STAT3191/6191: Group Project Submission

Group Number: Group 17

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

2025-09-26

Part A: Natural Variation in Completion Times

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

```
# 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)
```

```r
# 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()%>%
  kable_classic(full_width = F, html_font = "Cambria")
```

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
```

```r
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_trimmed
  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()%>%
  kable_classic(full_width = F, html_font = "Cambria")
```

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
```

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

```
[1] 1.085603
```

```
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
```

```
mcsd_median_cont/ mcsd_mean_cont
```

```
[1] 0.07533951
```

```
mcsd_trimmed_cont/mcsd_mean_cont
```

```
[1] 0.07591323
```

```
 # E(trimmed mean / median)
mcsd_median_cont/mcsd_trimmed_cont
```

```
[1] 0.9924425
```

Section 2 — Predicting Good Health (UNSDG 3)

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 $i = 1, \ldots, n$, 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

```
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
```

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

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
```

```r
beta_hat
```

```
beta0_hat beta1_hat
-20.38243   2.34889
```

```r
ll_max
```

```
[1] -42.44485
```

```r
conv
```

```
[1] 0
```

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

```r
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

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

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

```r
# 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.

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

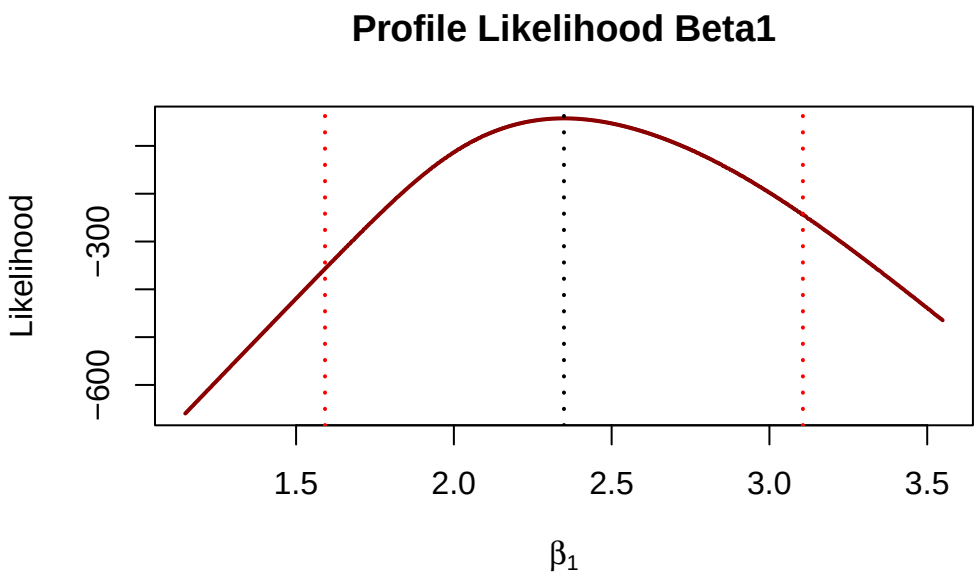
```
beta_hat_vec <- setNames(optim_fit$par, c("beta0","beta1"))
z <- 1.96
CI <- cbind(
lower = beta_hat_vec - z * se,
upper = beta_hat_vec + z * se
)
CI
```

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

PROFILING LOG LIKLIHOOD

```
# 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
```

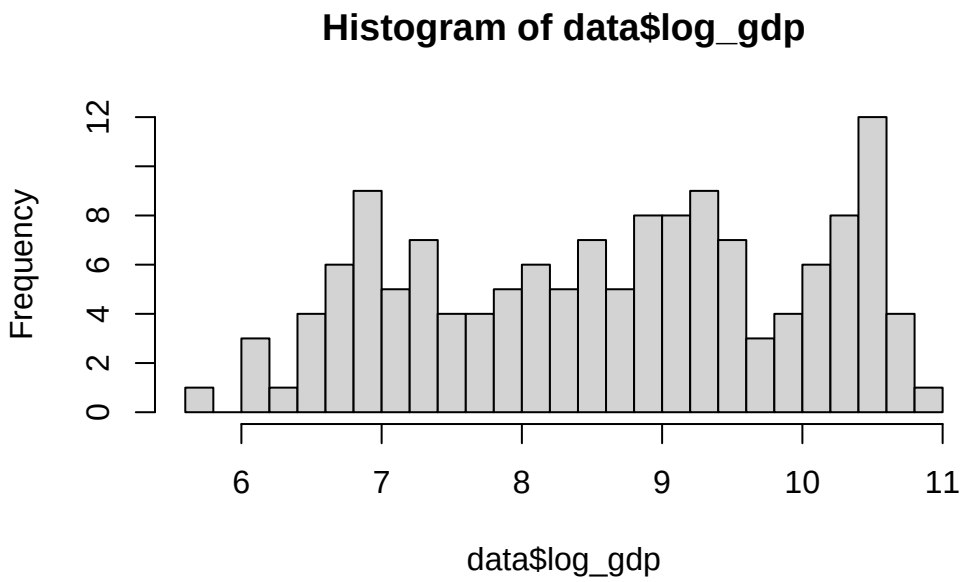
```
     xlab = expression(beta[1]),     # x-axis label with math notation
     ylab = "Likelihood",            # y-axis label
     main = "Profile Likelihood Beta1")   # plot title

# 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)
```



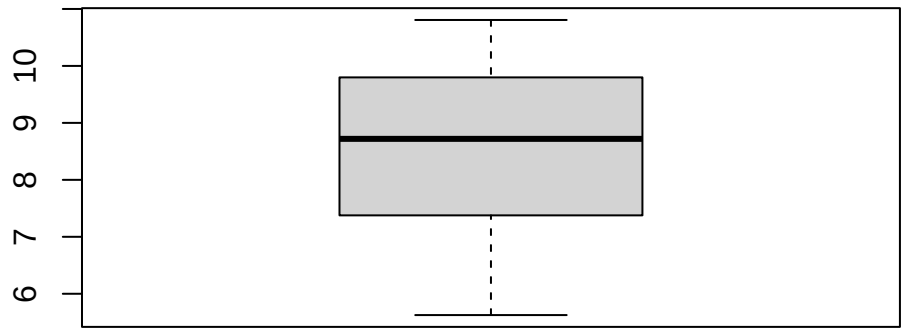

```
max_log_likelihood <- max(beta1_likelihood_grid)
```

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.

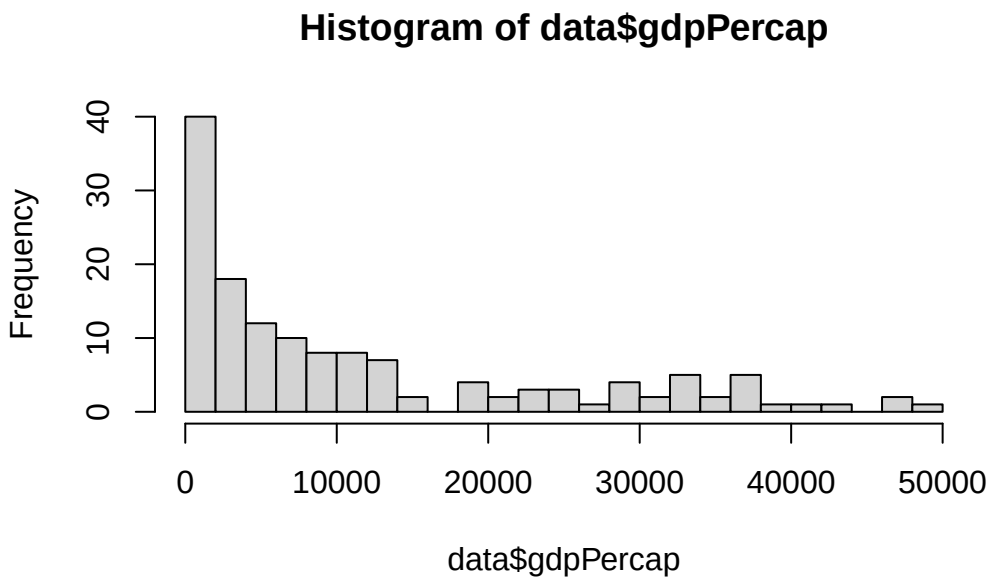
```r
hist(data$log_gdp, breaks = 30)
```




```r
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
```

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

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

```r
exp(2.349)
```

```
[1] 10.47509
```

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

```
CI
```

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

```
exp(1.59169* log(1.10)) - 1
```

```
[1] 0.163816
```

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

```
[1] 0.2509254
```

```
exp(3.106091* log(1.10)) - 1
```

```
[1] 0.3445267
```

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

exp(20.382/2.349)

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

```
[1] 5865.739
```

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

GDP = 5865.739 median

odd = 1

```
> 5865.739 - 5865.739 * 0.1
[1] 5279.165
> -20.382+ 2.349*log(5279.165)
[1] -0.247492
> exp(-0.247492)
[1] 0.7807565 # less likely below 5279.165 about 21% less likely to have higher life expectar
```

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

```
[1] 6124.371
```

```
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
```

```
ln_red1.10
```

```
[1] -0.2474919
```

```
exp(ln_red1.10) - 1
```

```
[1] -0.2192434
```

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

```
[1] 5511.934
```

```
ln_red2.10
```

```
[1] -0.1461382
```

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

```
[1] -0.1359617
```