Midterm Exam: POLI502

Benjamin Cook

2024-10-11

Quarto

```
# Load the necessary packages
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becon
```

```
install.packages("tidyverse")
```

```
Warning: package 'tidyverse' is in use and will not be installed
```

```
# Load the world dataset from local storage
 world.data <- read_csv("C:/Users/benco/OneDrive/Documents/data/world.csv")
```

```
Rows: 191 Columns: 62
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (24): country, colony, dem_other5, democ_regime, district_size3, enpp_3,...
```

```
dbl (37): confidence, decentralization, dem_other, durable, effectiveness, f...
lgl  (1): sources

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Let's check the column names in the dataset
colnames(world.data)
```

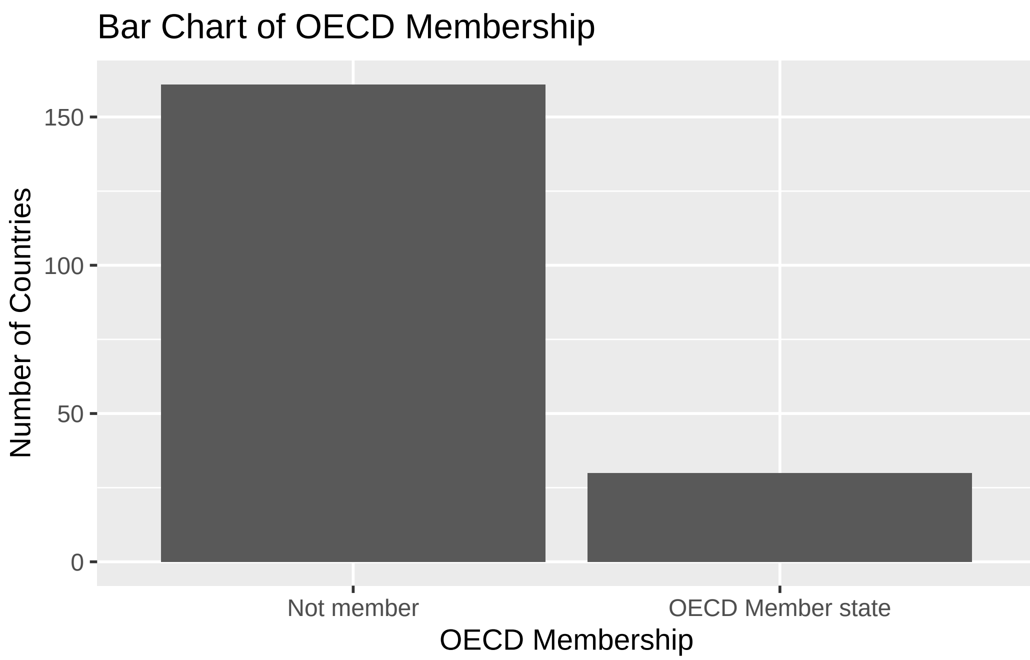
```
 [1] "country"          "colony"           "confidence"       "decentralization"
 [5] "dem_other"        "dem_other5"       "democ_regime"     "district_size3"
 [9] "durable"          "effectiveness"    "enpp_3"           "eu"
[13] "fhrate04_rev"     "fhrate08_rev"     "frac_eth"         "frac_eth3"
[17] "free_business"    "free_corrupt"     "free_finance"     "free_fiscal"
[21] "free_govspend"    "free_invest"      "free_labor"       "free_monetary"
[25] "free_overall"     "free_property"    "free_trade"       "gdp08"
[29] "gdp_10_thou"      "gdp_cap2"         "gdp_cap3"         "gdppcap08"
[33] "gender_equal3"    "gini04"           "gini08"           "hi_gdp"
[37] "indy"             "oecd"             "old2006"          "old2003"
[41] "pmat12_3"         "pop03"            "pop08"            "pop08_3"
[45] "popcat3"          "pr_sys"           "protact3"         "regime_type3"
[49] "region"           "sources"          "typerel"          "unions"
[53] "urban03"          "urban06"          "vi_rel3"          "votevap00s"
[57] "women05"          "women09"          "womyear"          "womyear2"
[61] "yng2003"          "young06"
```

```r
# Create a frequency table
ft.oecd <- world.data %>%
  count(oecd) %>%
  mutate(Percentage = (n / sum(n)) * 100) %>%
  rename(`OECD Member?` = oecd, Freq = n)

ft.oecd
```

```
# A tibble: 2 x 3
  `OECD Member?`      Freq Percentage
  <chr>              <int>      <dbl>
1 Not member           161       84.3
2 OECD Member state     30       15.7
```

```
# Task 3: Answers

# (A) Number of OECD member countries: 30
# (B) Number of non-OECD member countries: 161
# (C) Percentage of OECD member countries: 84.29
# (D) Percentage of non-OECD member countries: 30
```

```
# Create the bar chart for OECD membership
ggplot(ft.oecd, aes(x = `OECD Member?`, y = Freq)) +
  geom_bar(stat = "identity") +
  xlab("OECD Membership") +
  ylab("Number of Countries") +
  ggtitle("Bar Chart of OECD Membership")
```




```
# Task 5: List three OECD member countries and three non-democratic countries

# Filter for OECD member countries based on the actual coding
oecd_members <- world.data %>%
  filter(oecd == "OECD Member state") %>%
  select(country) %>%
  head(3)
```

```r
# Display three OECD member countries
oecd_members
```

```
# A tibble: 3 x 1
  country
  <chr>
1 Australia
2 Austria
3 Belgium
```

```r
# Check unique values for the democ_regime variable as well
unique(world.data$democ_regime)
```

```
[1] "No"  "Yes" NA
```

```r
# Filter for non-democratic countries based on the actual coding
non_democratic <- world.data %>%
  filter(democ_regime == "No") %>%
  select(country) %>%
  head(3)
```

```r
# Display three non-democratic countries
non_democratic
```

```
# A tibble: 3 x 1
  country
  <chr>
1 Afghanistan
2 Algeria
3 Angola
```

```r
# Task 6: Describe the GDP variable numerically

# Ensure the column name is correct by checking the dataset
colnames(world.data)
```

```
 [1] "country"           "colony"            "confidence"        "decentralization"
 [5] "dem_other"         "dem_other5"        "democ_regime"      "district_size3"
 [9] "durable"           "effectiveness"     "enpp_3"            "eu"
```

```
[13] "fhrate04_rev"     "fhrate08_rev"     "frac_eth"         "frac_eth3"
[17] "free_business"    "free_corrupt"     "free_finance"     "free_fiscal"
[21] "free_govspend"    "free_invest"      "free_labor"       "free_monetary"
[25] "free_overall"     "free_property"    "free_trade"       "gdp08"
[29] "gdp_10_thou"      "gdp_cap2"         "gdp_cap3"         "gdppcap08"
[33] "gender_equal3"    "gini04"           "gini08"           "hi_gdp"
[37] "indy"             "oecd"             "old2006"          "old2003"
[41] "pmat12_3"         "pop03"            "pop08"            "pop08_3"
[45] "popcat3"          "pr_sys"           "protact3"         "regime_type3"
[49] "region"           "sources"          "typerel"          "unions"
[53] "urban03"          "urban06"          "vi_rel3"          "votevap00s"
[57] "women05"          "women09"          "womyear"          "womyear2"
[61] "yng2003"          "young06"
```

```r
# Calculate the required statistics for gdp_10_thou
gdp_stats <- world.data %>%
  summarise(
    min_gdp = min(gdp_10_thou, na.rm = TRUE),
    max_gdp = max(gdp_10_thou, na.rm = TRUE),
    median_gdp = median(gdp_10_thou, na.rm = TRUE),
    mean_gdp = mean(gdp_10_thou, na.rm = TRUE),
    first_quartile = quantile(gdp_10_thou, 0.25, na.rm = TRUE),
    third_quartile = quantile(gdp_10_thou, 0.75, na.rm = TRUE),
    std_dev_gdp = sd(gdp_10_thou, na.rm = TRUE)
  )
```

```r
# Display the calculated statistics
gdp_stats
```

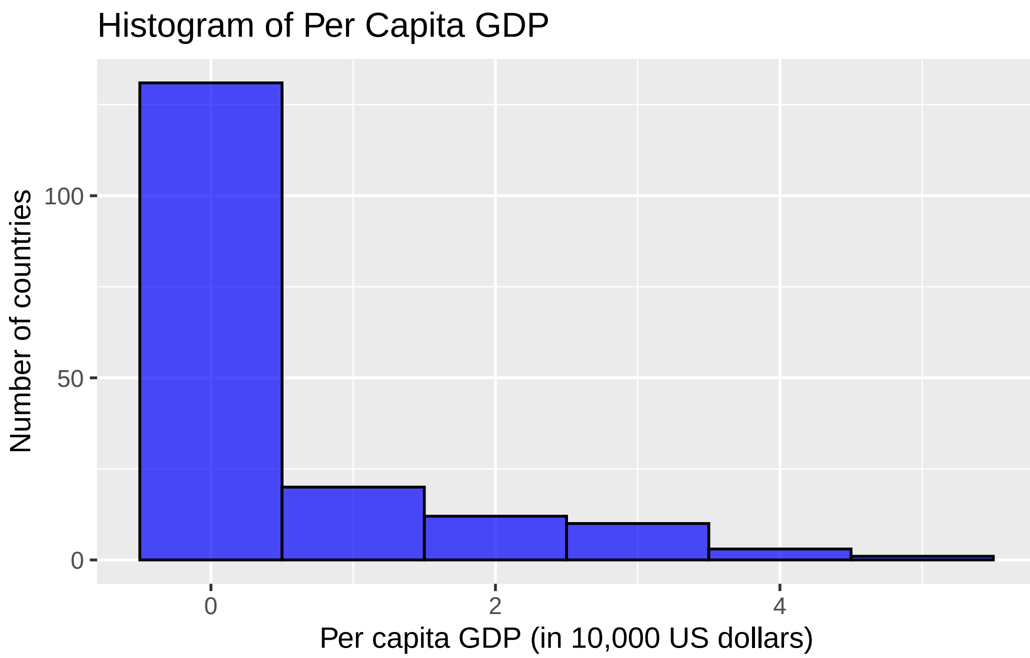
```
# A tibble: 1 x 7
  min_gdp max_gdp median_gdp mean_gdp first_quartile third_quartile std_dev_gdp
    <dbl>   <dbl>      <dbl>    <dbl>          <dbl>          <dbl>       <dbl>
1   0.009    4.74      0.190    0.602         0.0503          0.632       0.943
```

```r
# Task 7: Answer
# The distribution of the GDP per capita variable is positively skewed (skewed to the right)
```

```r
# Task 8: Draw a histogram for the per capita GDP variable

# Draw the histogram for the gdp_10_thou variable
ggplot(world.data, aes(x = gdp_10_thou)) +
```

```
  geom_histogram(binwidth = 1, fill = "blue", color = "black", alpha = 0.7) +
  xlab("Per capita GDP (in 10,000 US dollars)") +
  ylab("Number of countries") +
  ggtitle("Histogram of Per Capita GDP")
```

```
Warning: Removed 14 rows containing non-finite outside the scale range
(`stat_bin()`).
```




```
# Task 9: Identify two countries with per capita GDP greater than $40,000

# Filter countries with per capita GDP greater than 4 (which represents $40,000)
high_gdp_countries <- world.data %>%
  filter(gdp_10_thou > 4) %>%
  select(country, gdp_10_thou)
```

```
# Display the results
high_gdp_countries
```

```
# A tibble: 2 x 2
  country     gdp_10_thou
  <chr>             <dbl>
```

```
1 Luxembourg        4.74
2 Norway            4.20
```

```r
# Task 10: Calculate the standard error of the mean for gdp_10_thou

# Calculate the standard deviation (already calculated, but doing it again here)
std_dev_gdp <- sd(world.data$gdp_10_thou, na.rm = TRUE)
```

```r
# Number of observations (excluding missing values)
n <- 177
```

```r
# Calculate the standard error of the mean
standard_error <- std_dev_gdp / sqrt(n)
```

```r
# Display the standard error
standard_error
```

```
[1] 0.07091015
```

```r
# Task 11: Construct the 95% confidence interval for the mean of gdp_10_thou

# Calculate the mean of gdp_10_thou (if not already calculated)
mean_gdp <- mean(world.data$gdp_10_thou, na.rm = TRUE)
```

```r
# Use the standard error calculated from task 10
# Standard error calculated in Task 10
standard_error <- std_dev_gdp / sqrt(n)
```

```r
# Set the z-score for a 95% confidence interval
z_score <- 1.96
```

```r
# Calculate the lower and upper bounds of the confidence interval
lower_bound <- mean_gdp - z_score * standard_error
upper_bound <- mean_gdp + z_score * standard_error
```

```r
# Display the confidence interval
lower_bound
```

```
[1] 0.4628347
```

```
upper_bound
```

```
[1] 0.7408025
```

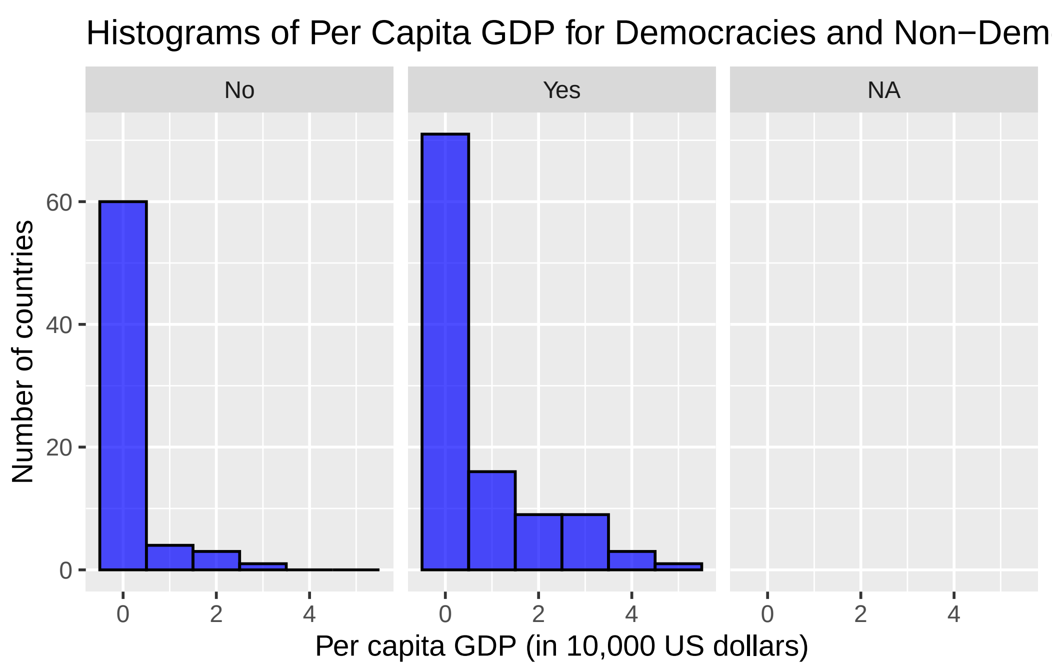
```
lower_bound <- mean_gdp - z_score * standard_error
```

```
lower_bound
```

```
[1] 0.4628347
```

```
# Task 12: Draw histograms for democracies and non-democracies

# Create the histogram for GDP, separating by democracy and non-democracy
ggplot(world.data, aes(x = gdp_10_thou)) +
  geom_histogram(binwidth = 1, fill = "blue", color = "black", alpha = 0.7) +
  facet_wrap(~ democ_regime) +
  xlab("Per capita GDP (in 10,000 US dollars)") +
  ylab("Number of countries") +
  ggtitle("Histograms of Per Capita GDP for Democracies and Non-Democracies")
```

```
Warning: Removed 14 rows containing non-finite outside the scale range
(`stat_bin()`).
```

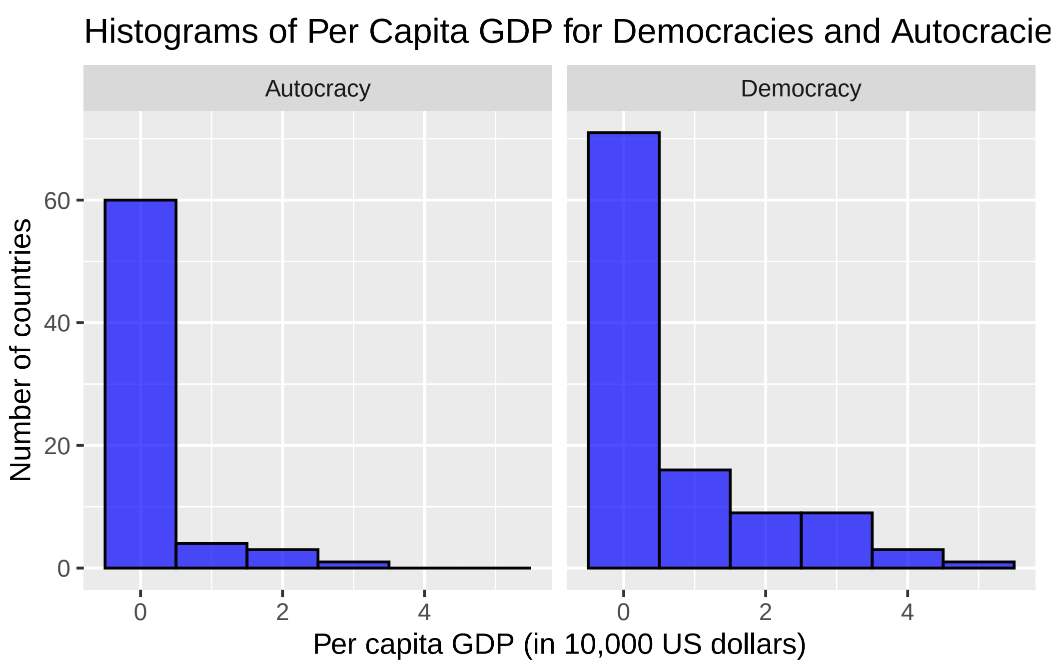


```r
# Task 13: Remove missing values and replace "Yes" and "No" with "Democracy" and "Autocracy"

# Create a new data frame excluding rows with NA in democ_regime
dem_gdp <- world.data %>%
  filter(!is.na(democ_regime))
```

```r
# Create a new variable with more intuitive labels
dem_gdp <- dem_gdp %>%
  mutate(dem_dum = ifelse(democ_regime == "Yes", "Democracy", "Autocracy"))
```

```r
# Draw the histogram for GDP, separating by democracy and autocracy
ggplot(dem_gdp, aes(x = gdp_10_thou)) +
  geom_histogram(binwidth = 1, fill = "blue", color = "black", alpha = 0.7) +
  facet_wrap(~ dem_dum) +
  xlab("Per capita GDP (in 10,000 US dollars)") +
  ylab("Number of countries") +
  ggtitle("Histograms of Per Capita GDP for Democracies and Autocracies")
```

```
Warning: Removed 12 rows containing non-finite outside the scale range
(`stat_bin()`).
```



```
# Task 14: Calculate mean and 95% confidence interval for democracies

# Filter the data for democracies
democracy_gdp <- dem_gdp %>%
  filter(dem_dum == "Democracy")
```

```
# Calculate the mean for democracies
mean_democracy_gdp <- mean(democracy_gdp$gdp_10_thou, na.rm = TRUE)
```

```
# Calculate the standard deviation for democracies
std_dev_democracy <- sd(democracy_gdp$gdp_10_thou, na.rm = TRUE)
```

```
# Number of democratic countries
n_democracy <- nrow(democracy_gdp)
```

```
# Calculate the standard error for democracies
standard_error_democracy <- std_dev_democracy / sqrt(n_democracy)
```

```
# Set the z-score for a 95% confidence interval
z_score <- 1.96
```

```
# Calculate the lower and upper bounds of the 95% confidence interval
lower_bound_democracy <- mean_democracy_gdp - z_score * standard_error_democracy
```

```
upper_bound_democracy <- mean_democracy_gdp + z_score * standard_error_democracy
```

```
# Display the mean and 95% confidence interval
mean_democracy_gdp
```

```
[1] 0.8013927
```

```
c(lower_bound_democracy, upper_bound_democracy)
```

```
[1] 0.6029297 0.9998557
```

```
# Task 15: Calculate mean and 95% confidence interval for autocracies

# Filter the data for autocracies
autocracy_gdp <- dem_gdp %>%
  filter(dem_dum == "Autocracy")
```

```r
# Calculate the mean for autocracies
mean_autocracy_gdp <- mean(autocracy_gdp$gdp_10_thou, na.rm = TRUE)
```

```r
# Calculate the standard deviation for autocracies
std_dev_autocracy <- sd(autocracy_gdp$gdp_10_thou, na.rm = TRUE)
```

```r
# Number of autocratic countries
n_autocracy <- nrow(autocracy_gdp)
```

```r
# Calculate the standard error for autocracies
standard_error_autocracy <- std_dev_autocracy / sqrt(n_autocracy)
```

```r
# Set the z-score for a 95% confidence interval
z_score <- 1.96
```

```r
# Calculate the lower and upper bounds of the 95% confidence interval
lower_bound_autocracy <- mean_autocracy_gdp - z_score * standard_error_autocracy
```

```r
upper_bound_autocracy <- mean_autocracy_gdp + z_score * standard_error_autocracy
```

```r
mean_autocracy_gdp
```

```
[1] 0.2819132
```

```r
c(lower_bound_autocracy, upper_bound_autocracy)
```

```
[1] 0.1610568 0.4027697
```

```r
# Task 16: Calculate probability of rain given dark clouds using Bayes' Theorem

# Given probabilities
P_R <- 0.30          # Probability of rain, P(R)
```

```r
P_not_R <- 1 - P_R   # Probability of no rain, P(¬R)
```

```r
P_C_given_R <- 0.95  # Probability of clouds given rain, P(C|R)
```

```r
P_C_given_not_R <- 0.25  # Probability of clouds given no rain, P(C|¬R)
```

```r
# Calculate P(C) using the total probability formula
P_C <- (P_C_given_R * P_R) + (P_C_given_not_R * P_not_R)
```

```r
# Apply Bayes' Theorem to calculate P(R|C)
P_R_given_C <- (P_C_given_R * P_R) / P_C
```

```r
# Display the probability of rain given dark clouds
P_R_given_C
```

```
[1] 0.6195652
```

```r
# Task 17: Bayesian inference with a Beta distribution

# Given parameters for the prior Beta(1.5, 1.5)
a <- 1.5
b <- 1.5
```

```r
# Number of students who answered "Yes" and "No"
n_yes <- 37
n_no <- 13
```

```r
# (a) Calculate the prior mean and prior standard deviation
prior_mean <- a / (a + b)
prior_var <- (a * b) / ((a + b)^2 * (a + b + 1))
prior_sd <- sqrt(prior_var)
```

```r
# Display prior mean and prior standard deviation
prior_mean
```

```
[1] 0.5
```

```r
prior_sd
```

```
[1] 0.25
```

```r
# (b) Find the prior probability that theta < 0.6
prior_prob <- pbeta(0.6, a, b)
```

```r
# Display the prior probability P(theta < 0.6)
prior_prob
```

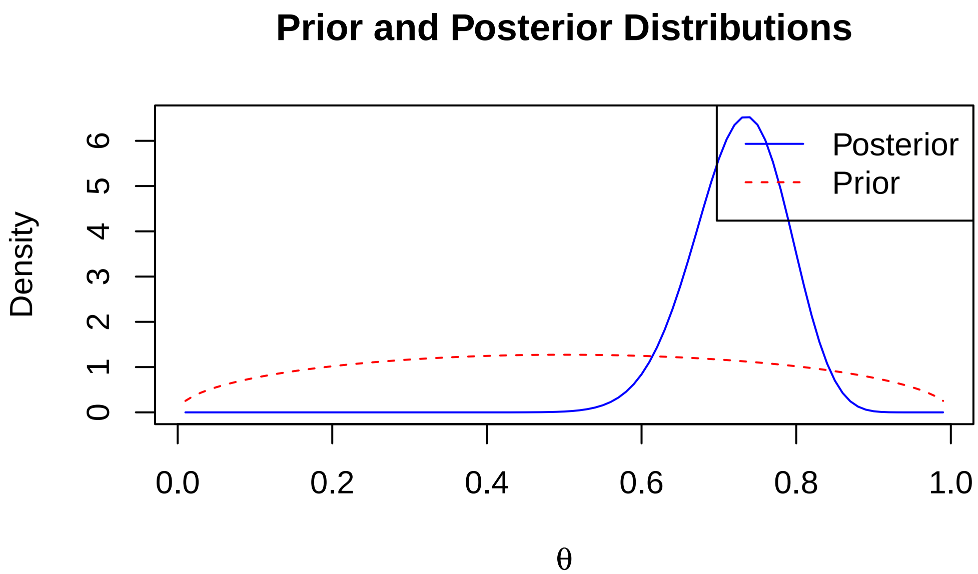
```
[1] 0.62647
```

```r
# (c) Likelihood function for the Beta distribution
# Likelihood is implicit in updating the posterior as Beta(a + n_yes, b + n_no)

# (d) Posterior distribution after observing the data
# Posterior will be Beta(a + n_yes, b + n_no)
posterior_a <- a + n_yes
```

```r
posterior_b <- b + n_no
```

```r
# Plotting the prior and posterior distributions
theta <- seq(0.01, 0.99, 0.01)
prior_density <- dbeta(theta, a, b)
posterior_density <- dbeta(theta, posterior_a, posterior_b)
```

```r
# Plot the prior and posterior distributions
plot(theta, posterior_density, type = "l", col = "blue",
     xlab = expression(theta), ylab = "Density",
     main = "Prior and Posterior Distributions")
lines(theta, prior_density, col = "red", lty = 2)

legend("topright", legend = c("Posterior", "Prior"), col = c("blue", "red"), lty = c(1, 2))
```


Prior and Posterior Distributions
Density
0
1
2
3
4
5
6
0.0
0.2
0.4
0.6
0.8
1.0
θ
Posterior
Prior