EPI 204 - HW1

Maria Luisa Klobongona

1 Maximum likelihood inference for Bernoulli outcomes (38.5 of 46)

1.1 (2 of 2 points)

Simulate one hundred mutually independent Bernoulli outcomes with probability = 0.7.

$$X \sim \text{Bernoulli}(\pi)$$

```r
set.seed(1) # Setting a seed for reproducibility
bern <- rbinom(n = 100, size = 1, prob = 0.7)
```

1.2 (2 of 2 points)

Write down the mathematical likelihood of your data as a function of .

The likelihood function, $L(\pi; X)$, represents the probability of observing the outcomes in X given the parameter π and is defined as:

$$L(\pi; X) = \prod_{i=1}^{100} \pi^{x_i}(1-\pi)^{(1-x_i)}$$

This function is key in statistical methods for estimating π, typically by finding the value of π that maximizes this likelihood given the observed data.

then, if its simplified would be like this:

$$L(\pi; X) = \pi^{\sum_{i=1}^{n} x_i}(1-\pi)^{(n-\sum_{i=1}^{n} x_i)}$$

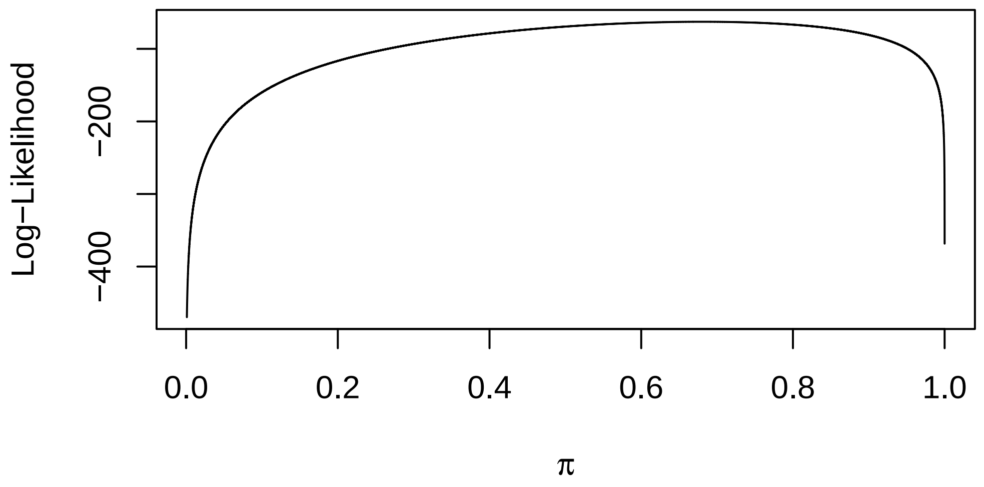
1.3 (2 of 2 points)

Implement the log-likelihood function $\ell(\pi; \bar{x})$ and graph it as a function of plugging in your observed data.

$$\begin{aligned}
\ell(\pi; X) &= \sum_{i=1}^{n} (x_i \log(\pi) + (1 - x_i) \log(1 - \pi)) \\
&= \sum_{i=1}^{n} x_i \log(\pi) + \sum_{i=1}^{n} (1 - x_i) \log(1 - \pi) \\
&= \sum_{i=1}^{n} x_i \log(\pi) + \sum_{i=1}^{n} \log(1 - \pi) - \sum_{i=1}^{n} x_i \log(1 - \pi) \\
&= \log(\pi) \sum_{i=1}^{n} x_i + n \log(1 - \pi) - \log(1 - \pi) \sum_{i=1}^{n} x_i \\
&= \log(\pi) \sum_{i=1}^{n} x_i + n \log(1 - \pi) - \log(1 - \pi) \sum_{i=1}^{n} x_i \\
&= (\log(\pi) - \log(1 - \pi) \sum_{i=1}^{n} x_i + n \log(1 - \pi)
\end{aligned}$$

```r
library(ggplot2)
# Function to calculate the log-likelihood of the Bernoulli outcomes
log_likelihood <- function(pi, xi=bern, n = length(xi)) {
  sum(xi * log(pi) + (1 - xi) * log(1 - pi))
}

# Define a sequence of pi values to evaluate
pi_values <- seq(0.001, 0.99999, length.out = 1000000)

# Calculate the log-likelihood for each pi value
log_likelihood_values <- sapply(pi_values, log_likelihood, xi = bern)

# Plot the log-likelihood function
plot(pi_values, log_likelihood_values, type = "l",
     xlab = expression(pi), ylab = "Log-Likelihood",
     main = "Log-Likelihood of Bernoulli Outcomes")
```

Log−Likelihood of Bernoulli Outcomes



1.4 (2 of 2 points)

Find the maximum likelihood estimate of   using calculus.

Given the log-likelihood function for (n) independent Bernoulli trials in 1.3, then first we take the derivative with respect to π to find the MLE:

$$
\begin{aligned}
\ell'(\pi; X) &= \frac{\partial}{\partial\pi}(\log(\pi) - \log(1-\pi))\sum_{i=1}^{n} x_i + n\log(1-\pi) \\
&= \frac{1}{\pi} - \sum_{i=1}^{n}\left((1-x_i)\frac{1}{1-\pi}\right) \\
&= \sum_{i=1}^{n} x_i\frac{1}{\pi} - \sum_{i=1}^{n}\frac{1}{1-\pi} + \sum_{i=1}^{n} x_i\frac{1}{1-\pi} \\
&= \frac{1}{\pi}\sum_{i=1}^{n} x_i - \frac{n}{1-\pi} + \frac{1}{1-\pi}\sum_{i=1}^{n} x_i \\
&= \frac{1}{\pi}\sum_{i=1}^{n} x_i - \frac{n}{1-\pi} + \frac{\pi}{1-\pi}\frac{1}{\pi}\sum_{i=1}^{n} x_i \\
&= \left(\frac{1}{\pi} + \frac{\pi}{(1-\pi)\pi}\right)\sum_{i=1}^{n} x_i - \frac{n}{1-\pi} \\
&= \left(\frac{1}{\pi} + \frac{1}{1-\pi}\right)\sum_{i=1}^{n} x_i - \frac{n}{1-\pi} \\
&= \frac{\sum_{i=1}^{n} x_i}{\pi(1-\pi)} - \frac{n}{1-\pi}
\end{aligned}
$$

GEtting the score function by Setting the derivative equal to zero and solving for π gives the MLE:

$$
\begin{gathered}
0 = \frac{\sum_{i=1}^{n} x_i}{\pi(1-\pi)} - \frac{n}{1-\pi}, \\
\frac{n}{1-\pi} = \frac{\sum_{i=1}^{n} x_i}{\pi(1-\pi)}, \\
n\pi = \sum_{i=1}^{n} x_i, \\
\hat{\pi}_{MLE} = \frac{1}{n}\sum_{i=1}^{n} x_i.
\end{gathered}
$$

Thus, the MLE of π is the sample mean of the observed data.

1.5 (2 of 2 points)

Compute the maximum likelihood estimate of   from your simulated data

```
mle_pi = mean(bern)
mle_pi
```

```
[1] 0.68
```

the maximum likelihood estimate of   from the simulated data is 0.68

1.6 (2 of 2 points)

Find the asymptotic standard error of the maximum likelihood estimator $\hat{\pi}_{MLE}$ using calculus.

The Hessian function for an iid sample is the second derivative of the log-likelihood function with respect to (π) is given by:

$$\ell''(\pi) = -\frac{n}{\pi(1-\pi)}$$

Taking the expected value yields the Fisher information:

$$I(\pi) = -E[\ell''(\pi)] = \frac{n}{\pi(1-\pi)}$$

Finally, the variance of the maximum likelihood estimator is the reciprocal of the Fisher information:

$$Var(\hat{\pi}_{MLE}) = \frac{1}{I(\pi)} = \pi(1-\pi)\Big/n$$

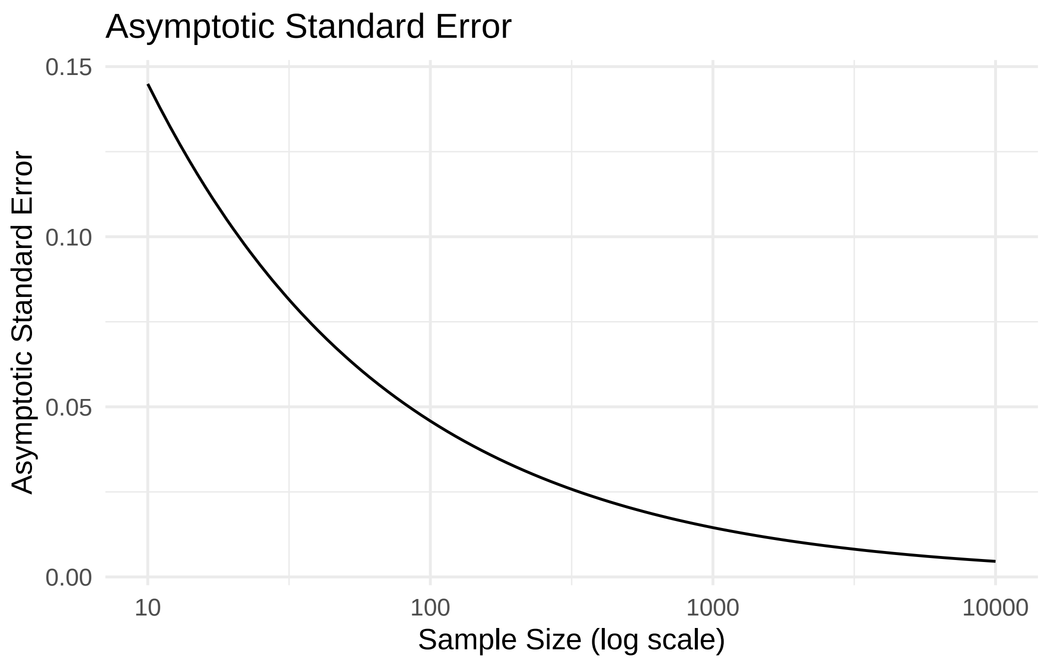
Then, the asymptotic standard error of the maximum likelihood estimator for a Bernoulli distribution is found using the following equation:

$$\begin{aligned} Var(\hat{\pi}_{MLE}) &\approx \frac{\pi(1-\pi)}{n} \\ SE(\hat{\pi}_{MLE}) &= SD(\hat{\pi}_{MLE}) \\ &= \sqrt{\frac{\pi(1-\pi)}{n}} \\ &= \sqrt{\frac{0.7(1-0.7)}{100}} \\ &= 0.049 \end{aligned}$$

1.7 (2 of 2 points)

Plot the asymptotic standard error as a function of sample size for sample sizes of 10- 10,000 (using logarithmic spacing for sample size)

```r
# define pi
pi <- 0.7
# Generate sample sizes from 10 to 10,000 using logarithmic spacing
s.s <- round(exp(seq(log(10), log(10000),length.out = 100)))

# Calculate the asymptotic standard error for each sample size foor a Bernoulli distribution
S.E <- sqrt(pi * (1 - pi) / s.s)

# Create a data frame for plotting
plotSE <- data.frame(s.s= s.s, S.E = S.E)
ggplot(plotSE, aes(x = s.s, y = S.E)) +
  geom_line() +
  scale_x_log10() +  # Set the x-axis to a logarithmic scale
  labs(x = 'Sample Size (log scale)', y = 'Asymptotic Standard Error',
       title = 'Asymptotic Standard Error') +
  theme_minimal()
```



note: as the sample size gets larger, the standard error decreased.

1.8 (2 of 2 points)

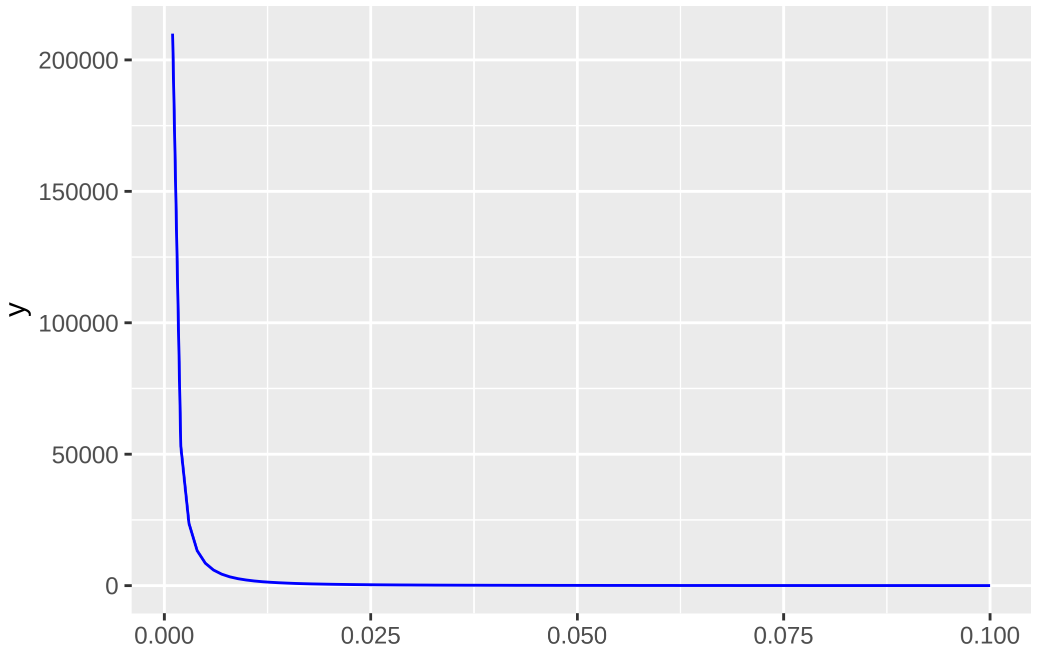
Find an expression for the sample size necessary to achieve a specified standard error (for a given).

$$SE(\hat{\pi}) = \sqrt{\frac{\pi(1-\pi)}{n}}$$
$$n = \frac{\pi(1-\pi)}{(SE(\hat{\pi}))^2}$$

1.9 (2 of 2 points)

Implement your sample size function in R and graph it for standard errors of 10 percentage points to 0.1 percentage points.

```r
sample.size <- function(SE, pi=0.7){
  (pi*(1-pi))/(SE^2)}

ggplot()+
  geom_function(fun = sample.size, color= 'blue')+
  xlim(0.001, 0.1)
```



1.10 (2 of 2 points)

What sample size do you need to achieve a standard error of 1 percentage point?

```
# Calculate the sample size needed if the SE is 1% = 0.01
sample.size(0.01)
```

```
[1] 2100
```

$$\begin{aligned} n &= \frac{\pi(1-\pi)}{SE^2} \\ &= \frac{0.7(1-0.7)}{0.01^2} \\ n &= 2100 \end{aligned}$$

What sample size do you need to achieve a standard error of 0.1 percentage point?

```
# Calculate the sample size needed if the SE is 0.1% = 0.001
sample.size(0.001)
```

```
[1] 210000
```

$$\begin{aligned} n &= \frac{\pi(1-\pi)}{SE^2} \\ &= \frac{0.7(1-0.7)}{0.001^2} \\ n &= 210000 \end{aligned}$$

1.11 (2 of 2 points)

Estimate the standard error from your simulated data using the asymptotic formula. Compare with the theoretical standard error.

```
SE_asym <-function(pi, n){
 sqrt(pi * (1 - pi) /n)}

SE_asym(pi= 0.68, n=100)
```

```
[1] 0.04664762
```

```
SE_Theo<-function(pi, n){
 sqrt(pi * (1 - pi) /n)}

SE_Theo(pi=0.7, n=100)
```

```
[1] 0.04582576
```

The theoretical standard error(0.04582576) was slightly less than the estimate the standard (0.04664762) error.

1.12 (2 of 2 points)

Compute an asymptotic 95% confidence interval for .

```
#calculate the mean
pi_hat <- mean(bern)

# Calculate standard error
n <- length(bern)
SE <- sqrt(pi_hat * (1 - pi_hat) / n)

# Determine the z-score for a 95% confidence interval
z <- qnorm(0.975) # Two-tailed

# Calculate the confidence interval
CI_lower <- pi_hat - z * SE
CI_upper <- pi_hat + z * SE

# Display the confidence interval
CI <- c(CI_lower, CI_upper)
CI
```

```
[1] 0.5885724 0.7714276
```

The asymptotic 95% confidence interval for was between 0.5885724 and 0.7714276

1.13 (1 of 2 points)

Calculate an asymptotic p-value for the null hypothesis $H_0 \quad = 0.5$.

```r
# Null hypothesis value
pi_0 <- 0.5
pi_hat = 0.68

# Sample size from 100 simulation
n <- length(bern)

# Standard error under H0
SE_hat <- sqrt(pi_hat * (1 - pi_hat) / n)

# Test statistic
Z <- (pi_hat - pi_0) / SE_hat

# Calculate two-tailed p-value
p_value <- 2 * (1 - pnorm(abs(Z)))

# Output the p-value
p_value
```

```
[1] 0.0001139833
```

The correct answer, since the calculated p-value is under the null H0= 0.5, we should calculate the test statistic using the standard error under the null.

```r
# Null hypothesis value
pi_0 <- 0.5
pi_hat = 0.68

# Sample size from 100 simulation
n <- length(bern)

# Standard error under H0
SE_0 <- sqrt(pi_0 * (1 - pi_0) / n)

# Test statistic
Z <- (pi_hat - pi_0) / SE_0

# Calculate two-tailed p-value
```

```r
p_value <- 2 * (1 - pnorm(abs(Z)))

# Output the p-value
p_value
```

```
[1] 0.0003182172
```

The asymptotic p-value for the null hypothesis should be 0.0003182172 (not 0.0001139833)

1.14 (1 of 2 points)

Interpret both results in scientific terms.

- The 95% of the confidence intervals calculated would contain the true proportion , and they are expected to fall within the range of 0.589 to 0.771.Since the interval does not contain the null hypothesis value of 0.5, we have evidence to suggest that the true proportion is statistically significantly different from 0.5 at the 5% significance level.
- The p-value is approximately 0.0001139833, which is much less than the significance level of 0.05, indicating that if the null hypothesis H_0 = 0.5 were true, there would be a very low probability of observing a sample proportion as extreme as the one in this data or more extreme by random chance alone is 0.0001139833.

Correction: The interpretation is almost correct only missed the conclusion and the p-value used here is not the correct p-value.

1.15 (2 of 2 points)

Find the set of binomial outcomes (values of $\sum_{i=1}^{n} X_i$) for which you would reject the null hypothesis.

As n is large, the binomial distribution can be approximate to normality. Assuming an

$$\alpha = 0.05$$

, the rejection region is defined by the null distribution

$$\pi = 0.5$$

$$Bin(n, \pi) \sim \mathcal{N}(n\pi, n\pi(1-\pi))$$

where:

$$\sum_{i=1}^{n} x_i = n\pi$$

Under null hypothesis:

$$\sum_{i=1}^{n} x_i = 100 \cdot 0.5 = 50$$

$$\sigma^2 = 100 \cdot 0.5 \cdot (1 - 0.5) = 25$$

$$\sigma = \sqrt{\sigma^2} = \sqrt{25} = 5$$

```r
# Set the number of trials and probability under H0
n <- 100 # for example
pi_0 <- 0.5 # under H0

# Mean and standard deviation under the null hypothesis
mean <- n * pi_0
sd <- sqrt(n * pi_0 * (1 - pi_0))

# Find the z-scores for alpha = 0.05 in a two-tailed test
alpha <- 0.05
z <- qnorm(1 - alpha / 2) # This gives the critical z value for a two-tailed test

# Convert z-scores to the binomial scale
# Note: use 'floor' for lower bound and 'ceiling' for the upper bound to be conservative
lower_critical <- floor(mean - z * sd)
upper_critical <- ceiling(mean + z * sd)

# Define the rejection region for the binomial outcomes
# The rejection region includes values below the lower critical value or above the upper cri
rejection_region <- c(lower_critical, upper_critical)
rejection_region
```

```
[1] 40 60
```

We reject H_0 when: If $0 \leq \sum_{i=1}^{n} X_i \leq 40$, and if $60 \leq \sum_{i=1}^{n} X_i \leq 100$

1.16 (1 of 2 points)

Compute the probability of rejecting the null hypothesis (power), if the data-generating value of equals your estimate. In other words, assume that $= \hat{\pi}$ and use the binomial distribution of $\sum_{i=1}^{n} X_i$ to evaluate the power of the normal-approximation test of H_0 $= 0.5$.

Assume that $= \hat{\pi}$, when $\hat{\pi}$ =0.68, then

```r
compute_power_binomial <- function(pi_hat, sample_size, pi_null = 0.5) {
  # Calculate the probability of rejecting the null hypothesis (power)
  power <- pbinom(qbinom(0.025, size = sample_size, prob = pi_null), size = sample_size, prol

  return(power)
}

# Example usage
pi_hat <- 0.68  # Replace with your estimate
sample_size <- 100  # Replace with your sample size

power <- compute_power_binomial(pi_hat, sample_size, pi_null = 0.5)
power
```

```
[1] 0.944154
```

Power = P(Reject $H_0 = 0.5 | H_A = \hat{\pi}) = 0.944154$.

Getting half point since the range is only for $0 \leq \sum_{i=1}^{n} X_i < 40$, and if $60 < \sum_{i=1}^{n} X_i \leq 100$. the power calculated above did not adjust for specific rejection region, meaning that I used 2.5th and 97.5th percentiles which generally apply for a two-sided test but without tailoring to your specific cutoffs of 40 and 60.

Correction: This adjustment ensures the function calculates power based on the correct rejection regions, by subtracting upper critical value (60) to 1 due to a quirk in `pbinom()`.

```r
compute_power_binomial1 <- function(pi_hat, sample_size, pi_null = 0.5) {
  # Calculate the lower and upper critical values based on pi_null
  lower_critical_value <- qbinom(0.025, size = sample_size, prob = pi_null)
  upper_critical_value <- qbinom(0.975, size = sample_size, prob = pi_null)

  # Calculate the power as the sum of probabilities of falling in the lower and upper reject
  power_lower <- pbinom(lower_critical_value, size = sample_size, prob = pi_hat)
  power_upper <- 1 - pbinom(upper_critical_value - 1, size = sample_size, prob = pi_hat)
  # Total power is the sum of the probabilities of rejecting in either region
  power <- power_lower + power_upper

  return(power)
}

# Example usage
pi_hat <- 0.68  # Probability of success under the alternative hypothesis
sample_size <- 100  # Number of trials

power <- compute_power_binomial1(pi_hat, sample_size)
print(power)
```

```
[1] 0.9638655
```

The power should be 0.9638655, or the probability of rejecting the null hypothesis if the data-generating value of equals your estimate was 0.9638655.

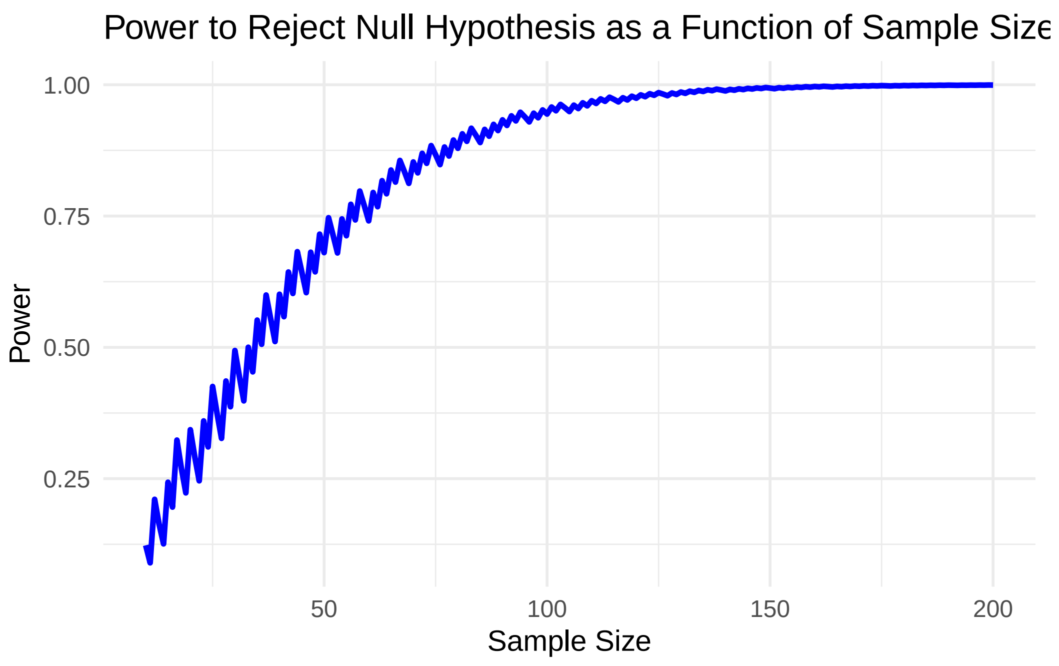
1.17 (1 of 2 points)

Graph the power to reject the null hypothesis as a function of sample size, using your sample estimate as the data-generating value. What sample size would you need to achieve 80% power? 90%? 95%? 99%?

```r
# Create a sequence of sample sizes
sample_sizes <- seq(10, 200, by = 1)

# Compute power for each sample size
powers <- sapply(sample_sizes, function(s) compute_power_binomial(pi_hat = 0.68, sample_size

# Create a dataframe for ggplot
data <- data.frame(SampleSize = sample_sizes, Power = powers)

# Plot using ggplot
plot <- ggplot(data, aes(x = SampleSize, y = Power)) +
  geom_line(color = "blue", size = 1) +  # Draw line
  ggtitle("Power to Reject Null Hypothesis as a Function of Sample Size") +
  xlab("Sample Size") +
  ylab("Power") +
  theme_minimal()  # Using a minimal theme for a clean look
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```

```
# Display the plot
print(plot)
```



```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
find_sample_size_for_power <- function(desired_power, pi_hat, pi_null = 0.5) {
  lower_bound <- 1
  upper_bound <- 1000
  while (lower_bound < upper_bound) {
    mid_point <- floor((lower_bound + upper_bound) / 2)
    if (compute_power_binomial(pi_hat, mid_point, pi_null) < desired_power) {
      lower_bound <- mid_point + 1
    } else {
      upper_bound <- mid_point
    }
  }
  return(upper_bound)
}

# Desired power levels
power_levels <- c(0.8, 0.9, 0.95, 0.99)

# The pi_hat value should be set according to your alternative hypothesis
pi_hat <- 0.68

# Calculate sample sizes for each power level
sample_sizes <- sapply(power_levels, function(p) find_sample_size_for_power(p, pi_hat))

# Creating a data frame to view the results
results_df <- data.frame(Power = power_levels, SampleSize = sample_sizes) %>%
  mutate(Power = scales::percent(Power))

print(results_df)
```

```
  Power SampleSize
1   80%         63
2   90%         81
3   95%        101
4   99%        141
```

Getting half point for not getting the correct sample size for the power provided.

```r
# Standard Error function for binomial distribution
std_err <- function(n, pi) {
  return(sqrt(pi * (1 - pi) / n))
}
```

```r
# Function to calculate power of a binomial test
power <- function(n = 100, null = 0.5, alt = 0.68) {
  # Ensure n is a whole number
  n <- floor(n)

  # Calculate standard error under the null hypothesis
  se_null <- std_err(n = n, pi = null)

  # Calculate rejection regions for a two-tailed test at alpha = 0.05
  reject_upper <- ceiling(n * (null + qnorm(0.975) * se_null))
  reject_lower <- floor(n * (null - qnorm(0.975) * se_null))

  # Probability of rejecting the null hypothesis when the alternative is true
  p_reject_high <- pbinom(q = reject_lower, size = n, prob = alt)
  p_reject_low <- pbinom(q = reject_upper - 1, size = n, prob = alt, lower.tail = FALSE)

  # Total power is the sum of the probabilities of rejecting the null hypothesis
  p_reject <- p_reject_high + p_reject_low
  return(p_reject)
}

# Function to find the sample size required for specified power levels
find_sample_size_for_power <- function(desired_power, null = 0.5, alt = 0.68) {
  sample_size <- 10  # Starting sample size
  while(TRUE) {
    current_power <- power(n = sample_size, null = null, alt = alt)
    if (current_power >= desired_power) {
      return(sample_size)
    }
    sample_size <- sample_size + 1
  }
}

# Desired power levels and their names
desired_powers <- c(0.8, 0.9, 0.95, 0.99)
power_names <- c("80%", "90%", "95%", "99%")

# Calculate the sample sizes for these power levels
sample_sizes <- sapply(desired_powers, find_sample_size_for_power)

# Create a data frame for better presentation
sample_sizes_df <- data.frame(
  "Power Level" = power_names,
  "Required Sample Size" = sample_sizes,
  stringsAsFactors = FALSE
)

# Print the data frame
print(sample_sizes_df)
```

```
  Power.Level Required.Sample.Size
1         80%                   55
2         90%                   75
3         95%                   93
4         99%                  133
```

1.18 (2 of 2 points)

Repeat the simulation 1000 times. Each time, record:

$\hat{\pi}$ (the MLE of)

$\hat{SE}(\hat{\pi}$ (the estimated standard error of the MLE of)

The p-value of the hypothesis test 0 $= 0.5$ vs 0.5

whether you rejected the null hypothesis at the $= 0.05$ level

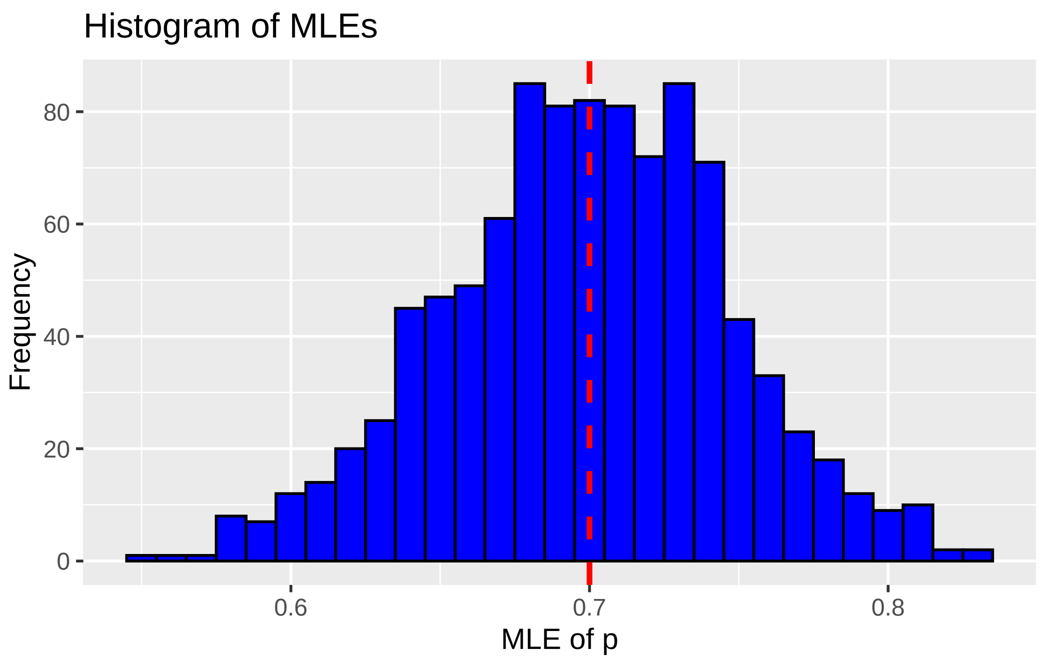
whether the confidence interval included the true value $= 0.7$

```r
pi_true = 0.7
n = 100
n_simulations = 1000
pi_MLE = numeric(n_simulations)
 standard_errors = numeric(n_simulations)
 p_values = numeric(n_simulations)
 reject_reg = numeric(n_simulations)
 conf_intervals= matrix(nrow = 1000, ncol = 2)
 conf_intervals_includes_true = numeric(n_simulations)

 # Perform simulations
for (i in 1:1000) {
  data <- rbinom(n, size = 1, prob = pi_true)
  set.seed(i)
  # Estimate
  pi_hat <- mean(data)
  pi_MLE [i] <- pi_hat
  # Calculate the standard error for the estimate of pi
  standard_errors[i] <- sqrt(pi_hat * (1 - pi_hat) / n)
  # Perform the hypothesis test and calculate p-value
  z_stat <- (pi_hat - 0.5) / standard_errors[i]
  p_values[i] <- 2 * (1 - pnorm(abs(z_stat)))
  # Determine if we reject the null hypothesis
  reject_reg[i] <- ifelse(p_values[i] < 0.05, 1, 0)
 # Calculate confidence interval
   z_critical <- qnorm(0.975)
    lower_bound <- pi_hat - z_critical * standard_errors[i]
    upper_bound <- pi_hat + z_critical * standard_errors[i]
    conf_intervals[i, ] <- c(lower_bound, upper_bound)

    # Check if the true value is within the confidence interval
    conf_intervals_includes_true[i] <- ifelse(pi_true >= lower_bound && pi_true <= upper_bou

#result
summary_df <- data.frame(
 Estimate = pi_MLE,
 StandardError = standard_errors,
  PValue = p_values,
  RejectedH0 = reject_reg,
  CIIncludesTrue = conf_intervals_includes_true
)
```

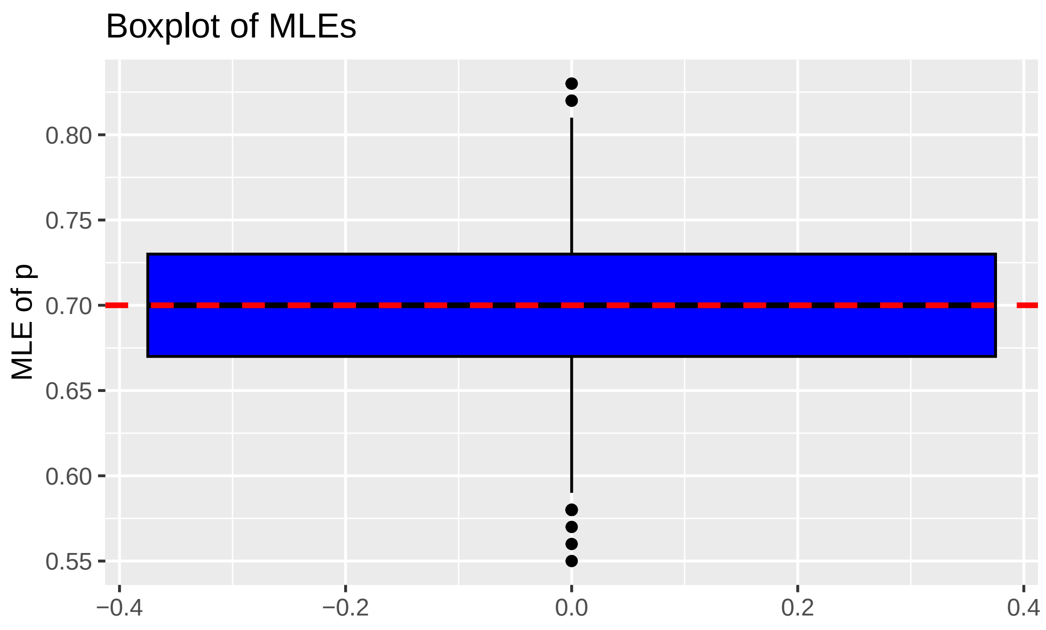
1.19 (2 of 2 points)

Create histograms and boxplots of the MLEs and estimated standard errors, with lines indicating the theoretical values.

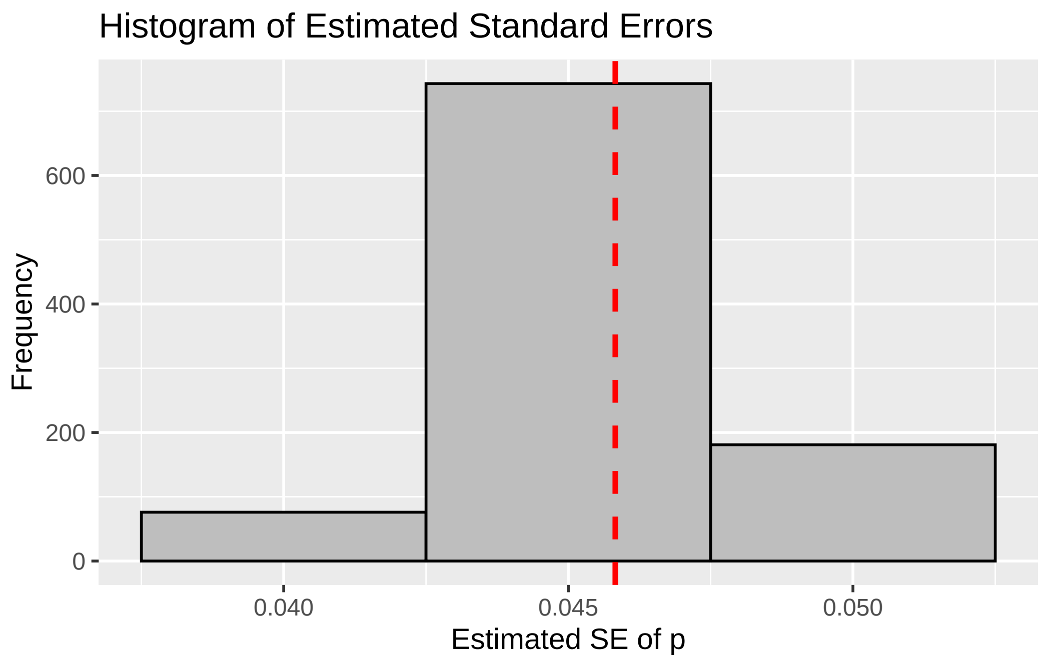
```
# Histogram of the MLEs with a line for the true value
hist_mle <- ggplot(data.frame(Estimate = pi_MLE), aes(x = Estimate)) +
  geom_histogram(binwidth = 0.01, fill = 'blue', color = 'black') +
  geom_vline(aes(xintercept = pi_true), color = "red", linetype = "dashed", size = 1) +
  labs(title = "Histogram of MLEs", x = "MLE of  ", y = "Frequency")
hist_mle
```



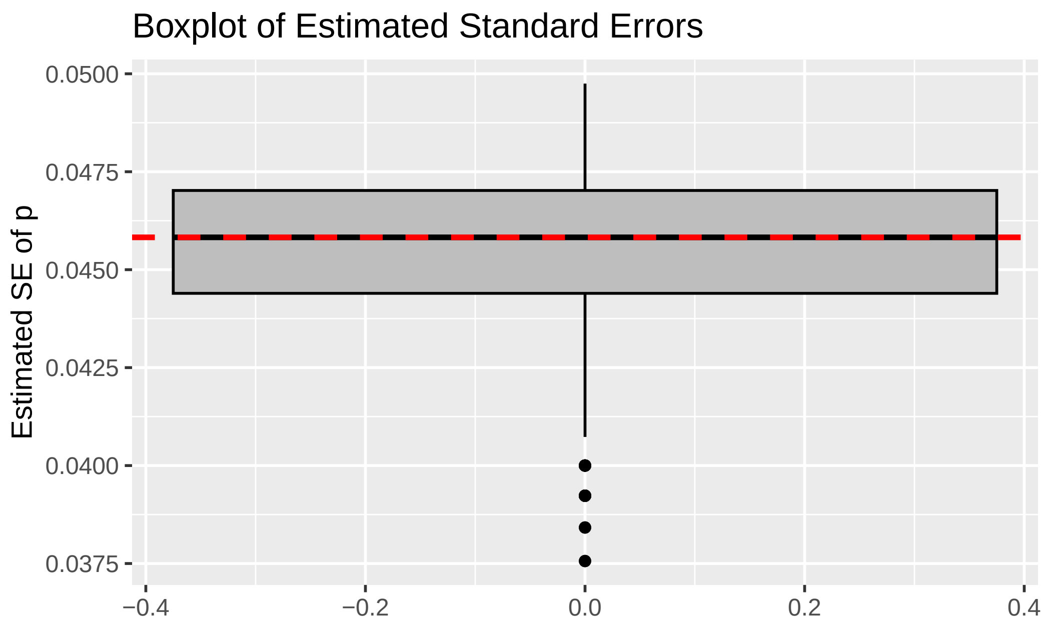
```r
# Boxplot of the MLEs with a line for the true value
boxplot_mle <- ggplot(data.frame(Estimate = pi_MLE), aes(y = Estimate)) +
  geom_boxplot(fill = 'blue', color = 'black') +
  geom_hline(aes(yintercept = pi_true), color = "red", linetype = "dashed", size = 1) +
  labs(title = "Boxplot of MLEs", y = "MLE of  ", x = "")
boxplot_mle
```



```
# Histogram of the estimated standard errors with a line for the theoretical standard error
# The theoretical standard error for a binomial distribution is sqrt(pi*(1-pi)/n)
pi_true = 0.7
theoretical_se <- sqrt(pi_true * (1 - pi_true) / n)
hist_se <- ggplot(data.frame(StandardErrors = standard_errors), aes(x = StandardErrors)) +
  geom_histogram(binwidth = 0.005, fill = 'gray', color = 'black') +
  geom_vline(aes(xintercept = theoretical_se), color = "red", linetype = "dashed", size = 1)
  labs(title = "Histogram of Estimated Standard Errors", x = "Estimated SE of  ", y = "Freque
hist_se
```



```r
# Boxplot of the estimated standard errors with a line for the theoretical standard error
boxplot_se <- ggplot(data.frame(StandardErrors = standard_errors), aes(y = StandardErrors)) +
  geom_boxplot(fill = 'gray', color = 'black') +
  geom_hline(aes(yintercept = theoretical_se), color = "red", linetype = "dashed", size = 1)
  labs(title = "Boxplot of Estimated Standard Errors", y = "Estimated SE of  ", x = "")
boxplot_se
```



1.20 (2 of 2 points)

Using your 1000 simulations, estimate:

```r
simulate_and_summarize <- function(sample_size, pi_true, n_simulations = 1000) {
  pi_MLE <- numeric(n_simulations)
  standard_errors <- numeric(n_simulations)
  p_values <- numeric(n_simulations)
  reject_reg <- numeric(n_simulations)
  conf_intervals <- matrix(nrow = n_simulations, ncol = 2)
  conf_intervals_includes_true <- numeric(n_simulations)

  for (i in 1:n_simulations) {
    data <- rbinom(sample_size, size = 1, prob = pi_true)
    set.seed(i)
    # Estimate
    pi_hat <- mean(data)
    pi_MLE[i] <- pi_hat

    # Calculate the standard error for the estimate of pi
    standard_errors[i] <- sqrt(pi_hat * (1 - pi_hat) / sample_size)
    # Perform the hypothesis test and calculate p-value
    z_stat <- (pi_hat - 0.5) / standard_errors[i]
    p_values[i] <- 2 * (1 - pnorm(abs(z_stat)))

    # Determine if we reject the null hypothesis
    reject_reg[i] <- as.numeric(p_values[i] < 0.05)

    # Calculate confidence interval
    z_critical <- qnorm(0.975)
    lower_bound <- pi_hat - z_critical * standard_errors[i]
    upper_bound <- pi_hat + z_critical * standard_errors[i]
    conf_intervals[i, ] <- c(lower_bound, upper_bound)

    # Check if the true value is within the confidence interval
    conf_intervals_includes_true[i] <- as.numeric(pi_true >= lower_bound && pi_true <= upper_
  }

  mean_MLE <- mean(pi_MLE)
  bias_mle <- mean_MLE - pi_true
  var_mle <- var(pi_MLE)
  se_mle <- sqrt(var_mle)
  mean_est_se <- mean(standard_errors)
  bias_est_se <- mean_est_se - sqrt(pi_true * (1 - pi_true) / sample_size)
  var_est_se <- var(standard_errors)
  se_est_se <- sqrt(var_est_se)
  coverage_prob <- mean((conf_intervals[,1] < pi_true) & (conf_intervals[,2] > pi_true))
  power <- mean(reject_reg)

  results <- list (
    MeanMLE = mean_MLE,
    BiasMLE = bias_mle,
    VarMLE = var_mle,
    SEMLE = se_mle,
    MeanEstSE = mean_est_se,
    BiasEstSE = bias_est_se,
    VarEstSE = var_est_se,
    SEEstSE = se_est_se,
    CoverageProb = coverage_prob,
    Power = power
  )

  return(results)
```

```
}
```

```
results100 <- simulate_and_summarize(sample_size = 100, pi_true = 0.7)
results100
```

```
$MeanMLE
[1] 0.69911

$BiasMLE
[1] -0.00089

$VarMLE
[1] 0.002251559

$SEMLE
[1] 0.0474506

$MeanEstSE
[1] 0.04556994

$BiasEstSE
[1] -0.0002558186

$VarEstSE
[1] 4.444164e-06

$SEEstSE
[1] 0.002108119

$CoverageProb
[1] 0.935

$Power
[1] 0.982
```

```
pi=0.7
n=100 #number of trials

# Theoretical mean
```

```r
# Theoretical variance of the mean estimator for sample size 100
variance_theoretical <- n * pi * (1 - pi)/100

# Theoretical standard error
se_theoretical <- sqrt(variance_theoretical / n)

# Print the theoretical values

cat("Theoretical variance:", variance_theoretical, "\n")
```

```
Theoretical variance: 0.21
```

```r
cat("Theoretical standard error:", se_theoretical, "\n")
```

```
Theoretical standard error: 0.04582576
```

1.21 (1 of 2 points)

Summarize the performance of your analyses, comparing empirical and theoretical results.

the estimated mean is very close to the true with minimal bias (-0.00089), theoretically, there should be no bias for the MLE of a binomial proportion, so this value should be close to zero. Theoretically, we use variance of the mean estimator to compare with the empirical variance of the MLE, here the theoretical variance(0.21) is larger than the empirical value (0.00221559). The empirical SE (0.0474506) which is larger than the theoretical SE (0.04582576) for the binomial distribution.

Correction: missing the conclusion about how these to model comparing empirical and theoretical results, as both values are very close which suggest asymptotic consistency of the MLE. Also, both SE prety close.

1.22 (1 of 2 points)

Repeat the simulation with a simulated sample sizes of 10^3 and 10^5 binary outcomes, and summarize the results. Compare the empirical results with the theoretical results above.

```r
# Run simulations for the specified sample sizes
results_1k <- simulate_and_summarize(sample_size = 1000, pi_true = 0.7)
results_10k <- simulate_and_summarize(sample_size = 10000, pi_true = 0.7)
```

```r
# Create a data frame to summarize the results for both sample sizes
simulation_results0 <- data.frame(
  Sample_Size = c(1000, 10000),
  MeanMLE = c(results_1k$MeanMLE, results_10k$MeanMLE),
  BiasMLE = c(results_1k$BiasMLE, results_10k$BiasMLE),
  VarMLE = c(results_1k$VarMLE, results_10k$VarMLE),
  SEMLE = c(results_1k$SEMLE, results_10k$SEMLE),
  MeanEstSE = c(results_1k$MeanEstSE, results_10k$MeanEstSE),
  BiasEstSE = c(results_1k$BiasEstSE, results_10k$BiasEstSE),
  VarEstSE = c(results_1k$VarEstSE, results_10k$VarEstSE),
  SEEstSE = c(results_1k$SEEstSE, results_10k$SEEstSE),
  CoverageProb = c(results_1k$CoverageProb, results_10k$CoverageProb),
  Power = c(results_1k$Power, results_10k$Power)
)

# View the results
print(simulation_results0)
```

```
  Sample_Size   MeanMLE   BiasMLE       VarMLE       SEMLE  MeanEstSE
1        1000 0.6997080 -2.92e-04 2.182250e-04 0.014772439 0.01448644
2       10000 0.7000281  2.81e-05 2.028877e-05 0.004504306 0.00458219
      BiasEstSE     VarEstSE      SEEstSE CoverageProb Power
1 -4.939105e-06 4.187434e-08 2.046322e-04        0.941     1
2 -3.858777e-07 3.859373e-10 1.964529e-05        0.952     1
```

for sample size = 1000 The estimated mean is very close to the true with minimal bias (-0.000292), theoretically, there should be no bias for the MLE of a binomial proportion, so this value should be close to zero. Theoretically, we use variance of the mean estimator to compare with the empirical variance of the MLE, here the theoretical variance(0.21) is larger than the empirical value (0.000218225). The empirical SE (0.01477244) which is smaller than the theoretical SE (0.04582576) for the binomial distribution.

for sample size = 10000 The estimated mean is slightly larger and very close to the true with minimal bias (2.81e-05), theoretically, there should be no bias for the MLE of a binomial proportion, so this value should be close to zero. Theoretically, we use variance of the mean estimator to compare with the empirical variance of the MLE, here the theoretical variance(0.21) is larger than the empirical value (2.028877e-05) which is ver small. The empirical SE (0.004504306) which is smaller than the theoretical SE (0.04582576) for the binomial distribution.

Correction: there is no comparison to the theoretical result and the interpretation was not clearly stated the conclusion. The take away from this is that as the simulated sample size increased (100- 1K- 10K), the estimates MLE closer to the true value of 0.7, and the error getting smaller closer to 0. The power also increased as sample size increased. Below is the theoretical result

```
library(dplyr)

# Function to compute theoretical values for SE, CI Coverage Probability, and Power
compute_theoretical_metrics <- function(n, pi, pi_null = 0.5, alpha = 0.05, pi_alt = 0.7) {
  # Standard Error
  se <- sqrt(pi * (1 - pi) / n)

  # Z-value for the desired confidence level (95% CI)
  z <- qnorm(1 - alpha/2)

  # Confidence Interval
  ci_lower <- pi - z * se
  ci_upper <- pi + z * se

  # Assuming normal distribution for CI coverage check (95% theoretical coverage)
  ci_coverage_prob <- ifelse(ci_lower <= pi & ci_upper >= pi, 0.95, NA)  # Simplified for the

  # Power calculation
  # Calculate the test statistic under H0: pi = pi_null and true pi = pi_alt
  se_null <- sqrt(pi_null * (1 - pi_null) / n)
  z_power <- (pi_alt - pi_null) / se_null
  beta <- pnorm(z - z_power) + pnorm(-z - z_power)  # Type II error rate
  power <- 1 - beta  # Power = 1 - beta

  # Return results as a tibble row
  tibble(n = n, se = se, ci_coverage_prob = ci_coverage_prob, power = power)
}

# Sample sizes to evaluate
sample_sizes <- c(1000, 10000)

# Actual pi value, null hypothesis pi, and alternative pi for power calculation
pi <- 0.7
pi_null <- 0.7
pi_alt <- 0.75

# Calculate metrics for each sample size
results <- purrr::map_df(sample_sizes, ~ compute_theoretical_metrics(.x, pi, pi_null, alpha =
# Print results
print(results)
```

```
# A tibble: 2 x 4
      n      se ci_coverage_prob power
  <dbl>   <dbl>            <dbl> <dbl>
1  1000 0.0145              0.95 0.932
2 10000 0.00458             0.95 1
```

1.23 (0.5 of 2 points)

Repeat the simulation at all three sample sizes for the scenario where the data-generating parameter = 0.5, thus matching the null hypothesis. Empirically assess the false positive rate of the hypothesis test.

```
# Run the simulations for each sample size
results_100 <- simulate_and_summarize(sample_size = 100, pi_true = 0.5)
results_1000 <- simulate_and_summarize(sample_size = 1000, pi_true = 0.5)
results_10000 <- simulate_and_summarize(sample_size = 10000, pi_true = 0.5)

# Create a data frame to summarize the results
simulation_results <- data.frame(
  Sample_Size = c(100, 1000, 10000),
  False_Positive_Rate = c(results_100$Power, results_1000$Power, results_10000$Power),
  Mean_MLE = c(results_100$MeanMLE, results_1000$MeanMLE, results_10000$MeanMLE),
  Bias_MLE = c(results_100$BiasMLE, results_1000$BiasMLE, results_10000$BiasMLE),
  Variance_MLE = c(results_100$VarMLE, results_1000$VarMLE, results_10000$VarMLE),
  SE_MLE = c(results_100$SEMLE, results_1000$SEMLE, results_10000$SEMLE),
  Mean_Est_SE = c(results_100$MeanEstSE, results_1000$MeanEstSE, results_10000$MeanEstSE),
  Bias_Est_SE = c(results_100$BiasEstSE, results_1000$BiasEstSE, results_10000$BiasEstSE),
  Variance_Est_SE = c(results_100$VarEstSE, results_1000$VarEstSE, results_10000$VarEstSE),
  SE_Est_SE = c(results_100$SEEstSE, results_1000$SEEstSE, results_10000$SEEstSE),
  Coverage_Prob = c(results_100$CoverageProb, results_1000$CoverageProb, results_10000$Covera
)

# View the results
print(simulation_results)
```

```
  Sample_Size False_Positive_Rate  Mean_MLE    Bias_MLE Variance_MLE      SE_MLE
1         100               0.061 0.4988600 -0.0011400 2.629130e-03 0.051275038
```

```
2        1000              0.064 0.5007020  0.0007020 2.626138e-04 0.016205364
3       10000              0.049 0.5001089  0.0001089 2.411092e-05 0.004910287
  Mean_Est_SE   Bias_Est_SE Variance_Est_SE    SE_Est_SE Coverage_Prob
1 0.049735156 -2.648438e-04    1.363786e-07 3.692948e-04         0.939
2 0.015803070 -8.318114e-06    1.288066e-10 1.134930e-05         0.936
3 0.004999759 -2.410046e-07    1.207272e-13 3.474581e-07         0.951
```

The false positive rate for sample size of 100, 1000, and 1000 are 0.061, 0.064 and 0.049.

Correction: half point because it clearly showed the false positive rate around 5% but did not explain and interpret the result also did not include the theoretical results. Here is the same as previous result that as the sample size increases the empirical MLE gets closer to the theoretical values, while standard error approches 0.And the truw value parameters fall in around 95% of the interval, and the false positive rate approximately 5%.