_results[[43]][["imp"]][["SBA"]]) %>%

slice(-43, -45) %>%

rbind(SBA\_results[[45]][["imp"]][["SBA"]]) %>%

slice(-44) %>%

rbind(SBA\_results[[46]][["imp"]][["SBA"]]) %>%

slice(-46, -47) %>%

rbind(SBA\_results[[47]][["imp"]][["SBA"]]) %>%

slice(-47, -48) %>%

rbind(SBA\_results[[48]][["imp"]][["SBA"]]) %>%

slice(-48, -49) %>%

rbind(SBA\_results[[49]][["imp"]][["SBA"]]) %>%

slice(-49, -50) %>%

rbind(SBA\_results[[50]][["imp"]][["SBA"]]) %>%

slice(-50, -51)

SBA\_results\_df <- mutate(SBA\_results\_df, mean = rowMeans(SBA\_results\_df),

se = rowSds(as.matrix(SBA\_results\_df)))

#Save results to computer a .csv

write.csv(SBA\_results\_df, "[filepath]/SBA-MICE-SUBESTIMATE-Results.csv")

```

#\*\*BA\*\*

```{r, message=FALSE, warning=FALSE, eval=FALSE}

#Now for the BA variable:

mice\_procedure <- function(x){

df\_mice1 <- df\_mice

df\_mice1[x, 6] <- NA

mice(df\_mice1, m = 100, maxit = 15, visitSequence = vis, method = "norm", pred = pred\_matrix, seed = 2019)

}

BA\_results <- lapply(1:length(df\_mice$BA), mice\_procedure)

```

#\*\*Organize results for BA and compute mean and variance\*\*

```{r, message=FALSE, warning=FALSE, eval=FALSE}

BA\_results\_df <- BA\_results[[1]][["imp"]][["BA"]] %>%

slice(1:2) %>%

rbind(BA\_results[[3]][["imp"]][["BA"]]) %>%

slice(-3, -5) %>%

rbind(BA\_results[[4]][["imp"]][["BA"]]) %>%

slice(-4, -6) %>%

rbind(BA\_results[[5]][["imp"]][["BA"]]) %>%

slice(-5, -7) %>%

rbind(BA\_results[[6]][["imp"]][["BA"]]) %>%

slice(-6, -8) %>%

rbind(BA\_results[[7]][["imp"]][["BA"]]) %>%

slice(-7, -9) %>%

rbind(BA\_results[[8]][["imp"]][["BA"]]) %>%

slice(-8, -10) %>%

rbind(BA\_results[[9]][["imp"]][["BA"]]) %>%

slice(-9, -11) %>%

rbind(BA\_results[[10]][["imp"]][["BA"]]) %>%

slice(-10, -12) %>%

rbind(BA\_results[[11]][["imp"]][["BA"]]) %>%

slice(-11, -13) %>%

rbind(BA\_results[[12]][["imp"]][["BA"]]) %>%

slice(-12, -14) %>%

rbind(BA\_results[[13]][["imp"]][["BA"]]) %>%

slice(-13, -15) %>%

rbind(BA\_results[[14]][["imp"]][["BA"]]) %>%

slice(-14, -16) %>%

rbind(BA\_results[[15]][["imp"]][["BA"]]) %>%

slice(-15, -17) %>%

rbind(BA\_results[[16]][["imp"]][["BA"]]) %>%

slice(-16, -18) %>%

rbind(BA\_results[[17]][["imp"]][["BA"]]) %>%

slice(-17, -19) %>%

rbind(BA\_results[[18]][["imp"]][["BA"]]) %>%

slice(-18, -20) %>%

rbind(BA\_results[[19]][["imp"]][["BA"]]) %>%

slice(-19, -21) %>%

rbind(BA\_results[[20]][["imp"]][["BA"]]) %>%

slice(-20, -22) %>%

rbind(BA\_results[[21]][["imp"]][["BA"]]) %>%

slice(-21, -23) %>%

rbind(BA\_results[[22]][["imp"]][["BA"]]) %>%

slice(-22, -24) %>%

rbind(BA\_results[[23]][["imp"]][["BA"]]) %>%

slice(-23, -25) %>%

rbind(BA\_results[[24]][["imp"]][["BA"]]) %>%

slice(-24, -26) %>%

rbind(BA\_results[[25]][["imp"]][["BA"]]) %>%

slice(-25, -27) %>%

rbind(BA\_results[[26]][["imp"]][["BA"]]) %>%

slice(-26, -28) %>%

rbind(BA\_results[[27]][["imp"]][["BA"]]) %>%

slice(-27, -29) %>%

rbind(BA\_results[[28]][["imp"]][["BA"]]) %>%

slice(-28, -30) %>%

rbind(BA\_results[[29]][["imp"]][["BA"]]) %>%

slice(-29, -31) %>%

rbind(BA\_results[[30]][["imp"]][["BA"]]) %>%

slice(-30, -32) %>%

rbind(BA\_results[[31]][["imp"]][["BA"]]) %>%

slice(-31, -33) %>%

rbind(BA\_results[[32]][["imp"]][["BA"]]) %>%

slice(-32, -34) %>%

rbind(BA\_results[[33]][["imp"]][["BA"]]) %>%

slice(-33, -35) %>%

rbind(BA\_results[[34]][["imp"]][["BA"]]) %>%

slice(-34, -36) %>%

rbind(BA\_results[[35]][["imp"]][["BA"]]) %>%

slice(-35, -37) %>%

rbind(BA\_results[[36]][["imp"]][["BA"]]) %>%

slice(-36, -38) %>%

rbind(BA\_results[[37]][["imp"]][["BA"]]) %>%

slice(-37, -39) %>%

rbind(BA\_results[[38]][["imp"]][["BA"]]) %>%

slice(-38, -40) %>%

rbind(BA\_results[[39]][["imp"]][["BA"]]) %>%

slice(-39, -41) %>%

rbind(BA\_results[[40]][["imp"]][["BA"]]) %>%

slice(-40, -42) %>%

rbind(BA\_results[[41]][["imp"]][["BA"]]) %>%

slice(-41, -43) %>%

rbind(BA\_results[[42]][["imp"]][["BA"]]) %>%

slice(-42, -44) %>%

rbind(BA\_results[[43]][["imp"]][["BA"]]) %>%

slice(-43, -45) %>%

rbind(BA\_results[[45]][["imp"]][["BA"]]) %>%

slice(-44) %>%

rbind(BA\_results[[46]][["imp"]][["BA"]]) %>%

slice(-46, -47) %>%

rbind(BA\_results[[47]][["imp"]][["BA"]]) %>%

slice(-47, -48) %>%

rbind(BA\_results[[48]][["imp"]][["BA"]]) %>%

slice(-48, -49) %>%

rbind(BA\_results[[49]][["imp"]][["BA"]]) %>%

slice(-49, -50) %>%

rbind(BA\_results[[50]][["imp"]][["BA"]]) %>%

slice(-50, -51)

BA\_results\_df <- mutate(BA\_results\_df, mean = rowMeans(BA\_results\_df),

se = rowSds(as.matrix(BA\_results\_df)))

#Save results to computer as a .csv

write.csv(BA\_results\_df, "[filepath]/BA-MICE-SUBESTIMATE-Results.csv")

```

#\*\*NB\*\*: MICE subsestimates have been computed for HS, SBA and BA

#\*\*Free up memory\*\*

```{r, message=FALSE, warning=FALSE, eval=FALSE}

rm(list = ls())

```

### FH subestimates

#\*\*First figure out which combinations of predictors per subgroup of interest yield the largest r-squared statistic.\*\*

#Merge NAEP-reported estimates of mean math achievement and state-level predictors

```{r, message=FALSE, warning=FALSE}

NR\_TV\_Mean\_SE <- read\_csv("[filepath]/NR\_TV\_Mean\_SE.csv")

PREDICTORS\_DF <- read\_csv("[filepath]/PREDICTORS-DF.csv")

NR\_STATEPRED\_DF <- full\_join(NR\_TV\_Mean\_SE, PREDICTORS\_DF, by="State")

#Define variables as numeric (except for the first one)

NR\_STATEPRED\_DF[,2:30] <- apply(NR\_STATEPRED\_DF[,2:30], 2, as.numeric)

NR\_STATEPRED\_DF <- as.data.frame(NR\_STATEPRED\_DF)

```

#NHS

```{r, message=FALSE, warning=FALSE}

NHS\_model <- lm(NR\_NHS\_Mean ~ B\_H\_AINA + FER + p\_EL + SQI, data = NR\_STATEPRED\_DF)

ols\_step\_all\_possible(NHS\_model)

#select predictors: FER p\_EL

```

#HS

```{r, message=FALSE, warning=FALSE}

HS\_model <- lm(NR\_HS\_Mean ~ B\_H\_AINA + FER + p\_EL + SQI, data = NR\_STATEPRED\_DF)

ols\_step\_all\_possible(HS\_model)

#select predictors: B\_H\_AINA p\_EL SQI

```

#SBA

```{r, message=FALSE, warning=FALSE}

SBA\_model <- lm(NR\_SBA\_Mean ~ B\_H\_AINA + FER + p\_EL + SQI, data = NR\_STATEPRED\_DF)

ols\_step\_all\_possible(SBA\_model)

#select predictors: B\_H\_AINA FER p\_EL

```

#BA

```{r, message=FALSE, warning=FALSE}

BA\_model <- lm(NR\_BA\_Mean ~ B\_H\_AINA + FER + p\_EL + SQI, data = NR\_STATEPRED\_DF)

ols\_step\_all\_possible(BA\_model)

#select predictors: B\_H\_AINA FER p\_EL

```

#B

```{r, message=FALSE, warning=FALSE}

B\_model <- lm(NR\_B\_Mean ~ p\_BA + FER + SQI + AA, data = NR\_STATEPRED\_DF)

ols\_step\_all\_possible(B\_model)

#select predictors: p\_BA SQI AA

```

#H

```{r, message=FALSE, warning=FALSE}

H\_model <- lm(NR\_H\_Mean ~ p\_BA + FER + p\_EL + SQI + MX, data = NR\_STATEPRED\_DF)

ols\_step\_all\_possible(H\_model)

#select predictors: SQI MX

```

#API

```{r, message=FALSE, warning=FALSE}

API\_model <- lm(NR\_API\_Mean ~ p\_BA + FER + p\_EL + SQI + A, data = NR\_STATEPRED\_DF)

ols\_step\_all\_possible(API\_model)

#select predictors: p\_BA p\_EL SQI A

```

#AINA

```{r, message=FALSE, warning=FALSE}

AINA\_model <- lm(NR\_AINA\_Mean ~ p\_BA + FER, data = NR\_STATEPRED\_DF)

ols\_step\_all\_possible(AINA\_model)

#select predictors: FER

```

#TP

```{r, message=FALSE, warning=FALSE}

TP\_model <- lm(NR\_TP\_Mean ~ p\_BA + B\_H\_AINA + FER + SQI, data = NR\_STATEPRED\_DF)

ols\_step\_all\_possible(TP\_model)

#select predictors: p\_BA B\_H\_AINA FER SQI

```

#EL

```{r, message=FALSE, warning=FALSE}

EL\_model <- lm(NR\_EL\_Mean ~ B\_H\_AINA + FER + SQI, data = NR\_STATEPRED\_DF)

ols\_step\_all\_possible(EL\_model)

#select predictors: FER SQI

```

\*\*Free up memory\*\*

```{r warning=FALSE, message=FALSE}

rm(list = ls())

```

#\*\*Execute FH procedure\*\*

#First import the FH\_df data file that was previously used for the FH approach

```{r, message=FALSE, warning=FALSE}

FH\_df <- read\_csv("[filepath]/FH-DF.csv")

FH\_df <- FH\_df[,-1]

```

#NHS

```{r warning=FALSE, message=FALSE}

options(digits = 5)

FH\_procedure <- function(x){

FH\_df1 <- FH\_df

FH\_df1[x, 30] <- FH\_df1[x, 2] #30 corresponds with NR\_NHS\_Mean, 2 with NHS\_direct\_est

FH\_df1[x, 31] <- FH\_df1[x, 3] #31 corresponds with NR\_NHS\_SE, 3 with NHS\_se

FH\_df1 <- filter(FH\_df1, NR\_NHS\_Mean != "NA") #drops non target value rows

attach(FH\_df1)

mseFH(NR\_NHS\_Mean ~ FER + p\_EL, NR\_NHS\_SE^2) #line changes per subgroup

}

NHS\_FH\_results <- lapply(1:length(FH\_df$NR\_NHS\_Mean), FH\_procedure)

###########################################################################

# Extract the data of interest from NHS\_FH\_results (saved as a list object)

###########################################################################

#Set up function to automate extraction of eblups

get\_eblup <- function(x){

NHS\_FH\_results[[x]]$est$eblup[x]

}

#Automate with lapply

NHS\_eblups <- lapply(1:48, get\_eblup) #line changes per subgroup

#Wrangle eblups into data frame and long format

NHS\_eblups <- t(as.data.frame(NHS\_eblups))

rownames(NHS\_eblups) <- c()

NHS\_eblups <- as.data.frame(NHS\_eblups)

names(NHS\_eblups) <- c("NHS eblups")

########################################################################################

#Set up function to automate extraction of mse values for eblups

get\_mse <- function(x){

NHS\_FH\_results[[x]]$mse[x]

}

#Automate with lapply

NHS\_eblup\_se <- lapply(1:48, get\_mse) #line changes per subgroup

#Wrangle eblup MSEs into data frame and long format

NHS\_eblup\_se <- t(as.data.frame(NHS\_eblup\_se))

rownames(NHS\_eblup\_se) <- c()

NHS\_eblup\_se <- as.data.frame(NHS\_eblup\_se)

names(NHS\_eblup\_se) <- c("NHS\_eblup\_se")

#Bind eblup and mse values, along with their corresponding state acronyms (in first column)

NHS\_NR <-

read\_csv("[filepath]/NHS.csv") %>%

filter(Mean != "NA")

NHS\_FH\_results <- cbind(NHS\_NR$State, NHS\_eblups, NHS\_eblup\_se)

names(NHS\_FH\_results) <- c("State", "NHS\_eblups", "NHS\_eblup\_se")

#One last step to make sure the \_se variable actually includes standard errors (not mean squared errors)

NHS\_FH\_results$NHS\_eblup\_se <- sqrt(NHS\_FH\_results$NHS\_eblup\_se)

```

#HS

```{r warning=FALSE, message=FALSE}

FH\_procedure <- function(x){

FH\_df1 <- FH\_df

FH\_df1[x, 32] <- FH\_df1[x, 4] #32 corresponds with NR\_HS\_Mean, 4 with HS\_direct\_est

FH\_df1[x, 33] <- FH\_df1[x, 5] #33 corresponds with NR\_HS\_SE, 5 with HS\_se

FH\_df1 <- filter(FH\_df1, NR\_HS\_Mean != "NA") #drops non target value rows

attach(FH\_df1)

mseFH(NR\_HS\_Mean ~ B\_H\_AINA + FER + SQI, NR\_HS\_SE^2) #line changes per subgroup

}

HS\_FH\_results <- lapply(1:length(FH\_df$NR\_HS\_Mean), FH\_procedure)

###########################################################################

# Extract the data of interest from HS\_FH\_results (saved as a list object)

###########################################################################

#Set up function to automate extraction of eblups

get\_eblup <- function(x){

HS\_FH\_results[[x]]$est$eblup[x]

}

#Automate with lapply

HS\_eblups <- lapply(1:48, get\_eblup) #line changes per subgroup

#Wrangle eblups into data frame and long format

HS\_eblups <- t(as.data.frame(HS\_eblups))

rownames(HS\_eblups) <- c()

HS\_eblups <- as.data.frame(HS\_eblups)

names(HS\_eblups) <- c("HS eblups")

########################################################################################

#Set up function to automate extraction of mse values for eblups

get\_mse <- function(x){

HS\_FH\_results[[x]]$mse[x]

}

#Automate with lapply

HS\_eblup\_se <- lapply(1:48, get\_mse) #line changes per subgroup

#Wrangle eblup MSEs into data frame and long format

HS\_eblup\_se <- t(as.data.frame(HS\_eblup\_se))

rownames(HS\_eblup\_se) <- c()

HS\_eblup\_se <- as.data.frame(HS\_eblup\_se)

names(HS\_eblup\_se) <- c("HS\_eblup\_se")

#Bind eblup and mse values, along with their corresponding state acronyms (in first column)

HS\_NR <-

read\_csv("[filepath]/HS.csv") %>%

filter(Mean != "NA")

HS\_FH\_results <- cbind(HS\_NR$State, HS\_eblups, HS\_eblup\_se)

names(HS\_FH\_results) <- c("State", "HS\_eblups", "HS\_eblup\_se")

#One last step to make sure the \_se variable actually includes standard errors (not mean squared errors)

HS\_FH\_results$HS\_eblup\_se <- sqrt(HS\_FH\_results$HS\_eblup\_se)

```

#SBA

```{r warning=FALSE, message=FALSE}

FH\_procedure <- function(x){

FH\_df1 <- FH\_df

FH\_df1[x, 34] <- FH\_df1[x, 6] #34 corresponds with NR\_SBA\_Mean, 6 with SBA\_direct\_est

FH\_df1[x, 35] <- FH\_df1[x, 7] #35 corresponds with NR\_SBA\_SE, 7 with SBA\_se

FH\_df1 <- filter(FH\_df1, NR\_SBA\_Mean != "NA") #drops non target value rows

attach(FH\_df1)

mseFH(NR\_SBA\_Mean ~ B\_H\_AINA + FER + p\_EL, NR\_SBA\_SE^2) #line changes per subgroup

}

SBA\_FH\_results <- lapply(1:length(FH\_df$NR\_SBA\_Mean), FH\_procedure)

###########################################################################

# Extract the data of interest from SBA\_FH\_results (saved as a list object)

###########################################################################

#Set up function to automate extraction of eblups

get\_eblup <- function(x){

SBA\_FH\_results[[x]]$est$eblup[x]

}

#Automate with lapply

SBA\_eblups <- lapply(1:48, get\_eblup) #line changes per subgroup

#Wrangle eblups into data frame and long format

SBA\_eblups <- t(as.data.frame(SBA\_eblups))

rownames(SBA\_eblups) <- c()

SBA\_eblups <- as.data.frame(SBA\_eblups)

names(SBA\_eblups) <- c("SBA eblups")

########################################################################################

#Set up function to automate extraction of mse values for eblups

get\_mse <- function(x){

SBA\_FH\_results[[x]]$mse[x]

}

#Automate with lapply

SBA\_eblup\_se <- lapply(1:48, get\_mse) #line changes per subgroup

#Wrangle eblup MSEs into data frame and long format

SBA\_eblup\_se <- t(as.data.frame(SBA\_eblup\_se))

rownames(SBA\_eblup\_se) <- c()

SBA\_eblup\_se <- as.data.frame(SBA\_eblup\_se)

names(SBA\_eblup\_se) <- c("SBA\_eblup\_se")

#Bind eblup and mse values, along with their corresponding state acronyms (in first column)

SBA\_NR <-

read\_csv("[filepath]/SBA.csv") %>%

filter(Mean != "NA")

SBA\_FH\_results <- cbind(SBA\_NR$State, SBA\_eblups, SBA\_eblup\_se)

names(SBA\_FH\_results) <- c("State", "SBA\_eblups", "SBA\_eblup\_se")

#One last step to make sure the \_se variable actually includes standard errors (not mean squared errors)

SBA\_FH\_results$SBA\_eblup\_se <- sqrt(SBA\_FH\_results$SBA\_eblup\_se)

```

#BA

```{r warning=FALSE, message=FALSE}

FH\_procedure <- function(x){

FH\_df1 <- FH\_df

FH\_df1[x, 36] <- FH\_df1[x, 8] #36 corresponds with NR\_BA\_Mean, 8 with BA\_direct\_est

FH\_df1[x, 37] <- FH\_df1[x, 9] #37 corresponds with NR\_BA\_SE, 9 with BA\_se

FH\_df1 <- filter(FH\_df1, NR\_BA\_Mean != "NA") #drops non target value rows

attach(FH\_df1)

mseFH(NR\_BA\_Mean ~ B\_H\_AINA + FER + p\_EL, NR\_BA\_SE^2) #line changes per subgroup

}

BA\_FH\_results <- lapply(1:length(FH\_df$NR\_BA\_Mean), FH\_procedure)

###########################################################################

# Extract the data of interest from BA\_FH\_results (saved as a list object)

###########################################################################

#Set up function to automate extraction of eblups

get\_eblup <- function(x){

BA\_FH\_results[[x]]$est$eblup[x]

}

#Automate with lapply

BA\_eblups <- lapply(1:48, get\_eblup) #line changes per subgroup

#Wrangle eblups into data frame and long format

BA\_eblups <- t(as.data.frame(BA\_eblups))

rownames(BA\_eblups) <- c()

BA\_eblups <- as.data.frame(BA\_eblups)

names(BA\_eblups) <- c("BA eblups")

########################################################################################

#Set up function to automate extraction of mse values for eblups

get\_mse <- function(x){

BA\_FH\_results[[x]]$mse[x]

}

#Automate with lapply

BA\_eblup\_se <- lapply(1:48, get\_mse) #line changes per subgroup

#Wrangle eblup MSEs into data frame and long format

BA\_eblup\_se <- t(as.data.frame(BA\_eblup\_se))

rownames(BA\_eblup\_se) <- c()

BA\_eblup\_se <- as.data.frame(BA\_eblup\_se)

names(BA\_eblup\_se) <- c("BA\_eblup\_se")

#Bind eblup and mse values, along with their corresponding state acronyms (in first column)

BA\_NR <-

read\_csv("[filepath]/BA.csv") %>%

filter(Mean != "NA")

BA\_FH\_results <- cbind(BA\_NR$State, BA\_eblups, BA\_eblup\_se)

names(BA\_FH\_results) <- c("State", "BA\_eblups", "BA\_eblup\_se")

#One last step to make sure the \_se variable actually includes standard errors (not mean squared errors)

BA\_FH\_results$BA\_eblup\_se <- sqrt(BA\_FH\_results$BA\_eblup\_se)

```

#B

```{r warning=FALSE, message=FALSE}

FH\_procedure <- function(x){

FH\_df1 <- FH\_df

FH\_df1[x, 38] <- FH\_df1[x, 10] #38 corresponds with NR\_B\_Mean, 10 with B\_direct\_est

FH\_df1[x, 39] <- FH\_df1[x, 11] #39 corresponds with NR\_B\_SE, 11 with B\_se

FH\_df1 <- filter(FH\_df1, NR\_B\_Mean != "NA") #drops non target value rows

attach(FH\_df1)

mseFH(NR\_B\_Mean ~ p\_BA + SQI + AA, NR\_B\_SE^2) #line changes per subgroup

}

B\_FH\_results <- lapply(1:length(FH\_df$NR\_B\_Mean), FH\_procedure)

###########################################################################

# Extract the data of interest from B\_FH\_results (saved as a list object)

###########################################################################

#Set up function to automate extraction of eblups

get\_eblup <- function(x){

B\_FH\_results[[x]]$est$eblup[x]

}

#Automate with lapply

B\_eblups <- lapply(1:39, get\_eblup) #line changes per subgroup

#Wrangle eblups into data frame and long format

B\_eblups <- t(as.data.frame(B\_eblups))

rownames(B\_eblups) <- c()

B\_eblups <- as.data.frame(B\_eblups)

names(B\_eblups) <- c("B eblups")

########################################################################################

#Set up function to automate extraction of mse values for eblups

get\_mse <- function(x){

B\_FH\_results[[x]]$mse[x]

}

#Automate with lapply

B\_eblup\_se <- lapply(1:39, get\_mse) #line changes per subgroup

#Wrangle eblup MSEs into data frame and long format

B\_eblup\_se <- t(as.data.frame(B\_eblup\_se))

rownames(B\_eblup\_se) <- c()

B\_eblup\_se <- as.data.frame(B\_eblup\_se)

names(B\_eblup\_se) <- c("B\_eblup\_se")

#Bind eblup and mse values, along with their corresponding state acronyms (in first column)

B\_NR <-

read\_csv("[filepath]/B.csv") %>%

filter(Mean != "NA")

B\_FH\_results <- cbind(B\_NR$State, B\_eblups, B\_eblup\_se)

names(B\_FH\_results) <- c("State", "B\_eblups", "B\_eblup\_se")

#One last step to make sure the \_se variable actually includes standard errors (not mean squared errors)

B\_FH\_results$B\_eblup\_se <- sqrt(B\_FH\_results$B\_eblup\_se)

```

#H

```{r warning=FALSE, message=FALSE}

FH\_procedure <- function(x){

FH\_df1 <- FH\_df

FH\_df1[x, 40] <- FH\_df1[x, 12] #40 corresponds with NR\_H\_Mean, 12 with H\_direct\_est

FH\_df1[x, 41] <- FH\_df1[x, 13] #41 corresponds with NR\_H\_SE, 13 with H\_se

FH\_df1 <- filter(FH\_df1, NR\_H\_Mean != "NA") #drops non target value rows

attach(FH\_df1)

mseFH(NR\_H\_Mean ~ SQI + MX, NR\_H\_SE^2) #line changes per subgroup

}

H\_FH\_results <- lapply(1:length(FH\_df$NR\_H\_Mean), FH\_procedure)

###########################################################################

# Extract the data of interest from H\_FH\_results (saved as a list object)

###########################################################################

#Set up function to automate extraction of eblups

get\_eblup <- function(x){

H\_FH\_results[[x]]$est$eblup[x]

}

#Automate with lapply

H\_eblups <- lapply(1:47, get\_eblup) #line changes per subgroup

#Wrangle eblups into data frame and long format

H\_eblups <- t(as.data.frame(H\_eblups))

rownames(H\_eblups) <- c()

H\_eblups <- as.data.frame(H\_eblups)

names(H\_eblups) <- c("H eblups")

########################################################################################

#Set up function to automate extraction of mse values for eblups

get\_mse <- function(x){

H\_FH\_results[[x]]$mse[x]

}

#Automate with lapply

H\_eblup\_se <- lapply(1:47, get\_mse) #line changes per subgroup

#Wrangle eblup MSEs into data frame and long format

H\_eblup\_se <- t(as.data.frame(H\_eblup\_se))

rownames(H\_eblup\_se) <- c()

H\_eblup\_se <- as.data.frame(H\_eblup\_se)

names(H\_eblup\_se) <- c("H\_eblup\_se")

#Bind eblup and mse values, along with their corresponding state acronyms (in first column)

H\_NR <-

read\_csv("[filepath]/H.csv") %>%

filter(Mean != "NA")

H\_FH\_results <- cbind(H\_NR$State, H\_eblups, H\_eblup\_se)

names(H\_FH\_results) <- c("State", "H\_eblups", "H\_eblup\_se")

#One last step to make sure the \_se variable actually includes standard errors (not mean squared errors)

H\_FH\_results$H\_eblup\_se <- sqrt(H\_FH\_results$H\_eblup\_se)

```

#API

```{r warning=FALSE, message=FALSE}

FH\_procedure <- function(x){

FH\_df1 <- FH\_df

FH\_df1[x, 42] <- FH\_df1[x, 14] #42 corresponds with NR\_API\_Mean, 14 with API\_direct\_est

FH\_df1[x, 43] <- FH\_df1[x, 15] #43 corresponds with NR\_API\_SE, 15 with API\_se

FH\_df1 <- filter(FH\_df1, NR\_API\_Mean != "NA") #drops non target value rows

attach(FH\_df1)

mseFH(NR\_API\_Mean ~ p\_BA + p\_EL + SQI + A, NR\_API\_SE^2) #line changes per subgroup

}

API\_FH\_results <- lapply(1:length(FH\_df$NR\_API\_Mean), FH\_procedure)

###########################################################################

# Extract the data of interest from API\_FH\_results (saved as a list object)

###########################################################################

#Set up function to automate extraction of eblups

get\_eblup <- function(x){

API\_FH\_results[[x]]$est$eblup[x]

}

#Automate with lapply

API\_eblups <- lapply(1:30, get\_eblup) #line changes per subgroup

#Wrangle eblups into data frame and long format

API\_eblups <- t(as.data.frame(API\_eblups))

rownames(API\_eblups) <- c()

API\_eblups <- as.data.frame(API\_eblups)

names(API\_eblups) <- c("API eblups")

########################################################################################

#Set up function to automate extraction of mse values for eblups

get\_mse <- function(x){

API\_FH\_results[[x]]$mse[x]

}

#Automate with lapply

API\_eblup\_se <- lapply(1:30, get\_mse) #line changes per subgroup

#Wrangle eblup MSEs into data frame and long format

API\_eblup\_se <- t(as.data.frame(API\_eblup\_se))

rownames(API\_eblup\_se) <- c()

API\_eblup\_se <- as.data.frame(API\_eblup\_se)

names(API\_eblup\_se) <- c("API\_eblup\_se")

#Bind eblup and mse values, along with their corresponding state acronyms (in first column)

API\_NR <-

read\_csv("[filepath]/API.csv") %>%

filter(Mean != "NA")

API\_FH\_results <- cbind(API\_NR$State, API\_eblups, API\_eblup\_se)

names(API\_FH\_results) <- c("State", "API\_eblups", "API\_eblup\_se")

#One last step to make sure the \_se variable actually includes standard errors (not mean squared errors)

API\_FH\_results$API\_eblup\_se <- sqrt(API\_FH\_results$API\_eblup\_se)

```

#AINA

```{r warning=FALSE, message=FALSE}

FH\_procedure <- function(x){

FH\_df1 <- FH\_df

FH\_df1[x, 44] <- FH\_df1[x, 16] #44 corresponds with NR\_AINA\_Mean, 16 with AINA\_direct\_est

FH\_df1[x, 45] <- FH\_df1[x, 17] #45 corresponds with NR\_AINA\_SE, 17 with AINA\_se

FH\_df1 <- filter(FH\_df1, NR\_AINA\_Mean != "NA") #drops non target value rows

attach(FH\_df1)

mseFH(NR\_AINA\_Mean ~ FER, NR\_AINA\_SE^2) #line changes per subgroup

}

AINA\_FH\_results <- lapply(1:length(FH\_df$NR\_AINA\_Mean), FH\_procedure)

###########################################################################

# Extract the data of interest from AINA\_FH\_results (saved as a list object)

###########################################################################

#Set up function to automate extraction of eblups

get\_eblup <- function(x){

AINA\_FH\_results[[x]]$est$eblup[x]

}

#Automate with lapply

AINA\_eblups <- lapply(1:13, get\_eblup) #line changes per subgroup

#Wrangle eblups into data frame and long format

AINA\_eblups <- t(as.data.frame(AINA\_eblups))

rownames(AINA\_eblups) <- c()

AINA\_eblups <- as.data.frame(AINA\_eblups)

names(AINA\_eblups) <- c("AINA eblups")

########################################################################################

#Set up function to automate extraction of mse values for eblups

get\_mse <- function(x){

AINA\_FH\_results[[x]]$mse[x]

}

#Automate with lapply

AINA\_eblup\_se <- lapply(1:13, get\_mse) #line changes per subgroup

#Wrangle eblup MSEs into data frame and long format

AINA\_eblup\_se <- t(as.data.frame(AINA\_eblup\_se))

rownames(AINA\_eblup\_se) <- c()

AINA\_eblup\_se <- as.data.frame(AINA\_eblup\_se)

names(AINA\_eblup\_se) <- c("AINA\_eblup\_se")

#Bind eblup and mse values, along with their corresponding state acronyms (in first column)

AINA\_NR <-

read\_csv("[filepath]/AINA.csv") %>%

filter(Mean != "NA")

AINA\_FH\_results <- cbind(AINA\_NR$State, AINA\_eblups, AINA\_eblup\_se)

names(AINA\_FH\_results) <- c("State", "AINA\_eblups", "AINA\_eblup\_se")

#One last step to make sure the \_se variable actually includes standard errors (not mean squared errors)

AINA\_FH\_results$AINA\_eblup\_se <- sqrt(AINA\_FH\_results$AINA\_eblup\_se)

```

#TP

```{r warning=FALSE, message=FALSE}

FH\_procedure <- function(x){

FH\_df1 <- FH\_df

FH\_df1[x, 46] <- FH\_df1[x, 18] #46 corresponds with NR\_TP\_Mean, 18 with TP\_direct\_est

FH\_df1[x, 47] <- FH\_df1[x, 19] #47 corresponds with NR\_TP\_SE, 19 with TP\_se

FH\_df1 <- filter(FH\_df1, NR\_TP\_Mean != "NA") #drops non target value rows

attach(FH\_df1)

mseFH(NR\_TP\_Mean ~ p\_BA + B\_H\_AINA + FER + SQI, NR\_TP\_SE^2) #line changes per subgroup

}

TP\_FH\_results <- lapply(1:length(FH\_df$NR\_TP\_Mean), FH\_procedure)

###########################################################################

# Extract the data of interest from TP\_FH\_results (saved as a list object)

###########################################################################

#Set up function to automate extraction of eblups

get\_eblup <- function(x){

TP\_FH\_results[[x]]$est$eblup[x]

}

#Automate with lapply

TP\_eblups <- lapply(1:24, get\_eblup) #line changes per subgroup

#Wrangle eblups into data frame and long format

TP\_eblups <- t(as.data.frame(TP\_eblups))

rownames(TP\_eblups) <- c()

TP\_eblups <- as.data.frame(TP\_eblups)

names(TP\_eblups) <- c("TP eblups")

########################################################################################

#Set up function to automate extraction of mse values for eblups

get\_mse <- function(x){

TP\_FH\_results[[x]]$mse[x]

}

#Automate with lapply

TP\_eblup\_se <- lapply(1:24, get\_mse) #line changes per subgroup

#Wrangle eblup MSEs into data frame and long format

TP\_eblup\_se <- t(as.data.frame(TP\_eblup\_se))

rownames(TP\_eblup\_se) <- c()

TP\_eblup\_se <- as.data.frame(TP\_eblup\_se)

names(TP\_eblup\_se) <- c("TP\_eblup\_se")

#Bind eblup and mse values, along with their corresponding state acronyms (in first column)

TP\_NR <-

read\_csv("[filepath]/2+.csv") %>%

filter(Mean != "NA")

TP\_FH\_results <- cbind(TP\_NR$State, TP\_eblups, TP\_eblup\_se)

names(TP\_FH\_results) <- c("State", "TP\_eblups", "TP\_eblup\_se")

#One last step to make sure the \_se variable actually includes standard errors (not mean squared errors)

TP\_FH\_results$TP\_eblup\_se <- sqrt(TP\_FH\_results$TP\_eblup\_se)

```

#EL

```{r warning=FALSE, message=FALSE}

FH\_procedure <- function(x){

FH\_df1 <- FH\_df

FH\_df1[x, 48] <- FH\_df1[x, 20] #48 corresponds with NR\_EL\_Mean, 20 with EL\_direct\_est

FH\_df1[x, 49] <- FH\_df1[x, 21] #49 corresponds with NR\_EL\_SE, 21 with EL\_se

FH\_df1 <- filter(FH\_df1, NR\_EL\_Mean != "NA") #drops non target value rows

attach(FH\_df1)

mseFH(NR\_EL\_Mean ~ FER + SQI, NR\_EL\_SE^2) #line changes per subgroup

}

EL\_FH\_results <- lapply(1:length(FH\_df$NR\_EL\_Mean), FH\_procedure)

###########################################################################

# Extract the data of interest from EL\_FH\_results (saved as a list object)

###########################################################################

#Set up function to automate extraction of eblups

get\_eblup <- function(x){

EL\_FH\_results[[x]]$est$eblup[x]

}

#Automate with lapply

EL\_eblups <- lapply(1:31, get\_eblup) #line changes per subgroup

#Wrangle eblups into data frame and long format

EL\_eblups <- t(as.data.frame(EL\_eblups))

rownames(EL\_eblups) <- c()

EL\_eblups <- as.data.frame(EL\_eblups)

names(EL\_eblups) <- c("EL eblups")

########################################################################################

#Set up function to automate extraction of mse values for eblups

get\_mse <- function(x){

EL\_FH\_results[[x]]$mse[x]

}

#Automate with lapply

EL\_eblup\_se <- lapply(1:31, get\_mse) #line changes per subgroup

#Wrangle eblup MSEs into data frame and long format

EL\_eblup\_se <- t(as.data.frame(EL\_eblup\_se))

rownames(EL\_eblup\_se) <- c()

EL\_eblup\_se <- as.data.frame(EL\_eblup\_se)

names(EL\_eblup\_se) <- c("EL\_eblup\_se")

#Bind eblup and mse values, along with their corresponding state acronyms (in first column)

EL\_NR <-

read\_csv("[filepath]/EL.csv") %>%

filter(Mean != "NA")

EL\_FH\_results <- cbind(EL\_NR$State, EL\_eblups, EL\_eblup\_se)

names(EL\_FH\_results) <- c("State", "EL\_eblups", "EL\_eblup\_se")

#One last step to make sure the \_se variable actually includes standard errors (not mean squared errors)

EL\_FH\_results$EL\_eblup\_se <- sqrt(EL\_FH\_results$EL\_eblup\_se)

```

\*\*Export FH subestimates\*\*

```{r warning=FALSE, message=FALSE}

#NHS

write.csv(NHS\_FH\_results, "[filepath]/NHS-FH-SUBESTIMATES.csv")

#HS

write.csv(HS\_FH\_results, "[filepath]/HS-FH-SUBESTIMATES.csv")

#SBA

write.csv(SBA\_FH\_results, "[filepath]/SBA-FH-SUBESTIMATES.csv")

#BA

write.csv(BA\_FH\_results, "[filepath]/BA-FH-SUBESTIMATES.csv")

#B

write.csv(B\_FH\_results, "[filepath]/B-FH-SUBESTIMATES.csv")

#H

write.csv(H\_FH\_results, "[filepath]/H-FH-SUBESTIMATES.csv")

#API

write.csv(API\_FH\_results, "[filepath]/API-FH-SUBESTIMATES.csv")

#AINA

write.csv(AINA\_FH\_results, "[filepath]/AINA-FH-SUBESTIMATES.csv")

#TP

write.csv(TP\_FH\_results, "[filepath]/TP-FH-SUBESTIMATES.csv")

#EL

write.csv(EL\_FH\_results, "[filepath]/EL-FH-SUBESTIMATES.csv")

```

### WPE subestimates

#Import and wrangle data

```{r, message=FALSE, warning=FALSE, eval=FALSE}

#Import data

SEDA\_geodist\_long\_NAEP\_v21 <- read\_csv("[filepath]/SEDA\_geodist\_long\_NAEP\_v21")

#rename to simplify

df <- SEDA\_geodist\_long\_NAEP\_v21

#subset

df <- df %>%

filter(grade == 8) %>%

filter(year == 2015) %>%

filter(subject == "math") %>%

filter(stateabb != "DC")

df <- dplyr::select(df, leaname, stateabb, totgyb\_all, mn\_all, sd\_all, mn\_all\_se, sd\_all\_se, totgyb\_asn, mn\_asn, sd\_asn, mn\_asn\_se, sd\_asn\_se, totgyb\_blk, mn\_blk, sd\_blk, mn\_blk\_se,

sd\_blk\_se, totgyb\_hsp, mn\_hsp, sd\_hsp, mn\_hsp\_se, sd\_hsp\_se)

#compute total sums into a tibble

df\_sums <- df %>%

group\_by(stateabb) %>%

summarize(state\_total\_sum = sum(totgyb\_all, na.rm = T))

#NB: There are 35 states that have data for this particular "gyb" (grade8-2015-math)

#merge in the sums for each state

df <- full\_join(df, df\_sums, by = "stateabb")

#create total weights

df <- df %>%

mutate(total\_weight = totgyb\_all/state\_total\_sum)

```

#"asn"

```{r, message=FALSE, warning=FALSE, eval=FALSE}

#compute total asn sums into a tibble

df\_asn\_sums <- df %>%

group\_by(stateabb) %>%

summarize(state\_asn\_sum = sum(totgyb\_asn, na.rm = T))

#merge in the sums for each state

df <- full\_join(df, df\_asn\_sums, by = "stateabb")

#create asn weights

df <- df %>%

mutate(asn\_weight = totgyb\_asn/state\_asn\_sum)

```

#"blk"

```{r, message=FALSE, warning=FALSE, eval=FALSE}

#compute total blk sums into a tibble

df\_blk\_sums <- df %>%

group\_by(stateabb) %>%

summarize(state\_blk\_sum = sum(totgyb\_blk, na.rm = T))

#merge in the sums for each state

df <- full\_join(df, df\_blk\_sums, by = "stateabb")

#create blk weights

df <- df %>%

mutate(blk\_weight = totgyb\_blk/state\_blk\_sum)

```

#"hsp"

```{r, message=FALSE, warning=FALSE, eval=FALSE}

#compute total hsp sums into a tibble

df\_hsp\_sums <- df %>%

group\_by(stateabb) %>%

summarize(state\_hsp\_sum = sum(totgyb\_hsp, na.rm = T))

#merge in the sums for each state

df <- full\_join(df, df\_hsp\_sums, by = "stateabb")

#create hsp weights

df <- df %>%

mutate(hsp\_weight = totgyb\_hsp/state\_hsp\_sum)

```

#\*\*Generate WPE means\*\*

```{r, message=FALSE, warning=FALSE, eval=FALSE}

#Create variables holding "contributions" of each strata (district)

df <- df %>%

mutate(tot\_con = mn\_all\*total\_weight, asn\_con = mn\_asn\*asn\_weight,

blk\_con = mn\_blk\*blk\_weight, hsp\_con = mn\_hsp\*hsp\_weight)

#Create tibble for WPE by state across subgroups

WPE\_means <- df %>%

group\_by(stateabb) %>%

summarize(tot\_wpe = sum(tot\_con, na.rm = T),

asn\_wpe = sum(asn\_con, na.rm = T),

blk\_wpe = sum(blk\_con, na.rm = T),

hsp\_wpe = sum(hsp\_con, na.rm = T))

```

#\*\*Generate WPE variances\*\*

```{r, message=FALSE, warning=FALSE, eval=FALSE}

#Create/Add variables to data frame that include the mean variance of achievement for each district across subgroups

df <- df %>%

mutate(tot\_mn\_var = mn\_all\_se^2,

asn\_mn\_var = mn\_asn\_se^2,

blk\_mn\_var = mn\_blk\_se^2,

hsp\_mn\_var = mn\_hsp\_se^2)

#Create/Add "n-1" variables for each strata (district) across subgroups

df <- df %>%

mutate(tot\_n1 = totgyb\_all-1,

asn\_n1 = totgyb\_asn-1,

blk\_n1 = totgyb\_blk-1,

hsp\_n1 = totgyb\_hsp-1)

#Create tibble for WPE variance by state across subgroups

WPE\_variances <- df %>%

group\_by(stateabb) %>%

summarize(tot\_var = sum(tot\_n1\*tot\_mn\_var, na.rm = T)/sum(tot\_n1, na.rm = T),

asn\_var = sum(asn\_n1\*asn\_mn\_var, na.rm = T)/sum(asn\_n1, na.rm = T),

blk\_var = sum(blk\_n1\*blk\_mn\_var, na.rm = T)/sum(blk\_n1, na.rm = T),

hsp\_var = sum(hsp\_n1\*hsp\_mn\_var, na.rm = T)/sum(hsp\_n1, na.rm = T))

```

#\*\*Merge WPE Means and Variances.\*\*

```{r, message=FALSE, warning=FALSE, eval=FALSE}

WPE\_subestimates <- full\_join(WPE\_means, WPE\_variances, by = "stateabb")

WPE\_subestimates <- dplyr::select(WPE\_subestimates, -"tot\_wpe", -"tot\_var")

WPE\_subestimates <- WPE\_subestimates %>%

rename(State = stateabb)

```

#\*\*Export WPE subestimates\*\*

```{r, message=FALSE, warning=FALSE, eval=FALSE}

write.csv(WPE\_subestimates, "[filepath]/WPE-SUBESTIMATES.csv")

```

### NNI subestimates

#The Euclidean distance between states is equal to the square root of the sum of squared differences between states’ values on measures of parental level of education (p\_BA), family economic resources (FER), race and ethnicity (B\_H\_AINA) and school quality (SQI). To limit the undue influence of the scale on which the variables’ values are measured, each of the data variables are standardized with a mean of 0 and a standard deviation of 1 so that the distribution of values for each variable are on the same scale.

```{r, message=FALSE, warning=FALSE, eval=FALSE}

#Import predictors

PREDICTORS\_DF <- read\_csv("[filepath]/PREDICTORS-DF.csv")

#Reduce dataset to B\_H\_AINA, FER, p\_BA, and SQI

NNI\_df <- dplyr::select(PREDICTORS\_DF, B\_H\_AINA, FER, p\_BA, SQI)

#Standardize variable values

NNI\_df <- apply(NNI\_df, 2, scale)

#Make sure NNI\_df is a data frame

NNI\_df <- as.data.frame(NNI\_df)

#Compute distance matrix

dist\_df <- dist(NNI\_df)

#define as data frame

dist\_df <- as.matrix(dist\_df)

dist\_df <- as.data.frame(dist\_df)

#TURNS OUT THERE ARE NO SIBLING STATES (distances of .20 or less)

#Doubling the threshold to less than or equal to .40 produces 5 pairs of "siblings." So 10 (one in five) states would have a sibling and thus an NNI subestimate. Seems like a reasonable proportion from my perspective. And HB (Advisor) agrees.

#Alabama & Oklahoma (0.25)

#Pennsylvania & Wisconsin (0.25)

#Michigan & Missouri (0.28)

#Connecticut & New Jersey (0.36)

#Kansas & Nebraska (0.36)

#Iowa & North Dakota (0.39)

```

#Import & Wrangle NAEP reported estimates

```{r, message=FALSE, warning=FALSE, eval=FALSE}

NR\_TV\_Mean\_SE <- read\_csv("C[filepath]/NR\_TV\_Mean\_SE.csv")

#Filter on states that have a sibling (according to 0.40 threshold)

siblings <- c("AL", "OK", "PA", "WI", "MI", "MO", "CT", "NJ", "KS", "NE", "IA", "ND")

siblings\_df <- NR\_TV\_Mean\_SE %>%

filter(State %in% siblings)

```

#Perform NNI substitutions (swaps)

```{r, message=FALSE, warning=FALSE, eval=FALSE}

#AL/OK (rows 1 and 10)

siblings\_df[1, 1] <- "OK"

siblings\_df[10, 1] <- "AL"

#PA/WI (rows 11 and 12)

siblings\_df[11, 1] <- "WI"

siblings\_df[12, 1] <- "PA"

#MI/MO (rows 5 and 6)

siblings\_df[5, 1] <- "MO"

siblings\_df[6, 1] <- "MI"

#CT/NJ (rows 2 and 8)

siblings\_df[2, 1] <- "NJ"

siblings\_df[8, 1] <- "CT"

#KS/NE (rows 4 and 7)

siblings\_df[4, 1] <- "NE"

siblings\_df[7, 1] <- "KS"

#IA/ND (rows 3 and 9)

siblings\_df[3, 1] <- "ND"

siblings\_df[9, 1] <- "IA"

```

#\*\*Export NNI subestimates\*\*

```{r, message=FALSE, warning=FALSE, eval=FALSE}

write.csv(siblings\_df, "[filepath]/NNI-SUBESTIMATES.csv")

```

### Wrangle final predicted values and their variance estimates into one data file

\*\*Free up memory\*\*

```{r warning=FALSE, message=FALSE}

rm(list = ls())

```

\*\*Import and merge subestimates\*\*

#NHS (Subsestimates: FH, NNI)

```{r warning=FALSE, message=FALSE}

#FH

NHS\_FH\_subs <- read\_csv("[filepath]/NHS-FH-SUBESTIMATES.csv") %>%

dplyr::select("State", "NHS\_eblups", "NHS\_eblup\_se")

#NNI

NHS\_NNI\_subs <- read\_csv("[filepath]/NNI-SUBESTIMATES.csv") %>%

dplyr::select("State", "NR\_NHS\_Mean", "NR\_NHS\_SE")

#merge with #full\_join()

NHS\_FLEXCS\_DF <- full\_join(NHS\_FH\_subs, NHS\_NNI\_subs, by = "State")

```

#HS (Subsestimates: MICE, FH, NNI)

```{r warning=FALSE, message=FALSE}

#First save abbreviations to an object called "State"

State <- read\_csv("[filepath]/2+.csv")

State <- State[,1]

#MICE

HS\_MICE\_subs <- read\_csv("[filepath]/HS-MICE-SUBESTIMATE-Results.csv")

HS\_MICE\_subs <- cbind(State, HS\_MICE\_subs)

HS\_MICE\_subs <- HS\_MICE\_subs %>%

dplyr::select("State", "mean", "se")

HS\_MICE\_subs[2,2:3] <- NA

HS\_MICE\_subs[44,2:3] <- NA

#FH

HS\_FH\_subs <- read\_csv("[filepath]/HS-FH-SUBESTIMATES.csv") %>%

dplyr::select("State", "HS\_eblups", "HS\_eblup\_se")

#NNI

HS\_NNI\_subs <- read\_csv("[filepath]/NNI-SUBESTIMATES.csv") %>%

dplyr::select("State", "NR\_HS\_Mean", "NR\_HS\_SE")

#merge with #full\_join()

HS\_FLEXCS\_DF <- full\_join(HS\_MICE\_subs, HS\_FH\_subs, by = "State") %>%

full\_join(HS\_NNI\_subs, by = "State")

```

#SBA (Subsestimates: MICE, FH, NNI)

```{r warning=FALSE, message=FALSE}

#MICE

SBA\_MICE\_subs <- read\_csv("[filepath]/SBA-MICE-SUBESTIMATE-Results.csv")

SBA\_MICE\_subs <- cbind(State, SBA\_MICE\_subs)

SBA\_MICE\_subs <- SBA\_MICE\_subs %>%

dplyr::select("State", "mean", "se")

SBA\_MICE\_subs[2,2:3] <- NA

SBA\_MICE\_subs[44,2:3] <- NA

#FH

SBA\_FH\_subs <- read\_csv("[filepath]/SBA-FH-SUBESTIMATES.csv") %>%

dplyr::select("State", "SBA\_eblups", "SBA\_eblup\_se")

#NNI

SBA\_NNI\_subs <- read\_csv("[filepath]/NNI-SUBESTIMATES.csv") %>%

dplyr::select("State", "NR\_SBA\_Mean", "NR\_SBA\_SE")

#merge with #full\_join()

SBA\_FLEXCS\_DF <- full\_join(SBA\_MICE\_subs, SBA\_FH\_subs, by = "State") %>%

full\_join(SBA\_NNI\_subs, by = "State")

```

#BA (Subsestimates: MICE, FH, NNI)

```{r warning=FALSE, message=FALSE}

#MICE

BA\_MICE\_subs <- read\_csv("[filepath]/BA-MICE-SUBESTIMATE-Results.csv")

BA\_MICE\_subs <- cbind(State, BA\_MICE\_subs)

BA\_MICE\_subs <- BA\_MICE\_subs %>%

dplyr::select("State", "mean", "se")

BA\_MICE\_subs[2,2:3] <- NA

BA\_MICE\_subs[44,2:3] <- NA

#FH

BA\_FH\_subs <- read\_csv("[filepath]/BA-FH-SUBESTIMATES.csv") %>%

dplyr::select("State", "BA\_eblups", "BA\_eblup\_se")

#NNI

BA\_NNI\_subs <- read\_csv("[filepath]/NNI-SUBESTIMATES.csv") %>%

dplyr::select("State", "NR\_BA\_Mean", "NR\_BA\_SE")

#merge with #full\_join()

BA\_FLEXCS\_DF <- full\_join(BA\_MICE\_subs, BA\_FH\_subs, by = "State") %>%

full\_join(BA\_NNI\_subs, by = "State")

```

#B (Subsestimates: FH, WPE, NNI)

```{r warning=FALSE, message=FALSE}

#FH

B\_FH\_subs <- read\_csv("[filepath]/B-FH-SUBESTIMATES.csv") %>%

dplyr::select("State", "B\_eblups", "B\_eblup\_se")

#WPE

B\_WPE\_subs <- read\_csv("[filepath]/WPE-SUBESTIMATES.csv") %>%

dplyr::select("State", "blk\_wpe", "blk\_var")

#NNI

B\_NNI\_subs <- read\_csv("[filepath]/NNI-SUBESTIMATES.csv") %>%

dplyr::select("State", "NR\_B\_Mean", "NR\_B\_SE")

#merge with #full\_join()

B\_FLEXCS\_DF <- full\_join(B\_FH\_subs, B\_WPE\_subs, by = "State") %>%

full\_join(B\_NNI\_subs, by = "State")

```

#H (Subsestimates: FH, WPE, NNI)

```{r warning=FALSE, message=FALSE}

#FH

H\_FH\_subs <- read\_csv("[filepath]/H-FH-SUBESTIMATES.csv") %>%

dplyr::select("State", "H\_eblups", "H\_eblup\_se")

#WPE

H\_WPE\_subs <- read\_csv("[filepath]/WPE-SUBESTIMATES.csv") %>%

dplyr::select("State", "hsp\_wpe", "hsp\_var")

#NNI

H\_NNI\_subs <- read\_csv("[filepath]/NNI-SUBESTIMATES.csv") %>%

dplyr::select("State", "NR\_H\_Mean", "NR\_H\_SE")

#merge with #full\_join()

H\_FLEXCS\_DF <- full\_join(H\_FH\_subs, H\_WPE\_subs, by = "State") %>%

full\_join(H\_NNI\_subs, by = "State")

```

#API (Subsestimates: FH, WPE, NNI)

```{r warning=FALSE, message=FALSE}

#FH

API\_FH\_subs <- read\_csv("[filepath]/API-FH-SUBESTIMATES.csv") %>%

dplyr::select("State", "API\_eblups", "API\_eblup\_se")

#WPE

API\_WPE\_subs <- read\_csv("[filepath]/WPE-SUBESTIMATES.csv") %>%

dplyr::select("State", "asn\_wpe", "asn\_var")

#NNI

API\_NNI\_subs <- read\_csv("[filepath]/NNI-SUBESTIMATES.csv") %>%

dplyr::select("State", "NR\_API\_Mean", "NR\_API\_SE")

#merge with #full\_join()

API\_FLEXCS\_DF <- full\_join(API\_FH\_subs, API\_WPE\_subs, by = "State") %>%

full\_join(API\_NNI\_subs, by = "State")

```

#AINA (Subsestimates: FH, NNI)

```{r warning=FALSE, message=FALSE}

#FH

AINA\_FH\_subs <- read\_csv("[filepath]/AINA-FH-SUBESTIMATES.csv") %>%

dplyr::select("State", "AINA\_eblups", "AINA\_eblup\_se")

#NNI

AINA\_NNI\_subs <- read\_csv("[filepath]/NNI-SUBESTIMATES.csv") %>%

dplyr::select("State", "NR\_AINA\_Mean", "NR\_AINA\_SE")

#merge with #full\_join()

AINA\_FLEXCS\_DF <- full\_join(AINA\_FH\_subs, AINA\_NNI\_subs, by = "State")

```

#TP (Subsestimates: FH, NNI)

```{r warning=FALSE, message=FALSE}

#FH

TP\_FH\_subs <- read\_csv("[filepath]/TP-FH-SUBESTIMATES.csv") %>%

dplyr::select("State", "TP\_eblups", "TP\_eblup\_se")

#NNI

TP\_NNI\_subs <- read\_csv("[filepath]/NNI-SUBESTIMATES.csv") %>%

dplyr::select("State", "NR\_TP\_Mean", "NR\_TP\_SE")

#merge with #full\_join()

TP\_FLEXCS\_DF <- full\_join(TP\_FH\_subs, TP\_NNI\_subs, by = "State")

```

#EL (Subsestimates: FH, NNI)

```{r warning=FALSE, message=FALSE}

#FH

EL\_FH\_subs <- read\_csv("[filepath]/EL-FH-SUBESTIMATES.csv") %>%

dplyr::select("State", "EL\_eblups", "EL\_eblup\_se")

#NNI

EL\_NNI\_subs <- read\_csv("[filepath]/NNI-SUBESTIMATES.csv") %>%

dplyr::select("State", "NR\_EL\_Mean", "NR\_EL\_SE")

#merge with #full\_join()

EL\_FLEXCS\_DF <- full\_join(EL\_FH\_subs, EL\_NNI\_subs, by = "State")

```

### Calculate FLEX CS mean estimates and variance estimates

#\*\*First, the mean estimates (precision-weighted averages)\*\*

#NHS

```{r}

#Will have 5 variables (State, NHS\_eblups, NHS\_eblup\_se, NR\_NHS\_Mean, NR\_NHS\_SE)

NHS\_FLEXCS\_DF$complete <- complete.cases(NHS\_FLEXCS\_DF)

#Numerator

NHS\_FLEXCS\_DF$Numerator <- ifelse(NHS\_FLEXCS\_DF$complete == TRUE,

(NHS\_FLEXCS\_DF$NHS\_eblups/NHS\_FLEXCS\_DF$NHS\_eblup\_se^2) + (NHS\_FLEXCS\_DF$NR\_NHS\_Mean/NHS\_FLEXCS\_DF$NR\_NHS\_SE^2),

(NHS\_FLEXCS\_DF$NHS\_eblups/NHS\_FLEXCS\_DF$NHS\_eblup\_se^2))

#Denominator

NHS\_FLEXCS\_DF$Denominator <- ifelse(NHS\_FLEXCS\_DF$complete == TRUE,

(1/NHS\_FLEXCS\_DF$NHS\_eblup\_se^2) + (1/NHS\_FLEXCS\_DF$NR\_NHS\_SE^2), 1/NHS\_FLEXCS\_DF$NHS\_eblup\_se^2)

#FLEX\_CS\_ESTIMATE

NHS\_FLEXCS\_DF$FLEX\_CS\_ESTIMATE <- NHS\_FLEXCS\_DF$Numerator/NHS\_FLEXCS\_DF$Denominator

```

#HS

```{r}

#Will have 7 variables (State, mean, se, HS\_eblups, HS\_eblup\_se, NR\_HS\_Mean, NR\_HS\_SE)

#Filter out cases with missing values (if there are any)

HS\_FLEXCS\_DF <- HS\_FLEXCS\_DF %>%

filter(HS\_eblups > 0)

HS\_FLEXCS\_DF$complete <- complete.cases(HS\_FLEXCS\_DF)

#Numerator

HS\_FLEXCS\_DF$Numerator <-

ifelse(HS\_FLEXCS\_DF$complete == TRUE,

(HS\_FLEXCS\_DF$mean/HS\_FLEXCS\_DF$se^2) + (HS\_FLEXCS\_DF$HS\_eblups/HS\_FLEXCS\_DF$HS\_eblup\_se^2) + (HS\_FLEXCS\_DF$NR\_HS\_Mean/HS\_FLEXCS\_DF$NR\_HS\_SE^2),

(HS\_FLEXCS\_DF$mean/HS\_FLEXCS\_DF$se^2) + (HS\_FLEXCS\_DF$HS\_eblups/HS\_FLEXCS\_DF$HS\_eblup\_se^2))

#Denominator

HS\_FLEXCS\_DF$Denominator <- ifelse(HS\_FLEXCS\_DF$complete == TRUE,

(1/HS\_FLEXCS\_DF$HS\_eblup\_se^2) + (1/HS\_FLEXCS\_DF$NR\_HS\_SE^2) + (1/HS\_FLEXCS\_DF$se^2), (1/HS\_FLEXCS\_DF$HS\_eblup\_se^2) + (1/HS\_FLEXCS\_DF$se^2))

#FLEX\_CS\_ESTIMATE

HS\_FLEXCS\_DF$FLEX\_CS\_ESTIMATE <- HS\_FLEXCS\_DF$Numerator/HS\_FLEXCS\_DF$Denominator

```

#SBA

```{r}

#Will have 7 variables (State, mean, se, SBA\_eblups, SBA\_eblup\_se, NR\_SBA\_Mean, NR\_SBA\_SE)

#Filter out cases with missing values (if there are any)

SBA\_FLEXCS\_DF <- SBA\_FLEXCS\_DF %>%

filter(SBA\_eblups > 0)

SBA\_FLEXCS\_DF$complete <- complete.cases(SBA\_FLEXCS\_DF)

#Numerator

SBA\_FLEXCS\_DF$Numerator <-

ifelse(SBA\_FLEXCS\_DF$complete == TRUE,

(SBA\_FLEXCS\_DF$mean/SBA\_FLEXCS\_DF$se^2) + (SBA\_FLEXCS\_DF$SBA\_eblups/SBA\_FLEXCS\_DF$SBA\_eblup\_se^2) + (SBA\_FLEXCS\_DF$NR\_SBA\_Mean/SBA\_FLEXCS\_DF$NR\_SBA\_SE^2),

(SBA\_FLEXCS\_DF$mean/SBA\_FLEXCS\_DF$se^2) + (SBA\_FLEXCS\_DF$SBA\_eblups/SBA\_FLEXCS\_DF$SBA\_eblup\_se^2))

#Denominator

SBA\_FLEXCS\_DF$Denominator <- ifelse(SBA\_FLEXCS\_DF$complete == TRUE,

(1/SBA\_FLEXCS\_DF$SBA\_eblup\_se^2) + (1/SBA\_FLEXCS\_DF$NR\_SBA\_SE^2) + (1/SBA\_FLEXCS\_DF$se^2), (1/SBA\_FLEXCS\_DF$SBA\_eblup\_se^2) + (1/SBA\_FLEXCS\_DF$se^2))

#FLEX\_CS\_ESTIMATE

SBA\_FLEXCS\_DF$FLEX\_CS\_ESTIMATE <- SBA\_FLEXCS\_DF$Numerator/SBA\_FLEXCS\_DF$Denominator

```

#BA

```{r}

#Will have 7 variables (State, mean, se, BA\_eblups, BA\_eblup\_se, NR\_BA\_Mean, NR\_BA\_SE)

#Filter out cases with missing values (if there are any)

BA\_FLEXCS\_DF <- BA\_FLEXCS\_DF %>%

filter(BA\_eblups > 0)

BA\_FLEXCS\_DF$complete <- complete.cases(BA\_FLEXCS\_DF)

#Numerator

BA\_FLEXCS\_DF$Numerator <-

ifelse(BA\_FLEXCS\_DF$complete == TRUE,

(BA\_FLEXCS\_DF$mean/BA\_FLEXCS\_DF$se^2) + (BA\_FLEXCS\_DF$BA\_eblups/BA\_FLEXCS\_DF$BA\_eblup\_se^2) + (BA\_FLEXCS\_DF$NR\_BA\_Mean/BA\_FLEXCS\_DF$NR\_BA\_SE^2),

(BA\_FLEXCS\_DF$mean/BA\_FLEXCS\_DF$se^2) + (BA\_FLEXCS\_DF$BA\_eblups/BA\_FLEXCS\_DF$BA\_eblup\_se^2))

#Denominator

BA\_FLEXCS\_DF$Denominator <- ifelse(BA\_FLEXCS\_DF$complete == TRUE,

(1/BA\_FLEXCS\_DF$BA\_eblup\_se^2) + (1/BA\_FLEXCS\_DF$NR\_BA\_SE^2) + (1/BA\_FLEXCS\_DF$se^2), (1/BA\_FLEXCS\_DF$BA\_eblup\_se^2) + (1/BA\_FLEXCS\_DF$se^2))

#FLEX\_CS\_ESTIMATE

BA\_FLEXCS\_DF$FLEX\_CS\_ESTIMATE <- BA\_FLEXCS\_DF$Numerator/BA\_FLEXCS\_DF$Denominator

```

#B

```{r}

#Will have 7 variables (State, B\_eblups, B\_eblup\_se, blk\_wpe, blk\_var, NR\_B\_Mean, NR\_B\_SE)

#Filter out cases with missing values (if there are any)

B\_FLEXCS\_DF <- B\_FLEXCS\_DF %>%

filter(B\_eblups > 0)

#Convert missing values (0) to NA

B\_FLEXCS\_DF$blk\_wpe[B\_FLEXCS\_DF$blk\_wpe==0] <- NA

#Numerator

B\_FLEXCS\_DF <- B\_FLEXCS\_DF %>%

mutate(eblup\_ratio = B\_eblups/B\_eblup\_se^2,

wpe\_ratio = blk\_wpe/blk\_var,

nni\_ratio = NR\_B\_Mean/NR\_B\_SE^2)

B\_FLEXCS\_DF$Numerator <- rowSums(B\_FLEXCS\_DF[,8:10], na.rm = T)

#Denominator

B\_FLEXCS\_DF <- B\_FLEXCS\_DF %>%

mutate(eblup\_var\_inv = 1/B\_eblup\_se^2,

wpe\_var\_inv = 1/blk\_var,

nni\_var\_inv = 1/NR\_B\_SE^2)

B\_FLEXCS\_DF$Denominator <- rowSums(B\_FLEXCS\_DF[,12:14], na.rm = T)

#FLEX\_CS\_ESTIMATE

B\_FLEXCS\_DF$FLEX\_CS\_ESTIMATE <- B\_FLEXCS\_DF$Numerator/B\_FLEXCS\_DF$Denominator

```

#H

```{r}

#Will have 7 variables (State, H\_eblups, H\_eblup\_se, hsp\_wpe, hsp\_var, NR\_H\_Mean, NR\_H\_SE)

#Filter out cases with missing values (if there are any)

H\_FLEXCS\_DF <- H\_FLEXCS\_DF %>%

filter(H\_eblups > 0)

#Convert missing values (0) to NA

H\_FLEXCS\_DF$hsp\_wpe[H\_FLEXCS\_DF$hsp\_wpe==0] <- NA

#Numerator

H\_FLEXCS\_DF <- H\_FLEXCS\_DF %>%

mutate(eblup\_ratio = H\_eblups/H\_eblup\_se^2,

wpe\_ratio = hsp\_wpe/hsp\_var,

nni\_ratio = NR\_H\_Mean/NR\_H\_SE^2)

H\_FLEXCS\_DF$Numerator <- rowSums(H\_FLEXCS\_DF[,8:10], na.rm = T)

#Denominator

H\_FLEXCS\_DF <- H\_FLEXCS\_DF %>%

mutate(eblup\_var\_inv = 1/H\_eblup\_se^2,

wpe\_var\_inv = 1/hsp\_var,

nni\_var\_inv = 1/NR\_H\_SE^2)

H\_FLEXCS\_DF$Denominator <- rowSums(H\_FLEXCS\_DF[,12:14], na.rm = T)

#FLEX\_CS\_ESTIMATE

H\_FLEXCS\_DF$FLEX\_CS\_ESTIMATE <- H\_FLEXCS\_DF$Numerator/H\_FLEXCS\_DF$Denominator

```

#API

```{r}

#Will have 7 variables (State, API\_eblups, API\_eblup\_se, asn\_wpe, asn\_var, NR\_API\_Mean, NR\_API\_SE)

#Filter out cases with missing values (if there are any)

API\_FLEXCS\_DF <- API\_FLEXCS\_DF %>%

filter(API\_eblups > 0)

#Convert missing values (0) to NA

API\_FLEXCS\_DF$asn\_wpe[API\_FLEXCS\_DF$asn\_wpe==0] <- NA

#Numerator

API\_FLEXCS\_DF <- API\_FLEXCS\_DF %>%

mutate(eblup\_ratio = API\_eblups/API\_eblup\_se^2,

wpe\_ratio = asn\_wpe/asn\_var,

nni\_ratio = NR\_API\_Mean/NR\_API\_SE^2)

API\_FLEXCS\_DF$Numerator <- rowSums(API\_FLEXCS\_DF[,8:10], na.rm = T)

#Denominator

API\_FLEXCS\_DF <- API\_FLEXCS\_DF %>%

mutate(eblup\_var\_inv = 1/API\_eblup\_se^2,

wpe\_var\_inv = 1/asn\_var,

nni\_var\_inv = 1/NR\_API\_SE^2)

API\_FLEXCS\_DF$Denominator <- rowSums(API\_FLEXCS\_DF[,12:14], na.rm = T)

#FLEX\_CS\_ESTIMATE

API\_FLEXCS\_DF$FLEX\_CS\_ESTIMATE <- API\_FLEXCS\_DF$Numerator/API\_FLEXCS\_DF$Denominator

```

#AINA

```{r}

#Will have 5 variables (State, AINA\_eblups, AINA\_eblup\_se, NR\_AINA\_Mean, NR\_AINA\_SE)

#Filter out cases with missing values (if there are any)

AINA\_FLEXCS\_DF <- AINA\_FLEXCS\_DF %>%

filter(AINA\_eblups > 0)

AINA\_FLEXCS\_DF$complete <- complete.cases(AINA\_FLEXCS\_DF)

#Numerator

AINA\_FLEXCS\_DF$Numerator <- ifelse(AINA\_FLEXCS\_DF$complete == TRUE,

(AINA\_FLEXCS\_DF$AINA\_eblups/AINA\_FLEXCS\_DF$AINA\_eblup\_se^2) + (AINA\_FLEXCS\_DF$NR\_AINA\_Mean/AINA\_FLEXCS\_DF$NR\_AINA\_SE^2),

(AINA\_FLEXCS\_DF$AINA\_eblups/AINA\_FLEXCS\_DF$AINA\_eblup\_se^2))

#Denominator

AINA\_FLEXCS\_DF$Denominator <- ifelse(AINA\_FLEXCS\_DF$complete == TRUE,

(1/AINA\_FLEXCS\_DF$AINA\_eblup\_se^2) + (1/AINA\_FLEXCS\_DF$NR\_AINA\_SE^2), 1/AINA\_FLEXCS\_DF$AINA\_eblup\_se^2)

#FLEX\_CS\_ESTIMATE

AINA\_FLEXCS\_DF$FLEX\_CS\_ESTIMATE <- AINA\_FLEXCS\_DF$Numerator/AINA\_FLEXCS\_DF$Denominator

```

#TP

```{r}

#Will have 5 variables (State, TP\_eblups, TP\_eblup\_se, NR\_TP\_Mean, NR\_TP\_SE)

#Filter out cases with missing values (if there are any)

TP\_FLEXCS\_DF <- TP\_FLEXCS\_DF %>%

filter(TP\_eblups > 0)

TP\_FLEXCS\_DF$complete <- complete.cases(TP\_FLEXCS\_DF)

#Numerator

TP\_FLEXCS\_DF$Numerator <- ifelse(TP\_FLEXCS\_DF$complete == TRUE,

(TP\_FLEXCS\_DF$TP\_eblups/TP\_FLEXCS\_DF$TP\_eblup\_se^2) + (TP\_FLEXCS\_DF$NR\_TP\_Mean/TP\_FLEXCS\_DF$NR\_TP\_SE^2),

(TP\_FLEXCS\_DF$TP\_eblups/TP\_FLEXCS\_DF$TP\_eblup\_se^2))

#Denominator

TP\_FLEXCS\_DF$Denominator <- ifelse(TP\_FLEXCS\_DF$complete == TRUE,

(1/TP\_FLEXCS\_DF$TP\_eblup\_se^2) + (1/TP\_FLEXCS\_DF$NR\_TP\_SE^2), 1/TP\_FLEXCS\_DF$TP\_eblup\_se^2)

#FLEX\_CS\_ESTIMATE

TP\_FLEXCS\_DF$FLEX\_CS\_ESTIMATE <- TP\_FLEXCS\_DF$Numerator/TP\_FLEXCS\_DF$Denominator

```

#EL

```{r}

#Will have 5 variables (State, EL\_eblups, EL\_eblup\_se, NR\_EL\_Mean, NR\_EL\_SE)

#Filter out cases with missing values (if there are any)

EL\_FLEXCS\_DF <- EL\_FLEXCS\_DF %>%

filter(EL\_eblups > 0)

EL\_FLEXCS\_DF$complete <- complete.cases(EL\_FLEXCS\_DF)

#Numerator

EL\_FLEXCS\_DF$Numerator <- ifelse(EL\_FLEXCS\_DF$complete == TRUE,

(EL\_FLEXCS\_DF$EL\_eblups/EL\_FLEXCS\_DF$EL\_eblup\_se^2) + (EL\_FLEXCS\_DF$NR\_EL\_Mean/EL\_FLEXCS\_DF$NR\_EL\_SE^2),

(EL\_FLEXCS\_DF$EL\_eblups/EL\_FLEXCS\_DF$EL\_eblup\_se^2))

#Denominator

EL\_FLEXCS\_DF$Denominator <- ifelse(EL\_FLEXCS\_DF$complete == TRUE,

(1/EL\_FLEXCS\_DF$EL\_eblup\_se^2) + (1/EL\_FLEXCS\_DF$NR\_EL\_SE^2), 1/EL\_FLEXCS\_DF$EL\_eblup\_se^2)

#FLEX\_CS\_ESTIMATE

EL\_FLEXCS\_DF$FLEX\_CS\_ESTIMATE <- EL\_FLEXCS\_DF$Numerator/EL\_FLEXCS\_DF$Denominator

```

\*\*Second, the variance estimates\*\*

#NHS

```{r, message=FALSE, warning=FALSE}

#Variables: State, NHS\_eblups, NHS\_eblup\_se, NR\_NHS\_Mean, NR\_NHS\_SE

#FLEX\_CS\_VAR\_ESTIMATE

NHS\_FLEXCS\_DF$NHS\_eblup\_var <- NHS\_FLEXCS\_DF$NHS\_eblup\_se^2

NHS\_FLEXCS\_DF$NR\_NHS\_var <- NHS\_FLEXCS\_DF$NR\_NHS\_SE^2

NHS\_FLEXCS\_DF$mean\_var <- rowMeans(NHS\_FLEXCS\_DF[,10:11], na.rm = TRUE)

NHS\_FLEXCS\_DF$NHS\_eblups\_temp <- NHS\_FLEXCS\_DF$NHS\_eblups

NHS\_FLEXCS\_DF$NR\_NHS\_Mean\_temp <- NHS\_FLEXCS\_DF$NR\_NHS\_Mean

NHS\_FLEXCS\_DF$sub\_var <- rowVars(as.matrix(NHS\_FLEXCS\_DF[,13:14]), na.rm = TRUE)

NHS\_FLEXCS\_DF$FLEX\_CS\_VAR\_ESTIMATE <- ifelse(NHS\_FLEXCS\_DF$complete==TRUE, NHS\_FLEXCS\_DF$mean\_var + NHS\_FLEXCS\_DF$sub\_var, NHS\_FLEXCS\_DF$mean\_var)

```

#HS

```{r, message=FALSE, warning=FALSE}

#Variables: State, mean, se, HS\_eblups, HS\_eblup\_se, NR\_HS\_Mean, NR\_HS\_SE

#FLEX\_CS\_VAR\_ESTIMATE

HS\_FLEXCS\_DF$mice\_var <- HS\_FLEXCS\_DF$se^2

HS\_FLEXCS\_DF$HS\_eblup\_var <- HS\_FLEXCS\_DF$HS\_eblup\_se^2

HS\_FLEXCS\_DF$NR\_HS\_var <- HS\_FLEXCS\_DF$NR\_HS\_SE^2

HS\_FLEXCS\_DF$mean\_var <- rowMeans(HS\_FLEXCS\_DF[,12:14], na.rm = TRUE)

HS\_FLEXCS\_DF$mean\_temp <- HS\_FLEXCS\_DF$mean

HS\_FLEXCS\_DF$HS\_eblups\_temp <- HS\_FLEXCS\_DF$HS\_eblups

HS\_FLEXCS\_DF$NR\_HS\_Mean\_temp <- HS\_FLEXCS\_DF$NR\_HS\_Mean

HS\_FLEXCS\_DF$sub\_var <- rowVars(as.matrix(HS\_FLEXCS\_DF[,16:18]), na.rm = TRUE)

HS\_FLEXCS\_DF$FLEX\_CS\_VAR\_ESTIMATE <-HS\_FLEXCS\_DF$mean\_var + HS\_FLEXCS\_DF$sub\_var

```

#SBA

```{r, message=FALSE, warning=FALSE}

#Variables: State, mean, se, SBA\_eblups, SBA\_eblup\_se, NR\_SBA\_Mean, NR\_SBA\_SE

#FLEX\_CS\_VAR\_ESTIMATE

SBA\_FLEXCS\_DF$mice\_var <- SBA\_FLEXCS\_DF$se^2

SBA\_FLEXCS\_DF$SBA\_eblup\_var <- SBA\_FLEXCS\_DF$SBA\_eblup\_se^2

SBA\_FLEXCS\_DF$NR\_SBA\_var <- SBA\_FLEXCS\_DF$NR\_SBA\_SE^2

SBA\_FLEXCS\_DF$mean\_var <- rowMeans(SBA\_FLEXCS\_DF[,12:14], na.rm = TRUE)

SBA\_FLEXCS\_DF$mean\_temp <- SBA\_FLEXCS\_DF$mean

SBA\_FLEXCS\_DF$SBA\_eblups\_temp <- SBA\_FLEXCS\_DF$SBA\_eblups

SBA\_FLEXCS\_DF$NR\_SBA\_Mean\_temp <- SBA\_FLEXCS\_DF$NR\_SBA\_Mean

SBA\_FLEXCS\_DF$sub\_var <- rowVars(as.matrix(SBA\_FLEXCS\_DF[,16:18]), na.rm = TRUE)

SBA\_FLEXCS\_DF$FLEX\_CS\_VAR\_ESTIMATE <-SBA\_FLEXCS\_DF$mean\_var + SBA\_FLEXCS\_DF$sub\_var

```

#BA

```{r, message=FALSE, warning=FALSE}

#Variables: State, mean, se, BA\_eblups, BA\_eblup\_se, NR\_BA\_Mean, NR\_BA\_SE

#FLEX\_CS\_VAR\_ESTIMATE

BA\_FLEXCS\_DF$mice\_var <- BA\_FLEXCS\_DF$se^2

BA\_FLEXCS\_DF$BA\_eblup\_var <- BA\_FLEXCS\_DF$BA\_eblup\_se^2

BA\_FLEXCS\_DF$NR\_BA\_var <- BA\_FLEXCS\_DF$NR\_BA\_SE^2

BA\_FLEXCS\_DF$mean\_var <- rowMeans(BA\_FLEXCS\_DF[,12:14], na.rm = TRUE)

BA\_FLEXCS\_DF$mean\_temp <- BA\_FLEXCS\_DF$mean

BA\_FLEXCS\_DF$BA\_eblups\_temp <- BA\_FLEXCS\_DF$BA\_eblups

BA\_FLEXCS\_DF$NR\_BA\_Mean\_temp <- BA\_FLEXCS\_DF$NR\_BA\_Mean

BA\_FLEXCS\_DF$sub\_var <- rowVars(as.matrix(BA\_FLEXCS\_DF[,16:18]), na.rm = TRUE)

BA\_FLEXCS\_DF$FLEX\_CS\_VAR\_ESTIMATE <-BA\_FLEXCS\_DF$mean\_var + BA\_FLEXCS\_DF$sub\_var

```

#B

```{r, message=FALSE, warning=FALSE}

#Variables: State, B\_eblups, B\_eblup\_se, blk\_wpe, blk\_var, NR\_B\_Mean, NR\_B\_SE

#FLEX\_CS\_VAR\_ESTIMATE

B\_FLEXCS\_DF$B\_eblup\_var <- B\_FLEXCS\_DF$B\_eblup\_se^2

B\_FLEXCS\_DF$blk\_var\_temp <- B\_FLEXCS\_DF$blk\_var

B\_FLEXCS\_DF$NR\_B\_var <- B\_FLEXCS\_DF$NR\_B\_SE^2

B\_FLEXCS\_DF$mean\_var <- rowMeans(B\_FLEXCS\_DF[,17:19], na.rm = TRUE)

B\_FLEXCS\_DF$B\_eblups\_temp <- B\_FLEXCS\_DF$B\_eblups

B\_FLEXCS\_DF$blk\_wpe\_temp <- B\_FLEXCS\_DF$blk\_wpe

B\_FLEXCS\_DF$NR\_B\_Mean\_temp <- B\_FLEXCS\_DF$NR\_B\_Mean

B\_FLEXCS\_DF$sub\_var <- rowVars(as.matrix(B\_FLEXCS\_DF[,21:23]), na.rm = TRUE)

B\_FLEXCS\_DF$FLEX\_CS\_VAR\_ESTIMATE <-

ifelse(is.na(B\_FLEXCS\_DF$sub\_var),

B\_FLEXCS\_DF$mean\_var,

B\_FLEXCS\_DF$mean\_var + B\_FLEXCS\_DF$sub\_var)

```

#H

```{r, message=FALSE, warning=FALSE}

#Variables: State, H\_eblups, H\_eblup\_se, hsp\_wpe, hsp\_var, NR\_H\_Mean, NR\_H\_SE

#FLEX\_CS\_VAR\_ESTIMATE

H\_FLEXCS\_DF$H\_eblup\_var <- H\_FLEXCS\_DF$H\_eblup\_se^2

H\_FLEXCS\_DF$hsp\_var\_temp <- H\_FLEXCS\_DF$hsp\_var

H\_FLEXCS\_DF$NR\_H\_var <- H\_FLEXCS\_DF$NR\_H\_SE^2

H\_FLEXCS\_DF$mean\_var <- rowMeans(H\_FLEXCS\_DF[,17:19], na.rm = TRUE)

H\_FLEXCS\_DF$H\_eblups\_temp <- H\_FLEXCS\_DF$H\_eblups

H\_FLEXCS\_DF$hsp\_wpe\_temp <- H\_FLEXCS\_DF$hsp\_wpe

H\_FLEXCS\_DF$NR\_H\_Mean\_temp <- H\_FLEXCS\_DF$NR\_H\_Mean

H\_FLEXCS\_DF$sub\_var <- rowVars(as.matrix(H\_FLEXCS\_DF[,21:23]), na.rm = TRUE)

H\_FLEXCS\_DF$FLEX\_CS\_VAR\_ESTIMATE <-

ifelse(is.na(H\_FLEXCS\_DF$sub\_var),

H\_FLEXCS\_DF$mean\_var,

H\_FLEXCS\_DF$mean\_var + H\_FLEXCS\_DF$sub\_var)

```

#API

```{r, message=FALSE, warning=FALSE}

#Variables: State, API\_eblups, API\_eblup\_se, asn\_wpe, asn\_var, NR\_API\_Mean, NR\_API\_SE

#FLEX\_CS\_VAR\_ESTIMATE

API\_FLEXCS\_DF$API\_eblup\_var <- API\_FLEXCS\_DF$API\_eblup\_se^2

API\_FLEXCS\_DF$asn\_var\_temp <- API\_FLEXCS\_DF$asn\_var

API\_FLEXCS\_DF$NR\_API\_var <- API\_FLEXCS\_DF$NR\_API\_SE^2

API\_FLEXCS\_DF$mean\_var <- rowMeans(API\_FLEXCS\_DF[,17:19], na.rm = TRUE)

API\_FLEXCS\_DF$API\_eblups\_temp <- API\_FLEXCS\_DF$API\_eblups

API\_FLEXCS\_DF$asn\_wpe\_temp <- API\_FLEXCS\_DF$asn\_wpe

API\_FLEXCS\_DF$NR\_API\_Mean\_temp <- API\_FLEXCS\_DF$NR\_API\_Mean

API\_FLEXCS\_DF$sub\_var <- rowVars(as.matrix(API\_FLEXCS\_DF[,21:23]), na.rm = TRUE)

API\_FLEXCS\_DF$FLEX\_CS\_VAR\_ESTIMATE <-

ifelse(is.na(API\_FLEXCS\_DF$sub\_var),

API\_FLEXCS\_DF$mean\_var,

API\_FLEXCS\_DF$mean\_var + API\_FLEXCS\_DF$sub\_var)

```

#AINA

```{r, message=FALSE, warning=FALSE}

#Variables: State, AINA\_eblups, AINA\_eblup\_se, NR\_AINA\_Mean, NR\_AINA\_SE

#FLEX\_CS\_VAR\_ESTIMATE

AINA\_FLEXCS\_DF$AINA\_eblup\_var <- AINA\_FLEXCS\_DF$AINA\_eblup\_se^2

AINA\_FLEXCS\_DF$NR\_AINA\_var <- AINA\_FLEXCS\_DF$NR\_AINA\_SE^2

AINA\_FLEXCS\_DF$mean\_var <- rowMeans(AINA\_FLEXCS\_DF[,10:11], na.rm = TRUE)

AINA\_FLEXCS\_DF$AINA\_eblups\_temp <- AINA\_FLEXCS\_DF$AINA\_eblups

AINA\_FLEXCS\_DF$NR\_AINA\_Mean\_temp <- AINA\_FLEXCS\_DF$NR\_AINA\_Mean

AINA\_FLEXCS\_DF$sub\_var <- rowVars(as.matrix(AINA\_FLEXCS\_DF[,13:14]), na.rm = TRUE)

AINA\_FLEXCS\_DF$FLEX\_CS\_VAR\_ESTIMATE <- ifelse(AINA\_FLEXCS\_DF$complete==TRUE, AINA\_FLEXCS\_DF$mean\_var + AINA\_FLEXCS\_DF$sub\_var, AINA\_FLEXCS\_DF$mean\_var)

```

#TP

```{r, message=FALSE, warning=FALSE}

#Variables: State, TP\_eblups, TP\_eblup\_se, NR\_TP\_Mean, NR\_TP\_SE

#FLEX\_CS\_VAR\_ESTIMATE

TP\_FLEXCS\_DF$TP\_eblup\_var <- TP\_FLEXCS\_DF$TP\_eblup\_se^2

TP\_FLEXCS\_DF$NR\_TP\_var <- TP\_FLEXCS\_DF$NR\_TP\_SE^2

TP\_FLEXCS\_DF$mean\_var <- rowMeans(TP\_FLEXCS\_DF[,10:11], na.rm = TRUE)

TP\_FLEXCS\_DF$TP\_eblups\_temp <- TP\_FLEXCS\_DF$TP\_eblups

TP\_FLEXCS\_DF$NR\_TP\_Mean\_temp <- TP\_FLEXCS\_DF$NR\_TP\_Mean

TP\_FLEXCS\_DF$sub\_var <- rowVars(as.matrix(TP\_FLEXCS\_DF[,13:14]), na.rm = TRUE)

TP\_FLEXCS\_DF$FLEX\_CS\_VAR\_ESTIMATE <- ifelse(TP\_FLEXCS\_DF$complete==TRUE, TP\_FLEXCS\_DF$mean\_var + TP\_FLEXCS\_DF$sub\_var, TP\_FLEXCS\_DF$mean\_var)

```

#EL

```{r, message=FALSE, warning=FALSE}

#Variables: State, EL\_eblups, EL\_eblup\_se, NR\_EL\_Mean, NR\_EL\_SE

#FLEX\_CS\_VAR\_ESTIMATE

EL\_FLEXCS\_DF$EL\_eblup\_var <- EL\_FLEXCS\_DF$EL\_eblup\_se^2

EL\_FLEXCS\_DF$NR\_EL\_var <- EL\_FLEXCS\_DF$NR\_EL\_SE^2

EL\_FLEXCS\_DF$mean\_var <- rowMeans(EL\_FLEXCS\_DF[,10:11], na.rm = TRUE)

EL\_FLEXCS\_DF$EL\_eblups\_temp <- EL\_FLEXCS\_DF$EL\_eblups

EL\_FLEXCS\_DF$NR\_EL\_Mean\_temp <- EL\_FLEXCS\_DF$NR\_EL\_Mean

EL\_FLEXCS\_DF$sub\_var <- rowVars(as.matrix(EL\_FLEXCS\_DF[,13:14]), na.rm = TRUE)

EL\_FLEXCS\_DF$FLEX\_CS\_VAR\_ESTIMATE <- ifelse(EL\_FLEXCS\_DF$complete==TRUE, EL\_FLEXCS\_DF$mean\_var + EL\_FLEXCS\_DF$sub\_var, EL\_FLEXCS\_DF$mean\_var)

```

### Export and prepare estimates for calculating accuracy statistics

```{r warning=FALSE, message=FALSE}

#NHS

write.csv(NHS\_FLEXCS\_DF, "[filepath]/NHS-FLEXCS-ESTIMATES.csv")

#HS

write.csv(HS\_FLEXCS\_DF, "[filepath]/HS-FLEXCS-ESTIMATES.csv")

#SBA

write.csv(SBA\_FLEXCS\_DF, "[filepath]/SBA-FLEXCS-ESTIMATES.csv")

#BA

write.csv(BA\_FLEXCS\_DF, "[filepath]/BA-FLEXCS-ESTIMATES.csv")

#B

write.csv(B\_FLEXCS\_DF, "[filepath]/B-FLEXCS-ESTIMATES.csv")

#H

write.csv(H\_FLEXCS\_DF, "[filepath]/H-FLEXCS-ESTIMATES.csv")

#API

write.csv(API\_FLEXCS\_DF, "[filepath]/API-FLEXCS-ESTIMATES.csv")

#AINA

write.csv(AINA\_FLEXCS\_DF, "[filepath]/AINA-FLEXCS-ESTIMATES.csv")

#TP

write.csv(TP\_FLEXCS\_DF, "[filepath]/TP-FLEXCS-ESTIMATES.csv")

#EL

write.csv(EL\_FLEXCS\_DF, "[filepath]/EL-FLEXCS-ESTIMATES.csv")

```

### Calculate wMAE for FLEX CS technique

\*\*Free up memory\*\*

```{r warning=FALSE, message=FALSE}

rm(list = ls())

```

\*\*Define a wMAE function\*\*

```{r warning=FALSE, message=FALSE}

wmae <- function(x){

w\_abs\_errors <- abs(x[,2] - x[,4])/x[,3]

mean(w\_abs\_errors[,1])

}

```

#NHS

```{r message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

NHS\_predicted <- read\_csv("[filepath]/NHS-FLEXCS-ESTIMATES.csv")

NHS\_predicted <- NHS\_predicted[,-1]

names(NHS\_predicted) <- c("State", "FLEXCS\_mean", "FLEXCS\_se")

NHS\_observed <- read\_csv("[filepath]/NHS.csv")

NHS\_observed <- NHS\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

NHS <- inner\_join(NHS\_observed, NHS\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(NHS)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, warning=FALSE, message=FALSE}

#Make NHS tidy

source <- rep(c("NAEP"), 48)

source <- as.data.frame(source)

part1 <- cbind(NHS$State, source, NHS$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("FLEXCS"), 48)

source <- as.data.frame(source)

part2 <- cbind(NHS$State, source, NHS$FLEXCS\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

NHS\_tidy <- rbind(part1, part2)

NHS\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

NHS\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

NHS\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#HS

```{r message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

HS\_predicted <- read\_csv("[filepath]/HS-FLEXCS-ESTIMATES.csv")

HS\_predicted <- HS\_predicted[,-1]

names(HS\_predicted) <- c("State", "FLEXCS\_mean", "FLEXCS\_se")

HS\_observed <- read\_csv("[filepath]/HS.csv")

HS\_observed <- HS\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

HS <- inner\_join(HS\_observed, HS\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(HS)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, warning=FALSE, message=FALSE}

#Make HS tidy

source <- rep(c("NAEP"), 48)

source <- as.data.frame(source)

part1 <- cbind(HS$State, source, HS$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("FLEXCS"), 48)

source <- as.data.frame(source)

part2 <- cbind(HS$State, source, HS$FLEXCS\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

HS\_tidy <- rbind(part1, part2)

HS\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

HS\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

HS\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#SBA

```{r message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

SBA\_predicted <- read\_csv("[filepath]/SBA-FLEXCS-ESTIMATES.csv")

SBA\_predicted <- SBA\_predicted[,-1]

names(SBA\_predicted) <- c("State", "FLEXCS\_mean", "FLEXCS\_se")

SBA\_observed <- read\_csv("[filepath]/SBA.csv")

SBA\_observed <- SBA\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

SBA <- inner\_join(SBA\_observed, SBA\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(SBA)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, warning=FALSE, message=FALSE}

#Make SBA tidy

source <- rep(c("NAEP"), 48)

source <- as.data.frame(source)

part1 <- cbind(SBA$State, source, SBA$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("FLEXCS"), 48)

source <- as.data.frame(source)

part2 <- cbind(SBA$State, source, SBA$FLEXCS\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

SBA\_tidy <- rbind(part1, part2)

SBA\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

SBA\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

SBA\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#BA

```{r message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

BA\_predicted <- read\_csv("[filepath]/BA-FLEXCS-ESTIMATES.csv")

BA\_predicted <- BA\_predicted[,-1]

names(BA\_predicted) <- c("State", "FLEXCS\_mean", "FLEXCS\_se")

BA\_observed <- read\_csv("[filepath]/BA.csv")

BA\_observed <- BA\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

BA <- inner\_join(BA\_observed, BA\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(BA)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, warning=FALSE, message=FALSE}

#Make BA tidy

source <- rep(c("NAEP"), 48)

source <- as.data.frame(source)

part1 <- cbind(BA$State, source, BA$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("FLEXCS"), 48)

source <- as.data.frame(source)

part2 <- cbind(BA$State, source, BA$FLEXCS\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

BA\_tidy <- rbind(part1, part2)

BA\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

BA\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

BA\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#B

```{r message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

B\_predicted <- read\_csv("[filepath]/B-FLEXCS-ESTIMATES.csv")

B\_predicted <- B\_predicted[,-1]

names(B\_predicted) <- c("State", "FLEXCS\_mean", "FLEXCS\_se")

B\_observed <- read\_csv("[filepath]/B.csv")

B\_observed <- B\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

B <- inner\_join(B\_observed, B\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(B)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r message=FALSE, warning=FALSE}

#Make B tidy

source <- rep(c("NAEP"), 39)

source <- as.data.frame(source)

part1 <- cbind(B$State, source, B$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("FLEXCS"), 39)

source <- as.data.frame(source)

part2 <- cbind(B$State, source, B$FLEXCS\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

B\_tidy <- rbind(part1, part2)

B\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

B\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

B\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#H

```{r message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

H\_predicted <- read\_csv("[filepath]/H-FLEXCS-ESTIMATES.csv")

H\_predicted <- H\_predicted[,-1]

names(H\_predicted) <- c("State", "FLEXCS\_mean", "FLEXCS\_se")

H\_observed <- read\_csv("[filepath]/H.csv")

H\_observed <- H\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

H <- inner\_join(H\_observed, H\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(H)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, warning=FALSE, message=FALSE}

#Make H tidy

source <- rep(c("NAEP"), 47)

source <- as.data.frame(source)

part1 <- cbind(H$State, source, H$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("FLEXCS"), 47)

source <- as.data.frame(source)

part2 <- cbind(H$State, source, H$FLEXCS\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

H\_tidy <- rbind(part1, part2)

H\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

H\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

H\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#API

```{r message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

API\_predicted <- read\_csv("[filepath]/API-FLEXCS-ESTIMATES.csv")

API\_predicted <- API\_predicted[,-1]

names(API\_predicted) <- c("State", "FLEXCS\_mean", "FLEXCS\_se")

API\_observed <- read\_csv("[filepath]/API.csv")

API\_observed <- API\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

API <- inner\_join(API\_observed, API\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(API)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, warning=FALSE, message=FALSE}

#Make API tidy

source <- rep(c("NAEP"), 30)

source <- as.data.frame(source)

part1 <- cbind(API$State, source, API$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("FLEXCS"), 30)

source <- as.data.frame(source)

part2 <- cbind(API$State, source, API$FLEXCS\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

API\_tidy <- rbind(part1, part2)

API\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

API\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

API\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#AINA

```{r message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

AINA\_predicted <- read\_csv("[filepath]/AINA-FLEXCS-ESTIMATES.csv")

AINA\_predicted <- AINA\_predicted[,-1]

names(AINA\_predicted) <- c("State", "FLEXCS\_mean", "FLEXCS\_se")

AINA\_observed <- read\_csv("[filepath]/AINA.csv")

AINA\_observed <- AINA\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

AINA <- inner\_join(AINA\_observed, AINA\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(AINA)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, warning=FALSE, message=FALSE}

#Make AINA tidy

source <- rep(c("NAEP"), 13)

source <- as.data.frame(source)

part1 <- cbind(AINA$State, source, AINA$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("FLEXCS"), 13)

source <- as.data.frame(source)

part2 <- cbind(AINA$State, source, AINA$FLEXCS\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

AINA\_tidy <- rbind(part1, part2)

AINA\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

AINA\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

AINA\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#TP

```{r message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

TP\_predicted <- read\_csv("[filepath]/TP-FLEXCS-ESTIMATES.csv")

TP\_predicted <- TP\_predicted[,-1]

names(TP\_predicted) <- c("State", "FLEXCS\_mean", "FLEXCS\_se")

TP\_observed <- read\_csv("[filepath]/2+.csv")

TP\_observed <- TP\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

TP <- inner\_join(TP\_observed, TP\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(TP)

```

#Create visuals and compute descriptive statistics to compare distributions

```{r, warning=FALSE, message=FALSE}

#Make TP tidy

source <- rep(c("NAEP"), 24)

source <- as.data.frame(source)

part1 <- cbind(TP$State, source, TP$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("FLEXCS"), 24)

source <- as.data.frame(source)

part2 <- cbind(TP$State, source, TP$FLEXCS\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

TP\_tidy <- rbind(part1, part2)

TP\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

TP\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

TP\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#EL

```{r message=FALSE, warning=FALSE}

#Import/Wrangle predicted and observed data

EL\_predicted <- read\_csv("[filepath]/EL-FLEXCS-ESTIMATES.csv")

EL\_predicted <- EL\_predicted[,-1]

names(EL\_predicted) <- c("State", "FLEXCS\_mean", "FLEXCS\_se")

EL\_observed <- read\_csv("[filepath]/EL.csv")

EL\_observed <- EL\_observed %>%

rename("NAEP\_mean" = Mean) %>%

rename("NAEP\_se" = SE)

EL <- inner\_join(EL\_observed, EL\_predicted, by = "State") %>%

filter(NAEP\_mean != "NA")

wmae(EL)

```

Create visuals and compute descriptive statistics to compare distributions

```{r, warning=FALSE, message=FALSE}

#Make EL tidy

source <- rep(c("NAEP"), 31)

source <- as.data.frame(source)

part1 <- cbind(EL$State, source, EL$NAEP\_mean)

names(part1) <- c("State", "method", "mean\_estimate")

source <- rep(c("FLEXCS"), 31)

source <- as.data.frame(source)

part2 <- cbind(EL$State, source, EL$FLEXCS\_mean)

names(part2) <- c("State", "method", "mean\_estimate")

EL\_tidy <- rbind(part1, part2)

EL\_tidy %>%

ggplot(aes(method, mean\_estimate, col=method)) +

geom\_boxplot() +

geom\_point(alpha = 0.5)

EL\_tidy %>%

ggplot(aes(mean\_estimate, col=method)) +

geom\_histogram() +

facet\_wrap(method~., ncol=1)

EL\_tidy %>%

group\_by(method) %>%

summarize(mean = mean(mean\_estimate),

median = median(mean\_estimate),

Std\_Dev = sd(mean\_estimate),

min = min(mean\_estimate),

max = max(mean\_estimate))

```

#\*\*Overall\*\*

```{r message=FALSE, warning=FALSE}

#Row bind subgroup datasets into an "overall dataset"

overall <- rbind(NHS, HS, SBA, BA, B, H, API, AINA, TP, EL)

#Calculate wMAE for FLEXCS across subgroups

wmae(overall)

```

### Calculate Coverage for FLEXCS Technique

#First, add the median of the NAEP-reported state-level standard deviations to each respective subgroup's data set (this permits calculation of the b-statistic).

```{r message=FALSE, warning=FALSE}

NHS$median\_sd <- 31.5

HS$median\_sd <- 32.6

SBA$median\_sd <- 30.6

BA$median\_sd <- 34.4

B$median\_sd <- 33.4

H$median\_sd <- 34.0

API$median\_sd <- 38.1

AINA$median\_sd <- 35.4

TP$median\_sd <- 35.2

EL$median\_sd <- 33.3

```

#\*\*By subgroup\*\*

#NHS

```{r message=FALSE, warning=FALSE}

NHS$b\_statisic <- abs(NHS$NAEP\_mean-NHS$FLEXCS\_mean)/NHS$median\_sd

mean(NHS$b\_statisic <= 0.20)

```

#NHS-visual

```{r message=FALSE, warning=FALSE}

NHS$lowerbound <- NHS$NAEP\_mean - 0.2\*NHS$median\_sd

NHS$upperbound <- NHS$NAEP\_mean + 0.2\*NHS$median\_sd

NHS <- NHS %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = NHS, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = NHS, aes(FLEXCS\_mean, State, col = "FLEXCS prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#HS

```{r message=FALSE, warning=FALSE}

HS$b\_statisic <- abs(HS$NAEP\_mean-HS$FLEXCS\_mean)/HS$median\_sd

mean(HS$b\_statisic <= 0.20)

```

#HS-visual

```{r message=FALSE, warning=FALSE}

HS$lowerbound <- HS$NAEP\_mean - 0.2\*HS$median\_sd

HS$upperbound <- HS$NAEP\_mean + 0.2\*HS$median\_sd

HS <- HS %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = HS, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = HS, aes(FLEXCS\_mean, State, col = "FLEXCS prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#SBA

```{r message=FALSE, warning=FALSE}

SBA$b\_statisic <- abs(SBA$NAEP\_mean-SBA$FLEXCS\_mean)/SBA$median\_sd

mean(SBA$b\_statisic <= 0.20)

```

#SBA-visual

```{r message=FALSE, warning=FALSE}

SBA$lowerbound <- SBA$NAEP\_mean - 0.2\*SBA$median\_sd

SBA$upperbound <- SBA$NAEP\_mean + 0.2\*SBA$median\_sd

SBA <- SBA %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = SBA, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = SBA, aes(FLEXCS\_mean, State, col = "FLEXCS prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#BA

```{r message=FALSE, warning=FALSE}

BA$b\_statisic <- abs(BA$NAEP\_mean-BA$FLEXCS\_mean)/BA$median\_sd

mean(BA$b\_statisic <= 0.20)

```

#BA-visual

```{r message=FALSE, warning=FALSE}

BA$lowerbound <- BA$NAEP\_mean - 0.2\*BA$median\_sd

BA$upperbound <- BA$NAEP\_mean + 0.2\*BA$median\_sd

BA <- BA %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = BA, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = BA, aes(FLEXCS\_mean, State, col = "FLEXCS prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#B

```{r message=FALSE, warning=FALSE}

B$b\_statisic <- abs(B$NAEP\_mean-B$FLEXCS\_mean)/B$median\_sd

mean(B$b\_statisic <= 0.20)

```

#B-visual

```{r message=FALSE, warning=FALSE}

B$lowerbound <- B$NAEP\_mean - 0.2\*B$median\_sd

B$upperbound <- B$NAEP\_mean + 0.2\*B$median\_sd

B <- B %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = B, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = B, aes(FLEXCS\_mean, State, col = "FLEXCS prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#H

```{r message=FALSE, warning=FALSE}

H$b\_statisic <- abs(H$NAEP\_mean-H$FLEXCS\_mean)/H$median\_sd

mean(H$b\_statisic <= 0.20)

```

#H-visual

```{r message=FALSE, warning=FALSE}

H$lowerbound <- H$NAEP\_mean - 0.2\*H$median\_sd

H$upperbound <- H$NAEP\_mean + 0.2\*H$median\_sd

H <- H %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = H, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = H, aes(FLEXCS\_mean, State, col = "FLEXCS prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#API

```{r message=FALSE, warning=FALSE}

API$b\_statisic <- abs(API$NAEP\_mean-API$FLEXCS\_mean)/API$median\_sd

mean(API$b\_statisic <= 0.20)

```

#API-visual

```{r message=FALSE, warning=FALSE}

API$lowerbound <- API$NAEP\_mean - 0.2\*API$median\_sd

API$upperbound <- API$NAEP\_mean + 0.2\*API$median\_sd

API <- API %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = API, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = API, aes(FLEXCS\_mean, State, col = "FLEXCS prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#AINA

```{r message=FALSE, warning=FALSE}

AINA$b\_statisic <- abs(AINA$NAEP\_mean-AINA$FLEXCS\_mean)/AINA$median\_sd

mean(AINA$b\_statisic <= 0.20)

```

#AINA-visual

```{r message=FALSE, warning=FALSE}

AINA$lowerbound <- AINA$NAEP\_mean - 0.2\*AINA$median\_sd

AINA$upperbound <- AINA$NAEP\_mean + 0.2\*AINA$median\_sd

AINA <- AINA %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = AINA, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = AINA, aes(FLEXCS\_mean, State, col = "FLEXCS prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#TP (2+)

```{r message=FALSE, warning=FALSE}

TP$b\_statisic <- abs(TP$NAEP\_mean-TP$FLEXCS\_mean)/TP$median\_sd

mean(TP$b\_statisic <= 0.20)

```

#TP-visual

```{r message=FALSE, warning=FALSE}

TP$lowerbound <- TP$NAEP\_mean - 0.2\*TP$median\_sd

TP$upperbound <- TP$NAEP\_mean + 0.2\*TP$median\_sd

TP <- TP %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = TP, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = TP, aes(FLEXCS\_mean, State, col = "FLEXCS prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#EL

```{r message=FALSE, warning=FALSE}

EL$b\_statisic <- abs(EL$NAEP\_mean-EL$FLEXCS\_mean)/EL$median\_sd

mean(EL$b\_statisic <= 0.20)

```

#EL-visual

```{r message=FALSE, warning=FALSE}

EL$lowerbound <- EL$NAEP\_mean - 0.2\*EL$median\_sd

EL$upperbound <- EL$NAEP\_mean + 0.2\*EL$median\_sd

EL <- EL %>%

mutate(State = fct\_reorder(State, desc(NAEP\_mean)))

ggplot() +

geom\_errorbar(data = EL, mapping=aes(xmin=lowerbound, xmax=upperbound, y=State,

col = "target interval")) +

geom\_point(data = EL, aes(FLEXCS\_mean, State, col = "FLEXCS prediction"),size=1.5) +

theme(text=element\_text(size=8, family="serif")) +

xlab("NAEP Score")

```

#\*\*Overall\*\*

#Row bind datasets

```{r message=FALSE, warning=FALSE}

#Row bind subgroup datasets into an "overall dataset"

overall <- rbind(NHS, HS, SBA, BA, B, H, API, AINA, TP, EL)

#Calculate coverage across groups

mean(overall$b\_statisic <= 0.20)

**Real World Application of FH for B subgroup (Chapter 4)**

The text that follows in courier font represents R script. Lines of script that begin with a pound/hashtag symbol (i.e., #) are comments (not code). Note, the name of local filepaths (e.g., “C:/Foldername/Foldername/…/Filename”) are replaced with [filepath] throughout. Three consecive backticks (```) demarcate the start and end of a chunk (i.e., section) of code).

**Program/File Type**: R Studio/.RMD File

**Dissertation Section: Applying the FH technique to unreported achievement data**

### Install/Load packages

```{r, message=FALSE, warning=FALSE}

if(!require(tidyverse)) install.packages("tidyverse",

repos = "http://cran.us.r-project.org")

if(!require(EdSurvey)) install.packages("EdSurvey",

repos = "http://cran.us.r-project.org")

if(!require(sae)) install.packages("sae",

repos = "http://cran.us.r-project.org")

```

### Import & Wrangle Administrative Data for B subgroup

#\*\*FER variable\*\*

#Composite measure of a state’s median household income and wealth (Family Economic Resources; “FER”)

#The income data come from page 3 of the following document on the US Census website:

https://www.census.gov/content/dam/Census/library/publications/2017/acs/acsbr16-02.pdf

#The wealth data come from page 9 of the following document on the US Census website:

https://www.census.gov/content/dam/Census/library/working-papers/2017/demo/FY2016-129.pdf

```{r, warning=FALSE, message=FALSE}

#First estabish a column for state abbreviations

State <- read\_csv("[filepath]/2+.csv")

State <- State[,1]

med\_income <- c(45182, 74165, 52062, 42530, 65087, 64598, 72121, 61882, 49852, 51753, 74451, 48728, 60094, 50896, 55172, 54520, 45541, 46106, 52111, 76596, 71146, 51584, 64188, 40910, 50642, 49924, 55474, 53320, 70813, 73242, 45710, 61311, 48420, 60944, 51610, 49062, 54748, 56207, 58826, 47790, 53746, 47818, 56139,

63794, 57565, 66916, 64764, 42620, 56115, 60570)

med\_wealth <- c(83349, 120365, 79785, 78554, 96190, 118180, 147278, 126219, 88938, 78710, 153570, 95389,

102768, 90247, 108512, 96608, 87998, 86574, 115971, 136853, 148838, 87983, 133224, 75772, 91123, 112580,

96347, 63224, 148468, 143831, 88135, 100543, 93956, 103615, 87717, 82256, 93621, 113131, 108967, 93925,

99726, 87508, 78825, 104950, 141716, 119459, 106626, 92262, 111986, 119763)

med\_income <- as.data.frame(med\_income)

med\_wealth <- as.data.frame(med\_wealth)

#bind

demographic\_df <- cbind(State, med\_income, med\_wealth)

#create composite variable (FER)

demographic\_df$FER <- (demographic\_df$med\_income + demographic\_df$med\_wealth)/2

#remove redundancies

demographic\_df <- dplyr::select(demographic\_df, "State", "FER")

#cleanup environment

rm(med\_income, med\_wealth, State)

```

#\*\*SQI variable\*\*

#The variable, SQI, is measured on a continuous scale with scores ranging 0.0 to 100.0, and reflects the average of states’ “Chance for Success” and “School Finance” ratings. The Chance for Success rating is meant to capture lifelong learning opportunities for students—beginning with early childhood, and progressing through K-12 education into adulthood. The School Finance rating is based on school spending patterns as well as how education dollars are distributed across each state (Education Week Research Center, 2015).

#The data are gathered from the following website:

#https://www.edweek.org/ew/qc/2015/2015-state-report-cards-map.html?intc=EW-QC15-LFTNAV

#NB: Hawaii is a single-district jurisdiction. As a result, it is not possible to calculate measures of financial equity, which capture the distribution of funding across districts within a state. Thus Hawaii does not receive a \*composite\* grade for school finance. Instead, its "Spending" score takes the place of school finance, which is one of the sub-components of the overall school finance measure in other states.

```{r message=FALSE, warning=FALSE}

chance\_for\_success <- c(71.2, 76.2, 71.2, 71.0, 72.8, 83.4, 87.5, 78.5, 75.5, 74.3, 80.0, 74.2, 80.1, 76.8, 84.1, 82.7, 75.3, 69.9, 80.0, 85.8, 91.9, 76.1, 87.3, 68.5, 77.4, 77.0, 83.7, 65.6, 89.6, 87.6, 67.2, 80.9, 76.2, 87.5, 76.6, 72.0, 74.8, 82.6, 79.7, 73.3, 78.9, 73.1, 73.4, 80.5, 85.7, 84.9, 80.0, 70.3, 82.6, 82.5)

school\_finance <- c(69.7, 81.7, 65.1, 73.1, 67.0, 68.8, 87.1, 83.2, 68.9, 71.2, 81.9, 59.1, 78.7, 73.3, 73.9, 75.5, 73.5, 75.8, 78.7, 86.8, 83.0, 74.6, 75.7, 66.9, 70.7, 71.7, 75.4, 62.8, 78.6, 86.8, 69.0, 88.8, 66.5, 74.3, 76.7, 66.6, 68.9, 82.2, 86.5, 70.8, 66.6, 68.4, 65.5, 61.7, 86.1, 76.9, 71.8, 89.2, 82.0, 89.3)

#bind

predictors\_df <- cbind(demographic\_df, chance\_for\_success, school\_finance)

#correct names of last two variables

names(predictors\_df) <- c("State", "FER", "chance\_for\_success", "school\_finance")

#Create SQI variable

predictors\_df$SQI <- (predictors\_df$chance\_for\_success + predictors\_df$school\_finance)/2

#remove irrelevant variables from df

predictors\_df <- dplyr::select(predictors\_df, State, FER, SQI)

#cleanup environment

rm(demographic\_df, chance\_for\_success, school\_finance)

```

#\*\*%BA variable\*\*

#This variable is operationalized as the percent of adults (25 and older) by state that have earned a bachelor’s or more advanced degree (%BA).

#Data are collected from the American Community Survey. Specifically, the following website:

#https://data.census.gov/cedsci/table?d=ACS%205-Year%20Estimates%20Data%20Profiles&table=DP02&tid=ACSDP5Y2015.DP02&y=2015&g=0400000US39,32,31,78,34,33,36,35,38,37,72,30,29,28,21,20,23,22,66,69,25,24,27,26,60,18,17,19,54,10,53,56,12,55,13,16,15,50,51,06,09,08,42,45,44,47,46,05,49,04,48,41,40,02\_0100000US&hidePreview=true&t=Educational%20Attainment&tp=true&moe=true

```{r, warning=FALSE, message=FALSE}

#Import

Percent\_BAplus\_df <- read\_csv("[filepath]/Percent\_BAplus\_df.csv")

#Merge in %BA variable

predictors\_df <- inner\_join(predictors\_df, Percent\_BAplus\_df, by = "State")

#cleanup environment

rm(Percent\_BAplus\_df)

```

#\*\*%AA variable\*\*

#Gathered from the American Community Survey (ACS), a program of the US Census. Specifically, from this website--

https://data.census.gov/cedsci/table?t=004%20-%20Black%20or%20African%20American%20alone&tid=ACSSPP1Y2015.S0201&hidePreview=true&g=0400000US02,04,05,06,08,09,10,12,13,15,16,17,18,19,20,21,22,23,24,26,25,27,28,29,31,30,32,33,34,35,36,37,38,39,40,41,42,44,46,01,45,47,48,49,51,50,53,54,55,56&moe=false&tp=true

#The data are filtered for the "Black of African American" population across across states.

#NB: This metric for 2015 (1-year estimate), is not available for 12/50 states. The data are simply missing in the ACS data set.

These states are the following: AK, HI, ID, ME, MT, NH, NM, ND, SD, UT, VT, & WY. These are states in which the prevalence of Black residents is relatively low.

#Instead, "5-year" estimates are used for these states. Estimates based on pooling data for these jurisdictions across multiple years. Specifically, from this website--

https://data.census.gov/cedsci/table?tid=ACSDT5Y2015.B05003B&t=Race%20and%20Ethnicity&vintage=2015&g=0400000US02,15,16,23,30,33,35,38,46,49,50,56&hidePreview=true&tp=false

#NB: This variable represents an estimate of the proportion of the Black population in each state born in the US.

```{r message=FALSE, warning=FALSE}

#Import data

Black\_df\_from\_Census <- read\_csv("[filepath]/Black\_df\_from\_Census.csv")

#Change from propotion to percent

Black\_df\_from\_Census$AA <- Black\_df\_from\_Census$AA\*100

#Bind AA variable

predictors\_df <- cbind(predictors\_df, Black\_df\_from\_Census$AA)

#correct variable names

names(predictors\_df) <- c("State", "FER", "SQI", "p\_BA", "AA")

#Clean up environment

rm(Black\_df\_from\_Census)

```

#\*\*Tidy up predictors dataset\*\*

```{r}

#Make sure all values are rounded to the nearest tenth (one decimal)

predictors\_df$AA <- format(round(predictors\_df$AA), nsmall=1)

#make sure all predictor variables are numeric

predictors\_df[, 2:5] <- apply(predictors\_df[, 2:5], 2, as.numeric)

```

### Combine direct estimate data and state-level predictors

```{r warning=FALSE, message=FALSE}

#import B\_DEs

B\_DEs <- read\_csv("[filepath]/B\_DEs.csv")

B\_DEs <- B\_DEs[,-1]

FH\_B\_df <- full\_join(B\_DEs, predictors\_df, by = "State")

#make sure merged dataset is 50 rows long

```

### Merge NAEP reported estimates of mean math achievent (for B) into the data set

```{r warning=FALSE, message=FALSE}

#Import the NAEP-reported means and standard errors for target values

NR\_TV\_Mean\_SE <- read\_csv("[filepath]/NR\_TV\_Mean\_SE.csv")

NR\_TV\_B\_MEAN\_SE <- dplyr::select(NR\_TV\_Mean\_SE, "State", "NR\_B\_Mean", "NR\_B\_SE")

#use full join

FH\_B\_df <- full\_join(FH\_B\_df, NR\_TV\_B\_MEAN\_SE, by = "State")

#verify that are still only 50 rows

#Export (save) FH\_df

write.csv(FH\_B\_df, "[filepath]/FH\_B\_df.csv")

```

#\*\*Free up memory\*\*

```{r warning=FALSE, message=FALSE}

rm(list = ls())

```

### Execute FH

#Import data set and combine DEs

```{r warning=FALSE, message=FALSE}

FH\_B\_df <- read\_csv("[filepath]/FH\_B\_df.csv")

FH\_B\_df <- FH\_B\_df[,-1]

FH\_B\_df <- FH\_B\_df %>%

unite("de", B\_direct\_est, NR\_B\_Mean, na.rm=T)

FH\_B\_df <- FH\_B\_df %>%

unite("se", B\_se, NR\_B\_SE, na.rm=T)

```

```{r warning=FALSE, message=FALSE}

options(digits = 5)

FH\_B\_df$de <- as.numeric(FH\_B\_df$de)

FH\_B\_df$se <- as.numeric(FH\_B\_df$se)

attach(FH\_B\_df)

B\_RW\_FH\_results <- mseFH(de ~ p\_BA + FER + SQI + AA, se^2)

FH\_B\_df$eblup <- B\_RW\_FH\_results[["est"]][["eblup"]]

FH\_B\_df$eblup\_mse <- B\_RW\_FH\_results[["mse"]]

FH\_B\_df$eblup\_se <- sqrt(FH\_B\_df$eblup\_mse)

FH\_B\_df$method <- "NAEP"

#HI

FH\_B\_df[11,11] <- "FH"

#ID

FH\_B\_df[12,11] <- "FH"

#ME

FH\_B\_df[19,11] <- "FH"

#MT

FH\_B\_df[26,11] <- "FH"

#NH

FH\_B\_df[29,11] <- "FH"

#NM

FH\_B\_df[31,11] <- "FH"

#OR

FH\_B\_df[37,11] <- "FH"

#SD

FH\_B\_df[41,11] <- "FH"

#UT

FH\_B\_df[44,11] <- "FH"

#VT

FH\_B\_df[45,11] <- "FH"

#WY

FH\_B\_df[50,11] <- "FH"

#Create final "Mean" and "SE" values (eblups for FH states)

FH\_B\_df$Mean <- ifelse(FH\_B\_df$method=="NAEP", FH\_B\_df$de, FH\_B\_df$eblup)

FH\_B\_df$StdErr <- ifelse(FH\_B\_df$method=="NAEP", FH\_B\_df$se, FH\_B\_df$eblup\_se)

```

### Create Visualization of Results

#First, descriptive statistics by method

```{r warning=FALSE, message=FALSE}

FH\_B\_df %>%

group\_by(method) %>%

summarize(Mean\_Est = mean(Mean),

Median\_Est = median(Mean),

SD = sd(Mean),

Min = min(Mean),

Max = max(Mean))

```

#Now, visualization

```{r warning=FALSE, message=FALSE}

FH\_B\_df$lower\_bound <- FH\_B\_df$Mean - 1.96\*FH\_B\_df$StdErr

FH\_B\_df$upper\_bound <- FH\_B\_df$Mean + 1.96\*FH\_B\_df$StdErr

FH\_B\_df <- FH\_B\_df %>%

mutate(State = fct\_reorder(State, desc(Mean)))

ggplot() +

geom\_errorbar(data=FH\_B\_df, mapping=aes(xmin=lower\_bound, xmax=upper\_bound, y=State, col = method)) +

xlab("NAEP score") +

theme(text=element\_text(size=8, family="serif",)) +

geom\_point(data=FH\_B\_df, aes(Mean, State), size=0.5) +

theme(axis.text.y=element\_text(colour=c("black", "red", "black", "black", "black", "black", "red", "black", "black", "black",

"black", "red", "black", "black", "black", "black", "red", "black", "red", "red",

"red", "black", "black", "red", "black", "black", "black", "red", "red", "black",

"black", "black", "black", "black", "black", "black", "red", "black", "black", "black",

"black", "black", "black", "black", "black", "black", "black", "black", "black", "black")))

```

#Extra visualization, whereby the differences between states' mean estimates and the mean of all mean estimates for the B subgroup are illustrated and color-coded by technique

```{r warning=FALSE, message=FALSE}

FH\_B\_df <- FH\_B\_df %>%

mutate(Difference = Mean - mean(Mean))

FH\_B\_df <- FH\_B\_df %>%

mutate(State = fct\_reorder(State, desc(Difference)))

ggplot(FH\_B\_df, aes(x=Difference, y=State, color=method)) +

geom\_point() +

geom\_vline(xintercept = 0) +

xlab("Difference from subgroup mean") +

theme(text=element\_text(size=8, family="serif")) +

theme(axis.text.y=element\_text(colour=c("black", "red", "black", "black", "black", "black", "red", "black", "black", "black",

"black", "red", "black", "black", "black", "black", "red", "black", "red", "red",

"red", "black", "black", "red", "black", "black", "black", "red", "red", "black",

"black", "black", "black", "black", "black", "black", "red", "black", "black", "black",

"black", "black", "black", "black", "black", "black", "black", "black", "black", "black")))```

#Extra visualization to demonstrate that none of the FH estimates of mean achievement are outliers.

```{r warning=FALSE, message=FALSE}

FH\_B\_df <- FH\_B\_df %>%

mutate(State = fct\_reorder(State, desc(Mean)))

FH\_B\_df %>%

ggplot(., aes(x=method, y=Mean, color=method)) +

geom\_dotplot(binwidth = .5, binaxis = "y", stackdir = "center") +

geom\_hline(yintercept = 273.9625, color="black") +

geom\_hline(yintercept = 245.0225, color="black") +

ylab("Score") +

theme(text=element\_text(size=14, family="serif"))

```

#Extra visualization that provides a state-by-state comparison of overall mean achievement (for all students) and mean achievement for Black students (color-coded by method-- NAEP vs FH).

```{r warning=FALSE, message=FALSE}

#Merge in achievement estimates for all\_students

All\_Students <- read\_csv("[pathfile]/All\_Students.csv")

FH\_B\_df <- FH\_B\_df %>%

inner\_join(All\_Students, by = "State")

#Redefine difference

FH\_B\_df <- FH\_B\_df %>%

mutate(Difference = Mean - All\_Students)

FH\_B\_df <- FH\_B\_df %>%

mutate(State = fct\_reorder(State, desc(Difference)))

ggplot(FH\_B\_df, aes(x=Difference, y=State, color=method)) +

geom\_point() +

xlab("Difference from mean for all students in state") +

theme(text=element\_text(size=8, family="serif")) +

theme(axis.text.y=element\_text(colour=c("red", "black", "red", "black", "black", "black", "black", "black", "black", "black",

"black", "black", "black", "black", "black", "black", "black", "black", "black", "black",

"black", "black", "black", "red", "black", "black", "black", "red", "red", "black",

"black", "red", "black", "red", "black", "black", "red", "red", "black", "black",

"red", "red", "black", "black", "black", "black", "black", "black", "black", "black")))