In-Class Lab 10

ECON 4223 (Prof. Tyler Ransom, U of Oklahoma)

February 21, 2019

The purpose of this in-class lab is to use R to practice computing weighted statistics and appropriately correcting for clustering in standard errors. The lab should be completed in your group. To get credit, upload your .R script to the appropriate place on Canvas.

For starters

First, install the `NHANES` package. You won't need the `wooldridge` package for this lab.

Open up a new R script (named `ICL10_XYZ.R`, where `XYZ` are your initials) and add the usual "preamble" to the top:

```
# Add names of group members HERE
library(tidyverse)
library(broom)
library(car)
library(lmtest)
library(magrittr)
library(NHANES)
library(estimatr)
```

Load the data

We'll use a well-known health data set called the National Health and Nutrition Examination Survey (NHANES). The data set contains 78 variables detailing the demographics and health status of 20,293 Americans.

The NHANES is *not* a random sample of the US population. Instead, the survey oversamples certain demographic groups so that it can obtain more precise measurements of their health status.

```
df <- as_tibble(NHANESraw)
```

Check out what's in the data by typing `View(df)` **in the console**. (Using `glimpse()` in this case is probably not a good idea because there are so many variables, but you should feel free to test it out and see if you find it useful.)

The main variables we're interested in are: BMI, SleepHrsNight, Age, Education, Gender, Race1, and WTINT2YR (a variable indicating each person's sampling weight). We also only want to look at observations only in the 2009-2010 survey wave.

Restrict to observations to the 2009-2010 survey wave and age 19+

Use a `filter()` statement to keep observations where `SurveyYr` equals `2009_10`. Because `SurveyYr` is a factor, the code is a bit tricky, so I'll put it below for your reference:[1]

[1] The trick is to convert the factor to a string variable so that you are able to match the label of the factor. Similarly, if the lables of the factor are integers, you should use `as.numeric(SurveyYr)==2009` in the `filter()` statement.

```
df %<>% filter(as.character(SurveyYr)=='2009_10' & Age>=19)
```

Get rid of variables you won't use

Use a `select()` statement to keep only the variables that will be used (refer to the list above; I won't put the code here)

Drop missing values

Finally, get rid of observations with missing BMI, Education, or Sleep:

```
df %<>% filter(!is.na(Education) & !is.na(BMI) & !is.na(SleepHrsNight))
# or
df %<>% drop_na(Education,BMI,SleepHrsNight)
```

Look at the data to make sure the code worked as expected. You should now have 5,971 observations and 7 variables.

Computing weighted summary stats

Suppose you are interested in the average BMI of the US population. You might be tempted to type

```
mean(df$BMI)
```

but you know that NHANES is not a random sample, and that there are sampling weights included in the data.

To compute the population average, you use the sampling weights like so:

```
weighted.mean(df$BMI, w=df$WTINT2YR)
```

1. How different are your two answers? What is the relevant population, given that you deleted so many observations on the way to estimating the population mean?

Regression-adjusted summary stats

Suppose now you are interested in the male-female BMI differential, correcting for other factors (like education, race, and sleep). The easiest way to do this to estimate a regression model

$$BMI = \beta_0 + \beta_1 male + \beta_2 race + \beta_3 sleep + u$$

```
est.unweighted <- lm(BMI ~ Gender + Race1 + SleepHrsNight, data=df)
```

2. Interpret your estimate of the `Gendermale` coefficient.

Now add weights so that your estimate lines up with the true difference in BMI in the population:

```
est.weighted <- lm(BMI ~ Gender + Race1 + SleepHrsNight, weights=WTINT2YR, data=df)
```

3. How does your answer change when you supply weights? What do you now conclude about the population gender differential in BMI?
4. What do you notice about differentials in race and amount of sleep?

Inference with Cluster-Robust Standard Errors

Now let's obtain standard errors from a different data set and regression model that are robust to heteroskedasticity. To do so, we use the `lm_robust()` function from the `estimatr` package.

Load new data

First load the data, which is a CSV file from my website:

```
df.auto <- read_csv('https://tyleransom.github.io/teaching/MetricsLabs/auto.csv')
```

The data set contains specifications for 74 different makes of automobiles. Estimate the following regression model:

$$\log(price) = \beta_0 + \beta_1 weight + \beta_2 foreign + u$$

```
df.auto %<>% mutate(log.price = log(price), foreign = as.factor(foreign))
est.auto <- lm(log.price ~ weight + foreign, data=df.auto)
```

Regular standard errors:

```
tidy(est.auto)
```

Now use the heteroskedasticty-robust SEs from last lab:

```
est.rob.auto <- lm_robust(log.price ~ weight + foreign, data=df.auto)
tidy(est.rob.auto)
```

Now use the cluster-robust SEs:

```
est.clust.auto <- lm_robust(log.price ~ weight + foreign, data=df.auto,
                            clusters=df.auto$manufacturer)
```

Notice that the SEs on each of the coefficients get bigger with each additional robustness option. The reason for this is that price is correlated within auto manufacturer (due to branding effects).

Finally, you can do an F-test as follows:

```
linearHypothesis(est.clust.auto,c("weight=0","foreignForeign=0"))
```

References