STAT3191/6191: Group Project Submission

Group Number: Group 17

(Group Leader) Vathana Khun (48302031)
Sadiqun Nur Ayan (48920959) Raja Pedapudi (46562753)

2025-09-26

Section 1: Estimating Typical Completion Time for Security Training

Section 1, Part A: Natural Variation in Completion Times

```r
# Load required library
library(matrixStats)
library(kableExtra)
```

```
Warning: package 'kableExtra' was built under R version 4.4.3
```

```r
# Simulation-Based Inference: Estimating Completion Time for Security Training
set.seed(42)

# Parameters
n <- 250                # Sample size per simulation
n_sim <- 1000           # Number of simulations
lambda <- 1             # Rate parameter for Exponential(1)

## Section 1, Part A : Natural Variation in Completion Times ##

# Step 1: Generate 1000 samples of size 250 from Exponential(1)
samples <- replicate(n_sim, rexp(n, rate = lambda))

# Step 2: Compute estimators for each sample
sample_means <- colMeans(samples)
sample_medians <- apply(samples, 2, median)
```

```
sample_trimmed_means <- apply(samples, 2, function(x) mean(x, trim = 0.1))

# Step 3: Calculate Bias, Monte Carlo Standard Deviation (MCSD), and Mean Squared Error (MSE
true_mean <- 1  # Theoretical mean of Exponential(1)

# Bias = mean of estimates - true mean
bias_mean <- mean(sample_means) - true_mean
bias_median <- mean(sample_medians) - true_mean
bias_trimmed <- mean(sample_trimmed_means) - true_mean

# Monte Carlo Standard Deviation = standard deviation of estimates
mcsd_mean <- sd(sample_means)
mcsd_median <- sd(sample_medians)
mcsd_trimmed <- sd(sample_trimmed_means)

# Mean Squared Error = mean squared difference from true mean
mse_mean <- mean((sample_means - true_mean)^2)
mse_median <- mean((sample_medians - true_mean)^2)
mse_trimmed <- mean((sample_trimmed_means - true_mean)^2)

# Step 4: Summarise results in a table
results_partA <- data.frame(
  Estimator = c("Mean", "Median", "Trimmed Mean (10%)"),
  Esimated = signif(c(mean(sample_means),mean(sample_medians), mean(sample_trimmed_means)),3
  Bias = signif(c(bias_mean, bias_median, bias_trimmed),3),
  MCSD = signif(c(mcsd_mean, mcsd_median, mcsd_trimmed),3),
  MSE = signif(c(mse_mean, mse_median, mse_trimmed),3)
)
results_partA%>%
  kable(caption = "Section 1 Part A: Monte Carlo summary (clean data)")%>%
  kable_classic(full_width = F, html_font = "Cambria")
```

Table 1: Section 1 Part A: Monte Carlo summary (clean data)

Estimator	Esimated	Bias	MCSD	MSE
Mean	0.998	-0.00173	0.0641	0.00411
Median	0.693	-0.30700	0.0638	0.09820
Trimmed Mean (10%)	0.830	-0.17000	0.0588	0.03240

Section 1, Part B: Logging Error Introduces Outlines

```r
set.seed(42)
# Step 1: Generate 1000 contaminated samples:
# 90% from Exp(1), 10% from Exp(0.02) to simulate contamination
lambda1 <- 1
lambda2 <- 0.02
contamination_rate <- 0.1

samples_contaminated <- matrix(NA, nrow = n, ncol = n_sim)

for (i in 1:n_sim) {
  n1 <- round(n * (1 - contamination_rate))  # number from main distribution
  n2 <- n - n1                               # number from contamination

  # Generate and combine samples
  samples_contaminated[, i] <- c(rexp(n1, rate = lambda1), rexp(n2, rate = lambda2))

  # Shuffle to mix contaminated values randomly
  samples_contaminated[, i] <- sample(samples_contaminated[, i])
}

# Step 2: Compute estimators for contaminated samples
sample_means_cont <- colMeans(samples_contaminated)
sample_medians_cont <- apply(samples_contaminated, 2, median)
sample_trimmed_means_cont <- apply(samples_contaminated, 2, function(x) mean(x, trim = 0.1))

# Step 3: Calculate Bias, MCSD, MSE for contaminated data
# True mean for mixture: 0.9 * 1/lambda1 + 0.1 * 1/lambda2
true_mean_cont <- 0.9 * (1 / lambda1) + 0.1 * (1 / lambda2)  # = 5.9

bias_mean_cont <- mean(sample_means_cont) - true_mean_cont
bias_median_cont <- mean(sample_medians_cont) - true_mean_cont
bias_trimmed_cont <- mean(sample_trimmed_means_cont) - true_mean_cont

mcsd_mean_cont <- sd(sample_means_cont)
mcsd_median_cont <- sd(sample_medians_cont)
mcsd_trimmed_cont <- sd(sample_trimmed_means_cont)

mse_mean_cont <- mean((sample_means_cont - true_mean_cont)^2)
mse_median_cont <- mean((sample_medians_cont - true_mean_cont)^2)
mse_trimmed_cont <- mean((sample_trimmed_means_cont - true_mean_cont)^2)
```

```
# Step 4: Summarise contaminated results in a table
results_partB <- data.frame(
  Estimator = c("Mean", "Median", "Trimmed Mean (10%)"),
  Esimated = signif(c(mean(sample_means_cont), mean(sample_medians_cont), mean(sample_trimme
  Bias = signif(c(bias_mean_cont, bias_median_cont, bias_trimmed_cont), 3),
  MCSD = signif(c(mcsd_mean_cont, mcsd_median_cont, mcsd_trimmed_cont), 3),
  MSE = signif(c(mse_mean_cont, mse_median_cont, mse_trimmed_cont), 3)
)


results_partB%>%
  kable(caption = "Section 1 Part B: Monte Carlo summary (contaminated data)")%>%
  kable_classic(full_width = F, html_font = "Cambria")
```

Table 2: Section 1 Part B: Monte Carlo summary (contaminated data)

Estimator	Esimated	Bias	MCSD	MSE
Mean	5.900	0.002	1.0100	1.02
Median	0.808	-5.090	0.0763	25.90
Trimmed Mean (10%)	1.080	-4.820	0.0769	23.20

```
results_partA
```

```
           Estimator Esimated     Bias   MCSD     MSE
1               Mean    0.998 -0.00173 0.0641 0.00411
2             Median    0.693 -0.30700 0.0638 0.09820
3 Trimmed Mean (10%)    0.830 -0.17000 0.0588 0.03240
```

```
 mcsd_median /mcsd_mean
```

```
[1] 0.99462
```

```
 mcsd_trimmed /mcsd_mean
```

```
[1] 0.916191
```

```r
# E(trimmed mean / median)
mcsd_median / mcsd_trimmed
```

```
[1] 1.085603
```

```r
results_partB
```

```
           Estimator Esimated   Bias   MCSD   MSE
1               Mean    5.900  0.002 1.0100  1.02
2             Median    0.808 -5.090 0.0763 25.90
3 Trimmed Mean (10%)    1.080 -4.820 0.0769 23.20
```

```r
mse_median_cont/ mse_mean_cont
```

```
[1] 25.31987
```

```r
mse_trimmed_cont/mse_mean_cont
```

```
[1] 22.65077
```

```r
mse_median_cont/mse_trimmed_cont
```

```
[1] 1.117837
```

Section 1, Part C: Reflection & Questions

This report analyzes the simulation results from Parts A and B, assessing the performance of the sample mean, median, and 10% trimmed mean as estimators for the typical completion time of security training.

1. Estimator Performance and Robustness

In the clean data scenario, the sample mean is the superior estimator. Its bias is negligible (-0.00173), and it achieves the lowest Mean Squared Error (MSE) of 0.00411, indicating high accuracy and precision. In contrast, the median and trimmed mean are noticeably biased downwards (-0.307 and -0.170, respectively).

The introduction of outliers in the contaminated data scenario dramatically alters this assessment. The sample mean's estimate jumped from 0.998 to 5.900, while its Monte Carlo

Standard Deviation (MCSD) went from 0.0641 to 1.024. This highlights its extreme sensitivity to outliers.

In this contaminated setting, the median and 10% trimmed mean proved to be far more robust. Their estimates shifted only slightly, and their MCSD remained small and stable (0.0752 for the median and 0.0771 for the trimmed mean). This stability in the face of extreme observations is the hallmark of a robust estimator and is clearly supported by the simulation results. Overall in the contaminated data, all the estimator have their values increase.

2. Relative Efficiency

An estimator's efficiency is its overall quality, considering both its precision (low variance) and accuracy (low bias). When estimators are biased, the Mean Squared Error (MSE), which accounts for both sources of error (MSE = Variance + Bias^2), is the most appropriate measure for comparing efficiency.

In the clean data scenario, the sample mean is the most efficient estimator. Its MSE of 0.00411 is nearly 8 times smaller than the trimmed mean's and 24 times smaller than the median's. In the absence of outliers, the mean's use of all data points makes it the superior choice.

In the contaminated data scenario, the 10% trimmed mean is the most efficient robust estimator. In this setting, the goal is to estimate the central tendency of the primary distribution, not the contaminated mixture. The large MSE values for the median and trimmed mean reflect their large bias relative to the mixture mean of 5.9, but this is by design. When comparing the two robust estimators for their intended purpose, the trimmed mean (MSE = 23.20) outperforms the median (MSE = 25.90). Although the median is marginally more precise (lower MCSD), the trimmed mean's smaller bias makes it the more efficient choice overall. The relative efficiency of the median to the trimmed mean is 23.20 / 25.90   0.89, indicating that the median is about 11% less efficient in this context.

Conclusion and Recommendation

The choice of estimator should be guided by the underlying data characteristics and the analytical goal. While the sample mean is optimal for clean, well-behaved data, it is unreliable when outliers are present.

Given that the IT security team's data contains known logging errors, a robust estimator is essential to avoid misleading conclusions. Our analysis demonstrates that both the median and the 10% trimmed mean are effective at resisting the influence of these outliers.

Based on statistical efficiency, the 10% trimmed mean is the recommended estimator, as it provides the best balance of accuracy and precision in the contaminated scenario.

Recommendation for the IT Security Team:

To report on the typical employee completion time, we recommend using the 10% trimmed mean, which yields an estimate of 1.08 days. An acceptable and more easily interpretable alternative is the median, which is 0.808 days. Both metrics provide a reliable and robust measure of central tendency that is not skewed by the erroneous data points. Reporting either of these values will lead to more accurate insights and better-informed policy decisions regarding employee training compliance.

Section 2: Predicting Good Health – A Data-Driven Perspective on UNSDG Goal 3

Section 2, Part A: Building the Model with Maximum Likelihood

Task 1 - Log-likelihood

We model whether life expectancy exceeds the median using a binary response and a single predictor (log GDP per capita).

- Data: (y_i, x_i) for $i = 1, \dots, n$, where $y_i \in \{0, 1\}$ and $x_i = \log(\text{gdpPercap}_i)$.
- Model: $\text{logit}(p_i) = \beta_0 + \beta_1 x_i$, where $p_i = \Pr(Y_i = 1 \mid x_i)$.

Likelihood

$$L(\beta_0, \beta_1) = \prod_{i=1}^{n} p_i^{y_i} \, (1 - p_i)^{1-y_i},$$

$$p_i = \frac{\exp(\beta_0 + \beta_1 x_i)}{1 + \exp(\beta_0 + \beta_1 x_i)}.$$

Log-likelihood

$$\ell(\beta_0, \beta_1) = \sum_{i=1}^{n} \Big[y_i(\beta_0 + \beta_1 x_i) \; - \; \log(1 + \exp(\beta_0 + \beta_1 x_i)) \Big].$$

Task 2 - Implement log-likelihood

```r
loglik_logistic <- function(par, y, x){
beta0 <- par[1]
beta1 <- par[2]
eta <- beta0 + beta1 * x
sum(y * eta - log(1 + exp(eta)))
}
```

This function `loglik_logistic` takes the parameter vector `par = c(beta0, beta1)` together with the data vectors `y` (binary 0/1) and `x = log(gdpPercap)`, forms the linear predictor $\eta_i = \beta_0 + \beta_1 x_i$, and returns the single numeric value of the logistic/Bernoulli log-likelihood

Data preparation

```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following object is masked from 'package:kableExtra':

    group_rows

The following object is masked from 'package:matrixStats':

    count

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

    filter, lag

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

    intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(gapminder)
```

```
Warning: package 'gapminder' was built under R version 4.4.3
```

```
data <- gapminder %>%
  filter(year == 2007) %>%
mutate(
high_lifeExp = ifelse(lifeExp > median(lifeExp), 1, 0),
log_gdp = log(gdpPercap)
)
```

Response and Predictor

```r
y <- data$high_lifeExp
x <- data$log_gdp
stopifnot(length(y) == length(x))
```

Task 3 - Maximise log-likelihood with `optim()`

```r
# Initial values for beta0 and beta1
par0 <- c(0, 0)

# Maximise the log likelihood
optim_fit <- optim(
par = par0,
fn = loglik_logistic,
y = y,
x = x,
method = "BFGS",
control = list(fnscale = -1),
hessian = TRUE
)

# Estimates and checks

beta_hat <- optim_fit$par
names(beta_hat) <- c("beta0_hat","beta1_hat")
ll_max <- optim_fit$value
conv <- optim_fit$convergence


beta_hat
```

```
beta0_hat beta1_hat
-20.38243   2.34889
```

```r
ll_max
```

```
[1] -42.44485
```

```
conv
```

```
[1] 0
```

Task 4 - Fit the model with `glm()`

```
model_glm <- glm(high_lifeExp ~ log_gdp, data = data, family = binomial)
coef_glm <- coef(model_glm)
coef_glm
```

```
(Intercept)     log_gdp
 -20.382309    2.348878
```

Task 5 - Compare `optim()` and `glm()` coefficients

```
# Comparison
comp <- rbind(optim = beta_hat,
glm = coef_glm)
comp
```

```
      beta0_hat beta1_hat
optim -20.38243  2.348890
glm   -20.38231  2.348878
```

```
# Numerical differences
diff <- comp["optim", ] - comp["glm", ]
diff
```

```
    beta0_hat     beta1_hat
-1.228718e-04  1.234431e-05
```

The estimates from `optim()` and `glm()` are essentially identical, differing only by numerical tolerance: about 10^{-4} for $\hat{\beta}_0$ and 10^{-5} for $\hat{\beta}_1$. This confirms that our log-likelihood implementation and maximisation with `optim()` (BFGS, `fnscale = -1`) recover the same MLEs as the built-in `glm()` fit.

Section 2, Part B: Estimating Uncertainty with Fisher Information

Task 1 - Extract and display the Hessian matrix

```
H <- optim_fit$hessian
H
```

```
            [,1]       [,2]
[1,]  -13.12452  -114.6084
[2,] -114.60837 -1007.5049
```

Task 2 - Compute and display the Fisher information matrix

```
I_hat <- -H
I_hat
```

```
          [,1]      [,2]
[1,]  13.12452  114.6084
[2,] 114.60837 1007.5049
```

Task 3 - Compute and display the standard errors

```
V_hat <- solve(I_hat)
se <- sqrt(diag(V_hat))
names(se) <- c("beta0","beta1")
se
```

```
    beta0     beta1
3.3848274 0.3863266
```

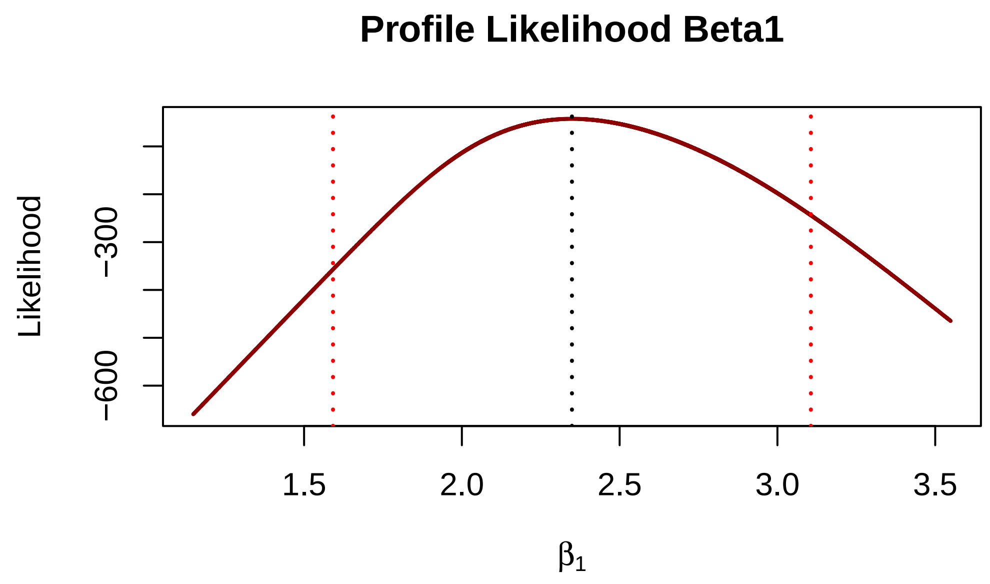
Task 4 - Manually construct the 95% confidence intervals

```
beta_hat_vec <- setNames(optim_fit$par, c("beta0","beta1"))
z <- 1.96
CI <- cbind(
lower = beta_hat_vec - z * se,
upper = beta_hat_vec + z * se
)
CI
```

```
          lower      upper
beta0 -27.01669 -13.748171
beta1   1.59169   3.106091
```

Section 2, Part C: Visualising Parameter Stability with Profile Likelihood (Hard)

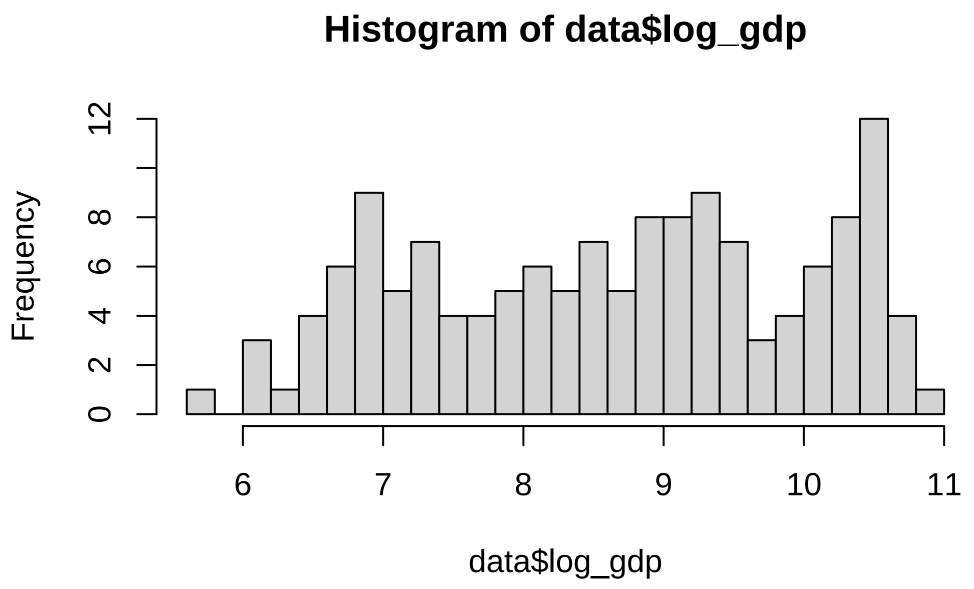
```
# LOG LIKLIHOOD PROFILE
length <- 50
beta1_hat <- model_glm$coefficients[2]
beta0_hat <- model_glm$coefficients[1]

upper_beta1 = beta1_hat +  1.2
lower_beta1 = beta1_hat -  1.2

beta1_grid <- seq(lower_beta1 , upper_beta1, 0.0001)
beta1_likelihood_grid <- numeric(length)

for(i in 1 : length(beta1_grid)) {
  beta1 <- beta1_grid[i]
  params <- c(beta0_hat,beta1)

  beta1_likelihood_grid[i] <- loglik_logistic(par = params,
                                                        y = data$high_lifeExp,
                                                        x = data$log_gdp)
}

plot(beta1_grid, beta1_likelihood_grid,
     type = "l",                          # line plot
     col = "darkred",                    # line color
     lwd = 2,                            # line width
     xlab = expression(beta[1]),         # x-axis label with math notation
     ylab = "Likelihood",                # y-axis label
     main = "Profile Likelihood Beta1")   # plot title

# Add a vertical dashed blue line at beta1 = 1.591683
#abline(v = 1.591683, col = "blue", lty = 3, lwd = 2)
#abline(v = 3.106073, col = "blue", lty = 3, lwd = 2)
abline(v =  CI[2,1], col = "red", lty = 3, lwd = 2)
abline(v = CI[2,2], col = "red", lty = 3, lwd = 2)
abline(v = beta1_hat, col = 'black',lty = 3, lwd = 2)
```



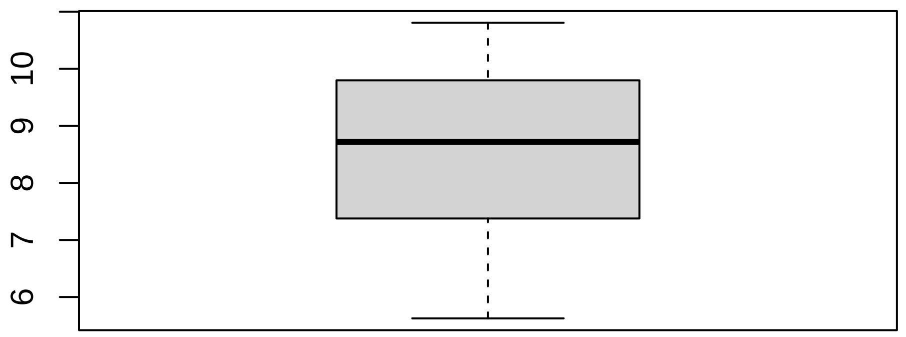
```r
max_log_likelihood <- max(beta1_likelihood_grid)
```

From the graphic, the profile likelihood curve for $\hat{\beta}_1$ shows a relatively sharp peak around its maximum. This indicates that the estimate is well-identified and has been estimated with high precision. This is further supported by the small standard error of **0.3863**, suggesting low variability in the estimate. The high precision implies that $\hat{\beta}_1$ is stable and not overly sensitive to fluctuations in the data. Additionally, the 95% confidence interval $[1.5917, 3.1061]$ is fairly narrow and does not include zero, confirming that the effect of GDP (on the log scale) on life expectancy is both statistically significant and reliably estimated. Together, these results suggest that the model's predictions regarding the effect of GDP on life expectancy are robust and trustworthy.

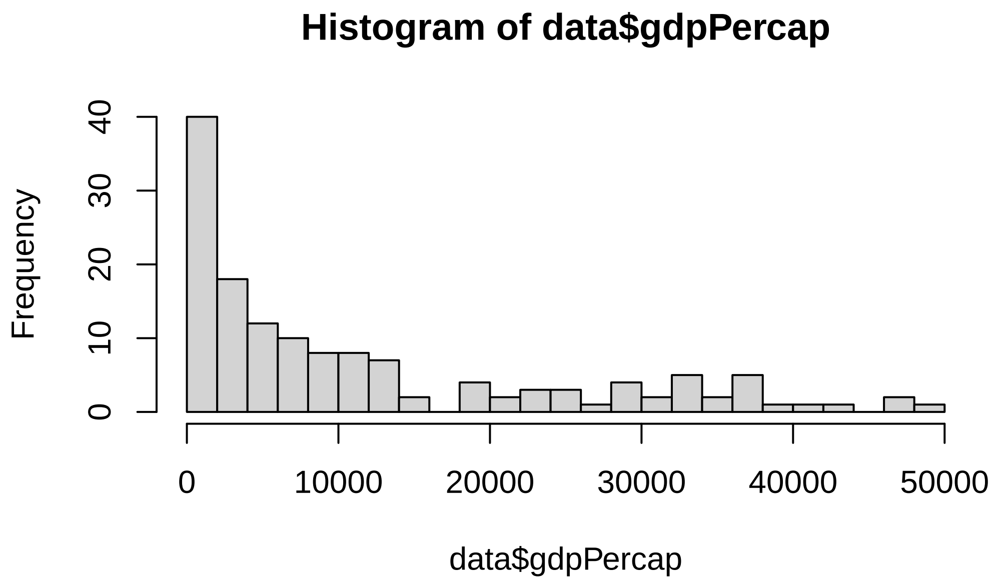
```r
hist(data$log_gdp, breaks = 30)
```



```
boxplot(data$log_gdp)
```



```r
hist(data$gdpPercap, breaks = 30)
```



```r
model_glm
```

```
Call:  glm(formula = high_lifeExp ~ log_gdp, family = binomial, data = data)

Coefficients:
(Intercept)      log_gdp  
    -20.382        2.349  

Degrees of Freedom: 141 Total (i.e. Null);  140 Residual
Null Deviance:      196.9 
Residual Deviance: 84.89    AIC: 88.89
```

$\log() = \beta_0 + \beta_1 \log(x)$

$logit(p) = -20.382 + 2.349 \times \log(x)$

$g(x_1) = \beta_0{+}\beta_1 \log x_1 g(x_2) = \beta_0{+}\beta_1 \log x_2 g(x_2){-}g(x_1) = \beta_0{+}\beta_1 \log x_2{-}\beta_0{-}\beta_1 \log x_1 g(x_2){-}g(x_1) = \beta_1(\log x_2{-}\mathrm{lo}$

```
exp(1.59169* log(1.10)) - 1 # upper range
```

```
[1] 0.163816
```

```
exp(2.349* log(1.10)) - 1
```

```
[1] 0.2509254
```

```
exp(3.106091* log(1.10)) - 1 # lower range
```

```
[1] 0.3445267
```

```
# 10 -30 range is good - 1
```

OR = 1.25, for every ln(1.1) = 10% increase in gdp per capita, you are 25% more likely to be higher life expectancy.

for every 10% you are (16,34) more more likely to be in higher life expectancy

Solving for when the odd are 1 or 50/50 of likely to be life expectancy

$-20.382 + 2.349 * \log(x) = 0$

$x = \exp(20.382/2.349)$

GDP = 5865.739 odd =1 point

odd = 1

```
# ln(odd) = 0 , odd = 1 so 50/50
exp(20.382/2.349)
```

```
[1] 5865.739
```

```
half_prop = exp(20.382/2.349)
red1.10 = half_prop*(1- 0.1)
ln_red1.10= -20.382+ 2.349*log(red1.10)
red1.10
```

```
[1] 5279.165
```

```r
ln_red1.10
```

```
[1] -0.2474919
```

```r
exp(ln_red1.10) - 1  # less likely below 5279.165 about 21% less likely to have higher life
```

```
[1] -0.2192434
```

```r
median(data$gdpPercap)
```

```
[1] 6124.371
```

```r
red2.10 = median(data$gdpPercap)*(1 - 0.1)
ln_red2.10 = -20.382+ 2.349*log(red2.10)
red2.10
```

```
[1] 5511.934
```

```r
ln_red2.10
```

```
[1] -0.1461382
```

```r
exp(ln_red2.10) - 1
```

```
[1] -0.1359617
```

Section 2, Part D: Reflection & Questions

This section interprets the findings from the logistic regression model, which predicts the likelihood of a country having a high life expectancy based on its GDP per capita. The results are contextualized for a non-technical audience with a focus on their real-world policy implications for United Nations Sustainable Development Goal 3 (UNSDG 3).

Explaining the Findings to a Non-Technical Audience

Our statistical analysis reveals a strong and reliable link between a nation's economic prosperity and the health of its population. The key findings can be summarized in two points:

1. **Wealthier is Healthier:** For every **10% increase** in a country's GDP per capita, the odds of its population having a high life expectancy (i.e., above the global median) **increase by approximately 25%**, (95% confidence interval between16% and 34% increase). This demonstrates a clear, positive relationship where economic growth is associated with significantly better health outcomes.
2. **A Critical Income Threshold:** We identified a "tipping point" for GDP per capita at approximately **$5,866**. Countries below this income level are substantially less likely to achieve a high life expectancy, while those above it are progressively more likely. For instance, a country with a GDP per capita 10% below this threshold is about 22% less likely to have a high life expectancy compared to a country at the threshold.

In simple terms, while not the only factor, economic health is a powerful predictor of a population's physical well-being.

Relationship Between GDP per Capita and Life Expectancy

The statistical model confirms a significant positive relationship between the logarithm of GDP per capita and the probability of having a high life expectancy. The estimated coefficient for log(GDP), $\widehat{\beta}_1 = 2.349$, is statistically significant, with a 95% confidence interval of [1.59, 3.11] that does not include zero. The sharpness of the profile likelihood curve further reinforces that this estimate is precise and stable.

The use of a logarithmic scale for GDP is important. It implies that the impact of economic growth on life expectancy is most pronounced at lower income levels. For example, an increase in GDP per capita from $1,000 to $2,000 has the same positive effect on the odds of high life expectancy as an increase from $10,000 to $20,000.

Implications for UNSDG Goal 3 and Global Health Investments

These findings have direct implications for achieving UNSDG Goal 3 ("ensure healthy lives and promote well-being for all"). The strong link between economic status and health suggests that global health strategies should be integrated with economic development initiatives.

This analysis can help guide global health investments in a targeted manner:

- **Prioritize Countries Near the Threshold:** The greatest return on investment for improving life expectancy may be found in countries with a GDP per capita just below the **$5,866 threshold**. For these nations, relatively modest economic assistance could push them past this tipping point, leading to substantial gains in health outcomes.

- **Tailor Strategies:** For countries far below this threshold, broad economic development and investment in fundamental infrastructure (e.g., sanitation, education, basic healthcare) are critical prerequisites for improving life expectancy. For nations already above the threshold, health investments might focus more on advanced healthcare, lifestyle-related diseases, and promoting well-being in aging populations.

By using data to identify this economic threshold, policymakers can better allocate resources to where they will have the most significant and immediate impact on global health.