Gregory Bruich, Ph.D.
Dept. of Economics, Harvard University

Lab 4: The Tennessee STAR Experiment

Methods/concepts: treatment effect estimation in stratified experiments, bar graphs, multivariable regression, statistical inference, statistical vs. practical significance

LAB DESCRIPTION

The Tennessee Student/Teacher Achievement Ratio (STAR) Experiment was implemented in 1985-1986 in 79 schools, involving more than 11,600 students. Both students and teachers were randomly assigned to small and regular size classes starting in kindergarten.

In this lab, you will measure the causal effect of class size on student achievement in kindergarten, as measured by year-end test scores for $N = 5{,}710$ kindergarten children. For more details on the variables included in these data, see Table 1. A list and description of each of the Stata and R commands needed for this lab are contained in Table 2 and Table 3, respectively. For more background on the experiment, see Krueger (1999) or Chetty et al. (2011).

QUESTIONS

1. In the Tennessee STAR Experiment, *both* students and teachers were randomly assigned to small and large classes. Explain briefly why it is important to randomly assign not just students but also teachers in order to determine the causal effect of class size.

2. Using the **star.dta** file, how does average class size (*class_size*) compare in small kindergarten classes vs. regular kindergarten classes (`small == 1` vs. `small == 0`)?

3. At the end of kindergarten school year, students took four Stanford Achievement Tests: Math-SAT *math*, Reading-SAT *read*, Word-SAT *wordskill*, and Listening-SAT *listen*. It is common in education research to convert test scores into more meaningful units. One way is to generate a new variable *sat_index* that combines the exam scores into one overall metric measured in "standard deviation units" (or σ's in the lingo of education researchers) as follows:[1]

   a. For each of the four exam scores, subtract the *control group mean* and divide by the *control group standard deviation* to define four "standardized" exam scores. Some pseudo code is: *standardized math score = (math score − control_mean(math score)) ÷ control_sd(math score)*, where *control_mean(math score)* and *control_sd(math score)* are calculated for observations with `small == 0`.

      Report summary statistics (mean, standard deviation, minimum, and maximum) for the four new variables that you generated.

   b. Then generate *sat_index* as the average of these four standardized exam scores. Some pseudo code is: *sat_index* = mean(*standardized math score*, *standardized reading score*, *standardized word score*, *standardized listening score*). Report summary statistics (mean, standard deviation, minimum, and maximum).

   c. Plot a histogram of *sat_index* for small kindergarten classes (`small == 1`) and for regular kindergarten classes (`small == 0`). What do you notice in the histograms?

[1] For example, this method was used to study multiple outcomes in the Moving to Opportunity Experiment by Larry Katz and co-authors.

4. Returning to question 1, we will assess whether the data are consistent with *teachers* having been randomly assigned to classrooms by testing for balance of teacher characteristics. The STAR experiment consisted of 325 teachers, but there are 5,710 students in these data. We will conduct this and all of our subsequent analyses in this lab at the teacher-level, rather than at the student-level.

    a. Aggregate the data by *teacher_id*, so that you end up with a 325 observation data set with information on *small, school_id, teacher_id, teacher_masters, teacher_white, teacher_black, teacher_experience* as well as the mean of *sat_index* and *class_size* across all the students in the teacher's class (which we'll use in question 5). Report means for all the variables in the resulting data set <u>separately</u> for small and large classes.

    b. Estimate a linear regression (`lm` in R or `regress` in stata) of *teacher_experience* on an intercept and *small*. Use the estimated coefficient on small to report the difference in average teacher experience in small vs. large classes. Calculate a 95% confidence interval for this difference: Regression coefficient on *small* $\pm$ 1.96 $\times$ standard error.

    c. Repeat question b for *teacher_masters, teacher_white,* and *teacher_black*.

    d. Are the differences in teacher characteristics in small vs. large classes *statistically significantly different from zero*? Are they practically significant? What do you conclude about whether the random assignment was successful in balancing teacher characteristics?

5. The STAR experiment was a *stratified randomized experiment*, also known as a *randomized block experiment*, because students were randomly assigned to classes at their own school. The *strata* were therefore the school. Intuitively, students could only be randomly assigned to a class at their school and not for example a school across town. The practical implication is that it was as-if each of the 79 schools conducted their own separate experiment.

    The most standard approach to obtain one overall estimate is to modify the regressions we ran in Lab 3 by adding indicator variables for each school as additional control variables. This is now a *multivariable regression*. Recall that we only care about the regression coefficient on the variable *small*, and can safely ignore the 79 other estimated coefficients.

    a. Using the teacher-level data with 325 observations, run a multivariable regression of *sat_index* on the small class indicator *small*, controlling for school fixed effects (e.g., `regress` with `i.school_id` in Stata; or `lm` with `factor(school_id)` in R).

    b. Use the estimated coefficient on the small class indicator *small* to report an estimate of the causal effect of the experiment.[2] Calculate a 95% confidence interval for this causal effect: Regression coefficient on *small* $\pm$ 1.96 $\times$ standard error.

    c. Visualize the estimated treatment effect using a bar graph, with one bar representing the control group and a second bar representing the treatment group. The height of the bar for the control group should equal the control group mean of *sat_index*. The height of the bar for the treatment group should equal the sum of the control group mean and regression coefficient on *small* from the regression in part a. Add square brackets to the treatment group bar to visualize the 95% confidence interval from part b.

[2] To help judge the magnitudes, recall from lab 1 that most of the data will usually be within 1 standard deviation of the mean and almost all the data will usually be within 2 standard deviations of the mean.

6. The files to submit for this lab are:

    a. Your well annotated do-file/.R file replicating all your analyses above (with enough comments that a principal investigator on a research project would be able to follow and understand what each step of the code is doing). You can submit this to Gradescope.

    b. For Stata users, a log-file with the log showing the output generated by your final do-file. You can submit this file to the same Gradescope assignment as the do-file.

    c. A PDF version of the solutions to the above questions. For graphs, you can save them as .png files and insert them into the document. You can submit this file to the same Gradescope assignment as the .R script/do-file and log-file. (Please do not submit a word document: we can only read PDFs in Gradescope).

Figure 1
Tennessee Student-Teacher Achievement Ratio (STAR) Experiment

Note: Image is from Mosteller (1995) in his review of the history of and results from the Tennessee STAR experiment.

DATA DESCRIPTION, FILE: star.dta

The data consist of $N = 5{,}710$ kindergarten children in the Tennessee Student/Teacher Achievement Ratio (STAR) Experiment. For more information about the STAR Experiment and these data, see Alan B. Krueger (1999) "Experimental Estimates of Education Production Functions," *Quarterly Journal of Economics* 114(2): 497-532; and Raj Chetty, John Friedman, Nathaniel Hilger, Emmanuel Saez, Diane Schanzenbach, and Danny Yagan (2011) "How Does Your Kindergarten Classroom Affect Your Earnings? Evidence from Project STAR," *Quarterly Journal of Economics* 126(4): 1593-1660. Various excellent textbooks also present analyses of the data from the STAR Experiment, including Stock and Watson (2019, Chapter 13), Angrist and Pischke (2009, Chapter 2), and Imbens and Rubin (2015, Chapter 9).

TABLE 1

Variable Definitions

	Variable (1)	Label (2)	Obs. (3)	Mean (4)	St. Dev. (5)	Min (6)	Max (7)
1	*student_id*	Student id	5,710	n/a	n/a	n/a	n/a
2	*school_id*	Kindergarten school id	5,710	n/a	n/a	n/a	n/a
3	*teacher_id*	Kindergarten teacher id	5,710	n/a	n/a	n/a	n/a
4	*class_size*	Class size in kindergarten	5,710	20.28	3.966	12	28
5	*read*	Kindergarten reading SAT test score	5,710	436.9	31.76	358	627
6	*math*	Kindergarten math SAT test score	5,710	485.8	47.75	320	626
7	*listen*	Kindergarten listening SAT test score	5,710	537.6	33.14	397	671
8	*wordskill*	Kindergarten word study skills SAT score	5,710	434.5	36.84	331	593
9	*small*	Small classroom in kindergarten	5,710	0.302	0.459	0	1
10	*female*	Student is female	5,710	0.487	0.500	0	1
11	*freelunch*	Student receives Free or Reduced Price Lunch	5,710	0.480	0.500	0	1
12	*teacher_masters*	Kindergarten Teacher has a Master's Degree	5,710	0.354	0.478	0	1
13	*teacher_white*	Kindergarten Teacher is White	5,710	0.839	0.368	0	1
14	*teacher_black*	Kindergarten Teacher is Black	5,710	0.158	0.364	0	1
15	*teacher_experience*	Kindergarten Teacher's Years of Experience	5,710	9.326	5.762	0	27

Note: Table describes variables in star.dta.

TABLE 2
Stata Commands

STATA command	Description
*clear the workspace clear all version 17 *change working directory and open data cd "C:\Users\gbruich\Ec 50\Lab 4\" use star.dta, clear *Display all variables in the data describe *Report detailed information on all variables codebook	This code shows how to clear the workspace, change the working directory, and open a Stata data file. To change directories on either a mac or windows PC, you can use the drop down menu in Stata. Go to file -> change working directory -> navigate to the folder where your data is located. The command to change directories will appear; it can then be copied and pasted into your .do file. The describe and codebook commands will report information on what is included in the data set loaded into memory.
*Summary stats for one variable sum yvar *Observations with treatment_group equal to 1 sum yvar if treatment_group == 1 *Observations with treatment_group equal to 0 sum yvar if treatment_group == 0	We used these commands in Lab 1. These commands report means, standard deviations, minimums, and maximums for *yvar*. The first line calculates these statistics across the full sample. The other lines illustrate how to calculate these statistics for observations meeting certain criteria: when another variable in the data is equal to 1, or equal to 0.
*Code to generate standardized version of variable *Step 1:Get summary statistics for control group sum yvar if treat == 0 *Store mean and standard deviation as scalar variables scalar yvar_mean = r(mean) scalar yvar_sd = r(sd) *Step 2: define z-score gen std_yvar = (yvar - yvar_mean) / yvar_sd	These commands show how to generate a new variable that equals yvar minus the control group mean and divided by the control group standard deviation. The first line reports summary statistics for the treatment group using the sum command. Immediately after running this command, Stata stores the mean and standard deviation temporally in memory as `r(mean)` and `r(sd)`. I then store these values as scalar variables. Finally, I use these variables to generate the new stadardized variable
*Code to draw histograms for two groups **#delimit ;** twoway (hist yvar if treat == 1, fcolor(gs12%50) lcolor(gs12)) (hist yvar if treat == 0, fcolor(red%50) lcolor(red)), legend(order(1 "Small Class" 2 "Regular Class")) ylabel(none) graphregion(color(white)) bgcolor(white) xtitle("End-of-Year KG Test") ; **#delimit cr** *Save graph graph export histogram_contrast.png, replace	These commands show how to draw histograms for different groups on the same axes. Similar to the bar graph code that we used on Lab 3, we use the #delimit command to reset the character that marks the end of a command to a semi colon `;` and later set it back to a carriage return `cr`. We do this because the options for the graph are quite complicated and spill over onto multiple lines. Everything from `twoway` through the semi colon in red is one command. We create the graph by overlaying two `histogram` type twoway graphs, one for the treatment group and one for the control group. The `fcolor()` options refer to the color of the histogram bars. The `lcolor()` options refer to the outline color of the bars. Specifying `%50` after `gs12` and `red` shade the bars in partially transparent gray and red, respectively. The `graph export` command saves the graph.

*Collapse data to teacher level collapse (mean) yvar xvar, by(teacher_id teacher_experience teacher_black teacher_white teacher_masters small school_id) *Look at the first 10 rows of the data list in 1/10	These commands show how to convert the data from student-level data to teacher-level data using the collapse command.The `(mean) yvar xvar` part of the code specifies that we would like the mean of variables called yvar and xvar in our data set. The `, by(teacher_id teacher_experience teacher_black teacher_white teacher_masters small school_id)` part of the code specifies that the means should be calculated separately by teacher. I also list inside the parentheses various variables that are always constant for all students taught by the same teacher (experience, race, education, small vs. large class, and school). These variables will be included in the collapsed data set.
*Estimate linear regression regress yvar treatment_group, robust *Estimate linear regression with school fixed effects regress yvar treatment_group i.school_id, robust	The first block of code reports estimated regression coefficients from a regression of *yvar* on an intercept and a variable *treatment_group*. The `, robust` option computes standard errors that allow for unequal variances in the two groups. The second block reports estimated regression coefficients from a regression of *yvar* on an intercept, a variable *treatment_group*, and school fixed effects. The `i.school_id` creates separate indicator variables for each school identifier. The `, robust` option computes standard errors that allow for unequal variances in the two groups.
* Opportunity Insights Style Bar Graphs clear all set obs 2 gen treat = 0 replace treat = 1 in 2 *Control group mean gen y = 0.1 in 1 *Treatment group mean replace y = 0.1 + 0.4 in 2 ***Add standard error for difference in means** **gen se = .** **replace se = 0.05 in 2** ***Compute 95% confidence interval range** **gen ub = y + 1.96*se** **gen lb = y - 1.96*se** ***Look at data set we have created** **list** **#delimit ;** twoway (bar y treat if treat == 0, barwidth(0.4) color(red)) (bar y treat if treat == 1, barwidth(0.4) color(blue)) **(rcap ub lb treat, color(black))** , legend(off) xlab(0 "Control Group" 1 "Treatment Group") xtitle("") ytitle("Moved Using Experimental Voucher" " ") xsc(range(-0.3 1.3)) ylab(0(.2).5,nogrid) graphregion(color(white)) bgcolor(white) **;** **#delimit cr** graph export fig1_compliance.png, replace	These commands show how to draw an Opportunity Insights style bar graph as in Lab 3, but with the addition of 95% confidence bars for the bar corresponding to the treatment group. The new part is in purple. I use rcap twoway graph type to create the bracket showing the 95% confidence interval.

*close any possibly open log-files cap log close *start a log file log using lab4.log, replace *commands go here *close and save log file log close	These commands show how to start and close a log file, which will save a text file of all the commands and output that appears on the command window in stata. The first line is short for "capture log close" which will close any open log files, and otherwise just proceed to the next step. Then the "log using lab4.log, replace" starts the log file and changes the default in two ways. First, it changes the file type to have a .log file extension, which creates a plain text log file (which is readable in Gradesope so is important!). Second, it also adds the ", replace" option which will save over any other log file that has the same name. This is usually what you want. The rest of your lab code can go below the "log using lab4.log, replace" line. At the end of your do-file you can include the last line which is "log close" which will close and save the log-file.

TABLE 3
R Commands

R command	Description

#clear the workspace
rm(list=ls()) # removes all objects from the environment
cat('\014') # clears the console
 	
#Install and load haven package
if (!require(haven)) install.packages("haven"); library(haven)
 	
#Change working directory and load stata data set
setwd("C:/Users/gbruich/Ec 50/Lab 4")
star <- read_dta("star.dta")
 	
#Report detailed information on all variables
summary(star)
	This sequence of commands shows how to open Stata datasets in R. The first block of code clears the work space. The second block of code installs and loads the "haven" package. The third block of code changes the working directory to the location of the data and loads in star.dta. To change the working directory in R Studio, you can also use the drop down menu. Go to session -> set working directory -> choose working directory.
	
	The easiest way to open a Stata data set in R Studio is to use the drop down menu. Go to file, then import data set, and finally browse to locate the file you want to open. This option will be available after you install the haven package.
	
	The summary command will report information on what is included in the data set loaded into memory, including information on the number of missing observations NAs for each variable.

#Code to report basic summary statistics for a variable
summary(star$yvar)
 	
#Get standard deviations too
sd(star$test_score_index)
	This code shows how to basic summary statistics such as mean, minimum, maximum, and number of NAs for variable yvar. The second line shows how to get the standard deviation, too.

#Summary stats for one variable
mean(star$yvar, na.rm=TRUE)
 	
#Summary stats for observations with treatment_group == 1
#Subset data
new_df <- subset(star, treatment_group == 1)
 	
#Report mean
mean(new_df$yvar, na.rm=TRUE)
 	
#Alternatively, do it all at once using the with() function
with(subset(star, treatment_group == 1), mean(yvar, na.rm=TRUE))
 	
#Summary stats for observations with treatment_group == 0
with(subset(star, treatment_group == 0), mean(yvar, na.rm=TRUE))
 	
#Alternatively, get both means using tapply()
tapply(star$yvar, star$treatment_group, mean)
 	
#Alternatively, get both means using by()
by(star$yvar, list(star$treatment_group), mean)
	We used these commands in previous labs. These commands report means for *yvar*. The first line calculates these statistics across the full sample.
	
	The other lines illustrate how to calculate these statistics for observations meeting certain criteria: when another variable in the data is equal to 1, or equal to 0.
	
	The first few examples use the subset() function to pick out only the observations in a data frame that meet certain criteria. We can combine this with the with() function. We also have seen how to use the tapply() function to report the mean of yvar grouped by another variable treatment_group. We can also use the by() function to do the same thing.

#Code to generate standardized version of variable
 	
#Subset data frame to control group
cntrl <- subset(star, small == 0)
 	
#Store mean and standard deviation of yvar
yvar_cntrl_mean <- mean(cntrl$yvar, na.rm = T)
yvar_cntrl_sd <- sd(cntrl$yvar, na.rm = T)
 	
#Generate standardized version of yvar and add to original df
star$yvar_std <- (star$yvar - yvar_cntrl_mean) / yvar_cntrl_sd
	These commands show how to generate a new variable that equals yvar minus the control group mean and divided by the control group standard deviation. I start by subsetting the data frame to just the control units. Then I store the mean and standard deviation of yvar computed in this data frame. Finally, I generate a new variable yvar_std in the original data frame that equals yvar minus the control group mean, and divided by the control group standard deviation.

#Load tidyverse
if (!require(tidyverse)) install.packages("tidyverse"); library(tidyverse)
 	
#Draw histograms for two groups
ggplot(star, aes(x=yvar,
        fill=factor(small, labels=c("Large", "Small")),
        y=..density..)) +
  geom_histogram(alpha=0.2, position="identity") +
  labs(x = "End-of-Year KG Test Index", fill = "Class Size")
 	
#Save the graph
ggsave("histogram.png")
	These commands show how to draw histograms for different groups on the same axes. I start by loading the tidyverse library. Then I use ggplot with geom_histogram() as in Lab 1.
	
	To get two histograms on the same axes, I specify certain options in the the `aes()` part of the main `ggplot()` part of the code. I tell it to plot a histogram of the variable yvar (`x=yvar`) and to do it on the density scale (`y=..denity..`). To plot two overlapping histograms, I specify `fill = factor(small)`. The `factor()` part of this code tells ggplot that the groups are defined by whether the variable small equals 1 or 0; otherwise it will treat small as a continuous variable.
	
	I also include `, labels=c("Large", "Small")` so that the graph will be labelled with Large and Small rather than just 0 and 1.
	
	In the `geom_hist()` part of the command, I specify the option `alpha=0.2` to refer to the opacity of the bars, allowing them to be partially see through. Values of alpha range from 0 to 1, with lower values corresponding to more transparent colors.
	
	I also specify the `position="identity"` option to get both histograms on the same axes.
	
	Finally, the labs() in the last line specifies the x-axis label and a label for the legend (the fill part).

#Load tidyverse
if (!require(tidyverse)) install.packages("tidyverse"); library(tidyverse)
 	
#Create grouped table by_class
by_class <- group_by(star,
          teacher_id,
          school_id,
          small,
          teacher_masters,
          teacher_white,
          teacher_black,
          teacher_experience)
 	
#Create new data frame called classes
classes <- summarise(by_class,
          yvar = mean(yvar, na.rm = TRUE),
          xvar = mean(xvar, na.rm = TRUE))
 	
#Describe new data frame that we have created
summary(classes)
	These commands show how to convert the data from student-level data to teacher-level data. We start by loading the tidyverse library.
	
	Then we use group_by() to create a new grouped table called by_class. This function takes an existing tbl and converts it into a grouped tbl where operations are performed "by group." The first argument of group_by() is the data frame to be grouped. The other part of the code specifies the grouping is by teacher_id. I also list various other variables that are always constant for all students taught by the same teacher (experience, race, education, small vs. large class, and school). These variables will be included as variables in the collapsed data frame.
	
	Then we use summarise() function to define a new data frame with the mean of variables called yvar and xvar grouped as specified by the by_class grouped table we created earlier.

#Report summary statistics split by different groups
 	
#Various ways to do this. First tapply()
tapply(classes$yvar, classes$treatment_group, mean)
tapply(classes$yvar, classes$treatment_group, sd)
 	
#Alternatively, by()
by(classes$yvar, list(classes$treatment_group), mean)
by(classes$yvar, list(classes$treatment_group), sd)
 	
#Third - Tidyverse summarise_all()
classes %>% group_by(treatment_group) %>% summarise_all("mean")
classes %>% group_by(treatment_group) %>% summarise_all("sd")
 	
#To report all variables, add this line before running:
options(dplyr.width = Inf)
classes %>% group_by(treatment_group) %>% summarise_all("mean")
classes %>% group_by(treatment_group) %>% summarise_all("sd")
	We used these commands in Lab 1. These commands shows how to report summary statistics separately by groups defined by another variable, enabling for example summary statistics to be computed separately for a treatment group and a control group.
	
	The first example uses the tapply() function to report the mean and standard deviation of yvar grouped by another variable treatment_group.
	
	The second example uses the by() function to do the same thing.
	
	The third example uses a combination of commands from the tidyverse library to report the means and standard deviations for all the variables in the data frame all at once with summarise_all() .
	
	By default, only the first several variables will be displayed. The options(dplyr.width = Inf) line will change the default to show summary statistics for all the variables

#Load packages
if (!require(sandwich)) install.packages("sandwich"); library(sandwich)
if (!require(lmtest)) install.packages("lmtest"); library(lmtest)
 	
#Estimate linear regression
mod1 <- lm(yvar ~ treatment_group, data=classes)
 	
#Report coefficients and standard errors
coeftest(mod1, vcov = vcovHC(mod1, type="HC1"))
 	
#Add school fixed effects
mod2 <- lm(yvar ~ treatment_group + factor(school_id), data= classes)
 	
#Report coefficients and standard errors
coeftest(mod2, vcov = vcovHC(mod2, type="HC1"))
	These commands report estimated regression coefficients from a regression of *yvar* on an intercept and a variable *treatment_group*. The sandwich and lmtest packages are used to report standard errors that allow unequal variances in the two groups via the option `type="HC1"`.
	
	The second block reports estimated regression coefficients from a regression of *yvar* on an intercept, a variable *treatment_group*, and school fixed effects. The `factor(school_id)` creates separate indicator variables for each school identifier.

<pre>#Bar graph
#Load tidyverse library
if (!require(tidyverse)) install.packages("tidyverse"); library(tidyverse)
 	
#Create a data frame with three columns
#Column 1 is the height of the two bars (in blue)
#Column 2 is the standard error (in purple)
#Column 3 is the group names (in red)
df <- data.frame(c(0.001, 0.4),
                 c(NA, 0.5),
                 c("Control group", "Treatment group"))
 	
# Change name of 1st column of df to "Moved" 	
names(df)[1] <- "Moved"
 	
# Change name of 2nd column of df to "se" 	
names(df)[2] <- "se"
 	
# Change name of 3rd column of df to "Group" 	
names(df)[3] <- "Group"
 	
#Add upper bound on 95% CI
df$ub <- df$Moved + 1.96*df$se
 	
#Add lower bound on 95% CI
df$lb <- df$Moved - 1.96*df$se
 	
# Bar graph displaying results 	
ggplot(data=df, aes(x=Group, y=Moved)) +
  geom_bar(stat="identity", fill="navy") +
  geom_errorbar(aes(ymin=lb, ymax=ub), width=.1, color="red") +
  labs(y = "Moved Using Experimental Voucher")
 	
ggsave("fig1_test.png")</pre>
	These commands show how to draw an Opportunity Insights style bar graph as in Lab 3, but with the addition of 95% confidence bars for the bar corresponding to the treatment group. The new part is in purple.
	
	We use geom_errorbar() in the ggplot line to create the bracket showing the 95% confidence interval.