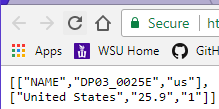
Commute time in the United States will be investigated in this handout. The goal here is to obtain a reasonable estimate for Mean Commute Time using data provided by the US Census Bureau.

|  |  |
| --- | --- |
| WNYC Average Commute Time by Zipcode  Source: <https://project.wnyc.org/commute-times-us/embed.html#5.00/38.272/-96.224> | Image result for commute |

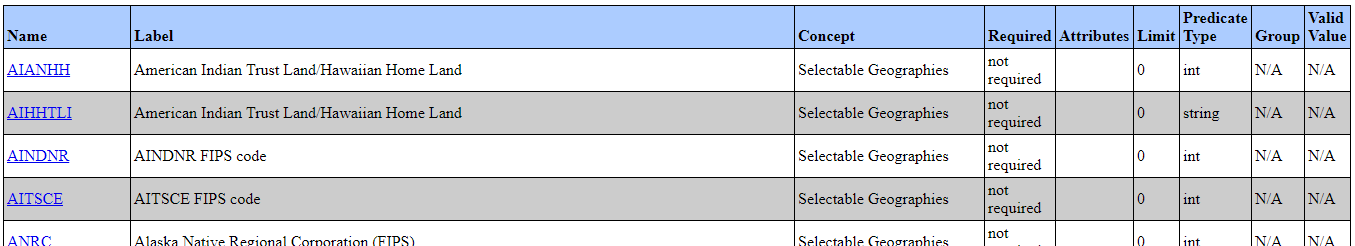
The Census Bureau now has an application program interface (API) that allows one to obtain census data. Here, the Mean Commute Time is being returned for commuters in the United States.

[https://api.census.gov/data/2015/acs5/profile?get=NAME,DP03\_0025E&for=us:\*](https://api.census.gov/data/2015/acs5/profile?get=NAME,DP03_0025E&for=us:*)

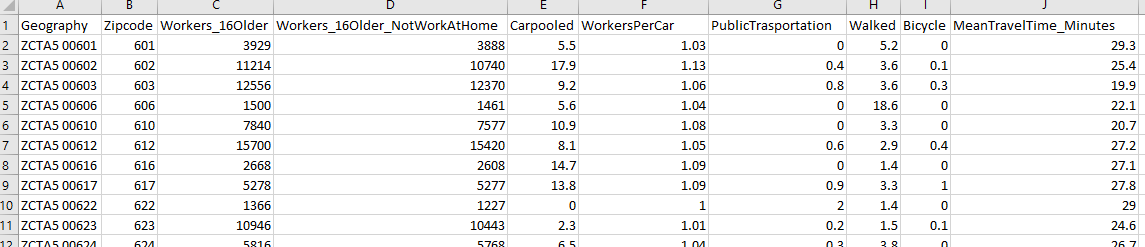


The possible list of variables for American Community Survey 5 Year dataset is provided by the following link.

ACS 5 Year List of Variables: <https://api.census.gov/data/2015/acs5/profile/variables.html>



The CommuteData\_Zipcode.csv file will be used as the sampling frame for this handout. A snipit of this data is shown here.

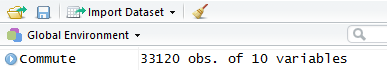


**Simple Random Sampling in R**

First, reading in the CommuteData\_Zipcode Data into R. Use the View() function to ensure the data was read in correctly.

Commute <- read.csv( file.choose() )

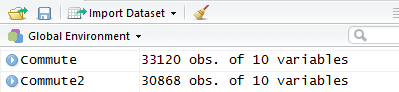
View(Commute)



Using the filter() function in dplyr package to remove missingness from variables of interest.

library(dplyr)

Commute2 <- filter(Commute,!is.na(MeanTravelTime\_Minutes))



Using the sample command to take a random sample of **n=500** from the Commute2 data.frame.

mysample<-sample(1:30868,size=500,replace=FALSE)

Getting the outcome of interest for the simple random sample that has been selected.

sample\_data <- Commute2 %>% filter(row\_number() %in% mysample) %>% select(MeanTravelTime\_Minutes)

Getting the estimated mean travel time from our simple random sample.

sample\_data %>% summarize('Mean'= mean(MeanTravelTime\_Minutes))

Creating an R function to do the simple random sampling from the Commute2 data.frame.

MySim\_SRS <- function(){

mysample<-sample(1:30868,size=500,replace=FALSE)

sample\_data <- Commute2 %>% filter(row\_number() %in% mysample) %>%   
 select(MeanTravelTime\_Minutes)

output<-sample\_data %>% summarize('Mean'= mean(MeanTravelTime\_Minutes))

return(output)

}

Using the replicate() function in R to run the MySim\_SRS() function. Here, 2 iterations are run.

> SRS\_Output <- data.frame( "Estimates" = unlist( replicate(2, MySim\_SRS() ) ) )

|  |  |
| --- | --- |
| R Code | Description |
| |  |  | | --- | --- | | > MySim\_SRS()  Mean  1 25.764  > MySim\_SRS()  Mean  1 25.9086 | Running the MySim\_SRS() function.  The output is being returned to the screen here. | | > replicate(2, MySim\_SRS() )  $Mean  [1] 26.2106  $Mean  [1] 25.6342  Confirmation that replicate() function indeed returns a list data type in R.  > testdata <- replicate(2,   MySim\_SRS() )  > str(testdata)  List of 2  $ Mean: num 25.8  $ Mean: num 25.3 | The replicate function will run the MySim\_SRS() repeatedly – only 2 iterations are being specified here.  The output is from the replicate() function is returned as a list. Notice the slightly different appearance of the output.  The structure, i.e. str(), function in identifies that this is indeed a list object in R. | | > unlist(replicate(2,MySim\_SRS() ))  Mean Mean  25.7254 26.1924 | The unlist() function here will reduce a list object to a simple vector. | | > SRS\_Output <- data.frame(  "Estimates" = unlist(  replicate(2, MySim\_SRS() )  )  )  SRS\_Output  Estimates  1 25.9318  2 26.0690 | The final step is the put the output back into a data.frame object in R so that standard  summaries and ggplot commands will work  correctly. | | |

The goal in sampling is to provide estimates that mimic the target population as closely as possible (no bias) with high precision (low sampling error).

|  |  |
| --- | --- |
| Target Population Quantity: 25.9 | Estimated Value from SRS: 25.87  > sample\_data %>% summarize('Mean'= mean(MeanTravelTime\_Minutes))  Mean  1 25.8655 |

Using the replicate() function to obtain several thousands repeated outcomes using the simple random sampling approach.

> SRS\_Output <- data.frame( "Estimates" = unlist( replicate(5000, MySim\_SRS() ) ) )

The bias in an estimate is computed as the difference between the estimate and the parameter.

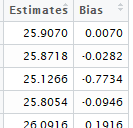
**Bias in Estimate = (Estimate – Parameter)**

Calculate the bias for the first SRS of your simulation with the parameter value of 25.9, i.e. the average of the entire target population.

**Bias =**

A more thorough approach might be to calculate the bias for each individual

> SRS\_Output = mutate(SRS\_Output, "Bias" = Estimates - 25.9)



Basic summaries for the amount of bias for simple random sample of size n=500.

> SRS\_Output %>% summarize("Overall Average Bias" = mean(Bias))

Overall Average Bias

1 -0.0301478

> SRS\_Output %>% summarize("Overall Median Bias" = median(Bias))

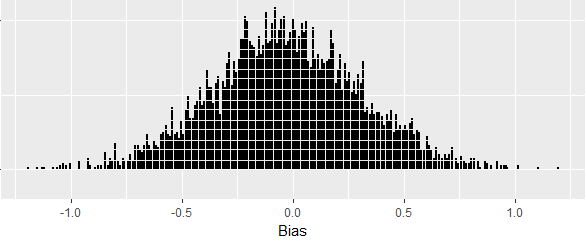
Median

1 -0.0372

Plotting the distribution of bias.

> library(ggplot2)

> ggplot(SRS\_Output, aes(Bias)) + geom\_dotplot(binwidth=0.01)



The following can be used to compute the proportion of estimates that under-estimated the actual target.

> length(which(SRS\_Output$Bias < 0)) / 5000

[1] 0.5444

The precision of the estimates can be determined by looking at the standard deviation of the Estimates.

> SRS\_Output %>% summarize("Std Dev of Estimates" = sd(Estimates))

Std Dev of Estimates

1 0.3336817

Precision of the estimates can be obtained by way of statistical theory as well.

Getting the standard deviation of the target population…

> Commute2 %>% summarize("Std Dev of Mean Travel Time" = sd(MeanTravelTime\_Minutes) )

Std Dev

1 7.447405

Now, using this to compute the Std Error of Mean Travel Time with sample size = 500

> Commute2 %>% summarize("Std Error of Mean Travel Time" = sd(MeanTravelTime\_Minutes) / sqrt(500) )

Std Error of Mean Travel Time

1 0.333058

Consider the following modification to the MySim\_SRS() function. This modification will allow us to easily change the number of observations being drawn from the “hat”. A default value of sample size = 100 is being specified here; thus, if the function is run without a sample size specification, the function will use 100.

MySim\_SRS <- function(samplesize=100){

mysample<-sample(1:30868,size=samplesize,replace=FALSE)

sample\_data <- Commute2 %>% filter(row\_number() %in% mysample) %>% select(MeanTravelTime\_Minutes)

output<-sample\_data %>% summarize('Mean'= mean(MeanTravelTime\_Minutes))

return(output)

}

|  |  |
| --- | --- |
| R Code | Description |
| |  |  | | --- | --- | | > MySim\_SRS(samplesize=250) | The mean from a simple random sample of  size = 250 will be returned from the  MySim\_SRS() function. | | > MySim\_SRS() | The mean from a simple random sample of  size = 100 (the function will use the default value) will be returned from the MySim\_SRS() function. | | |

The following command can be used to obtain 5000 repeated simulations, each sample taken is a simple random sample of size 250. The output should be labeled in such a way to indicate that sample size = 250.

> SRS\_Output250 <- data.frame( "Estimates" = unlist( replicate(5000, MySim\_SRS(samplesize=250) ) ) )

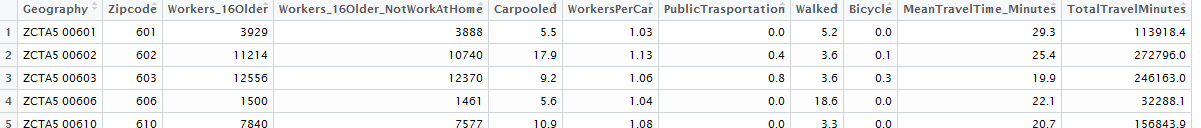
Tasks:

1. Obtain several thousand repeated simulations using a sample size of 250. Use ggplot() to plot the mean estimates. Discuss what you see in this plot? Does there appear to be bias present? Is the variability in these estimates higher/lower than the distribution when using sample size = 500.
2. Calculate the bias for each mean estimate. Use ggplot() to plot the bias distribution. Is this distribution centered about 0? Should it be centered about 0? Discuss.
3. Calculate the standard deviation in the mean estimates from your simulation. Calculate the standard error using the formula provided above. To what degree do these estimates agree? Discuss.
4. Repeat Tasks 1 -3 using a sample size = 100. Discuss the resulting bias and standard error from samples of size 100.
5. Someone not so familiar with math suggests that doubling the sample size should cut the standard error in half. Someone else suggest that quadrupling the sample size should cut the standard error in half. Who is correct and why?

**PROBLEM**: NOT ALL ZIPCODES HAVE SAME NUMBERS OF WORKERS; THUS, COMPUTING AN AVERAGE ACROSS ZIPCODES NEEDS TO BE CAREFULLY CONSIDERED.

Creating an updated data.frame that includes the Total Travel Time.

> Commute3 <- mutate(Commute2, 'TotalTravelMinutes' = Workers\_16Older\_NotWorkAtHome \* MeanTravelTime\_Minutes)



Updating the MySim\_SRS() function to reflect the need to compute an “improved” estimate.

> MySim\_SRS <- function(samplesize=500){

mysample<-sample(1:30868,size=samplesize,replace=FALSE)

output<-summarize(Commute3[mysample,], 'Mean'= sum(TotalTravelMinutes) /   
 sum(Workers\_16Older\_NotWorkAtHome))

return(output)

}

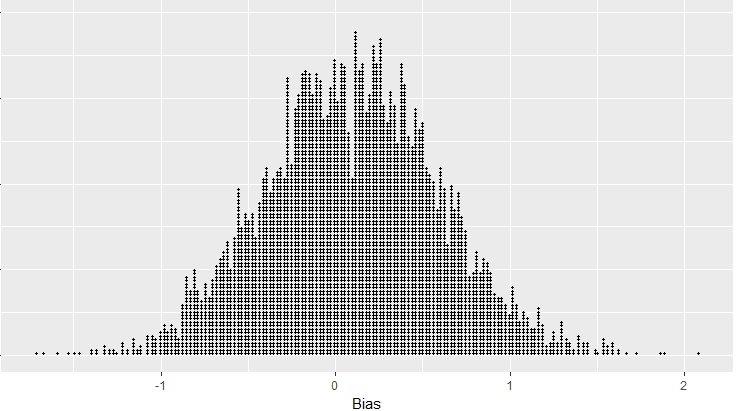
Again, getting 5000 repeated simulations…

> SRS\_Output500 <- data.frame( "Estimates" = unlist( replicate(5000, MySim\_SRS(samplesize=500) ) ) )

> SRS\_Output500 = mutate(SRS\_Output500, "Bias" = Estimates - 25.87)

Creating a plot of the bias…

> ggplot(SRS\_Output500, aes(Bias)) + geom\_dotplot(binwidth=.02)



> length(which(SRS\_Output500$Bias < 0)) / 5000

[1] 0.4286

The bias appears to be ***larger*** (not smaller) than using the unweighted approach. What might be causing this problem?