Review Exercise

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

Introduction

For the review 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* in April 2017.

In this article, Edelman, Luca, and Svirsky conduct an experiment wherein they apply for Airbnb apartments using guest names that have distinctively white or African American sounding 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 an R Markdown file for a PDF-type document, in a new folder of your course repository.
- As you go through each step, either add in the instructions found here or add your comments explaining what you are doing (*outside of the code chunk*).
- Commit (and push) your work often.

Importing Data

- 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, change text to lower case using the tolower() function.
- 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_df(tolower)

## Here is a better approach using map_if
# main_data <- main_data %>% map_if(is.numeric, tolower)
#
## Here is a function using an anonymous map function
# main_data <- main_data %>% map_df(function(var)
#   if(is.character(var)){
```

```r
#     tolower(var)
#   }
#   else{
#     var
#   }
# )

# Replace missing values and convert to a tibble
main_data <- main_data %>% map_df(na_if,"null")
main_data <- main_data %>% map_df(na_if,"\\n")
main_data <- main_data %>% map_df(na_if,"-1")
main_data <- main_data %>% map_df(na_if,-1)
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. - You may want to change the format of *datanames* to matrix after you import it.

```r
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")
datanames <-  import("./data/datanames.csv")
datanames <- as.matrix(datanames)
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.

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

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

```
# # Alternative approach that preserves vector type - avoiding unlist()
# # 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*.

```
 main_data <- main_data %>% mutate(
   namebycity = glue("{guest_first_name} {city}"))
```

Import and merge survey results

- Import the file "name_survey_results.xlsx"
- Again apply tolower() to the *guest_first_name* variable.
- 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")
survey_results$guest_first_name <- tolower(survey_results$guest_first_name)

# 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, sex, and age 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.
- *young* equal to 1 if *host_age* is equal to any of "young","young/uu", "uu/young", "young/na", or "na/young".

```
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),
  young = ifelse(host_age %in% c("young","young/uu", "uu/young",
                    "young/na", "na/young"), 1,0)
)
```

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.
- *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".

```
merged_data <- merged_data %>%
  mutate(ten_reviews = ifelse(number_of_reviews >= 10, "yes", "no"),
    multiple_listings = ifelse(number_of_listings > 1, "yes", "no"),
    shared_property = ifelse((property_setup == "private room" |
                                property_setup == "shared room"), "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: The Impact of Race on Likelihood of Acceptance

- 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.
  - Cluster on *namebycity*
  - The syntax of cluster.vcov is:

```
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.

```
# 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)	0.4980690	0.0133467	37.317648	0.0000000
guest_raceblack	-0.0801920	0.0169482	-4.731605	0.0000023
host_race_black	0.0685129	0.0231873	2.954756	0.0031409
host_male	-0.0510422	0.0137169	-3.721109	0.0002001

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

term	estimate	std.error	statistic	p.value
(Intercept)	0.7568326	0.0674929	11.213509	0.0000000
guest_raceblack	-0.0876302	0.0173621	-5.047198	0.0000005
host_race_black	0.0927733	0.0231558	4.006474	0.0000624
host_male	-0.0478424	0.0141825	-3.373345	0.0007472
multiple_listingsyes	0.0630171	0.0146443	4.303178	0.0000171
as.factor(shared_property)yes	-0.0690273	0.0165691	-4.166034	0.0000314
ten_reviewsyes	0.1193189	0.0131627	9.064957	0.0000000
log(price)	-0.0626511	0.0127763	-4.903707	0.0000010

Reproduce Figure 2: Host Responses by Race

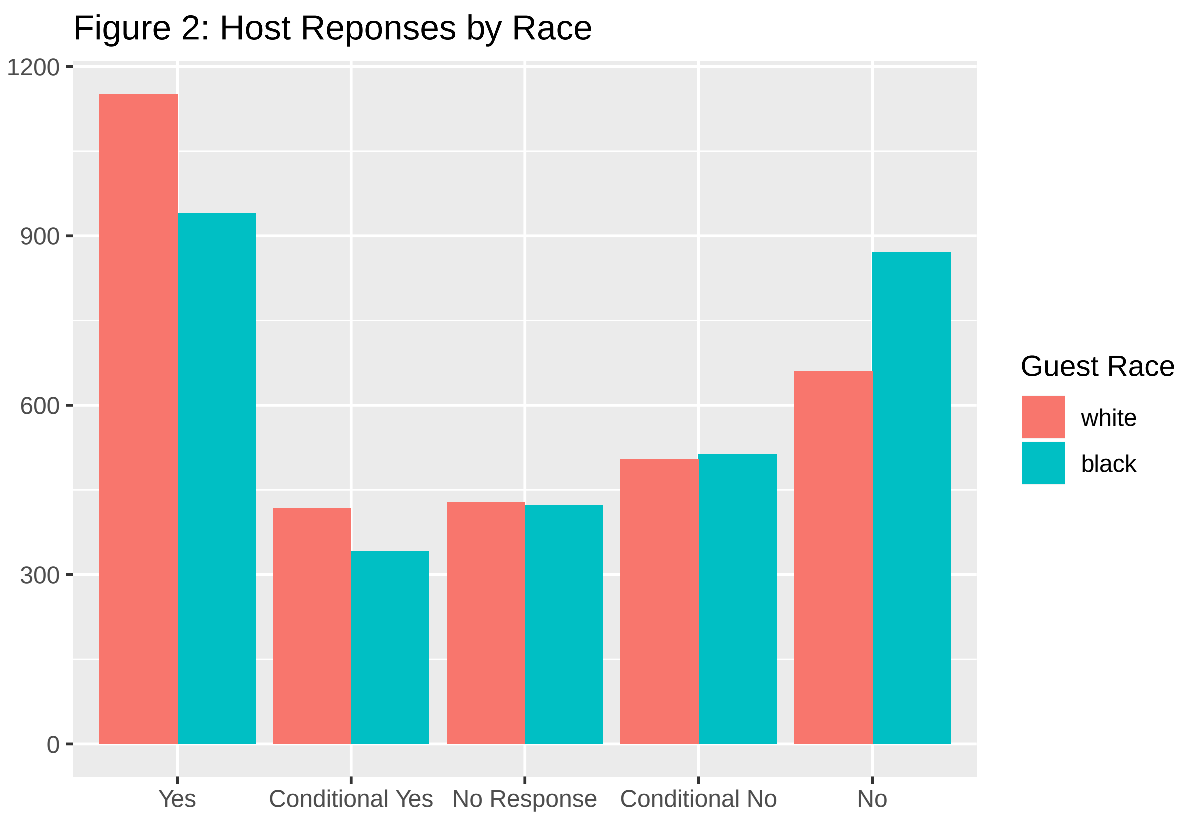
Create a grouped bar plot of host responses by Race, as in Figure 2 of Edelman, Luca, and Svirsky.

- 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("")  +
  labs(fill='Guest Race')
```



[Bonus!] 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

term	estimate	std.error	statistic	p.value
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

```r
# 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)	0.4980139	0.0122740	40.5747085	0.0000000
guest__raceblack	-0.0760515	0.0175374	-4.3365348	0.0000147
young	-0.0275293	0.0182933	-1.5048895	0.1324032
guest__raceblack:young	-0.0102320	0.0247659	-0.4131481	0.6795123

```r
# 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