# Problem Set 5

Daniel Shapiro

9/29/2022

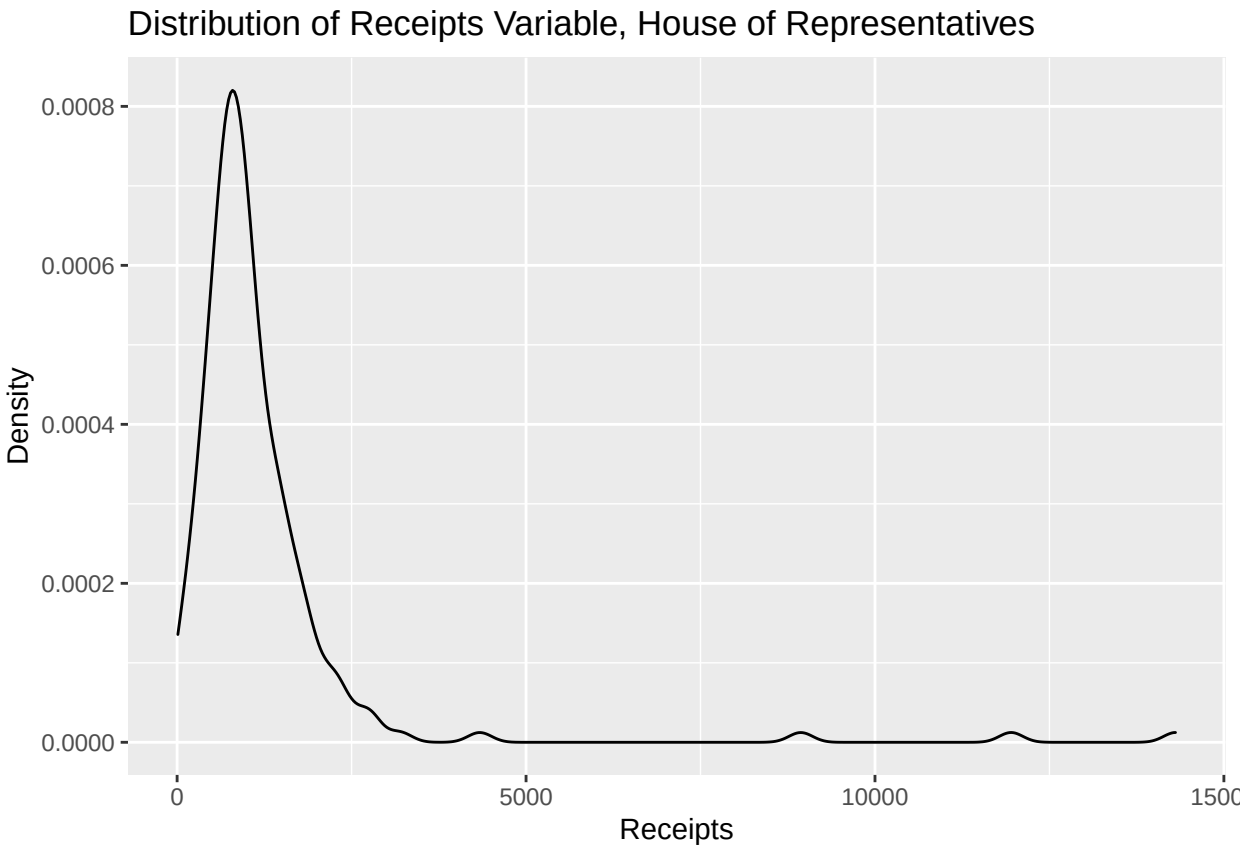
**Question 1 Background:**

*For this problem we will use data on funds raised by a subset of incumbent Democrats in the US House of Representatives for the 2006 election cycle. Download the data,* ***house.csv****. In the dataset, you will find the name of the representatives (****house$candidate****) and the funds they raised in thousands of dollars (****house$receipts****).*

```
house <- read.csv("house.csv")
```

**1a) First, use `ggplot()` and `geom_density()` to plot the density of `house$receipts` and briefly describe the distribution of the receipts variable in the House of Representatives dataset.**

```
ggplot(house, aes(x = receipts)) +
  geom_density() +
  labs(title = "Distribution of Receipts Variable, House of Representatives",
       x = "Receipts",
       y = "Density")
```




The distribution of the house$receipts variable is located for the most part between 0 and 2500; however, there are a few values that occur later on – one just below 5000, another around 8500, another around 12000, and another around 14000.

**1b) Find the mean and standard deviation of `house$receipts`, and the standard error of the mean. Write your own function for the standard deviation – do not used a canned `R` function.**

```
mean(house$receipts)
```

```
## [1] 1190.31
```

```
deviation <- function(data){

  # Define the mean. I put this with an extra letter just in case it gets confused
  # with a function.

  meann <- mean(data)

  # Add an empty vector to do a for loop over

  vector <- c(rep(0, times = length(data)))

  for(i in 1:length(data)){
    vector[i] <- (data[i] - meann)^2
```

```
  }

  summ <- sum(vector)
  variance <- summ/(length(vector) - 1)
  deviation <- sqrt(variance)

deviation
}

deviation(house$receipts)
```

```
## [1] 1500.763
```

The mean is listed as 1190.3095183 and the standard deviation is 1500.7633085.

```
se <- deviation(house$receipts)/sqrt(length(house$receipts))
se
```

```
## [1] 108.5914
```

This 108.5914219 is the standard error of the mean.

**1c) Find 95% confidence intervals for your estimate of the mean. Interpret this CI (in words).**

The function to find the confidence interval can be expressed as: $\overline{X} \pm t_{\alpha/2,N-1} S_{\overline{x}}$

We already have $\overline{X}$; that's just the mean of the string. We also already have $S_{\overline{x}}$; that's just the standard error. So now we need to get $\alpha$ and the t score. Then we can plug everything in.

```
# We make this equal to .05 because we are looking for the 95% CI. Will
# have to divide by two later. All of this is just setup.

alpha <- .05

degrees_freedom <- length(house$receipts) - 1

# Here, we use the quantile function from the stats package. Honestly, I've never
# been super sure how the quantile function works. But I know I've used it
# in R before for this sort of thing, so I'm doing it again here.

ts <- qt(p = alpha/2, df = degrees_freedom, lower.tail = FALSE)

# Regardless, now we have the T score. I set lower.tail to FALSE because otherwise
# it came out flipped.

margin <- ts * se

lower <- mean(house$receipts) - margin
upper <- mean(house$receipts) + margin

c(lower, upper)
```

```
## [1]  976.1099 1404.5092
```

These two values, pictured above, are the bounds of our 95% confidence interval for the string house$receipts. This means that we can be 95% confident that the mean candidate receipt value lies between 976.1099 and 1404.5092.

**1d) Confirm, through simulation, that if you redraw 1,000 samples from your data of the size of your data (with replacement), about 95% of them fall into your confidence interval.**

```
# A favorite of mine -- rep_sample_n from the infer package.

sample <- house %>% infer::rep_sample_n(size = 191, replace = TRUE, reps = 1000) %>%
  select(-candidate)

# Map the mean() to the data. I didn't need the candidate column either.

samplemean <- sample %>%
  group_by(replicate) %>%
  nest() %>%
  mutate(mean = map(.x = data, .f = ~mean(.x$receipts, na.rm = TRUE))) %>%
  select(-data) %>%
  ungroup()

testdata <- samplemean %>%
  filter(mean > 976.1099) %>%
  filter(mean < 1404.5092)

nrow(testdata) / nrow(samplemean)
```

```
## [1] 0.955
```

Confirmed!

**Question 2 Background:**

*Now, we will use the house data to examine how well different estimators recover the true population mean of funds raised. We will now assume that our sample represents the full population (e.g., the mean found above is the true $\mu$). Our parade of estimators will involve drawing 1,000 random samples (without replacement) of size n from the population. Let $X = \{X_1, \ldots, X_N\}$ denote the vector of sample observations, so $X_i$ denotes the i-th observation drawn in each sample with $i = 1, \ldots, N$. We consider the following estimators:*

- The Sample Mean: $\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} X_i$
- The Sample Median: If your sample is ordered such that $X_1 \leq X_2 \leq X_3 ... \leq X_n$, the median is the midpoint value.
- The Deletion Estimator I: Order the sample observations such that $X_1 \leq X_2 \leq X_3 ... \leq X_n$. Discard the minimum value $min(X) = X_1$ from the sample, and compute the mean of the remaining observations.
- The Deletion Estimator II: Order the sample observations such that $X_1 \leq X_2 \leq X_3 ... \leq X_n$. Discard the two lowers values $X_1, X_2$ and the two highest values $(X_{n-1}, X_n)$ from the sample, and compute the mean of the remaining observations.

**2a) Write a function that takes data and returns a dataframe of values for the mean, median, and two deletion estimators (eg, it should take data and return a dataframe with 4 rows and 2 columns). Hint: start your function by defining a dataframe with an "estimator" and "result" column. Use `sort()` to sort your data. Make sure your function works by evaluating the following toy sample: $X = 2, 1, 100, 3, 4, 3, 2$. Your function should return the following:**

| Estimator | Sample Mean | Sample Median | Deletion Estimator I | Deletion Estimator II |
|---|---|---|---|---|
| $\hat{\mu}$ | 16.429 | 3 | 19 | 2.667 |

```
tablefunc <- function(data){

  estimator <- c("Sample Mean", "Sample Median", "Deletion Estimator I", "Deletion Estimator II")

# Empty vector time.

  result <- c(rep(0), times = 4)

  deletion <- data %>%
    sort(decreasing = FALSE)
  deletion1 <- deletion[-1] %>%
    mean()

# A bit round-about, but it does the job.

  deletionintermediate <- deletion[-c(1, 2)]

  deletionlast <- deletionintermediate %>%
    sort(decreasing = TRUE)

  deletion2 <- deletionlast[-c(1, 2)] %>%
    mean()

  for(i in 1:4){
    if(estimator[i] == "Sample Mean"){result[i] <- sum(data)/length(data)
    }
    else if(estimator[i] == "Sample Median"){result[i] <- median(data)
    }
    else if(estimator[i] == "Deletion Estimator I"){result[i] <- deletion1}
    else(result[i] <- deletion2)

  }
  data.frame(estimator, result)
  }

vector <- as.numeric(c(2, 1, 100, 3, 4, 3, 2))
tablefunc(vector)
```

```
##                estimator    result
## 1           Sample Mean 16.428571
## 2         Sample Median  3.000000
## 3  Deletion Estimator I 19.000000
## 4 Deletion Estimator II  2.666667
```

**2b) Using the function you defined above, simulate the sampling distribution for all the estimators from part a) for the receipts variable. Your sample size in this case should be $n = 100$, and you should conduct 1,000 simulations (hint: create an empty dataframe, then use `rbind()` to append the results of each iteration of your simulation). Create a single density plot using `ggplot()` and `geom_density()`, including a vertical line at the true population mean, and with each line in a different color. Axes and legends should be appropriately titled.**

```
# I don't do this exactly how the problem instructs, but I prefer my way.

sampling <- house %>% infer::rep_sample_n(size = 100, replace = FALSE, reps = 1000)

# Each time, I'm going to pull the desired coefficient from my nested estimator
# dataframes. So for "mean", I'll pull the mean. I can use the handy pluck()
# function.

samplestats <- sample %>%
  group_by(replicate) %>%
  nest() %>%
    mutate(stats = map(.x = data, .f = ~tablefunc(.x$receipts))) %>%

# I mapped my function over every sample. Issue is, now I have dataframes
# mapped within my column. So I need to do a series of "plucking" to get that
# solved.

  mutate(statistics = map(stats, ~pluck(.x, var = 2))) %>%
  mutate(mean = map(statistics, ~pluck(.x, 1))) %>%
  mutate(median = map(statistics, ~pluck(.x, 2))) %>%
  mutate(deletion1 = map(statistics, ~pluck(.x, 3))) %>%
  mutate(deletion2 = map(statistics, ~pluck(.x, 4))) %>%

# Don't need my huge nested columns anymore, so I'm going to get rid of those.

  select(-c(data, stats, statistics))

# Last, I need to make my columns numeric.

samplestats$mean <- as.numeric(samplestats$mean)
samplestats$median <- as.numeric(samplestats$median)
samplestats$deletion1 <- as.numeric(samplestats$deletion1)
samplestats$deletion2 <- as.numeric(samplestats$deletion2)

# Now we're good to start setting up the plot!
```

```
# Had to do some tricky manual labor here to get the legend with the right colors.

fills <- c("Del. Estimator II" = "lightblue", "Del. Estimator I" = "red", "Mean" = "yellow", "Median" =

# Have to make a dataframe for geom_text().

xint <- mean(house$receipts)

xint <- data.frame(xint)
```

```
ggplot(samplestats) +
  geom_density(aes(x = mean, fill = "Mean"), color = "orange", alpha = 0.4) +
  geom_density(aes(x = median, fill = "Median"), color = "darkgreen", alpha = 0.4) +
  geom_density(aes(x = deletion1, fill = "Del. Estimator I"), color = "darkred", alpha = 0.4) +
  geom_density(aes(x = deletion2, fill = "Del. Estimator II"), color = "darkblue", alpha = 0.4) +
  geom_vline(xintercept = xint$xint, color = "red") +
   geom_text(data = xint, x = xint$xint, y = 0.010, label = "True Mean", size = 3, angle = 90, vjust =
  labs(x = "Funds Raised",
       y = "Density",
       title = "Estimator Sampling Distributions",
       subtitle = "Sample Size: 100",
       fill = "Legend")
```