Economics 50/HKS SUP 135

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

Lab 1: Introductory Statistical Concepts and Statistical Computing

Methods/concepts: histograms, means, quantiles, indicator variables, rank transformation, bin scatters, and random assignment

LAB DESCRIPTION

The goal of this first lab is to acquire some familiarity working with data in statistical software.

- For FAS students, Stata 17 is available from the software download page on the Harvard University Information Technology website. If the link does not work, try changing your internet browser.
- To use R, you should install both R and RStudio on your Computer. You can download R from https://cran.r-project.org and you can download RStudio Desktop from https://www.rstudio.com/products/rstudio/download/
- Alternatively, the FAS OnDemand link along the left-hand side of our class canvas site will allow you to access Stata and R from inside of your browser window via the Harvard RCE. Note that the GUI won't work with Safari, so please use either Firefox or Chrome.

This lab uses an extract from the National Longitudinal Survey of Youth called **nlsy97.dta**. 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.

QUESTIONS

1. A *histogram* is a simple way to visualize the distribution of a variable in a data set. Strictly speaking, a histogram is drawn so that area (base times height) represents the probability of the variable falling into the range indicated by the values along the x-axis. The area of the rectangles adds up to 1. Plot a histogram of *kid_income*.

2. The sample *mean* or arithmetic average is the "balance point" of the histogram: "Think of the rectangles as weights and the x-axis of the histogram as a wooden board. Below the wooden board is a steel rod. Move the rod back and forth. Where the board balances is the mean of the histogram" (Brightman 1986). It is often represented with a bar over the variable, such as $\bar{Y}$. It equals the sum of the values of the variable Y_i for each observation divided by the total number of observations: $\bar{Y} = \frac{1}{N}\sum_{i=1}^{N} Y_i$. Calculate and report the sample mean of *kid_income*.

3. What fraction of observations have *kid_income* below its mean? Why is this fraction *not* 50%? *Hint:* Generate a new variable called *below_mean* that equals 1 if *kid_income* is (strictly) less than its mean, and 0 if it is greater than or equal to its mean. The arithmetic average of the *indicator variable* *below_mean* is the fraction of observations with *kid_income* below its mean.

4. The sample *median* is the value for which 50% of the observations have a value above the value and 50% have a value below the value. The median is also called the 50th percentile, or sometimes abbreviated p50. Calculate and report the sample median of *kid_income*.

5. The sample *standard deviation* is a measure of spread, sometimes abbreviated SD. It equals the square root of the sample *variance*, which is the sum of the squared differences between

the values of the variable Y_i for each observation and its mean $\bar{Y} = \frac{1}{N}\sum_{i=1}^{N} Y_i$ divided by the total number of observations: $SD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Y_i - \bar{Y})^2}$. Calculate and report the sample standard deviation of *kid_income*. Note that the units will be in dollars.

6. What fraction of observations are within one SD of the mean of *kid_income*? What fraction of observations are within two SDs of the mean of *kid_income*?

7. Now generate a new variable *kid_inc_rank* that equals *kid_income* converted to *percentile ranks*, normalized so that the highest rank is 100. Plot a histogram of *kid_inc_rank*. Notice that the histogram is approximately uniformly distributed between 0 and 100. Verify that the sample mean of *kid_inc_rank* approximately equals its sample median.

8. Most of social science research is concerned with the relationship *between* two or more variables. Using *binned scatter plots*, can you find some variables in Table 1 that are *linearly* related to *kid_income*? Can you find some variables that are related to *kid_income* in a *non-linear* way?

9. In Lecture 2, Professor Chetty argued that random assignment is the ideal solution to the "fundamental problem of causal inference," because it ensures comparability between treatment and control groups, and eliminates confounding. To conclude this week's lab, we will explore how random assignment works in these data.

    a. First, set the "seed" using your Harvard University ID number (`set seed 12345` in stata or `set.seed(12345)` in R), which will make your simulation reproducible, but different from your classmates' simulations. Then assign to each observation a random number drawn uniformly between 0 and 1.

    b. Generate a new variable *treatment_group* that equals 1 ("treatment group") if the number generated in part a is greater than or equal to 0.5, and otherwise equals 0 when the number is less than 0.5 ("control group"). How many observations are in the treatment group? How many are in your control group?

    c. Compute the sample mean and sample standard deviation of all the variables listed in Table 1 separately for observations in your treatment group and your control group.

    d. Go to this Google form and enter the treatment group and control group sample sizes, means, and standard deviations for the variable *parent_inc* from your simulation.

       https://forms.gle/pHHwuH9XEDpMLUeo6

    e. What is the purpose of random assignment in an experiment? In this simulation, random assignment balances all the variables listed in Table 1. Would you prefer to use random assignment to achieve comparability? Or would you prefer to allocate observations *yourself* to treatment and control groups to ensure that both groups have exactly the same composition of the variables listed in Table 1? Explain why. Include your answer on the Google form.

10. Create a do-file or .R script file that can replicate all your analyses above. This will be the final file that you submit on Canvas. The motivation for using do-files and .R script files is described below, which has been adapted from training materials used by Innovations for Poverty Action (IPA) and the Abdul Latif Jameel Poverty Action Lab (J-PAL).

WHAT ARE DO-FILES AND .R FILES AND WHY DO WE NEED ONE?

Let's imagine the following situation - you just found out you have to present your results to a partner– all the averages you produced and comparisons you made. Suppose you also found out that the data you had used to produce all these results was not completely clean, and have only just fixed it. You now have incorrect numbers and need to re-do everything.

How would you go about it? Would you reproduce everything you did for Lab 1 from scratch? Can you do it? How long would it take you to do? Just re-typing all those commands into Stata or R in order and checking them would take an hour.

An important feature of any good research project is that the results should be reproducible. For Stata and R the easiest way to do this is to create a text file that lists all your commands in order, so anyone can re-run all your Stata or R work on a project anytime. Such text files that are produced within Stata or linked to Stata are called do-files, because they have an extension .do (like intro_exercise.do). Similarly, in R, these files are called .R files because they have an extension of .R. These files feed commands directly into Stata or R without you having to type or copy them into the command window.

An added bonus is that having do-files and .R files makes it very easy to fix your typos, re-order commands, and create more complicated chains of commands that wouldn't work otherwise. You can now quickly reproduce your work, correct it, adjust it, and build on it.

Finally, do-files and .R files make it possible for multiple people to work on a project, which is necessary for collaborating with others or when you hand off a project to someone else.

DATA DESCRIPTION, FILE: nlsy97.dta

The data consist of $N = 5{,}486$ children from the National Longitudinal Survey of Youth 1997, born between 1980 and 1984. The sample and income definitions are chosen to match Chetty et al. (2014) as closely as possible. I measure the income of the children in 2013 and 2015, when they have grown up and are in their early 30s. I measure their parents' income in 1997 and 1998 in the first two waves of the survey.

TABLE 1
Variable Definitions

	Variable	Description	Obs.	Mean	St. Dev.	Min	Max
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	*id_num*	Individual identifier	5,486	n/a	n/a	3	9022
2	*kid_income*	Child's income, averaged across 2013 and 2015	5,486	$70,500	$59,552	0	$329,331
3	*incarcerated*	Child was incarcerated at least once by 2015	5,486	0.0995	0.299	0	1
4	*child_education*	Child's years of education in 2017	5,486	13.77	3.002	5	20
5	*child_college*	Child has college degree in 2017	5,486	0.295	0.456	0	1
6	*child_sat*	Child's SAT score (math plus verbal)	2,456	1,187	198.9	600	1,600
7	*parent_inc*	Parents' income, averaged across 1997 and 1998	5,486	$46,413	$46,089	0	$425,586
8	*mother_education*	Mother years of education	5,486	12.67	2.490	5	20
9	*father_education*	Father years of education	5,486	12.70	2.362	5	20
10	*female*	Child is female	5,486	0.501	0.500	0	1
11	*black*	Child is Black	5,486	0.265	0.441	0	1
12	*hispanic*	Child is Hispanic	5,486	0.199	0.399	0	1
13	*white*	Child is White	5,486	0.600	0.490	0	1
14	*region*	Census Region of residence in 1997, defined as: 1 = Northeast (CT, ME, MA, NH, NJ, NY, PA, RI, VT) 2 = North Central (IL, IN, IA, KS, MI, MN, MO, NE, OH, ND, SD, WI) 3 = South (AL, AR, DE, DC, FL, GA, KY, LA, MD, MS, NC, OK, SC, TN, TX, VA, WV) 4 = West (AK, AZ, CA, CO, HI, ID, MT, NV, NM, OR, UT, WA, WY)	5,486	2.655	0.987	1	4
15	*age2015*	Child's age in 2015	5,486	32.96	1.399	31	35
16	*cohort*	Child's year of birth	5,486	1982	1.399	1980	1984

Note: Child's SAT score is missing (indicated by a period "." in Stata) for 55% of observations in these data.

TABLE 2
Stata Commands

STATA command	Description
*clear the workspace clear all version 17 cap log close *change working directory and open data cd "C:\Users\gbruich\Ec 50\Lab 1\" use nlsy97.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.
*Histogram histogram yvar graph export histogram_yvar.png, replace *Histogram, changing number of bins to 50 histogram yvar, bin(50) graph export histogram_yvar.png, replace	These commands create and save histograms of a variable "yvar" which is a placeholder for the name of a variable in your data set. The first line creates a histogram (letting Stata decide how many bins to use). The second line saves the graph as a .png file. The second block of code changes the options by adding ", bin(50)" which will override the default binning and group the data into 50 buckets.
*Summary stats for one variable sum yvar *Summary stats for observations with wvar<=55 sum yvar if wvar <= 55 *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 *Summary stats by each value of treatment_group bys treatment_group: sum yvar *Summary statistics for all variables in data sum *Detailed summary statistics sum yvar, d	These commands report means and standard deviations 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 less than or equal to 55, equal to 1, or equal to 0. In these examples, the following are placeholders for variables in your data: yvar, treatment_group, wvar, xvar. To report summary statistics for multiple variables, list them next to each other with no commas: *sum yvar wvar xvar* To report means, standard deviations, medians, and other quantiles and statistics use the ", d" option to report detailed summary statistics
*Create new indicator variables gen dvar= 0 replace dvar = 1 if abs(xvar - 42.1) < 10 gen dvar= 0 replace dvar = 1 if xvar >= 0.5	These commands show how to generate a new indicator variable called dvar. In the first example, dvar equals 1 if another variable xvar is within 10 units of the number 42.1. We start by generating a variable that always equals 0. Then we replace it with 1 for observations that meet the criteria that the absolute value of the difference between xvar minus 42.1 is less than 10. The function abs() calculates the absolute value. The second block of code illustrates another example, where the indicator instead equals 1 when xvar is greater than or equal to 0.5.

*Create variable in percentile ranks egen yvar_rank = rank(yvar) *Get maximum rank, automatically stored as r(max) sum yvar_rank *Normalize rank so that maximum is 100 replace yvar_rank = 100* yvar_rank /r(max)	These commands show how to convert a variable yvar into percentile ranks, normalized so that the highest rank is 100. We start using egen and the rank() function to generate a new variable that rank orders yvar. Then to normalize the variable, we divide it by the maximum rank and multiply by 100. The maximum rank is saved temporarily as r(max) after the sum command.
*install bin scatter command ssc install binscatter *Bin scatter plot – connected dots binscatter wage age, linetype(connect) graph export figure2_connected.png, replace *Bin scatter plot – linear best fit line binscatter wage age, linetype(lfit) graph export figure2_linear.png, replace	These commands show how to create binned scatter plots. The first line installs the command from the SSC. The second block of code shows how to create a binned scatter plot where a variable yvar is along the y-axis and a variable xvar is along the x-axis. It will connect the dots with a line. The third block of code shows how to create a binned scatter plot where a variable yvar is along the y-axis and a variable xvar is along the x-axis. It will also plot a linear best fit line. The commands beginning with "graph export" save the graphs as .png files.
*Set seed to make reproducible set seed 505050505 *Generate uniform random number between 0-1 gen random_number = runiform()	These commands show how to randomly assign a number between 0 and 1 to each observation. We start by setting the "seed". Then we generate a new variable called random_number that equals runiform(). runiform() is a function that creates a pseudo random number that has a uniform distribution.
*start a log file log using lab1.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.

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 1")
nlsy <- read_dta("nlsy97.dta")
	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 nlsy97.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.

#Histogram in base R
hist(nlsy$yvar, probability = T)
 	
#Histogram using ggplot
if (!require(tidyverse)) install.packages("tidyverse"); library(tidyverse)
if (!require(ggplot2)) install.packages("ggplot2"); library(ggplot2)
 	
ggplot(nlsy) + geom_histogram(aes(x=yvar, y=..density..))
ggsave("histogram_yvar.png")
 	
#Use 50 bins, overriding default
ggplot(nlsy) + geom_histogram(aes(x=kid_inc_rank, y=..density..),
bins = 50)
	These commands create and save histograms of a variable "yvar" which is a placeholder for the name of a variable in your data set. The first line creates a histogram (letting R decide how many bins to use) using base R.
	
	The second block of code shows how to do this using ggplot. First start by installing the tidyverse library. Then use ggplot to draw the graph. The ggsave() line saves the graph as a .png file.
	
	The last line overrides the default to show 50 bins by adding the bins = 50 option.

# summary stats, unweighted 	
summary(nlsy$yvar)
mean(nlsy$yvar, na.rm=TRUE)
sd(nlsy$yvar, na.rm=TRUE)
median((nlsy$yvar, na.rm=TRUE)
	These commands show how to calculate unweighted summary statistics. The variable yvar is a placeholder for a variable in a data frame called nlsy. The ", na.rm=TRUE" argument takes care of missing values.

#Create new indicator variable called dvar nlsy$dvar <- 0 nlsy$dvar[which(nlsy$xvar >= 0.5)] <- 1 #Alternatively, use ifelse() function nlsy$dvar <- ifelse(nlsy$dvar >= 0.5, 1, 0) #Another example: nlsy$dvar <- ifelse(abs(nlsy$xvar - 42.1) < 10, 1, 0)	These commands illustrate an example of how to generate an indicator that equals 1 when xvar is greater than or equal to 0.5. The first two lines show how to start a variable that always equals 0 and then replace it equal to 1 if a logical condition is satisfied (i.e., xvar >= 0.5) An alternative way to do it uses the ifelse() function, which takes three arguments: the logical condition, the value if the condition is satisfied, and the value of the condition is not satisfied. The last example shows how to generate a new indicator variable called dvar that equals 1 if another variable xvar is within 10 units of the number 42.1. I use the ifelse() function. The first argument is the criteria that the absolute value of the difference between xvar minus 42.1 is less than 10. The function abs() calculates the absolute value.

#Various ways to do this. First tapply()
tapply(nlsy$yvar, nlsy$treatment_group, mean)
tapply(nlsy$yvar, nlsy$treatment_group, sd)
 	
#Alternatively, by()
by(nlsy$yvar, list(nlsy$treatment_group), mean)
by(nlsy$yvar, list(nlsy$treatment_group), sd)
 	
#Third - Tidyverse summarise_all()
nlsy %>% group_by(treatment_group) %>% summarise_all("mean")
nlsy %>% group_by(treatment_group) %>% summarise_all("sd")
 	
#To report all variables, add this line before running:
options(dplyr.width = Inf)
nlsy %>% group_by(treatment_group) %>% summarise_all("mean")
nlsy %>% group_by(treatment_group) %>% summarise_all("sd")
	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

#Create variable in percentile ranks
nlsy$yvar_rank <- rank(nlsy$yvar)
 	
#Normalize rank so that maximum is 100
nlsy$yvar_rank <- 100*nlsy$yvar_rank /max(nlsy$yvar_rank)
 	
# Create Function that will Calculate Percentile Ranks with NAs 	
 	
#Define function for percentile ranking
percentile_rank<-function(variable){
 	
#Convert to ranks, taking care of potential missing values
r <- ifelse(is.na(variable), NA, rank(variable, ties.method = "average"))
 	
#Return percentile rank = rank normalized so max is 100
100*r/max(r, na.rm = T)
}
 	
#Example using Function to Define ranks
nlsy$yvar_rank <-with(nlsy, percentile_rank(yvar))
	These commands show how to convert a variable yvar into percentile ranks, normalized so that the highest rank is 100. We start using the rank() function to generate a new variable that rank orders yvar. Then to normalize the variable, we divide it by the maximum rank and multiply by 100. The code uses the max() function in R in the denominator to do the normalization.
	
	Unfortunately, the rank() function does not work as desired for data with missing values (NAs). But we can create our own function to do what we want that will work as intended in more complex data sets. This second block of code shows how to define a new function called percentile_rank() that will generate percentile ranks that assign missing values to NAs, and returns the percentile rank normalized to have a maximum rank of 100.
	
	The last line shows how to use the function to create the variable yvar_rank. The with() function in R takes two arguments: a data frame and an expression. The data frame argument is nlsy and the expression applies the new function we wrote to the variable yvar: percentile_rank(yvar).

#install statar package if (!require(statar)) install.packages("statar"); library(statar) #Bin scatter plot – connected dots ggplot(nlsy, aes(x = xvar , y = yvar)) + stat_binmean(n = 20, geom = "line") + stat_binmean(n = 20, geom = "point") #Bin scatter plot – linear best fit line ggplot(nlsy, aes(x = xvar , y = yvar)) + stat_smooth(method = "lm", se = FALSE) + stat_binmean(n = 20, geom = "point")	These commands show how to create binned scatter plots. The first line installs the statar package so that we can use the stat_binmean() function with ggplot. The second block of code shows how to create a binned scatter plot where a variable yvar is along the y-axis and a variable xvar is along the x-axis. It will connect the dots with a line. The third block of code shows how to create a binned scatter plot where a variable yvar is along the y-axis and a variable xvar is along the x-axis. It will also plot a linear best fit line.

#Set seed so that simulations are replicable
set.seed(505050505)
 	
#Uniformly distributed random number between 0 and 1
nlsy$random_number <- runif(length(nlsy$yvar))
	These commands show how to randomly assign a number between 0 and 1 to each observation. We start by setting the "seed".
	
	Then we generate a new variable called random_number that equals runif(). runif() is a function that Stata uses to create a pseudo random number that has a uniform distribution.
	
	Inside the argument of runif() is length(nlsy$yvar), which counts the number of observations. The length() function returns the length of an object in R.

#Report total number of observations in treatment group
sum(nlsy$treatment_group)
 	
#Report total number of observations in control group
sum(1-nlsy$treatment_group)
	This code shows how to count how many observations have treatment_group equal to 1 and how many observations have treatment_group equal to 0.
	
	The sum() command adds up the indicator variable across all observations, yielding the total number of observations with treatment_group equal to 1. Summing the values of 1 minus treatment_group gives the total number of observations with treatment_group equal to 0.

sink(file="lab1_log.txt", split=TRUE) sink()	The first line starts a log file. The last line closes and saves the log file.