Understanding Political Surveys and Their Implications

Prof. Jonathan Cervas

2024-09-20

```
# Remove all objects just to be safe.
     rm(list=ls(all=TRUE)) # Remove all objects from R history
     options(scipen=999) # Turn off Scientific Notation
     set.seed(666) # Setting a seed for reproducibility
```

Introduction

This assignment will guide you through the principles of political surveys, including the importance of sample size, the calculation of margin of error, and the impact of nonresponse bias.

We know from our reading and logic that the margin of error in a poll goes down as sample size goes up. We can show this in `R` using the following:

The margin of error for a proportion `p` is typically calculated using the formula:

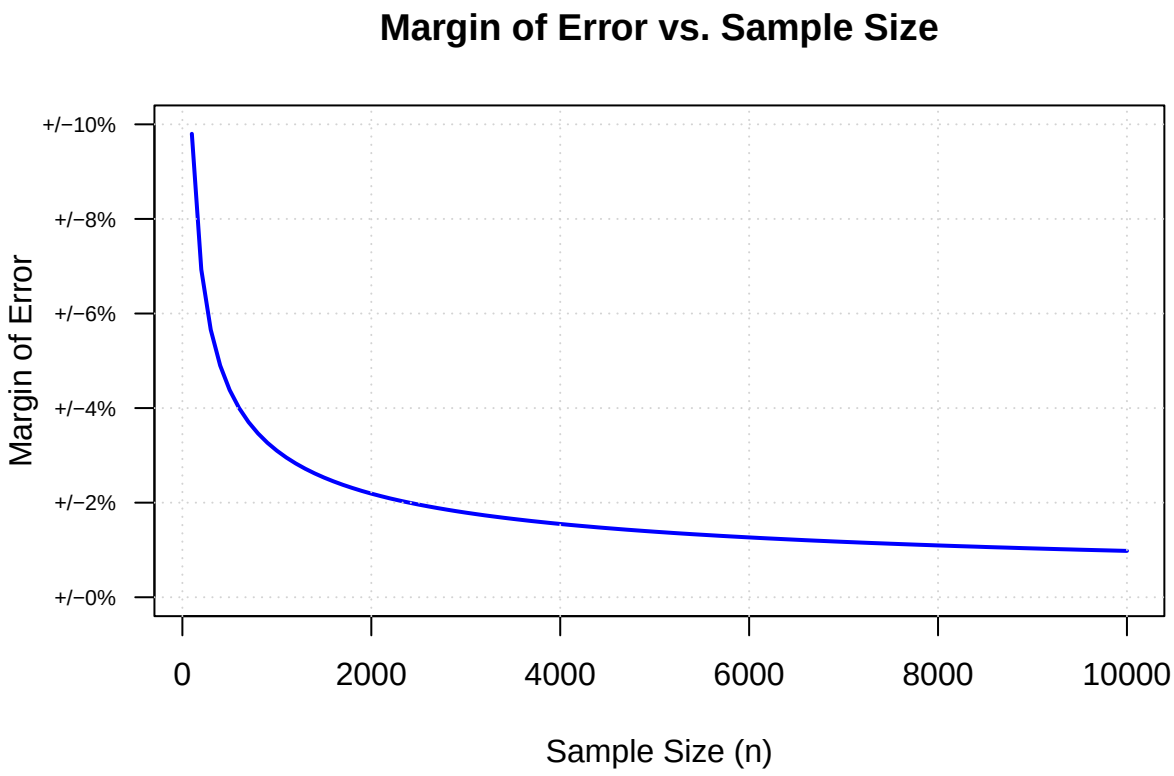
$$\text{Margin of Error} = Z \times \sqrt{\frac{\hat{p} \times (1 - \hat{p})}{n}}$$

`z` is the z-score corresponding to the desired confidence level (e.g., 1.96 for a 95% confidence level),

`p` is the proportion (e.g., 0.5 for the maximum margin of error),

`n` is the sample size.

```
# Set parameters
z <- 1.96  # z-score for 95% confidence level
p <- 0.5   # worst-case scenario for proportion
n <- seq(100, 10000, by=100)  # sample sizes from 100 to 10,000

# Calculate margin of error for each sample size
margin_of_error <- z * sqrt(p * (1 - p) / n)

# Plot
plot(n, margin_of_error, type="l", col="blue", lwd=2,
     xlab="Sample Size (n)", ylab="Margin of Error",
     main="Margin of Error vs. Sample Size",
     ylim = c(0,0.1),
     yaxt = "n")

y_values <- seq(0,0.1,0.02)
axis(side=2, las=2, at=y_values, label=paste0("+/-", y_values * 100, "%"), cex.axis = 0.65)

grid()
```



Part 1: Simulating a Political Survey

Activity 1: Population Simulation

In the 2020 election, the candidates received these number of votes:

```
(biden <- 81284666)
```

```
## [1] 81284666
```

```
(trump <- 74224319)
```

```
## [1] 74224319
```

(**Source: The New York Times 2020**)

This implies that Trump won what share of the two-party vote?

```
(trump_proportion <- trump/(biden + trump)) # Two-Party Share of the Vote
```

```
## [1] 0.4772992
```

We will explore how random sampling can produce relatively accurate polls, assuming no errors beyond sampling error.

Simulated 2020 Elections

Let's simulate a population of 100,000 voters based on this proportion:

In order to better understand how polls are conducted and how margin of error is calculated, we can create a simulated world reflecting the US in 2020.

Let's first create a vector of the 'population' of voters, and in that assign Trump voters the value `1` and Biden voters a value `0`

We can simulate this in `R` using `rbinom()`, which will create a vector of 0 and 1s to some size in proportion to the value we give. Let's start with a smaller electorate (100000) but with the same underlying vote choices.

```
population_size <- 100000
population <- rbinom(population_size, 1, trump_proportion)
```

We can see if it is right.

```
mean(population)
```

```
## [1] 0.47983
```

It is approximately right.

Alternative Way to Simulate the Real World

We can also just do it directly by brute force.

```
population_1 <- c(
  rep(1,trump),
  rep(0,biden))
```

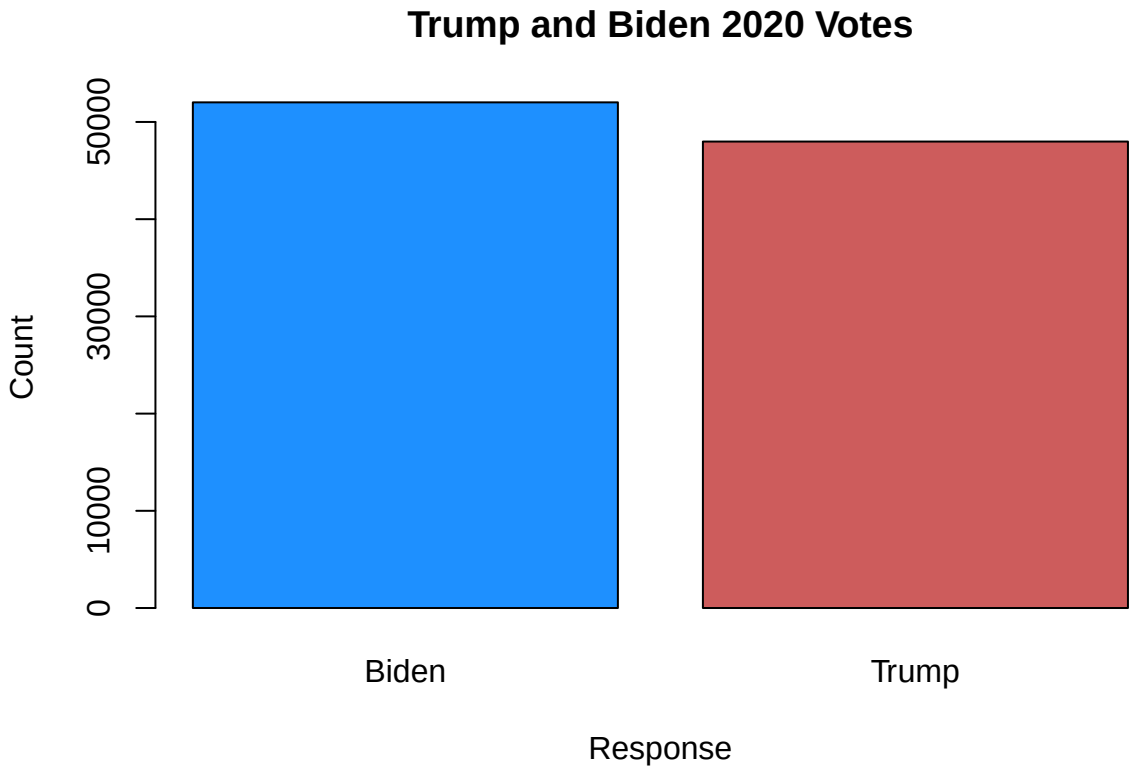
```
mean(population_1)
```

```
## [1] 0.4772992
```

It is perfectly accurate. But not necessary for these purposes.

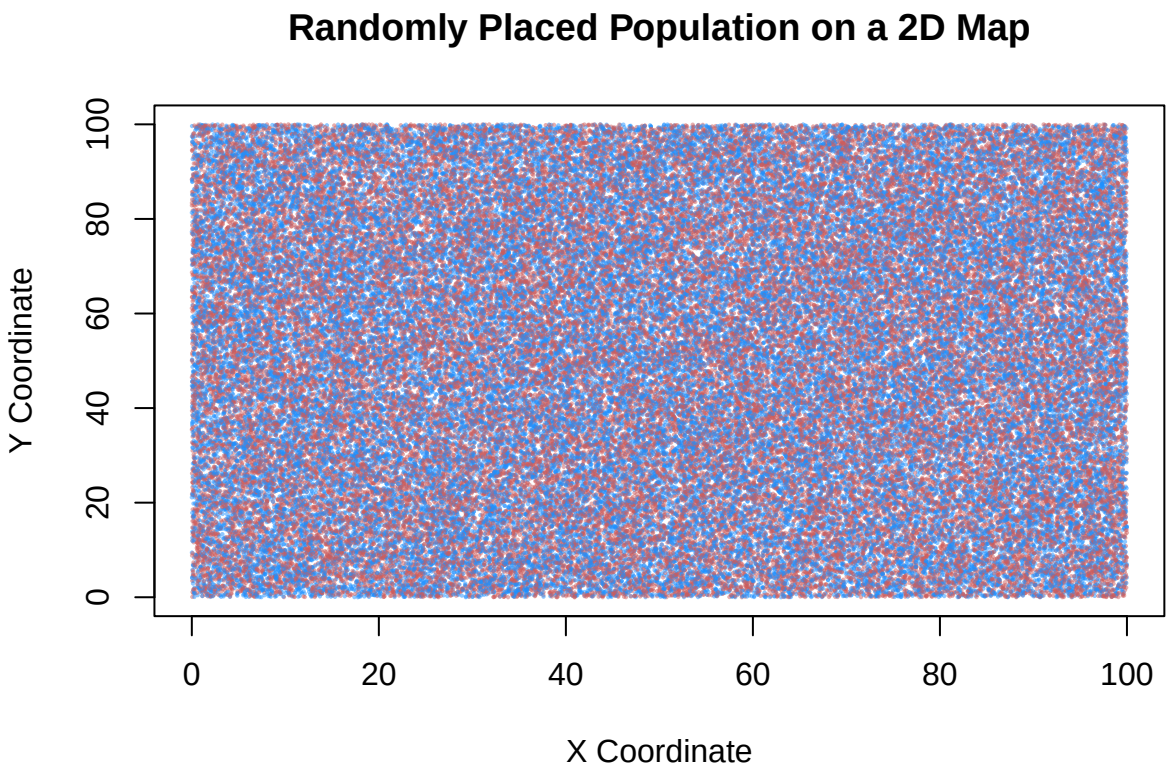
Now let us visualize this using a barplot

```r
# Colors
dodgerblue.transparent <- rgb(30, 144, 255, 127.5, max =255)
dodgerblue <- rgb(30, 144, 255, max =255)
indianred.transparent <- rgb(205, 92, 92, 127.5, max =255)
indianred <- rgb(205, 92, 92, max =255)

# Plotting the population data using a bar plot
population_table <- table(population)
barplot(population_table, beside=TRUE, col=c(dodgerblue, indianred),
        names.arg = c("Biden", "Trump"),
        main = "Trump and Biden 2020 Votes",
        xlab = "Response", ylab = "Count")
```



Bonus, let's try to "map" the voters for fun.

```r
# Generate random x and y coordinates for each individual
x_coords <- runif(population_size, min = 0, max = 100)
y_coords <- runif(population_size, min = 0, max = 100)

# Create a scatterplot with different colors for "yes" and "no" responses
plot(x_coords, y_coords, col = ifelse(population == 1, dodgerblue.transparent, indianred.transparent),
     pch = 16, cex = 0.3,
     xlab = "X Coordinate", ylab = "Y Coordinate",
     main = "Randomly Placed Population on a 2D Map")
```



Probably not really what any city looks like given spatial patterns of living and vote choice.

Sampling from the population

We can simulate to see how accurate our poll is. First, let us draw **ten** voters, completely randomly, from our population.

```
(ten_voters <- sample(population, 10))
```

```
##  [1] 1 1 1 0 0 0 0 0 0 1
```

We can take the `mean()` to see what percentage voted for Trump, since our "population" is made up of 0 and 1s.s

```
mean(ten_voters)
```

```
## [1] 0.4
```

How accurate was it? We can actually find the deviation, since we know the true value (**47.7%**).

```r
# Our sample missed the true value by
deviation <- mean(ten_voters) - trump_proportion
paste0(round(deviation, d=3) * 100, "%")
```

```
## [1] "-7.7%"
```

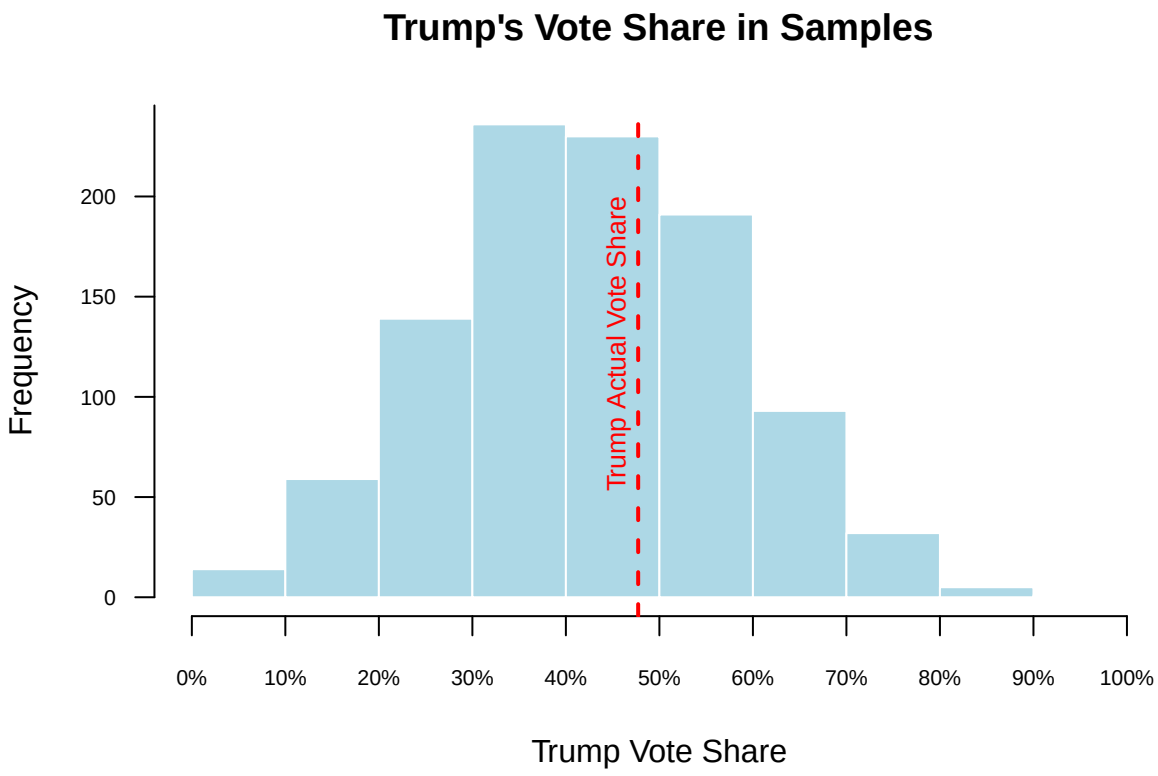
Now let's poll **ten** random people `1,000` times and see how accurate each is, both individually and in the aggregate.

```r
nsims <- 1000
sample_values_10 <- list()
sample_mean_10 <- numeric(length(nsims))
 for (i in 1:nsims) {
     sample_values_10[[i]] <- sample(population, 10)
     sample_mean_10[i] <- mean(sample_values_10[[i]])
     }
```

How accurate were these polls?

Plot Trump's vote share in each simulation (hint: Histogram is a good way to visualize this kind of data)

```r
# Create the histogram
hist(sample_mean_10,
     main = "Trump's Vote Share in Samples",
     xlab = "Trump Vote Share",
     col = "lightblue",
     border = "white",
     xlim = c(0,1),
     yaxt = "n",
     xaxt = "n")

# Add the custom x-axis
axis(side = 1, at = seq(0,1,0.1), labels = paste0(seq(0,1,0.1)*100,"%"), cex.axis = 0.65)
# Add the custom y-axis
axis(side = 2, las = 2, at = seq(0,300,50), labels = seq(0,300,50), cex.axis = 0.65)

# Add a vertical line at Trump's Vote Share
abline(v = trump_proportion, col = "red", lwd = 2, lty = 2)

# Add text to label the line
text(x = trump_proportion, y = 200,
     labels = "Trump Actual Vote Share",
     pos = 2, col = "red", cex = 0.8, srt = 90)
```

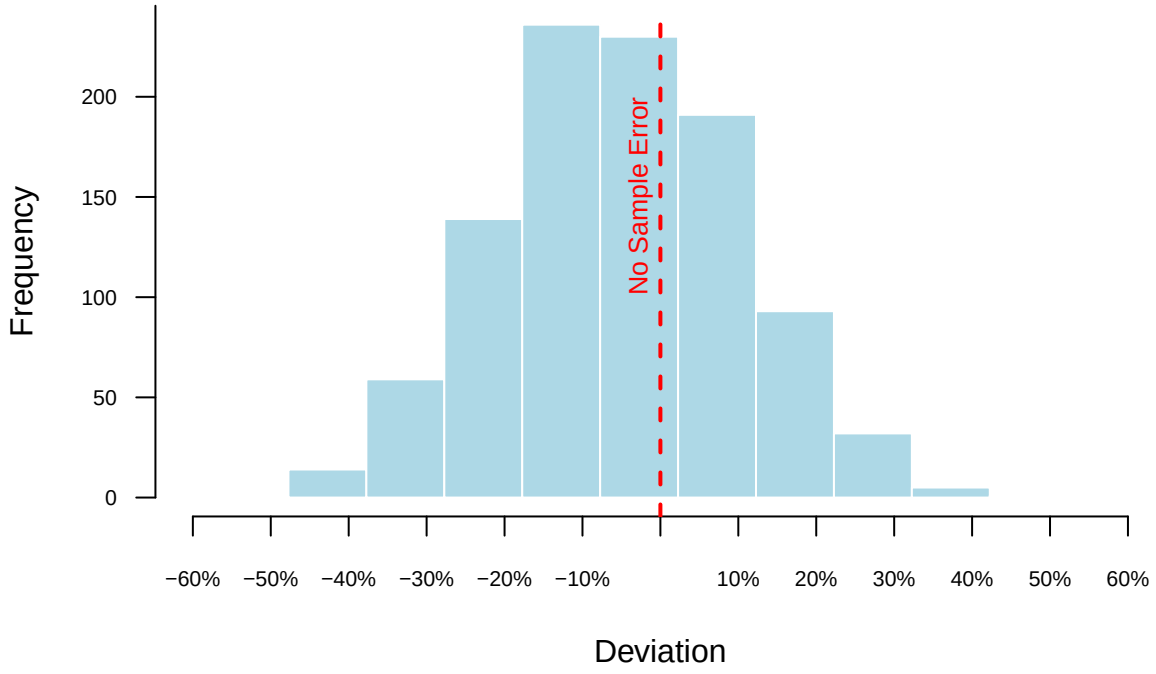



```
# # Add a vertical line at average of samples
# abline(v = mean(sample_mean_10), col = "gray30", lwd = 2, lty = 3)

# # Add text to label the line
# text(x = trump_proportion, y = 200,
#      labels = "Average of Samples",
#      pos = 1, col = "gray30", cex = 0.8, srt = 90)
```

Now plot the accuracy of the poll compared to Trump's actual vote share.

```
deviation <- sample_mean_10 - trump_proportion
hist(deviation,
     breaks = seq(min(deviation), max(deviation) + 0.1, by = 0.1),
     main = "",
     xlab = "Deviation",
     col = "lightblue",
     border = "white",
     xlim = c(-0.6,0.6),
     yaxt = "n",
     xaxt = "n")

# Add the custom x-axis
axis(side = 1, at = seq(-0.6,0.6,0.1), labels = paste0(seq(-0.6,0.6,0.1)*100,"%"), cex.axis = 0.65)
# Add the custom y-axis
axis(side = 2, las = 2, at = seq(0,300,50), labels = seq(0,300,50), cex.axis = 0.65)
```

```r
# Add a vertical line at Trump's Vote Share
abline(v = 0, col = "red", lwd = 2, lty = 2)

# Add text to label the line
text(x = 0, y = 200,
     labels = "No Sample Error",
     pos = 2, col = "red", cex = 0.8, srt = 90)
```




```r
(mean(deviation))
```

```
## [1] -0.002399231
```

The histogram looks the same as the previous one, but showing the information in a different way.

On *average*, the mean error is low. But we rarely run `1,000` polls!

Ideally, we sample **one** subset of the population and produce error this low on the first try!

Sample a larger number of people

Let's see if our error goes down as we sample a larger number of people.

In `R`, we can first write a `function` to do a simulation. In the function, allow for a parameter to vary the number of people sampled, and for changing the number of simulations for greatest flexibility.

```
sim_sample <- function(nsims, sample_sizes, population) {
sample_values <- list()
sample_mean <- numeric(length(sample_sizes))
 for (i in 1:nsims) {
     sample_values[[i]] <- sample(population, sample_sizes)
     }
  return(sample_values)
}
```

Now run your function to sample for 1,000 people.

```
sim_1000 <- sim_sample(1000,1000,population)
sim_1000_mean <- unlist(lapply(sim_1000, mean))
```

Let's see what the 95% confidence intervals are:

```
c(quantile(sample_mean_10, 0.025), quantile(sample_mean_10, 0.975))
```

```
##  2.5% 97.5%
##   0.2   0.8
```

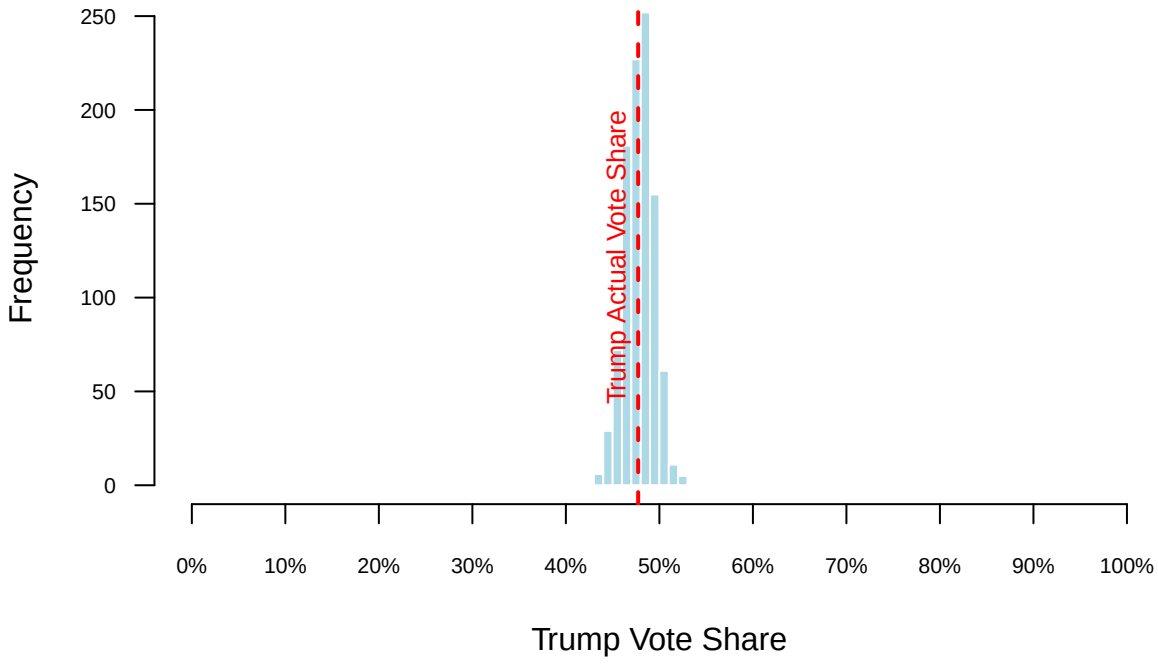
```
c(quantile(sim_1000_mean, 0.025), quantile(sim_1000_mean, 0.975))
```

```
##  2.5% 97.5%
## 0.447 0.508
```

We can see a significant reduction in uncertainty about Trump's vote share when we ask 1,000 people compared to when we only ask 10 people.

Plot the new histogram

```
# Create the histogram
hist(sim_1000_mean,
     # main = "Histogram of Polsby-Popper Compactness\n (compared to 500,000 simulations)",
     main = "",
     xlab = "Trump Vote Share",
     col = "lightblue",
     border = "white",
     xlim = c(0,1),
     yaxt = "n",
     xaxt = "n")

# Add the custom x-axis
axis(side = 1, at = seq(0,1,0.1), labels = paste0(seq(0,1,0.1)*100,"%"), cex.axis = 0.65)
# Add the custom y-axis
axis(side = 2, las = 2, at = seq(0,250,50), labels = seq(0,250,50), cex.axis = 0.65)

# Add a vertical line at Trump's Vote Share
abline(v = trump_proportion, col = "red", lwd = 2, lty = 2)

# Add text to label the line
text(x = trump_proportion, y = 200,
     labels = "Trump Actual Vote Share",
     pos = 2, col = "red", cex = 0.8, srt = 90)
```



Law of large numbers

If you are running a campaign and only have so many resources, you may not want to poll more people than necessary to understand the current state of the election. So how many peole should you poll?

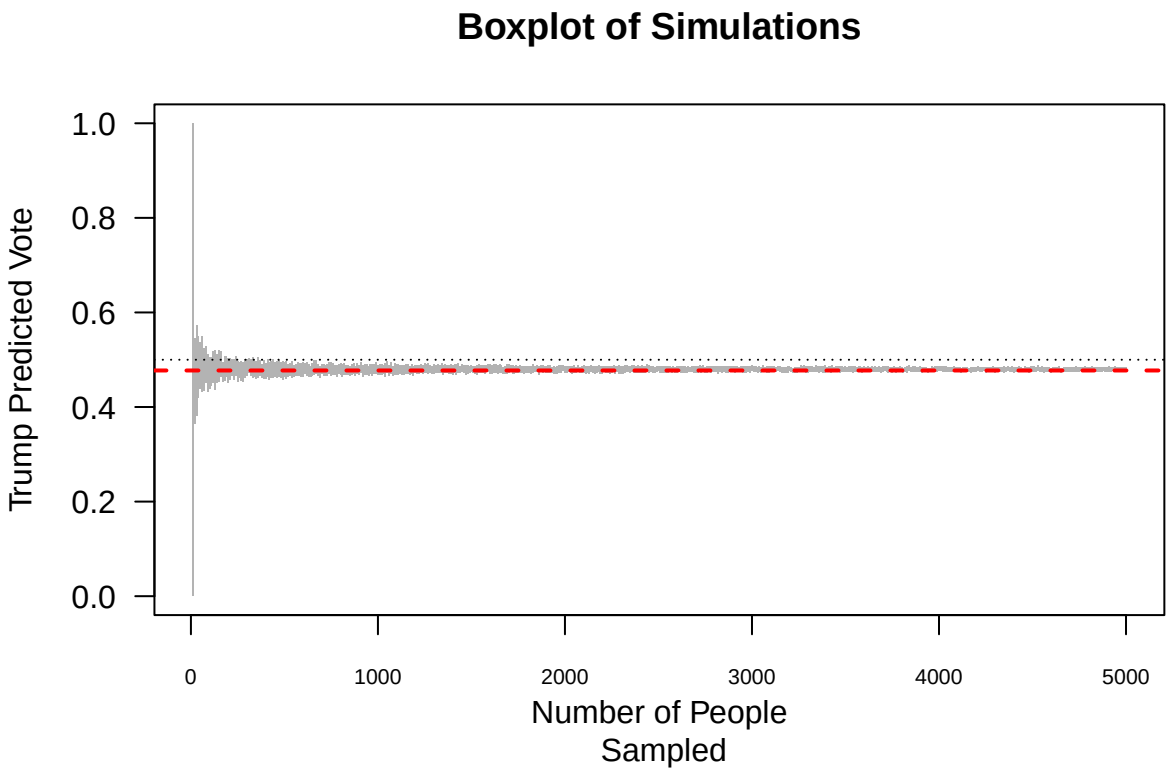
Simulate polls with sample sizes ranging from 10 to 5000

```
sim_n <- seq(1,5000,10)
sim_full_i <- list()
mean_sim <- list()
for (i in seq_along(sim_n)) {
     sim_full_i[[i]] <- sim_sample(100,sim_n[i],population)
     mean_sim[[i]] <- unlist(lapply(sim_full_i[[i]], mean))
}
```

Create a box and whisker plot to demonstrate the reduction in error as the sample size increases

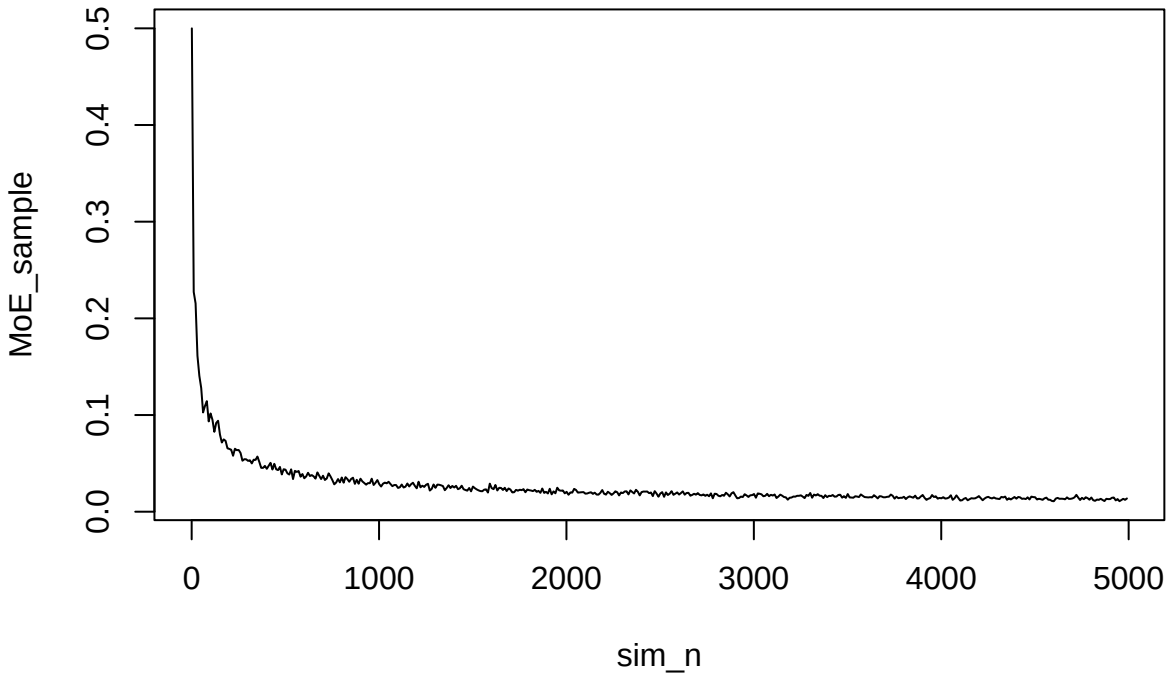
```
boxplot(as.data.frame(mean_sim),
        main = "Boxplot of Simulations",
        xlab = "Number of People\n Sampled",
        ylab = "Trump Predicted Vote",
        col = "gray70",
        border = F,
        las = 2, # Rotate x-axis labels
        xaxt = "n",
```

```
        outline = FALSE) # Avoid showing outliers as individual points

# Add a horizontal line at 50% support and Trump Proportion
abline(h = trump_proportion, col = "red", lwd = 2, lty = 2)
abline(h = 0.5, col = "black", lwd = 1, lty = 3)

# Add the custom x-axis
axis(side = 1, at = seq(0,500,100), labels = seq(0,5000,1000), cex.axis = 0.65)
```



It is also useful to see the reduction in error as a function of sample size. Create a line plot showing the difference between the 2.5 percentile and the 97.5 percentile.

```
# 95 percentile for each number of persons sampled
confidence_sample <- lapply(mean_sim, FUN=function(x) c(quantile(x, 0.025), quantile(x, 0.975)))
     confidence_sample <- do.call(rbind,confidence_sample)

     MoE_sample <- (confidence_sample[,2]-confidence_sample[,1])/2

     plot(sim_n,MoE_sample, type="l")
```



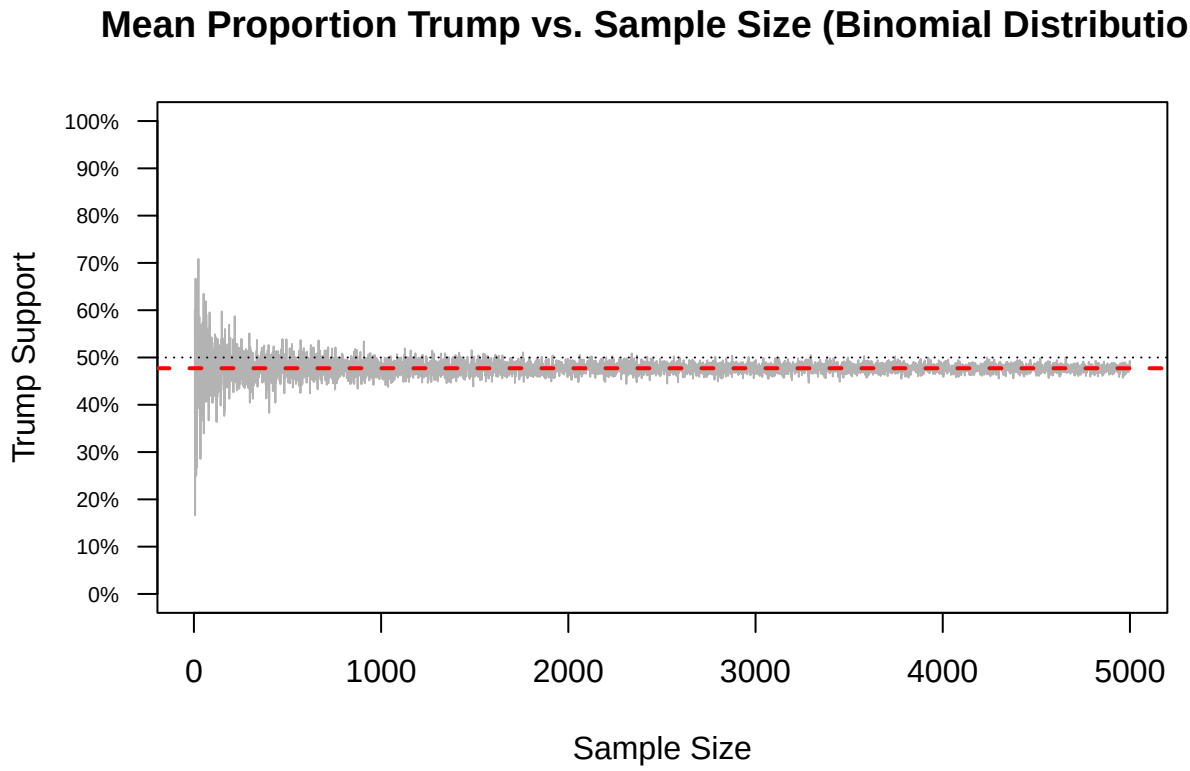
Alternative Way to do this, calculating the margin of error through simulation

Write a function to calculate the margin of error and mean of each simulation:

```
calculate_binomial_margin_of_error <- function(
    sample_sizes,
    trump_proportion,
    confidence_level = 0.95) {
  Z <- qnorm((1 + confidence_level) / 2)  # Z-score for the confidence level
  MoE <- numeric(length(sample_sizes))
  sample <- list()
  for (i in seq_along(sample_sizes)) {
    n <- sample_sizes[i]
    sample[[i]] <- rbinom(n, 1, trump_proportion)
    sample_proportion <- mean(sample[[i]])
    MoE[i] <- Z * sqrt((sample_proportion * (1 - sample_proportion)) / n)
  }
  return(list(sample,MoE))
}

sample_sizes <- seq(5, 5000, by = 1)
MoE <- calculate_binomial_margin_of_error(sample_sizes, trump_proportion)
```

Plot the mean of each sample, with the sample size on the x-axis and the mean Trump support on the y-axis.

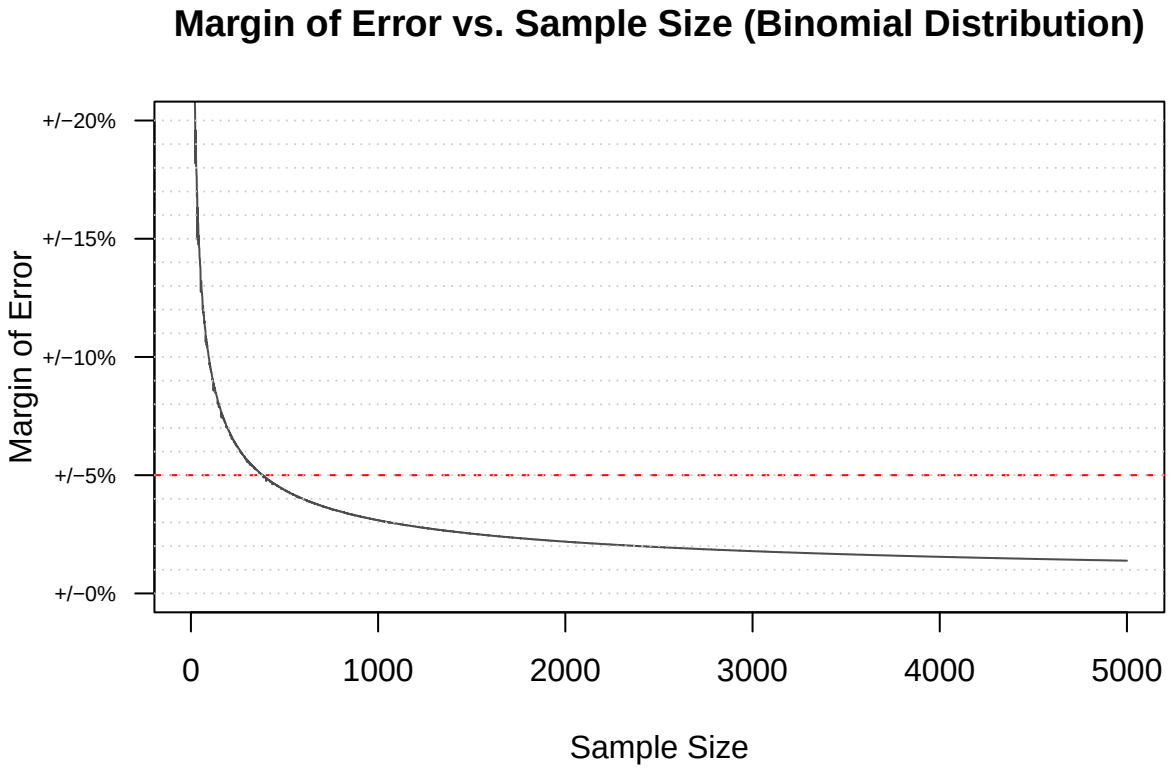
```r
  sample_mean <- unlist(lapply(MoE[[1]], FUN= mean))
# Plotting the Means
plot(sample_sizes, sample_mean, type = "l", col = "gray70",
     xlab = "Sample Size", ylab = "Trump Support",
     main = "Mean Proportion Trump vs. Sample Size (Binomial Distribution)",
     ylim = c(0,1),
     yaxt='n'
     )
y_values <- seq(0,1,0.1)
axis(side=2, las=2, at=y_values, label=paste0(y_values * 100, "%"), cex.axis = 0.65)

# Add a horizontal line at 50% support and Trump Proportion
abline(h = 0.5, col = "black", lwd = 1, lty = 3)
abline(h = trump_proportion, col = "red", lwd = 2, lty = 2)
```



Plot the margin of error as the sample size increases

```r
# Plotting the Margin of Error
plot(sample_sizes, MoE[[2]], type = "l", col = "gray30",
     xlab = "Sample Size", ylab = "Margin of Error",
     main = "Margin of Error vs. Sample Size (Binomial Distribution)",
     ylim = c(0,0.2),
     yaxt='n'
     )

y_values <- seq(0,0.2,0.05)
```

```r
axis(side=2, las=2, at=y_values, label=paste0("+/-", y_values * 100, "%"), cex.axis = 0.65)

# Add a horizontal line representing a common threshold (e.g., 5%)
abline(h = 0.05, col = "red", lty = 2)

abline(h = seq(0,0.20,0.01), col="gray80", lty=3)
```



Non-response

We also learned about non-response and how it might bias our sample. How might it matter if 40% of those who support Trump fail to respond to the pollster?

```r
nonresponse_rate <- 0.4  # 40% of those who oppose the policy do not respond
response <- ifelse(population == 0, 1, rbinom(population_size, 1, 1 - nonresponse_rate))
responded_population <- population[response == 1]

# Compare proportions
true_proportion <- mean(population)
responded_proportion <- mean(responded_population)

cat("True Proportion of 'Yes' Responses in Population:", true_proportion, "\n")
```

```
## True Proportion of 'Yes' Responses in Population: 0.47983
```

```
cat("Proportion of 'Yes' Responses in Responded Population:", responded_proportion, "\n")
```

```
## Proportion of 'Yes' Responses in Responded Population: 0.3550839
```