Capstone Exercise

Reproducing Edelman, Luca, and Svirsky (AEJ Applied 2017)

February 28, 2018

Introduction

For the capstone exercise, you will be applying what you've learned over the past several modules to conduct a replication of a recent journal article. Specifically, you will be reproducing the main results from "Racial Discrimination in the Sharing Economy: Evidence from a Field Experiment," by Benjamin Edelman, Michael Luca, and Dan Svirsky and published in the *American Economic Journal: Applied Economics* last April.

In this article, Edelman, Luca, and Svirsky conduct an experiment wherein they apply for Airbnb apartments using guest names that have distinctively African American names. Using this experiment, they then investigate racial discrimination based on a number of host and location characteristics.

Data Preparation

Preliminaries

- To begin the replication, download the paper and data from here.
- Create a new Project for the exercise in RStudio with version-control (preferably connected to a GitHub repository).
  - You should put the data from Edelman, Luca, and Svirsky in something like a *data* subfolder in the project directory.
- Within the project, create a new RMarkdown document.
- As you go through each step, either add in the instructions found here or add your comments explaining what you are doing.
- Commit (and Push) your work every 15 minutes or so.

Importing Data

- If you are not working inside of RMarkdown / RStudio Project, set the work directory.
- Import the data set "main_data.csv"

```
# Import Data
#import delimited "main_data.csv", delimiter(comma) bindquote(strict)
main_data <- import("./data/main_data.csv", header=FALSE)
```

Recode missing values and convert to a tibble

- Using a map or for-loop, recode the following values to missing throughout the dataset: "\N", "Null", "-1".
- Then convert the dataframe to a tibble.

```
main_data <- main_data %>% map(na_if,"Null")
main_data <- main_data %>% map(na_if,"NULL")
main_data <- main_data %>% map(na_if,"\\N")
main_data <- main_data %>% map(na_if,"-1")
main_data <- main_data %>% map(na_if,-1)
main_data <- main_data %>% map(na_if,"N")
main_data <- main_data %>% as.tibble()
```

Rename the variables

Import the "datanames" csv file and assign it as the column names of the main dataset.

```
datanames <- c("host_response", "response_date", "number_of_messages",
  "automated_coding", "latitude", "longitude", "bed_type", "property_type",
  "cancellation_policy", "number_guests", "bedrooms", "bathrooms",
  "cleaning_fee", "price", "apt_rating", "property_setup", "city", "date_sent",
  "listing_down", "number_of_listings", "number_of_reviews", "member_since",
  "verified_id", "host_race", "super_host", "host_gender", "host_age",
  "host_gender_1", "host_gender_2", "host_gender_3", "host_race_1",
  "host_race_2", "host_race_3", "guest_first_name", "guest_last_name",
  "guest_race", "guest_gender", "guest_id", "population", "whites", "blacks",
  "asians", "hispanics", "available_september", "up_not_available_september",
  "september_price", "census_tract", "host_id", "new_number_of_listings")
datanames <- as.matrix(datanames)
export(datanames, "./data/datanames.csv")
colnames(main_data) <- datanames
```

Convert columns to correct class

Using for-loops, change the class of columns in the main dataset as follows:

- **Covert to numeric columns:** 3-6, 10-14, 19-21, 39-46, and 49.
- **Convert to factor columns:** 1, 7-9, 15-17, 23-33, 36-38, and 47.

```
# Convert numeric columns
for (i in c(3:6,10:14,19:21,39:46,49)) {
  main_data[[i]] <- as.numeric(main_data[[i]])
}

# Convert factor columns
for (i in c(1, 7:9, 15:17, 23:33, 36:38, 47)) {
  main_data[[i]] <- as.factor(main_data[[i]])
}
```

Set reference groups

- For the variable *guest_race*, set the reference group to the value "white".
- For the variable *guest_gender*, set the reference group to the value "male".

```
main_data$guest_race <- main_data$guest_race %>% relevel(ref="white")
main_data$guest_gender <- main_data$guest_gender %>% relevel(ref="male")
```

Create a guest_name by city variable to identify individual guests

For clustering of standard errors in the regression analysis, create a variable *namebycity* that concatenates the values from *guest_first_name* and *city*.

- Use the paste() function to do this.

```
main_data <- main_data %>% mutate(
  namebycity = paste(guest_first_name,city))
```

Import and merge survey results

- Import the file "name_survey_results.xlsx"
- Merge in additional variables from this dataset for observations from the main dataset, using the key *guest_first_name*.

```
# Import
survey_results <- import("./data/name_survey_results.xlsx")

# Merge
merged_data <- left_join(main_data, survey_results, by="guest_first_name")
```

Change the values of *guest_race_continuous*

Change the value of *guest_race_continuous* by subtracting one from it's current value, so that it's range is 0 to 1 instead of 1 to 2.

```
# replace guest_race_continuous = guest_race_continuous - 1

merged_data <- merged_data %>%
  mutate(guest_race_continuous = guest_race_continuous - 1)
```

Make host race and sex variables

Create the following indicator variables:

- *host_race_black* equal to 1 if the host's race is "black" according to the *host_race* variable.
- *host_race_white* equal to 1 if the host's race is "white" according to the *host_race* variable.
- *host_male* equal to 1 if the host's race is "M" according to the *host_gender* variable.

```
merged_data <- merged_data  %>% mutate(
  host_race_black = ifelse(host_race=="black", 1,0),
  host_race_white = ifelse(host_race=="white", 1,0),
  host_male = ifelse(host_gender =="M", 1,0)
)
```

Make a categorical host age variable

Make a categorical host age variable, *host_age_cat*, with values as follows:

- Value of 0 if *host_age* is equal to any of "young","young/UU", "UU/young", "young/NA", or "NA/young".
- Value of 1 if *host_age* is equal to any of "middle/young",or "young/middle".

- Value of 2 if *host_age* is equal to any of “middle”,“middle/UU”,“UU/middle”,“middle/NA”, or “NA/middle”.
- Value of 3 if *host_age* is equal to any of “middle/old” or “old/middle”.
- Value of 4 if *host_age* is equal to any of “old/middle”,“old”,“old/UU”,“UU/old”,“old/NA”, or “NA/old”.

```r
merged_data <- merged_data %>%
  mutate(host_age_cat = case_when(
    host_age %in% c("young","young/UU", "UU/young",
                    "young/NA", "NA/young") ~ 0,
    host_age %in% c("middle/young","young/middle") ~ 1,
    host_age %in% c("middle","middle/UU","UU/middle",
                    "middle/NA","NA/middle") ~ 2,
    host_age %in% c("middle/old","old/middle") ~ 3,
    host_age %in% c("old/middle","old","old/UU",
                    "UU/old","old/NA","NA/old") ~ 3,
    ))
```

Make binary variables for other host characteristics:

Create the following binary variables:

- *ten_reviews* indicating whether or not *number_of_reviews* is greater than or equal to 10.
- *five_star_property* indicating whether or not *apt_rating* is equal to five.
- *multiple_listings* indicating whether or not *number_of_listings* is greater than 1.
- *shared_property* indicating whether *property_setup* is ***either*** “Private Room” or “Shared Room”.
- *shared_bathroom* for the conditions that *bathrooms* is less than 1.5 and the property is shared according to your variable above.
- *has_cleaning_fee* indicating whether *cleaning_fee* is not missing.
- *strict_cancellation* indicating whether *cancellation_policy* is equal to “Strict”.
- *young* indicating whether *host_age_cat* is equal to zero.
- *middle* indicating whether *host_age_cat* is equal to one or two.
- *old* indicating whether *host_age_cat* is equal to three or four.

```r
merged_data <- merged_data %>%
  mutate(ten_reviews = ifelse(number_of_reviews >= 10, "yes", "no"),
    five_star_property = ifelse(apt_rating == 5, "yes", "no"),
    multiple_listings = ifelse(number_of_listings > 1, "yes", "no"),
    shared_property = ifelse((property_setup == "Private Room" |
                              property_setup == "Shared Room"), "yes", "no"),
    shared_bathroom = ifelse((shared_property=="yes"
                            & bathrooms<1.5), "yes", "no"),
    has_cleaning_fee = ifelse(!is.na(cleaning_fee), "yes", "no"),
    strict_cancellation = ifelse(cancellation_policy == "Strict", "yes", "no"),
    young = ifelse(host_age_cat == 0, "yes", "no"),
    middle = ifelse((host_age_cat == 1 | host_age_cat == 2), "yes", "no"),
    old = ifelse((host_age_cat == 3 | host_age_cat == 4), "yes", "no"))
```

Crate a simplified host response variable

Create a new variable *simplified_response* that has the following values:

- “No Response” if *host_response* is equal to NA.
- “Yes” if *host_response* is equal to 1.
- “No” if *host_response* is equal to 0.

- "Conditional Yes" if *host_response* is equal to 4, 5,6, 7 or 8.
- "Conditional No" if *host_response* is equal to 2,3, 9, 10, or 11.

```
merged_data <- merged_data %>% mutate(simplified_response = case_when(
  is.na(host_response) ~  "No Response",
  host_response == 0 ~  "No",
  host_response == 1 ~  "Yes",
  host_response %in% c(4,5,6,7,8) ~ "Conditional Yes",
  host_response %in% c(2,3,9,10,11) ~ "Conditional No"
  ))
merged_data$simplified_response <- as.factor(merged_data$simplified_response)
```

Create a binary host response variable

Create a new variable *yes* that that is equal to: - 1 if if *host_response* is equal to 1,4, or 6. - 0 if if *host_response* is equal to 0,2,3,7,8,9,10,11,12, or if *host_response* is missing.

```
merged_data <- merged_data %>% mutate(yes = case_when(
  host_response %in% c(1,4,6) ~ 1,
  host_response %in% c(0,2,3,7,8,9,10,11,12) ~ 0,
  is.na(host_response) ~ 0
  ))
```

Drop observations in Tampa and Atlanta

The experiment could not be completed in Tampa or Atlanta, so drop the observations where *city* is equal to either of these two values.

```
merged_data <- merged_data %>% filter(!((city == "Tampa") | (city == "Atlanta")))
```

Merge in data on past guests

- Import the dataset "hosts.dta" and add in variables from this dataset to the observations from the main dataset using the key *host_id*.

```
hosts <- import("./data/hosts.dta")
hosts$host_id <- as.character(hosts$host_id)
final_data <- merged_data %>% left_join(hosts, by="host_id")
```

Main Analysis

Reproduce estimates from Table 2

- Perform separate regressions corresponding to each of the columns of Table 2 and save the regressions objects.
- For the first regression:
  - Obtain the cluster-robust standard errors and test-statistics using the function cluster.vcov from the multiwayvcov package.
  - The syntax of cluster.vcov is:

```r
cluster_obj <- cluster.vcov(reg_object, cluster=data$clustervar)
```

- - Print a *tidy-ed* version of the estimates from each regression using the cluster-robust standard errors.
- After the first regression:
  - create a function that takes a regression objects, obtains the clustered-standard errors, performs t-tests using the clustered standard errors, and then saves the tidy-ed version of those estimates.
  - Use the function to get the estimates from columns 2 and 3.

```r
# Column 1
table2_c1 <- lm(data = final_data, yes ~ guest_race)
cluster_t2c1 <-  cluster.vcov(table2_c1, cluster=final_data$namebycity)
t2s1_clustered <- tidy(coeftest(table2_c1, cluster_t2c1))
t2s1_clustered
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.4880915	0.0119568	40.821398	0.0e+00
guest_raceblack	-0.0797825	0.0170352	-4.683394	2.9e-06

```r
# Function to get cluster-robust results
clustered_tstats <- function(regobj) {
  cluster_est <-  cluster.vcov(regobj, cluster=final_data$namebycity)
  results_clustered <- tidy(coeftest(regobj, cluster_est))
  results_clustered
}

# Column 2
table2_c2 <- lm(data = final_data, yes ~ guest_race + host_race_black + host_male)
t2c2_clustered <-  clustered_tstats(table2_c2)
t2c2_clustered
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.5035630	0.0136890	36.785978	0.0000000
guest_raceblack	-0.0866329	0.0175020	-4.949887	0.0000008
host_race_black	0.0663135	0.0233024	2.845786	0.0044463
host_male	-0.0531374	0.0138374	-3.840128	0.0001243

```r
# Column 3
table2_c3 <- lm(data = final_data, yes ~ guest_race + host_race_black +
                  host_male + multiple_listings + shared_property + ten_reviews
                + log(price))
t2c3_clustered <-  clustered_tstats(table2_c3)
t2c3_clustered
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.8118504	0.0600317	13.523706	0.0000000
guest_raceblack	-0.0926542	0.0179445	-5.163377	0.0000003
host_race_black	0.0901740	0.0237494	3.796899	0.0001481
host_male	-0.0467097	0.0142251	-3.283600	0.0010313

term	estimate	std.error	statistic	p.value
multiple_listingsyes	0.0537906	0.0145758	3.690407	0.0002260
shared_propertyyes	-0.0701524	0.0153722	-4.563583	0.0000051
ten_reviewsyes	0.1184834	0.0144882	8.177907	0.0000000
log(price)	-0.0730090	0.0113730	-6.419501	0.0000000

Reproduce Figure 2

Create a grouped bar plot of host responses by Race, as in Figure 2 of Edelman, Luca, and Svirsky.

- First create a summary data frame that counts the number of observations grouped by *guest_race* and *simplified_response*.
- Then create a bar plot that is ***grouped** by specifying* fill* color according to *guest_race* inside of the base aesthetic, with the argument `position="dodge"` inside of **geom_bar** (otherwise you'd get a stacked bar plot).

```r
figure2_df <- final_data %>% filter(!is.na(simplified_response)) %>%
  group_by(guest_race, simplified_response) %>%
  summarize(responses = n())
figure2_df$simplified_response <- factor(figure2_df$simplified_response,
                    levels(figure2_df$simplified_response)[c(5,2, 4,1,3)])

# Grouped
ggplot(figure2_df, aes(fill=guest_race, y=responses, x=simplified_response)) +
    geom_bar(position="dodge", stat="identity") +
  ggtitle("Figure 2: Host Reponses by Race") + xlab("") + ylab("")
```

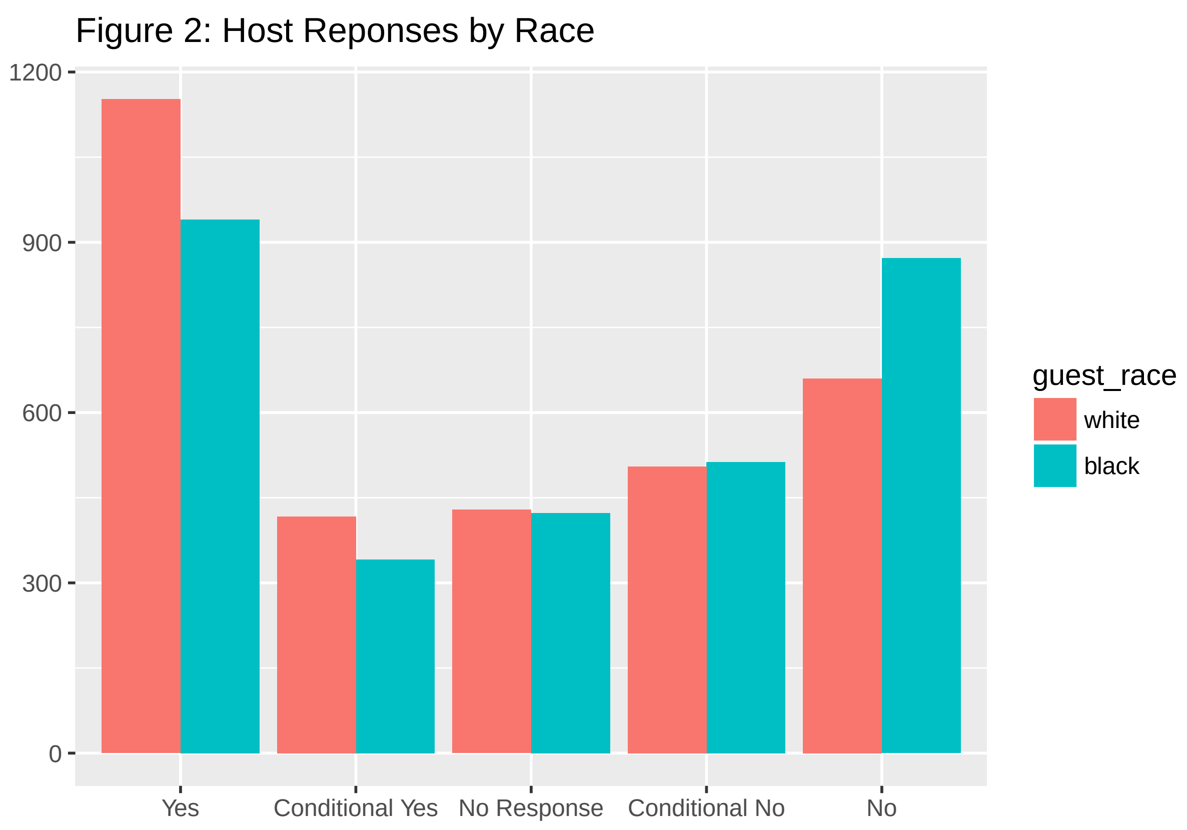

Table 5. Are Effects Driven by Host Characteristics?

Reproduce columns 4 and 5 from Table 5 (again using your helper function for cluster-robust test statistics).

- "Host has 1+ reviews from an African American guest" is represented by the *any_black* variable.

```r
# Column 1
table5_c1 <- lm(data = final_data, yes ~ guest_race + shared_property +
                  shared_property*guest_race)
table5_c1_clustered <-  clustered_tstats(table5_c1)
table5_c1_clustered
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.5000000	0.0128412	38.9371403	0.0000000
guest_raceblack	-0.0771176	0.0202810	-3.8024620	0.0001446
shared_propertyyes	-0.0113788	0.0142875	-0.7964216	0.4258177
guest_raceblack:shared_propertyyes	-0.0148820	0.0252719	-0.5888744	0.5559671

```r
# Column 2
table5_c2 <- lm(data = final_data, yes ~ guest_race + multiple_listings +
                  multiple_listings*guest_race)
table5_c2_clustered <-  clustered_tstats(table5_c2)
table5_c2_clustered
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.4564509	0.0144848	31.5123195	0.0000000
guest_raceblack	-0.0794017	0.0192992	-4.1142531	0.0000393
multiple_listingsyes	0.0994374	0.0227288	4.3749510	0.0000123
guest_raceblack:multiple_listingsyes	-0.0037954	0.0267263	-0.1420092	0.8870773

```
# Column 4
table5_c5 <- lm(data = final_data, yes ~ guest_race + young +
                  young*guest_race)
table5_c5_clustered <-  clustered_tstats(table5_c5)
table5_c5_clustered
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.5057263	0.0129932	38.9224319	0.0000000
guest_raceblack	-0.0863513	0.0191020	-4.5205314	0.0000063
youngyes	-0.0352418	0.0197470	-1.7846601	0.0743715
guest_raceblack:youngyes	0.0000678	0.0259588	0.0026133	0.9979150

```
# Column 5
table5_c5 <- lm(data = final_data, yes ~ guest_race + any_black +
                  any_black*guest_race)
table5_c5_clustered <-  clustered_tstats(table5_c5)
table5_c5_clustered
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.4602324	0.0112774	40.809984	0.0000000
guest_raceblack	-0.0945876	0.0177580	-5.326479	0.0000001
any_black	0.0962989	0.0137393	7.009011	0.0000000
guest_raceblack:any_black	0.0560169	0.0232483	2.409503	0.0160031

[Bonus] Table 6: Are Effects Driven by Location Characteristics?

Data Preperation

Make a variable that lists the number of properties within the census tract

- Using the *group_by* and *summarize* function, first create a variable that tallies the number of Airbnb listings in each tract using the summary condition:

```
tract_listings = sum(latitude > 0)
```

- Use a join operation to add this data to the main dataset.

```
tract_listings_df <- final_data %>% group_by(census_tract) %>%
  summarize(tract_listings = sum(latitude > 0))

final_data <- final_data %>% left_join(tract_listings_df, by="census_tract")
```

Generate Price Variables

Generate *price_geq_median* indicating whether or not the apartment price is greater than equal to the median of apartment prices, according to *price*.

```r
top_decile_price <- quantile(final_data$price, .90, na.rm=TRUE)
final_data <-  final_data %>% mutate(
  pricey = ifelse(price >= top_decile_price, "yes", "no"))

price_median <- median(final_data$price, na.rm=TRUE)
final_data <- final_data %>% mutate(
  price_geq_median = ifelse(price >= price_median, "yes", "no"))
```

Generate racial composition of Census tract variables

Create the racial composition variables as follows:

- *white_proportion* equal to the variable *whites* divided by *population*.
- *black_proportion* equal to the variable *blacks* divided by *population*.
- *asian_proportion* equal to the variable *asians* divided by *population*.
- *hispanic_proportion* equal to the variable *hispanics* divided by *population*.

```r
final_data <-  final_data %>% mutate(
  white_proportion = whites/population,
  black_proportion = blacks/population,
  asian_proportion = asians/population,
  hispanic_proportion = hispanics/population
  )
```

Analysis

Reproduce columns 1 through 3 of Table 6.

```r
# Column 1
table6_c1 <- lm(data = final_data, yes ~ guest_race + price_geq_median +
                  price_geq_median*guest_race)
table6_c1_clustered <-  clustered_tstats(table6_c1)
table6_c1_clustered
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.5215124	0.0175918	29.6451445	0.0000000
guest_raceblack	-0.0862489	0.0232822	-3.7045025	0.0002137
price_geq_medianyes	-0.0533190	0.0212859	-2.5048943	0.0122744
guest_raceblack:price_geq_medianyes	0.0029735	0.0297654	0.0998989	0.9204278

```r
# Column 2
table6_c2 <- lm(data = final_data, yes ~ guest_race + black_proportion +
                  black_proportion*guest_race)
table6_c2_clustered <-  clustered_tstats(table6_c2)
table6_c2_clustered
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.4819048	0.0131208	36.7283127	0.0000000
guest_raceblack	-0.0824496	0.0183119	-4.5025085	0.0000068

term	estimate	std.error	statistic	p.value
black_proportion	0.0451258	0.0459949	0.9811043	0.3265795
guest_raceblack:black_proportion	0.0197870	0.0668193	0.2961277	0.7671425

```r
# Column 3
table6_c3 <- lm(data = final_data, yes ~ guest_race + tract_listings +
                  tract_listings*guest_race)
table6_c3_clustered <-  clustered_tstats(table6_c3)
table6_c3_clustered
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.4944132	0.0164256	30.1002040	0.0000000
guest_raceblack	-0.0888135	0.0239174	-3.7133410	0.0002063
tract_listings	-0.0006798	0.0009567	-0.7105879	0.4773662
guest_raceblack:tract_listings	0.0009721	0.0015046	0.6460577	0.5182658