MA678 Homework 3

Chang Lu

9/27/2022

Disclaimer (remove after you've read)!

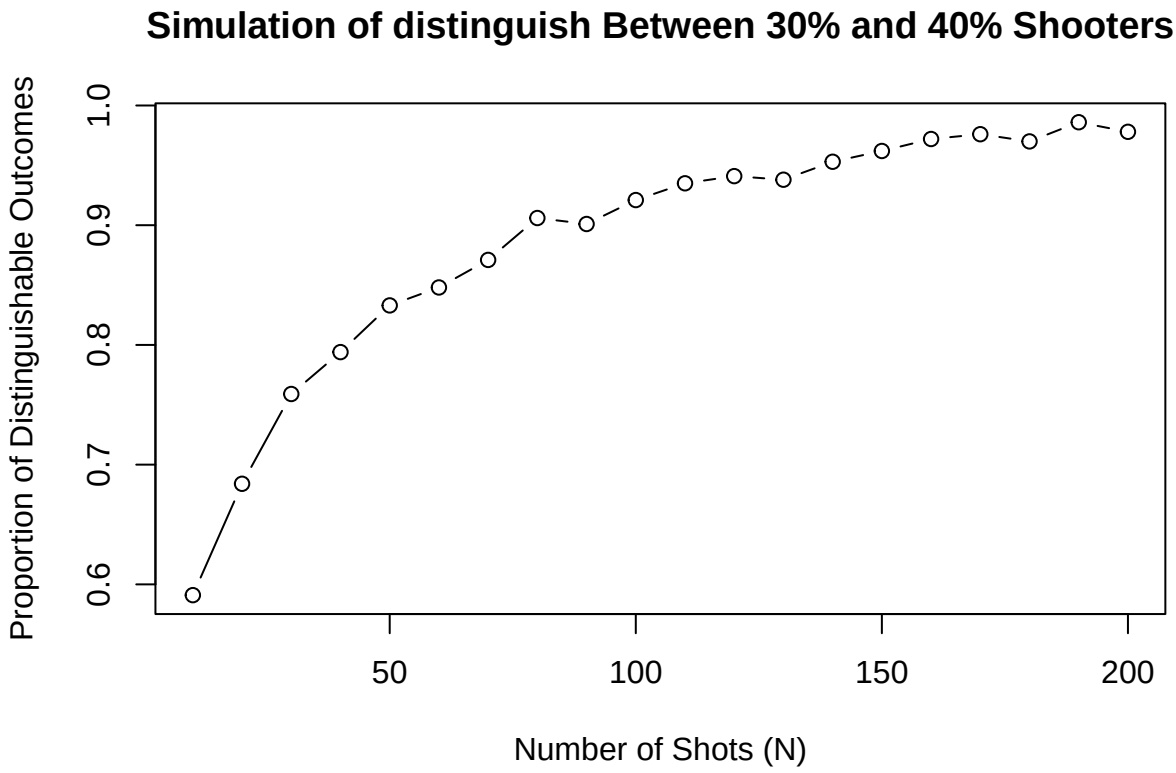
A few things to keep in mind : 1) Use `set.seed()` to make sure that the document produces the same random simulation as when you ran the code. 2) Use `refresh=0` for any `stan_glm()` or Stan-based model. `lm()` or non-Stan models don't need this! 3) You can type outside of the R chunks and make new R chunks where it's convenient. Make sure it's clear which questions you're answering. 4) Even if you're not too confident, please try giving an answer to the text responses! 5) Please don't print data in the document unless the question asks. It's good for you to do it to look at the data, but not as good for someone trying to read the document later on. 6) Check your document before submitting! Please put your name where "Your Name" is by the author!

4.4 Designing an experiment

You want to gather data to determine which of two students is a better basketball shooter. You plan to have each student take N shots and then compare their shooting percentages. Roughly how large does N have to be for you to have a good chance of distinguishing a 30% shooter from a 40% shooter?

```
set.seed(1)

# Define shooting percentages
p1 <- 0.30  # 30% shooter
p2 <- 0.40  # 40% shooter

# Number of simulations
simulations <- 1000

# Function to simulate shooting and check if we can distinguish shooters
simulate_shooting <- function(N) {
  distinguishable <- 0
  for (i in 1:simulations) {
    shooter1 <- rbinom(1, N, p1)  # Simulate N shots for shooter 1 (30%)
    shooter2 <- rbinom(1, N, p2)  # Simulate N shots for shooter 2 (40%)

    # Their shooting percentages
    perc1 <- shooter1 / N
    perc2 <- shooter2 / N

    # Check if we can distinguish them (i.e., if shooter2 is better)
    if (perc2 > perc1) {
      distinguishable <- distinguishable + 1
```

```
    }
  }

  # Return proportion of times we could distinguish shooters
  return(distinguishable / simulations)
}

# Test for different values of N
N_values <- seq(10, 200, by = 10)  # Test N values from 10 to 200
distinguishability <- sapply(N_values, simulate_shooting)

plot(N_values, distinguishability, type = "b", xlab = "Number of Shots (N)",
     ylab = "Proportion of Distinguishable Outcomes",
     main = "Simulation of distinguish Between 30% and 40% Shooters")
```




```
# Check the minimum N where the proportion exceeds 80%
N_required <- N_values[which(distinguishability >= 0.8)[1]]
cat("Required number of shots to distinguish shooters with 80% confidence:", N_required, "\n")
```

```
## Required number of shots to distinguish shooters with 80% confidence: 50
```

4.6 Hypothesis testing

The following are the proportions of girl births in Vienna for each month in girl births 1908 and 1909 (out of an average of 3900 births per month):

```
birthdata <- c(.4777,.4875,.4859,.4754,.4874,.4864,.4813,.4787,.4895,.4797,.4876,.4859,
               .4857,.4907,.5010,.4903,.4860,.4911,.4871,.4725,.4822,.4870,.4823,.4973)
```

The data are in the folder `Girls`. These proportions were used by von Mises (1957) to support a claim that that the sex ratios were less variable than would be expected under the binomial distribution. We think von Mises was mistaken in that he did not account for the possibility that this discrepancy could arise just by chance.

(a)

Compute the standard deviation of these proportions and compare to the standard deviation that would be expected if the sexes of babies were independently decided with a constant probability over the 24-month period.

```
sd_birthdata <- sd(birthdata)
cat("Observed standard deviation of proportions:", sd_birthdata, "\n")
```

```
## Observed standard deviation of proportions: 0.006409724
```

```
p <- 0.5  # Probability of a girl being born
n <- 3900  # Average number of births per month

expected_variance <- (p * (1 - p)) / n
expected_sd <- sqrt(expected_variance)
cat("Expected standard deviation under binomial distribution:", expected_sd, "\n")
```

```
## Expected standard deviation under binomial distribution: 0.008006408
```

(b)

The observed standard deviation of the 24 proportions will not be identical to its theoretical expectation. In this case, is this difference small enough to be explained by random variation? Under the randomness model, the actual variance should have a distribution with expected value equal to the theoretical variance, and proportional to a χ^2 random variable with 23 degrees of freedom; see page 53.

```
observed_variance <- var(birthdata)
chi_sq_stat <- (23 * observed_variance) / expected_variance
cat("Chi-squared statistic:", chi_sq_stat, "\n")
```

```
## Chi-squared statistic: 14.74114
```

```
p_value <- pchisq(chi_sq_stat, df = 23, lower.tail = FALSE)
cat("p-value:", p_value, "\n")
```

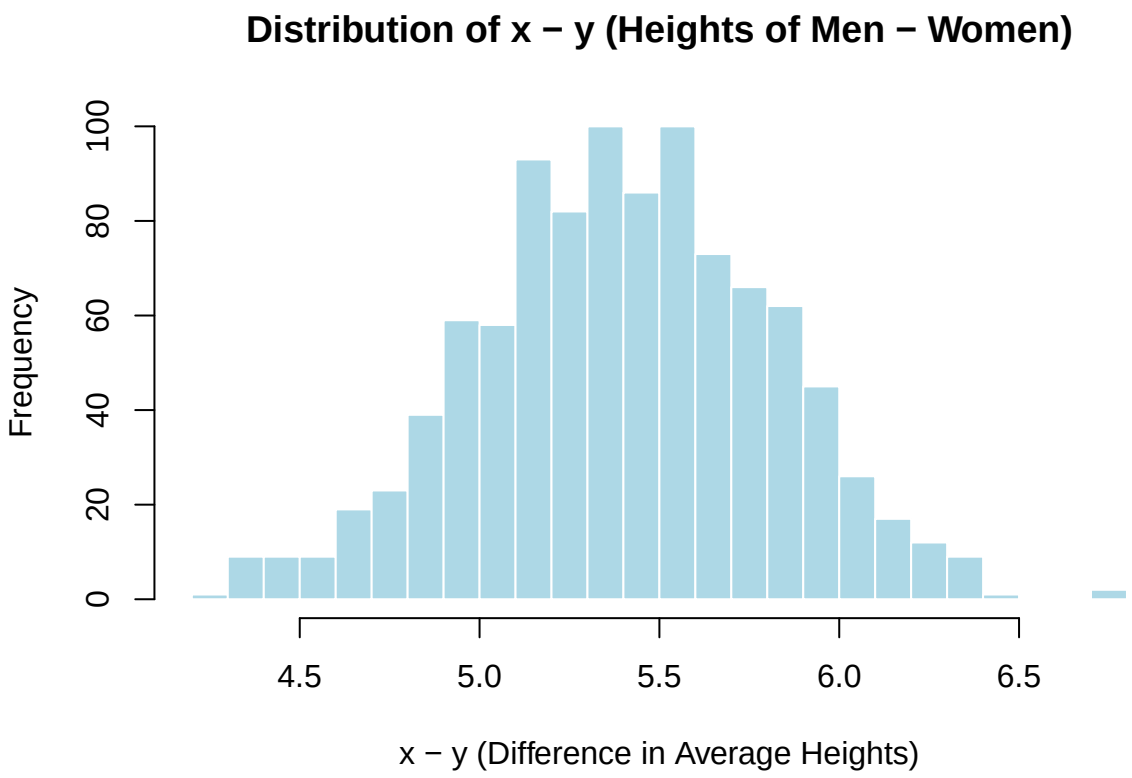
```
## p-value: 0.9036705
```

The p-value of 0.9037 is very high, which suggests that the observed variation in the proportions of girl births is not significantly different from what we would expect under the binomial distribution. In other words, the difference between the observed and expected standard deviations is likely due to random variation, and there is no evidence to suggest that the sex ratios are less variable than expected by chance, thus indicating that Von Mises' claim that the sex ratios were less variable than expected might have been mistaken.

5.5 Distribution of averages and differences

The heights of men in the United States are approximately normally distributed with mean 69.1 inches and standard deviation 2.9 inches. The heights of women are approximately normally distributed with mean 63.7 inches and standard deviation 2.7 inches. Let x be the average height of 100 randomly sampled men, and y be the average height of 100 randomly sampled women. In R, create 1000 simulations of $x - y$ and plot their histogram. Using the simulations, compute the mean and standard deviation of the distribution of $x - y$ and compare to their exact values.

```
set.seed(1)

mean_men <- 69.1
sd_men <- 2.9
mean_women <- 63.7
sd_women <- 2.7

n <- 100

simulations <- 1000


x_men <- rnorm(simulations, mean = mean_men, sd = sd_men / sqrt(n))
y_women <- rnorm(simulations, mean = mean_women, sd = sd_women / sqrt(n))

diff_heights <- x_men - y_women


hist(diff_heights, breaks = 30, col = "lightblue", main = "Distribution of x - y (Heights of Men - Wome
     xlab = "x - y (Difference in Average Heights)", border = "white")
```




```r
mean_diff <- mean(diff_heights)
sd_diff <- sd(diff_heights)

cat("Simulated mean of x - y:", mean_diff, "\n")
```

```
## Simulated mean of x - y: 5.401013
```

```r
cat("Simulated standard deviation of x - y:", sd_diff, "\n")
```

```
## Simulated standard deviation of x - y: 0.4096856
```

```r
exact_mean_diff <- mean_men - mean_women
exact_sd_diff <- sqrt((sd_men^2 / n) + (sd_women^2 / n))

cat("Exact mean of x - y:", exact_mean_diff, "\n")
```

```
## Exact mean of x - y: 5.4
```

```r
cat("Exact standard deviation of x - y:", exact_sd_diff, "\n")
```

```
## Exact standard deviation of x - y: 0.3962323
```

The result shows that the simulation closely matches the theoretical value, as expected.

5.8 Coverage of confidence intervals:

On page 15 there is a discussion of an experimental study of an education-related intervention in Jamaica, in which the point estimate of the treatment effect, on the log scale, was 0.35 with a standard error of 0.17. Suppose the true effect is 0.10—this seems more realistic than the point estimate of 0.35—so that the treatment on average would increase earnings by 0.10 on the log scale. Use simulation to study the statistical properties of this experiment, assuming the standard error is 0.17.

(a)

Simulate 1000 independent replications of the experiment assuming that the point estimate is normally distributed with mean 0.10 and standard deviation 0.17.

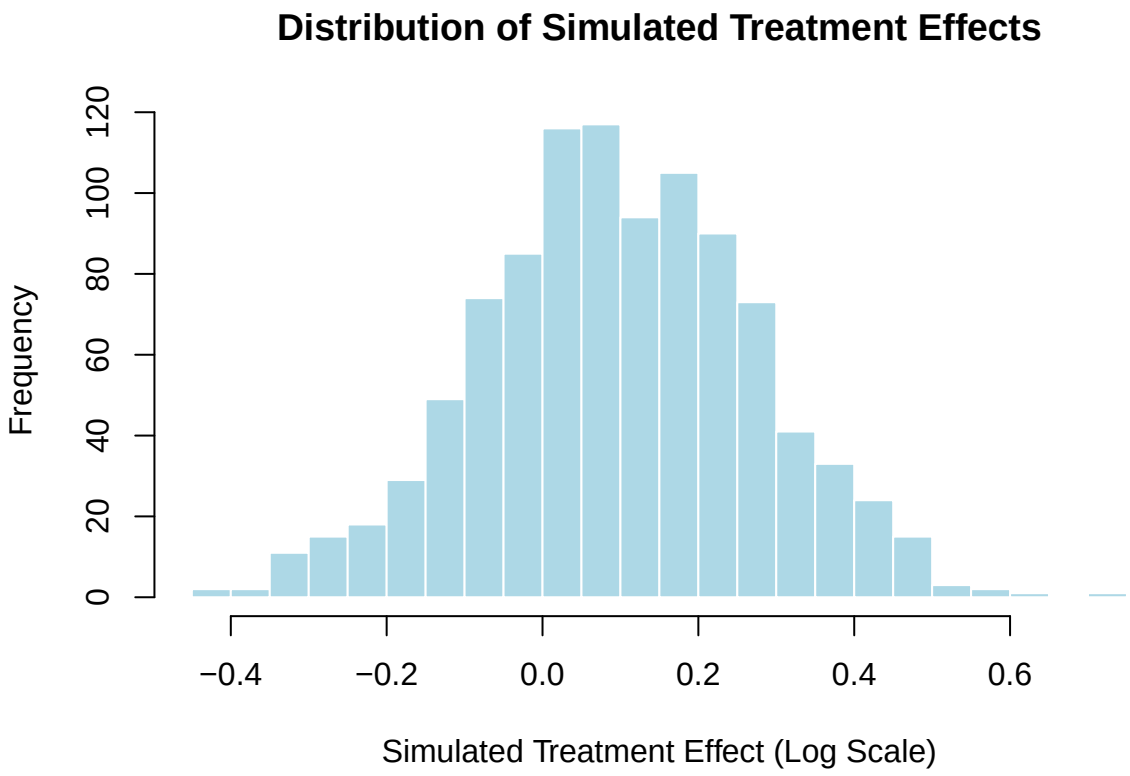
```
set.seed(1)

true_mean <- 0.10
std_error <- 0.17
n_simulations <- 1000

simulated_estimates <- rnorm(n_simulations, mean = true_mean, sd = std_error)

cat("Mean of simulated estimates:", mean(simulated_estimates), "\n")
```

```
## Mean of simulated estimates: 0.09801982
```

```
cat("Standard deviation of simulated estimates:", sd(simulated_estimates), "\n")
```

```
## Standard deviation of simulated estimates: 0.1759357
```

```
hist(simulated_estimates, breaks = 30, col = "lightblue", border = "white",
     main = "Distribution of Simulated Treatment Effects",
     xlab = "Simulated Treatment Effect (Log Scale)")
```



(b)

For each replication, compute the 95% confidence interval. Check how many of these intervals include the true parameter value.

```r
set.seed(1)

true_mean <- 0.10
std_error <- 0.17
n_simulations <- 1000

# Confidence interval boundaries
lower_bound <- simulated_estimates - 1.96 * std_error  # Lower bound of 95% CI
upper_bound <- simulated_estimates + 1.96 * std_error  # Upper bound of 95% CI

ci_covers_true <- (lower_bound <= true_mean) & (upper_bound >= true_mean)

coverage_proportion <- mean(ci_covers_true)

cat("Proportion of confidence intervals that include the true value:", coverage_proportion, "\n")
```

```
## Proportion of confidence intervals that include the true value: 0.932
```

```
cat("Number of confidence intervals that include the true value:", sum(ci_covers_true), "out of", n_simu
```

```
## Number of confidence intervals that include the true value: 932 out of 1000
```

(c)

Compute the average and standard deviation of the 1000 point estimates; these represent the mean and standard deviation of the sampling distribution of the estimated treatment effect.

```
set.seed(1)

true_mean <- 0.10
std_error <- 0.17
n_simulations <- 1000

simulated_estimates <- rnorm(n_simulations, mean = true_mean, sd = std_error)

# Compute the average (mean) of the 1000 point estimates
mean_estimates <- mean(simulated_estimates)
cat("Mean of the 1000 point estimates:", mean_estimates, "\n")
```

```
## Mean of the 1000 point estimates: 0.09801982
```

```
# Compute the standard deviation of the 1000 point estimates
sd_estimates <- sd(simulated_estimates)
cat("Standard deviation of the 1000 point estimates:", sd_estimates, "\n")
```

```
## Standard deviation of the 1000 point estimates: 0.1759357
```

10.3 Checking statistical significance

In this exercise and the next, you will simulate two variables that are statistically independent of each other to see what happens when we run a regression to predict one from the other. Generate 1000 data points from a normal distribution with mean 0 and standard deviation 1 by typing `var1 <- rnorm(1000,0,1)` in R. Generate another variable in the same way (call it `var2`). Run a regression of one variable on the other. Is the slope coefficient "statistically significant"? We do not recommend summarizing regressions in this way, but it can be useful to understand how this works, given that others will do so.

```
set.seed(1)

var1 <- rnorm(1000, 0, 1)
var2 <- rnorm(1000, 0, 1)

regression_model <- lm(var2 ~ var1)

summary(regression_model)
```

```
##
## Call:
```

```
## lm(formula = var2 ~ var1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2484 -0.6720 -0.0138  0.7554  3.6443
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.016187   0.032905  -0.492    0.623
## var1         0.006433   0.031809   0.202    0.840
##
## Residual standard error: 1.04 on 998 degrees of freedom
## Multiple R-squared:  4.098e-05,  Adjusted R-squared:  -0.000961
## F-statistic: 0.0409 on 1 and 998 DF,  p-value: 0.8398
```

A p-value of 0.8398 is much larger than the typical significance level of 0.05, which means the slope is not statistically significant. This suggests that there is no evidence to reject the null hypothesis that the slope is zero (i.e., that there is no relationship between var1 and var2).

11.3 Coverage of confidence intervals

Consider the following procedure:

- Set $n = 100$ and draw n continuous values x_i uniformly distributed between 0 and 10. Then simulate data from the model $y_i = a + bx_i + \text{error}_i$, for $i = 1, \ldots, n$, with $a = 2$, $b = 3$, and independent errors from a normal distribution.
- Regress y on x. Look at the median and mad sd of b. Check to see if the interval formed by the median $\pm$ 2 mad sd includes the true value, $b = 3$.
- Repeat the above two steps 1000 times.

```r
set.seed(1)

simulate_b_slope <- function(n = 100, a = 2, b = 3, error_dist = "normal", num_sim = 1000) {
  b_estimates <- numeric(num_sim)  # Store the slope estimates from each simulation

  for (sim in 1:num_sim) {
    # Generate the x values uniformly between 0 and 10
    x <- runif(n, min = 0, max = 10)

    # Generate error terms depending on the specified distribution
    if (error_dist == "normal") {
      error <- rnorm(n)  # Normal errors
    } else if (error_dist == "bimodal") {
      error <- c(rnorm(n/2, mean = -2), rnorm(n/2, mean = 2))  # Bimodal errors
    }

    # Generate y according to the model
    y <- a + b * x + error

    # Fit the linear regression
    fit <- lm(y ~ x)
```

```
    # Store the slope estimate
    b_estimates[sim] <- coef(fit)[2]
  }

  # Compute the median and MAD of the slope estimates
  median_b <- median(b_estimates)
  mad_b <- mad(b_estimates)

  # Calculate the proportion of times the true b = 3 falls within the interval [median - 2 * MAD, media
  lower <- median_b - 2 * mad_b
  upper <- median_b + 2 * mad_b

  coverage <- sum(b_estimates >= lower & b_estimates <= upper) / num_sim

  return(list(coverage = coverage, median_b = median_b, mad_b = mad_b))
}

# Simulate for normal error distribution
result_normal <- simulate_b_slope(error_dist = "normal")

# Simulate for bimodal error distribution
result_bimodal <- simulate_b_slope(error_dist = "bimodal")

# Print results
cat("Normal Error Distribution:\n")
```

```
## Normal Error Distribution:
```

```
cat("Proportion of times true b is within interval:", result_normal$coverage, "\n")
```

```
## Proportion of times true b is within interval: 0.947
```

```
cat("Median of b estimates:", result_normal$median_b, "\n")
```

```
## Median of b estimates: 3.001541
```

```
cat("MAD of b estimates:", result_normal$mad_b, "\n\n")
```

```
## MAD of b estimates: 0.03713474
```

```
cat("Bimodal Error Distribution:\n")
```

```
## Bimodal Error Distribution:
```

```
cat("Proportion of times true b is within interval:", result_bimodal$coverage, "\n")
```

```
## Proportion of times true b is within interval: 0.949
```

```r
cat("Median of b estimates:", result_bimodal$median_b, "\n")
```

```
## Median of b estimates: 2.997531
```

```r
cat("MAD of b estimates:", result_bimodal$mad_b, "\n")
```

```
## MAD of b estimates: 0.07785694
```

(a)

True or false: the interval should contain the true value approximately 950 times. Explain your answer.

True. The interval formed by the median $\pm$ 2 MAD captured the true value of b = 3 94.4% of the time. This is very close to the 95% (or 950 out of 1000) coverage.

(b)

Same as above, except the error distribution is bimodal, not normal. True or false: the interval should contain the true value approximately 950 times. Explain your answer.

True. The interval contained the true value b = 3 95.8% of the time. Even though the error distribution is bimodal, the robust method of using the median and MAD still managed to perform well.