Assignment 2 code

Roudranil Das, Shreyan Chakraborty, Saikat Bera

2022-11-16

Problem 1

Suppose X denote the number of goals scored by home team in premier league. We can assume X is a random variable. Then we have to build the probability distribution to model the probability of number of goals. Since X takes value in $\mathbb{N} = \{0, 1, 2, \cdots\}$, we can consider the geometric progression sequence as possible candidate model, i.e.,

$$S = \{a, ar, ar^2, ar^3, \cdots\}.$$

But we have to be careful and put proper conditions in place and modify S in such a way so that it becomes proper probability distributions.

Part 1

Figure out the necesary conditions and define the probability distribution model using S.

We are given a random variable X with a supposed prob. distribution $\{a, ar, ar^2, ...\}$. Hence, we can write:

$$\mathbb{P}(X = i) = ar^i$$

Now, because $\sum_i \mathbb{P}(X = i) = 1$ for a probability distribution, we must have

$$\begin{aligned}
\sum_i \mathbb{P}(X = i) &= 1 \\
\implies \sum_i ar^i &= 1 \\
\implies \frac{a}{1-r} &= 1 \\
\implies a &= 1 - r
\end{aligned}$$

We have ***assumed*** that $|r| < 1$, but because probabilities are non-negative, it's sufficient to have: $0 \le r < 1$.

Hence, we finally have

$$\mathbb{P}(X = i) = (1 - r)r^i, \quad 0 \le r < 1$$

Parts 2 and 3

2. Check if mean and variance exists for the probability model.

3. Can you find the analytically expression of mean and variance.

Once again, we ***assume*** $|r| < 1$, but because probabilities are non-negative, it's sufficient to have: $0 \le r < 1$.

We now have:

$$\begin{aligned}
\mathbb{E}(X) &= \sum_i i\mathbb{P}(X = i) \\
&= \sum_i i(1-r)r^i \\
\implies \frac{\mathbb{E}(X)}{1-r} &= \sum_i ir^i \\
\implies \frac{\mathbb{E}(X)}{1-r} &= \frac{r}{(1-r)^2} \\
\implies \mathbb{E}(X) &= \frac{r}{1-r}
\end{aligned}$$

Observe that

$$\begin{aligned}
\mathbb{E}(X^2) &= \sum_i i^2\mathbb{P}(X = i) \\
&= \sum_i i^2(1-r)r^i
\end{aligned}$$

Calculating the sum, we obtain:

$$\mathbb{E}(X^2) = \frac{r(r+1)}{(1-r)^2}$$

To calculate the variance, now:

$$\begin{aligned}
\mathrm{Var}(X) &= \mathbb{E}(X^2) - [\mathbb{E}(X)]^2 \\
&= \frac{r(r+1)}{(1-r)^2} - \frac{r^2}{(1-r)^2} \\
&= \frac{r}{(1-r)^2}
\end{aligned}$$

Part 4

From historical data we found the following summary statistics

mean	median	variance	total number of matches
1.5	1	2.25	380

Using the summary statistics and your newly defined probability distribution model find the following: a. What is the probability that home team will score at least one goal? b. What is the probability that home team will score at least one goal but less than four goal?

We are given that the mean is 1.5 and the variance is 2.25. This amounts to solving the following set of equations:

$$\frac{r}{1-r} = 1.5 \tag{1}$$

$$\frac{r}{(1-r)^2} = 2.25 \tag{2}$$

But, one can easily verify that this system of equations is inconsistent. So, we cannot solve for the value of r using this method. To resolve this issue, we are going to find the ***maximum likelihood estimate*** of r.

Denote by $\mathbf{x}$ the observed values in a random sample $x_1, x_2, \cdots, x_n$. The likelihood function for the geometric distribution can be expressed as:

$$L(r|\mathbf{x}) = \prod_{i=1}^{n} (1-r) r^{x_i} = (1-r)^n \, r^{\sum_{i=1}^{n} x_i}$$

Taking the natural logarithm of the likelihood function gives:

$$\ln L(r|\mathbf{x}) = \ln \left[(1-r)^n \, r^{\sum_{i=1}^{n} x_i} \right] = n \ln(1-r) + \ln(r) \sum x_i \tag{a}$$

Let's take the first-order partial derivative of $\ln L(r|\mathbf{x})$ with respect to r and set the answer equal to zero:

$$\frac{\partial \ln L(r|\mathbf{x})}{\partial r} = -\frac{n}{1-r} + \frac{\sum x_i}{r} \overset{set}{=} 0$$

The solution is given by $\hat{r} = \frac{\sum x_i}{\sum x_i + n}$. It's easy to check that the second-order partial derivative of the log-likelihood function is negative at $r = \hat{r}$.

For our problem, let's find this value from the summary of data we're given:

$$\hat{r} = \frac{380 * 1.5}{380 * 1.5 + 380} = \frac{3}{5}$$

Consequently,

a. $\mathbb{P}(X \geq 1) = 1 - \mathbb{P}(X = 0) = 1 - (1 - r) = 0.6$;
b. $\mathbb{P}(1 \leq X < 4) = \mathbb{P}(X = 1) + \mathbb{P}(X = 2) + \mathbb{P}(X = 3) = (1 - r)\left[r + r^2 + r^3\right] = 0.4704$

Part 5

Suppose on another thought you want to model it with off-the shelf Poisson probability models. Under the assumption that underlying distribution is Poisson probability find the above probabilities, i.e., a. What is the probability that home team will score at least one goal? b. What is the probability that home team will score at least one goal but less than four goal?

For a Poisson distribution, the mean and variance is equal to λ. Furthermore, we know that the ***maximum likelihood estimate*** for λ is the sample mean. So, we will take $\lambda = 1.5$.

a. $\mathbb{P}(X \geq 1) = 1 - \mathbb{P}(X = 0) = 1 - e^{-\lambda} = 0.78$;
b. $\mathbb{P}(1 \leq X < 4) = \mathbb{P}(X = 1) + \mathbb{P}(X = 2) + \mathbb{P}(X = 3) = e^{-\lambda}\left[\lambda + \frac{\lambda^2}{2} + \frac{\lambda^3}{6}\right] = 0.71$

Part 6

Which probability model you would prefer over another?

When we first see the observed statistics such as mean and variance, we see that sample variance is more than that of sample mean.

Generally speaking,geometric distribution has its variance more than that of mean.

But we see that,on further inspection the sample mean and sample variance is inconsistent.Therefore the given geometric distribution is not that of great fit.

Looking at the observed probabilities for both (a) and (b) in Parts 4 and 5, we are inclined towards the Poisson model as likely to better fit the data.

Part 7

Write down the likelihood functions of your newly defined probability models and Poisson models. Clearly mention all the assumptions that you are making.

For the Poisson distribution:

$$L(\lambda|\mathbf{x}) = \prod_{i=1}^{n} \frac{e^{-\lambda}\lambda^{x_i}}{x_i!} = e^{-n\lambda} \prod_{i=1}^{n} \frac{\lambda^{x_i}}{x_i!}$$

For the geometric distribution:

$$L(r|\mathbf{x}) = \prod_{i=1}^{n} (1-r) r^{x_i} = (1-r)^n \, r^{\sum_{i=1}^{n} x_i}$$

Problem 2

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Part 1

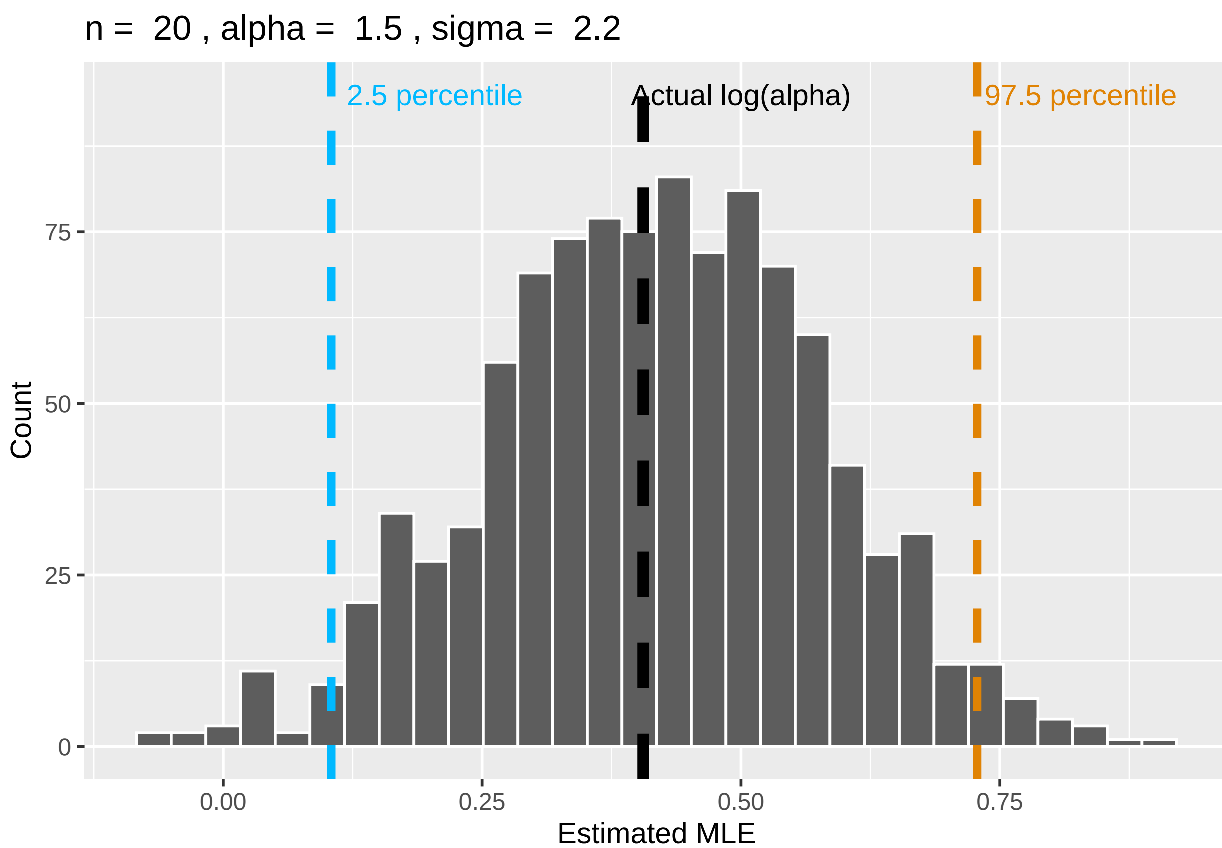
```
mle <- function(log_alpha, data, sigma) {
    l = sum(log(dgamma(data, shape = exp(log_alpha), scale = sigma)))
    # print(paste('l is ', l))
    return(-l)
}

MyMLE <- function(data, sigma) {
    log_alpha_initial <- log(mean(data)^2/var(data))
    # print(paste('log alpha initial is ',
    # log_alpha_initial))
    estimator <- optim(log_alpha_initial, mle, data = data, sigma = sigma)
    log_alpha_hat <- estimator$par
    return(log_alpha_hat)
}
```

```
get_estimates <- function(n, alpha, sigma) {
    estimates <- c()
    for (i in 1:1000) {
        samples <- rgamma(n, shape = alpha, scale = sigma)
        # print(paste('some of the samples are ',
        # samples[1:5]))
        estimates <- append(estimates, MyMLE(data = samples,
            sigma = sigma))
    }
    return(estimates)
}
```

Part 2

```
n = 20
alpha = 1.5
sigma = 2.2

estimated_mle <- tibble(get_estimates(n = n, alpha = alpha, sigma = sigma))
colnames(estimated_mle) <- c("estimate")

perc_2.5 <- quantile(estimated_mle$estimate, probs = 0.025, names = FALSE)
perc_97.5 <- quantile(estimated_mle$estimate, probs = 0.975,
    names = FALSE)

estimated_mle %>%
    ggplot(aes(estimate)) + geom_histogram(color = "white", fill = "#5D5D5D") +
    geom_vline(xintercept = log(alpha), size = 2, linetype = "dashed") +
    annotate("text", label = "Actual log(alpha)", x = 0.5, y = 95,
        color = "black") + geom_vline(xintercept = perc_2.5,
    color = "#00B9FF", size = 1.5, linetype = "dashed") + annotate("text",
    label = "2.5 percentile", x = perc_2.5 + 0.1, y = 95, color = "#00B9FF") +
    geom_vline(xintercept = perc_97.5, color = "#E08304", size = 1.5,
        linetype = "dashed") + annotate("text", label = "97.5 percentile",
    x = perc_97.5 + 0.1, y = 95, color = "#E08304") + labs(title = paste("n = ",
    n, ", alpha = ", alpha, ", sigma = ", sigma), x = "Estimated MLE",
    y = "Count")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



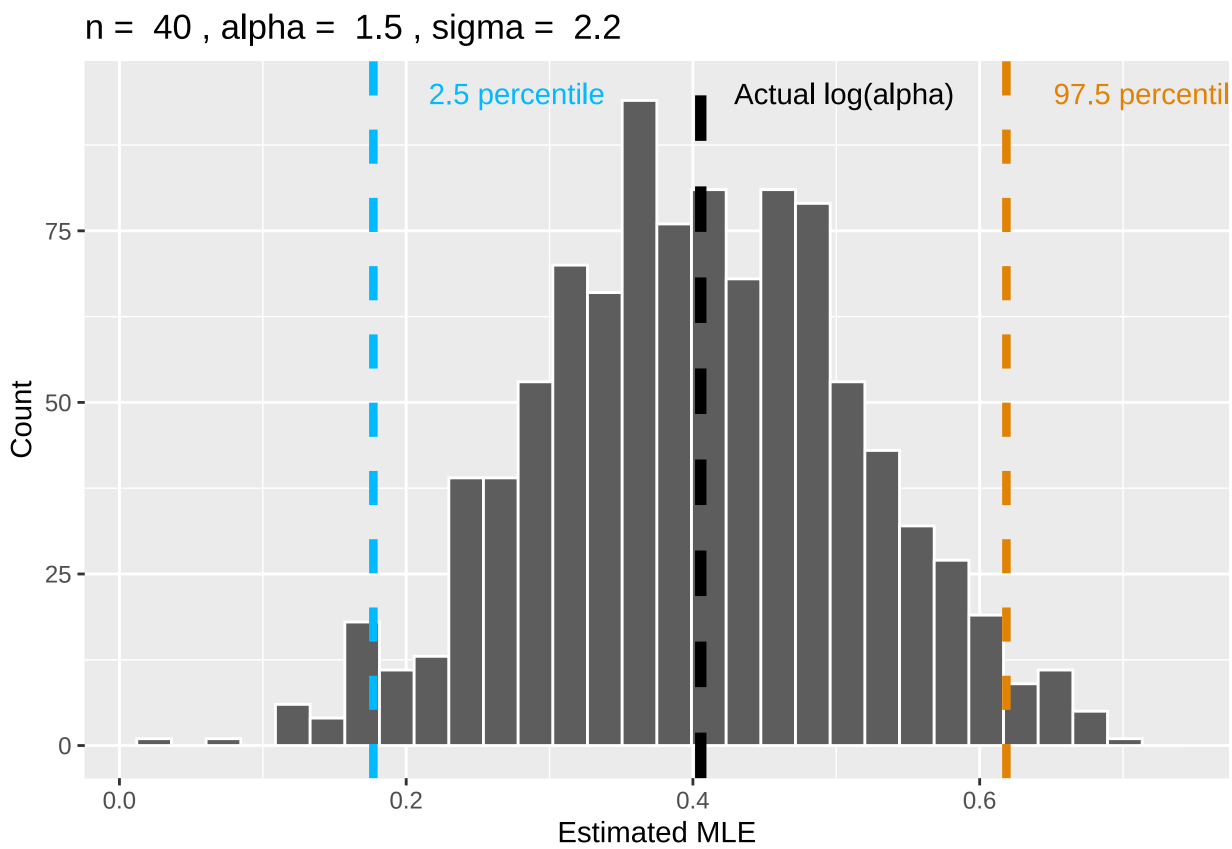

```
diff_20 <- perc_97.5 - perc_2.5
```

Part 3

```
n = 40
alpha = 1.5
sigma = 2.2

estimated_mle <- tibble(get_estimates(n = n, alpha = alpha, sigma = sigma))
colnames(estimated_mle) <- c("estimate")

perc_2.5 <- quantile(estimated_mle$estimate, probs = 0.025, names = FALSE)
perc_97.5 <- quantile(estimated_mle$estimate, probs = 0.975,
    names = FALSE)

estimated_mle %>%
    ggplot(aes(estimate)) + geom_histogram(color = "white", fill = "#5D5D5D") +
    geom_vline(xintercept = log(alpha), size = 2, linetype = "dashed") +
    annotate("text", label = "Actual log(alpha)", x = log(alpha) +
        0.1, y = 95, color = "black") + geom_vline(xintercept = perc_2.5,
    color = "#00B9FF", size = 1.5, linetype = "dashed") + annotate("text",
    label = "2.5 percentile", x = perc_2.5 + 0.1, y = 95, color = "#00B9FF") +
    geom_vline(xintercept = perc_97.5, color = "#E08304", size = 1.5,
        linetype = "dashed") + annotate("text", label = "97.5 percentile",
    x = perc_97.5 + 0.1, y = 95, color = "#E08304") + labs(title = paste("n = ",
    n, ", alpha = ", alpha, ", sigma = ", sigma), x = "Estimated MLE",
```

```
    y = "Count")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



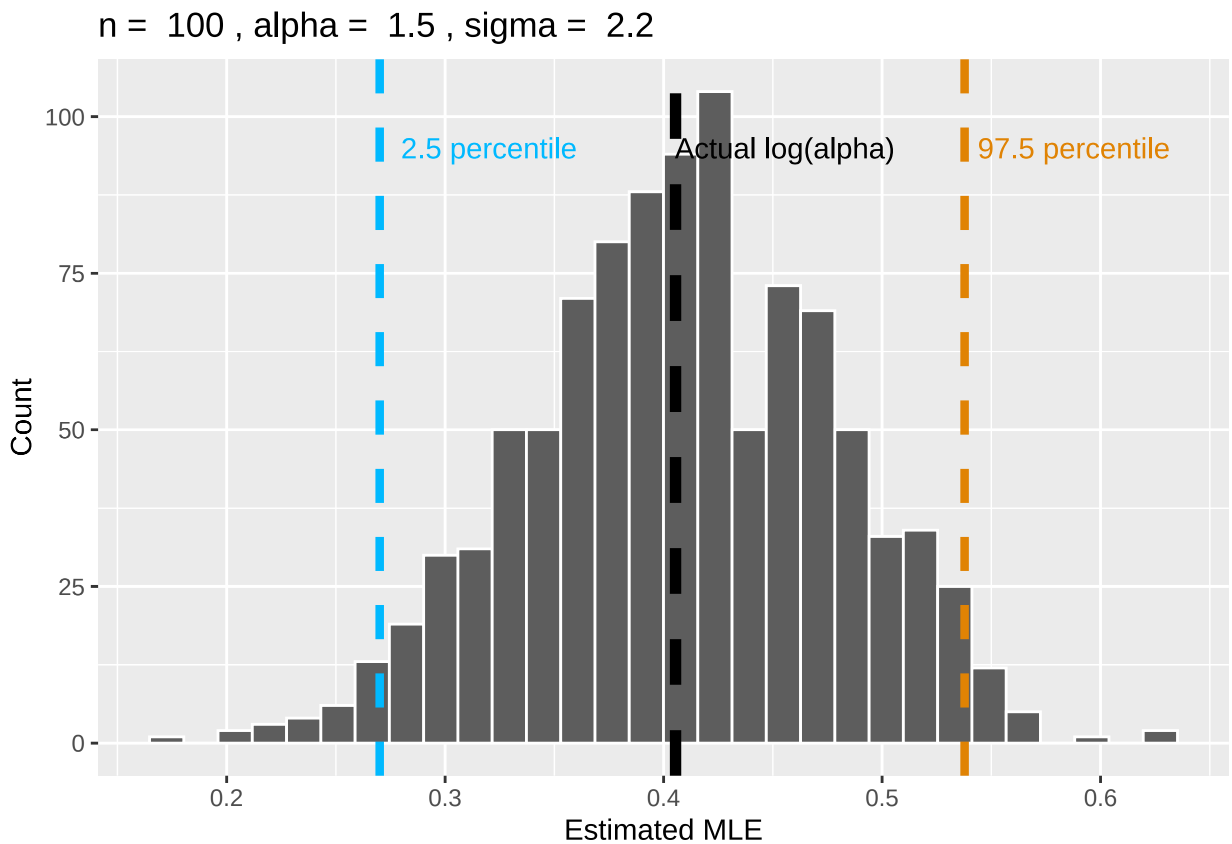

```
diff_40 <- perc_97.5 - perc_2.5
```

Part 4

```
n = 100
alpha = 1.5
sigma = 2.2

estimated_mle <- tibble(get_estimates(n = n, alpha = alpha, sigma = sigma))
colnames(estimated_mle) <- c("estimate")

perc_2.5 <- quantile(estimated_mle$estimate, probs = 0.025, names = FALSE)
perc_97.5 <- quantile(estimated_mle$estimate, probs = 0.975,
    names = FALSE)

estimated_mle %>%
    ggplot(aes(estimate)) + geom_histogram(color = "white", fill = "#5D5D5D") +
    geom_vline(xintercept = log(alpha), size = 2, linetype = "dashed") +
    annotate("text", label = "Actual log(alpha)", x = log(alpha) +
        0.05, y = 95, color = "black") + geom_vline(xintercept = perc_2.5,
    color = "#00B9FF", size = 1.5, linetype = "dashed") + annotate("text",
    label = "2.5 percentile", x = perc_2.5 + 0.05, y = 95, color = "#00B9FF") +
```

```r
    geom_vline(xintercept = perc_97.5, color = "#E08304", size = 1.5,
        linetype = "dashed") + annotate("text", label = "97.5 percentile",
    x = perc_97.5 + 0.05, y = 95, color = "#E08304") + labs(title = paste("n = ",
    n, ", alpha = ", alpha, ", sigma = ", sigma), x = "Estimated MLE",
    y = "Count")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
diff_100 <- perc_97.5 - perc_2.5
```

```r
diff_20
```

```
## [1] 0.6237406
```

```r
diff_40
```

```
## [1] 0.4415459
```

```r
diff_100
```

```
## [1] 0.2677258
```

We can see that the gap between the percentile points is decreasing as the sample size increases.

Problem 3

```r
library(tidyverse)
library(scales)
```

```
##
## Attaching package: 'scales'
## The following object is masked from 'package:purrr':
##
##     discard
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
# library(AICcmodavg)
```

```
data_q3 <- faithful %>%
    as_tibble()

x <- sort(data_q3$waiting)

# hist(x, xlab = 'waiting', probability = T, col='pink',
# main='')
```

comparing 3 models

```
# model 1

p <- length(x[x < 65])/length(x)

as <- mean(x[x < 65])
ass <- var(x[x < 65])
s <- ass/as
a <- as/s

mu <- mean(x[x >= 65])
sigma <- sd(x[x >= 65])

theta_inital <- c(p, a, s, mu, sigma)

neg_log_likelihood <- function(theta, data) {
    n = length(data)

    p = theta[1]
    a = theta[2]
    s = theta[3]
    mu = theta[4]
    sigma = theta[5]

    l = 0
    for (i in 1:n) {
        l = l + log(p * dgamma(data[i], shape = a, scale = s) +
            (1 - p) * dnorm(data[i], mean = mu, sd = sigma))
    }
    return(-l)
}

fit = optim(theta_inital, neg_log_likelihood, data = x, control = list(maxit = 1500),
    lower = c(0, 0, 0, -Inf, 0), upper = c(1, Inf, Inf, Inf,
```

```
        Inf), method = "L-BFGS-B")

theta_1 = fit$par
theta_1
```

```
## [1]   0.3574036 101.4861312   0.5371140  80.0176038   5.9487453
```

```
p = theta_1[1]
a = theta_1[2]
s = theta_1[3]
mu = theta_1[4]
sigma = theta_1[5]

model_1 = p * dgamma(x, shape = a, scale = s) + (1 - p) * dnorm(x,
    mean = mu, sd = sigma)

aic_1 <- 2 * 5 + 2 * neg_log_likelihood(theta_1, x)

# hist(x, xlab = 'waiting', probability = T, col='pink',
# main='') lines(x, model_1)
```

```
# model 2

p <- length(x[x < 65])/length(x)

as_1 <- mean(x[x < 65])
ass_1 <- var(x[x < 65])
s_1 <- ass_1/as_1
a_1 <- as_1/s_1

as_2 <- mean(x[x >= 65])
ass_2 <- var(x[x >= 65])
s_2 <- ass_2/as_2
a_2 <- as_2/s_2

theta_inital <- c(p, a_1, s_1, a_2, s_2)

neg_log_likelihood <- function(theta, data) {
    n <- length(data)

    p <- theta[1]
    a_1 <- theta[2]
    s_1 <- theta[3]
    a_2 <- theta[4]
    s_2 <- theta[5]

    l <- 0
    for (i in 1:n) {
        l = l + log(p * dgamma(data[i], shape = a_1, scale = s_1) +
            (1 - p) * dgamma(data[i], shape = a_2, scale = s_2))
    }
    return(-l)
}
```

```r
fit = optim(theta_inital, neg_log_likelihood, data = x, control = list(maxit = 1500),
    lower = c(0, 0, 0, 0, 0), upper = c(1, Inf, Inf, Inf, Inf),
    method = "L-BFGS-B")

theta_2 <- fit$par
theta_2
```

```
## [1]   0.3582592 101.5126436   0.5371146 169.2757878   0.4728250
```

```r
p <- theta_2[1]
a_1 <- theta_2[2]
s_1 <- theta_2[3]
a_2 <- theta_2[4]
s_2 <- theta_2[5]

model_2 <- p * dgamma(x, shape = a_1, scale = s_1) + (1 - p) *
    dgamma(x, shape = a_2, scale = s_2)

aic_2 <- 2 * 5 + 2 * neg_log_likelihood(theta_2, x)

# hist(x, xlab = 'waiting', probability = T, col='pink',
# main='') lines(x, model_2)
```

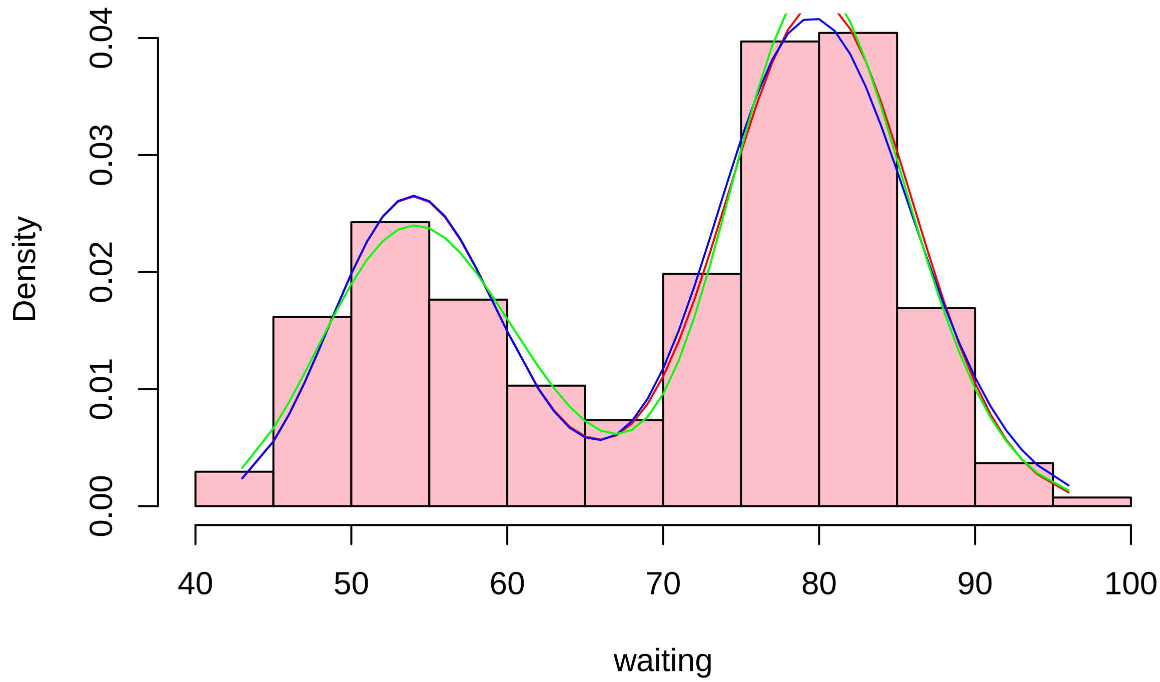
```r
# model 3

p <- length(x[x < 65])/length(x)

m_1 <- mean(x[x < 65])
v_1 <- var(x[x < 65])
sigma2_1 <- log((v_1/m_1^2) + 1)
mu_1 <- log(m_1) - sigma2_1/2

m_2 <- mean(x[x >= 65])
v_2 <- var(x[x >= 65])
sigma2_2 <- log((v_2/m_2^2) + 1)
mu_2 <- log(m_2) - sigma2_2/2

theta_inital <- c(p, mu_1, sqrt(sigma2_1), mu_2, sqrt(sigma2_2))

neg_log_likelihood <- function(theta, data) {
    n <- length(data)

    p <- theta[1]
    mu_1 <- theta[2]
    sigma_1 <- theta[3]
    mu_2 <- theta[4]
    sigma_2 <- theta[5]

    l <- 0
    for (i in 1:n) {
        l = l + log(p * dlnorm(data[i], meanlog = mu_1, sdlog = sigma_1) +
            (1 - p) * dlnorm(data[i], meanlog = mu_2, sdlog = sigma_2))
    }
```

```r
    return(-l)
}

fit = optim(theta_inital, neg_log_likelihood, data = x, control = list(maxit = 1500),
    lower = c(0, -Inf, 0, -Inf, 0), upper = c(1, Inf, Inf, Inf,
        Inf), method = "L-BFGS-B")

theta_3 <- fit$par
theta_3
```

```
## [1] 0.37613816 4.00383608 0.11485512 4.38430182 0.06973823
```

```r
p <- theta_3[1]
mu_1 <- theta_3[2]
sigma_1 <- theta_3[3]
mu_2 <- theta_3[4]
sigma_2 <- theta_3[5]

model_3 <- p * dlnorm(x, meanlog = mu_1, sdlog = sigma_1) + (1 -
    p) * dlnorm(x, meanlog = mu_2, sdlog = sigma_2)

aic_3 <- 2 * 5 + 2 * neg_log_likelihood(theta_3, x)

# hist(x, xlab = 'waiting', probability = T, col='pink',
# main='') lines(x, model_3)
```

```r
hist(x, xlab = "waiting", probability = T, col = "pink", main = "")
lines(x, model_1, col = "red")
lines(x, model_2, col = "blue")
lines(x, model_3, col = "green")
```




```
results <- data.frame(models = c("Gamma + Normal", "Gamma + Gamma",
    "Lognormal + Lognormal"), AIC = c(aic_1, aic_2, aic_3))
results
```

```
##                  models      AIC
## 1        Gamma + Normal 2077.495
## 2         Gamma + Gamma 2078.725
## 3 Lognormal + Lognormal 2075.420
```

Based on the AIC value of all the models, we choose the third model as it has the lowest AIC value.

The required probability $\mathbb{P}[60 < \texttt{waiting} < 70]$ is:

```
p <- theta_3[1]
mu_1 <- theta_3[2]
sigma_1 <- theta_3[3]
mu_2 <- theta_3[4]
sigma_2 <- theta_3[5]

reqd_prob <- (p * plnorm(70, meanlog = mu_1, sdlog = sigma_1) +
    (1 - p) * plnorm(70, meanlog = mu_2, sdlog = sigma_2)) -
    (p * plnorm(60, meanlog = mu_1, sdlog = sigma_1) + (1 - p) *
        plnorm(60, meanlog = mu_2, sdlog = sigma_2))
```

Hence $\mathbb{P}[60 < \texttt{waiting} < 70] = 0.0908132$

Problem 4

Part A

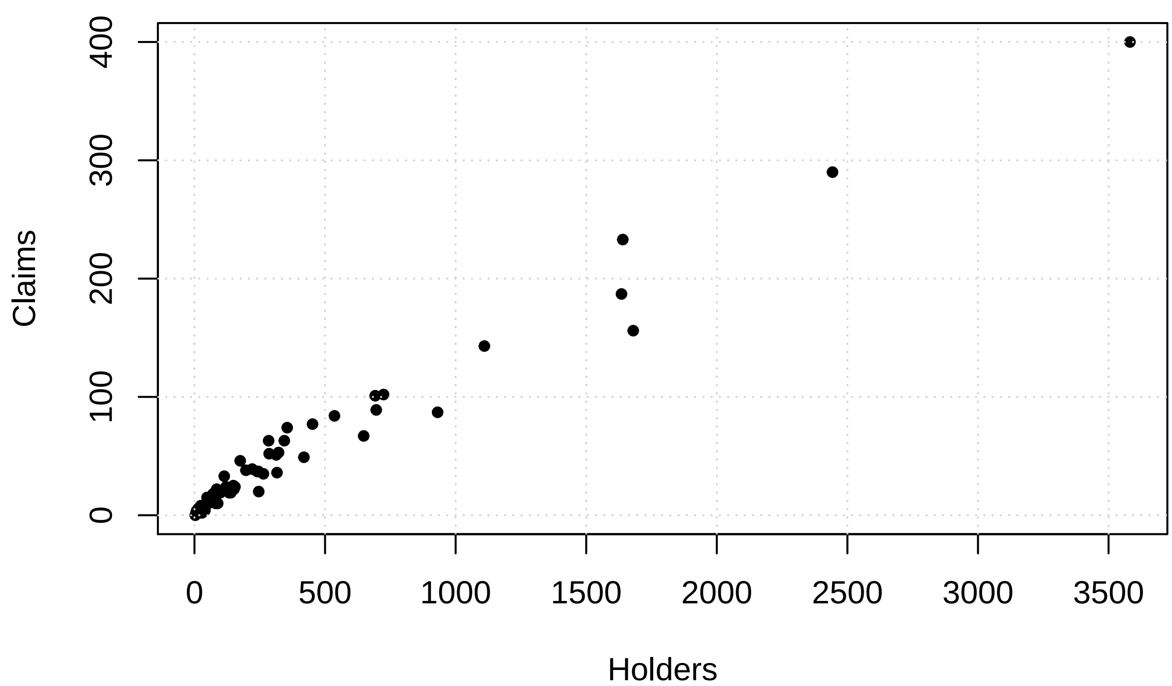
(i)

```
library(tidyverse)
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

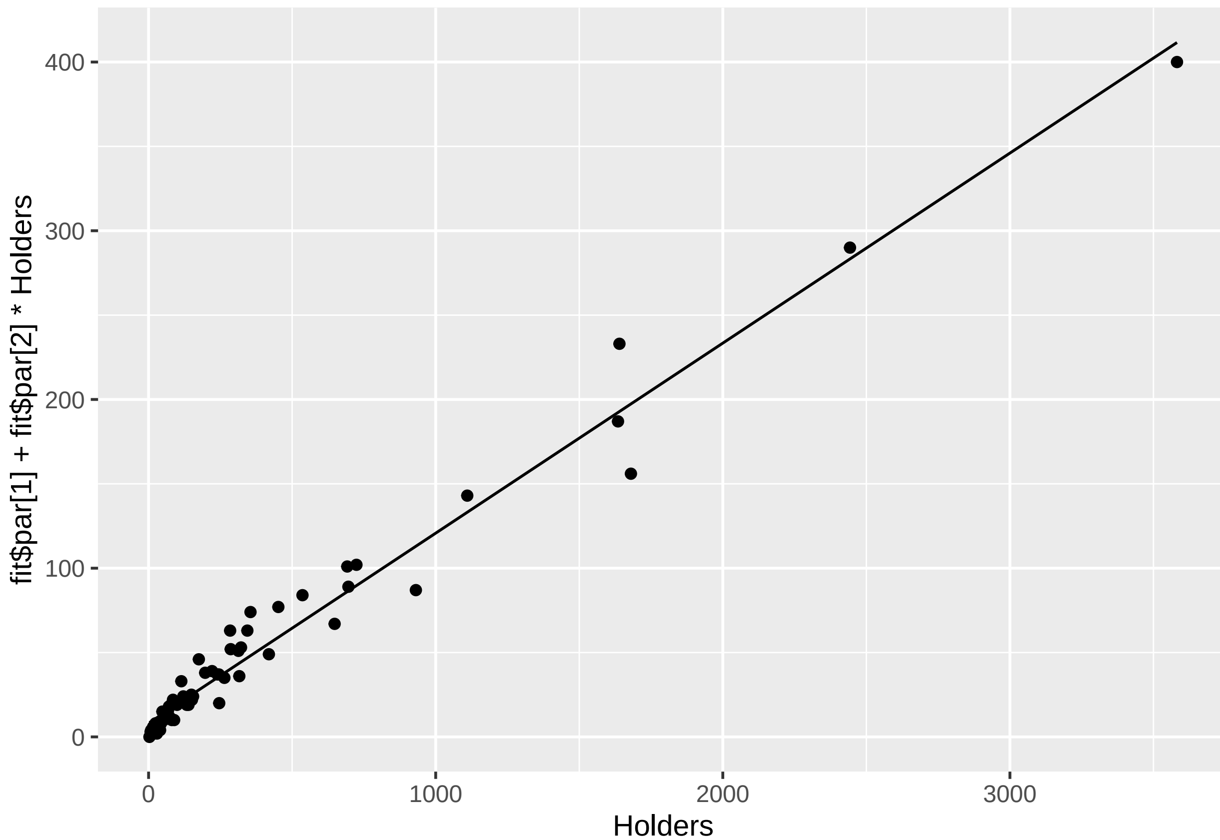
```
plot(Insurance$Holders, Insurance$Claims, xlab = "Holders", ylab = "Claims",
    pch = 20)
grid()
```




```
NegLogLikeA = function(theta, data) {
    sigma = exp(theta[3])
    n = nrow(data)
    l = 0
    for (i in 1:n) {
        mu = theta[1] + theta[2] * data$Holders[i]
        l = l + log(dnorm(data$Claims[i], mean = mu, sd = sigma))
```

```
    }
    return(-l)
}


theta_initial = c(0.01, 0.1, log(10))
NegLogLikeA(theta_initial, Insurance)

fit = optim(theta_initial, NegLogLikeA, data = Insurance)
```

```
ggplot(data = Insurance) + geom_line(aes(Holders, fit$par[1] +
    fit$par[2] * Holders)) + geom_point(aes(Holders, Claims))
```




```
theta_hat = fit$par
theta_hat
```

```
## [1] 8.1271165 0.1126338 2.4741587
```

(ii)

```
BIC = 2 * NegLogLikeA(theta_hat, Insurance) + log(nrow(Insurance)) *
    3
```

```
BIC
```

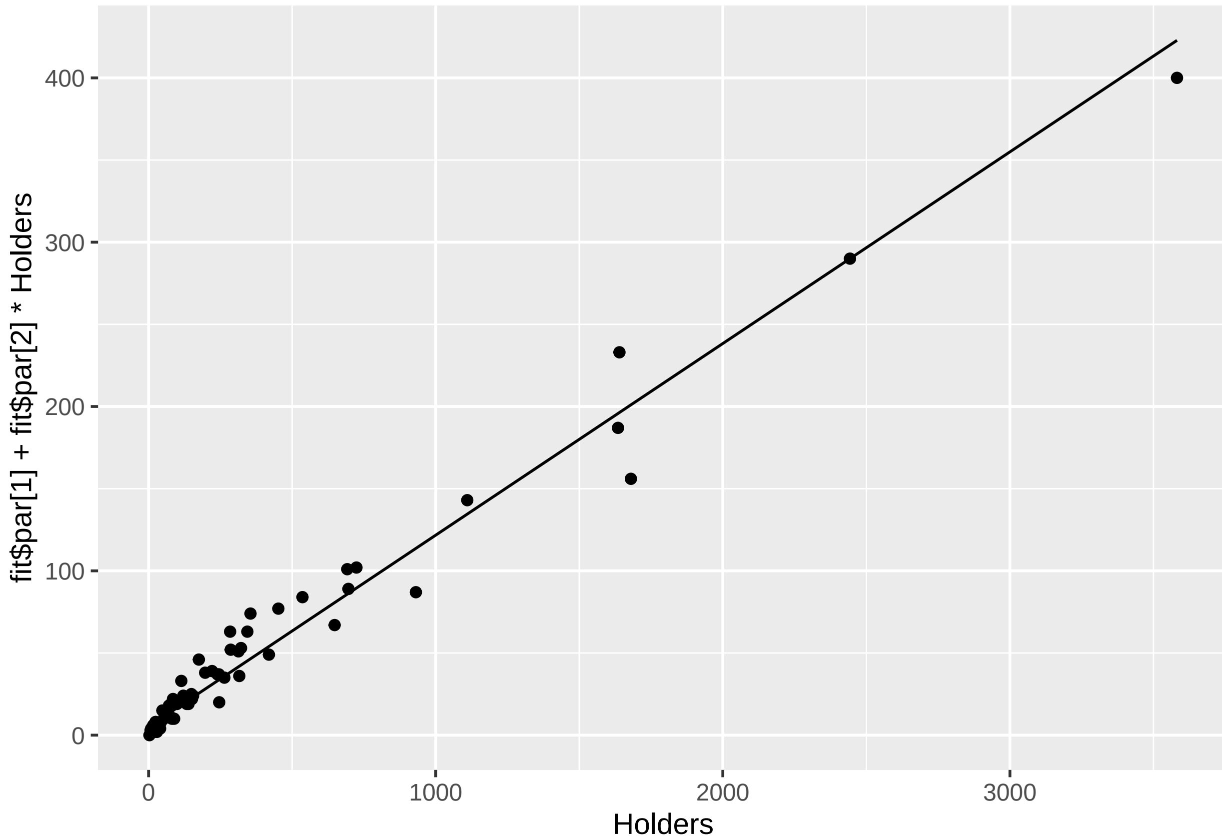
```
## [1] 510.7587
```

Part B

(i)

```
dlaplace = function(x, mu, sigma) {
    exp(-abs(x - mu)/sigma)/(2 * sigma)
}


NegLogLikeB = function(theta, data) {
    sigma = theta[3]
    n = nrow(data)
    l = 0
    for (i in 1:n) {
        mu = theta[1] + theta[2] * data$Holders[i]
        l = l + log(dlaplace(data$Claims[i], mu, sigma))
        print(l)
    }
    return(-l)
}


theta_initial = c(0.01, 0.1, 10)
NegLogLikeB(theta_initial, Insurance)

fit = optim(theta_initial, NegLogLikeB, data = Insurance)
```

```
ggplot(data = Insurance) + geom_line(aes(Holders, fit$par[1] +
    fit$par[2] * Holders)) + geom_point(aes(Holders, Claims))
```




```
theta_hat = fit$par
theta_hat
```

```
## [1] 5.0843671 0.1166253 8.2060147
```

(ii)

```
BIC = 2 * NegLogLikeB(theta_hat, Insurance) + log(nrow(Insurance)) *
    3
```

```
BIC
```

```
## [1] 498.6869
```

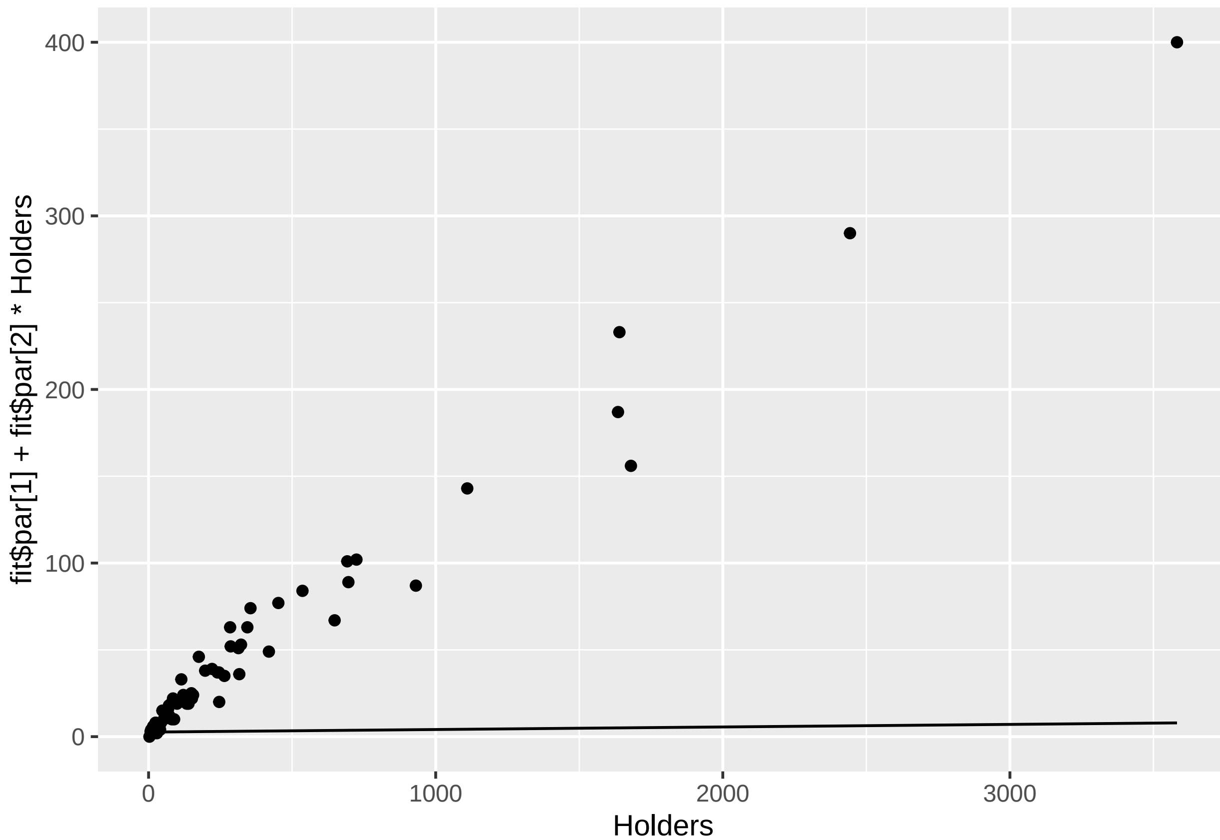
Part C

(i)

```
NegLogLikeC = function(theta, data) {
    sigma = theta[3]
    n = nrow(data)
    l = 0
    for (i in 1:n) {
        if (data$Claims[i] > 0) {
            mu = theta[1] + theta[2] * data$Holders[i]
            l = l + log(dlnorm(data$Claims[i], meanlog = mu,
                sdlog = sigma))
```

```
            print(l)
        }
    }
    return(-l)
}


theta_initial = c(1, 0, 1)
NegLogLikeC(theta_initial, Insurance)

fit = optim(theta_initial, NegLogLikeC, data = Insurance)
```

```
ggplot(data = Insurance) + geom_line(aes(Holders, fit$par[1] +
    fit$par[2] * Holders)) + geom_point(aes(Holders, Claims))
```




```
theta_hat = fit$par
theta_hat
```

```
## [1] 2.638505797 0.001474601 0.822601510
```

(ii)

```
BIC = 2 * NegLogLikeC(theta_hat, Insurance) + log(nrow(Insurance)) *
    3
```

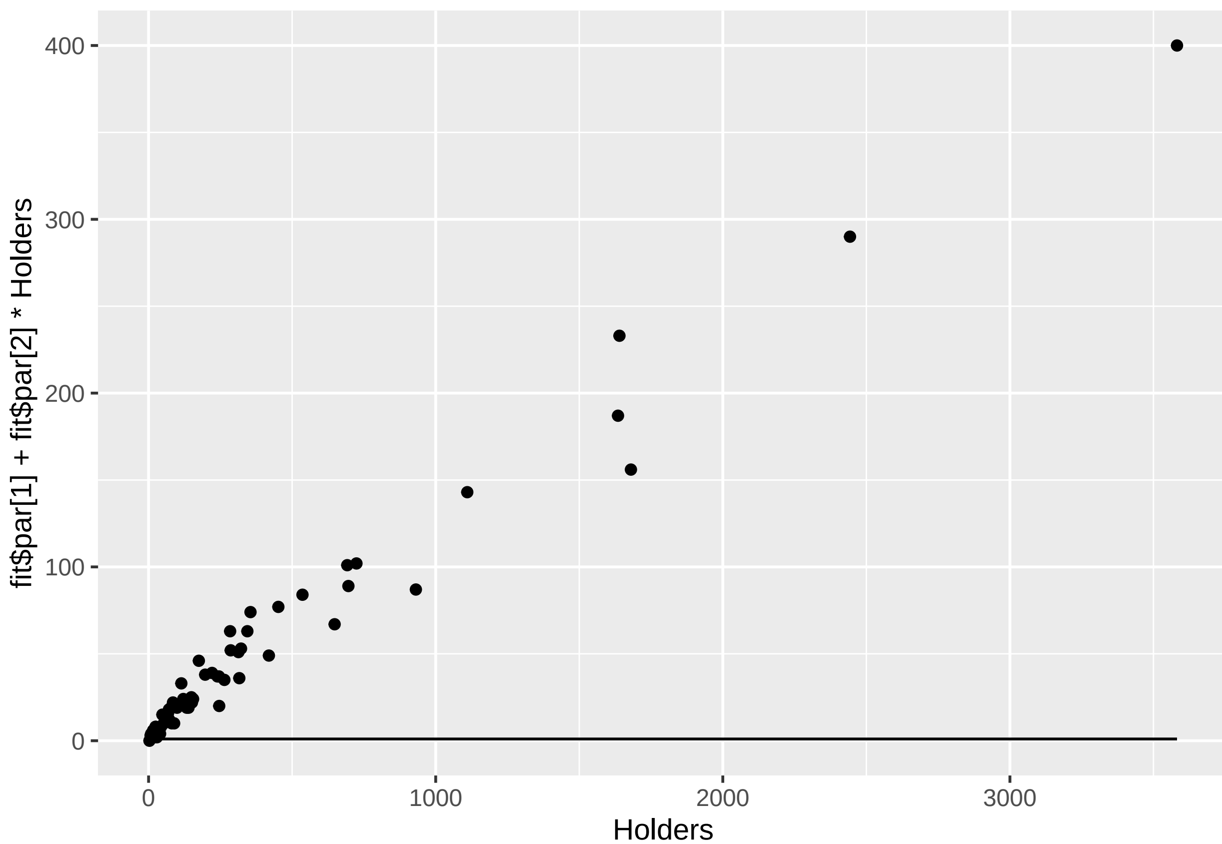
```
BIC
```

```
## [1] 568.0196
```

Part D

```r
NegLogLikeD = function(theta, data) {
    scale = theta[3]
    n = nrow(data)
    l = 0
    for (i in 1:n) {
        shape = theta[1] + theta[2] * data$Holders[i]
        l = l + log(dgamma(data$Claims[i], shape = shape, scale = scale))
        print(l)
    }
    return(-l)
}


theta_initial = c(1, 0, 1)
NegLogLikeD(theta_initial, Insurance)

fit = optim(theta_initial, NegLogLikeD, data = Insurance)
```

```r
ggplot(data = Insurance) + geom_line(aes(Holders, fit$par[1] +
    fit$par[2] * Holders)) + geom_point(aes(Holders, Claims))
```



```r
theta_hat = fit$par
theta_hat
```

```
## [1] 1.000000e+00 2.960595e-17 1.100000e+00
```

```r
BIC = 2 * NegLogLikeD(theta_hat, Insurance) + log(nrow(Insurance)) *
    3
```

```r
BIC
```

```
## [1] 5753.767
```

Comparing BIC of all models

Comparing BIC of all the models, we see that BIC of model 2 is least.
So, we conclude that model 2(Laplace Distribution) is the best fit.

Problem 5

```r
library(tidyverse)
library(quantmod)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
##
## Attaching package: 'xts'
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, last
```

```
## Loading required package: TTR
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
getSymbols("TCS.NS")
```

```
## [1] "TCS.NS"
```

```r
tail(TCS.NS)
```

```
##            TCS.NS.Open TCS.NS.High TCS.NS.Low TCS.NS.Close TCS.NS.Volume
## 2022-11-07      3229.0     3242.80    3195.10      3233.70       1474498
## 2022-11-09      3249.8     3249.80    3201.65      3216.05       1162267
## 2022-11-10      3170.0     3225.00    3170.00      3205.65       1573092
## 2022-11-11      3269.6     3341.60    3255.05      3315.95       3265394
## 2022-11-14      3324.0     3349.00    3309.00      3335.50       1342074
## 2022-11-15      3321.0     3339.95    3292.00      3332.60       1400708
```

```
##            TCS.NS.Adjusted
## 2022-11-07         3233.70
## 2022-11-09         3216.05
## 2022-11-10         3205.65
## 2022-11-11         3315.95
## 2022-11-14         3335.50
## 2022-11-15         3332.60
```

```
plot(TCS.NS$TCS.NS.Adjusted)
```


```
getSymbols("^NSEI")
```

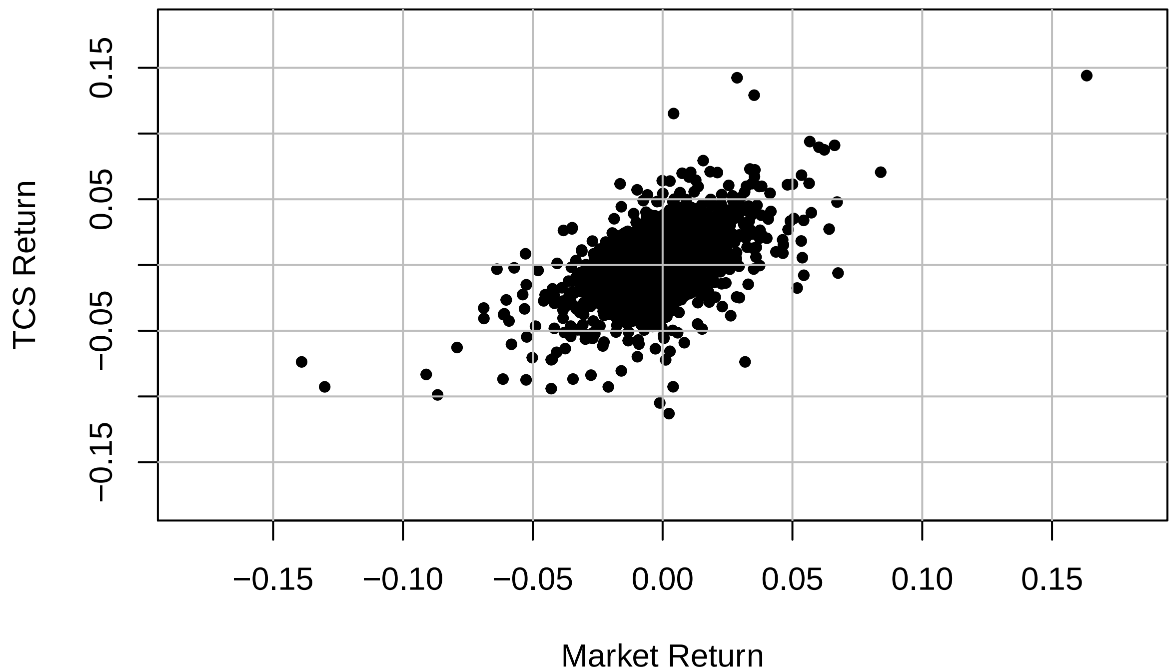
```
## [1] "^NSEI"
```

```
tail(NSEI)
```

```
##            NSEI.Open NSEI.High NSEI.Low NSEI.Close NSEI.Volume NSEI.Adjusted
## 2022-11-07  18211.75  18255.50 18064.75   18202.80      314800      18202.80
## 2022-11-09  18288.25  18296.40 18117.50   18157.00      307200      18157.00
## 2022-11-10  18044.35  18103.10 17969.40   18028.20      256500      18028.20
## 2022-11-11  18272.35  18362.30 18259.35   18349.70      378500      18349.70
## 2022-11-14  18376.40  18399.45 18311.40   18329.15      301400      18329.15
## 2022-11-15  18362.75  18427.95 18282.00   18403.40      250900      18403.40
```

```
plot(NSEI$NSEI.Adjusted)
```


```
TCS_rt = diff(log(TCS.NS$TCS.NS.Adjusted))
Nifty_rt = diff(log(NSEI$NSEI.Adjusted))
retrn = cbind.xts(TCS_rt, Nifty_rt)
retrn = na.omit(data.frame(retrn))
plot(retrn$NSEI.Adjusted, retrn$TCS.NS.Adjusted, pch = 20, xlab = "Market Return",
    ylab = "TCS Return", xlim = c(-0.18, 0.18), ylim = c(-0.18,
        0.18))
grid(col = "grey", lty = 1)
```



where $\mathbb{E}(\varepsilon) = 0$ and $\mathbb{V}ar(\varepsilon) = \sigma^2$.

Part 1 and 2

```
# For method of moment estimation, the equations are as
# follows: E(TCS)=alpha +beta*E(Nifty)
# E(Nifty(TCS-alpha-beta*Nifty))=0
# E(TCS-alpha-beta*Nifty)^2=sigma^2
attach(retrn)
tail(retrn)
```

```
##            TCS.NS.Adjusted NSEI.Adjusted
## 2022-11-07    0.0050534278   0.004716446
## 2022-11-09   -0.0054730636  -0.002519310
## 2022-11-10   -0.0032390663  -0.007119006
## 2022-11-11    0.0338292966   0.017676028
## 2022-11-14    0.0058784492  -0.001120473
## 2022-11-15   -0.0008697836   0.004042741
```

```
# creating a 2X2 coefficient matrix for solving of system
# of equations:
x <- array(0, 4)
x[1] = 1
x[2] <- mean(NSEI.Adjusted)
x[3] <- mean(NSEI.Adjusted)
x[4] <- mean((NSEI.Adjusted) * (NSEI.Adjusted))
A <- matrix(x, nrow = 2)
```

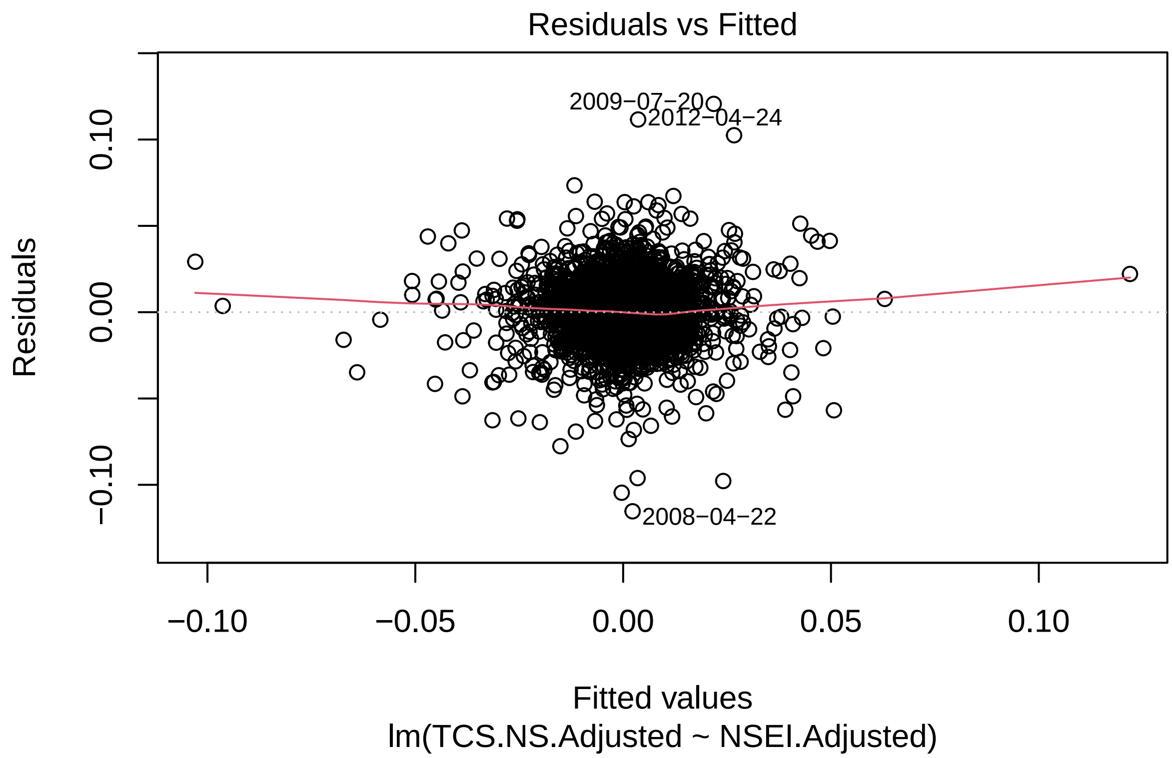
```r
# inserting another column containing TCS*NSE value named
# as 'C'
retrn <- retrn %>%
    mutate(C = TCS.NS.Adjusted * NSEI.Adjusted)
attach(retrn)
```

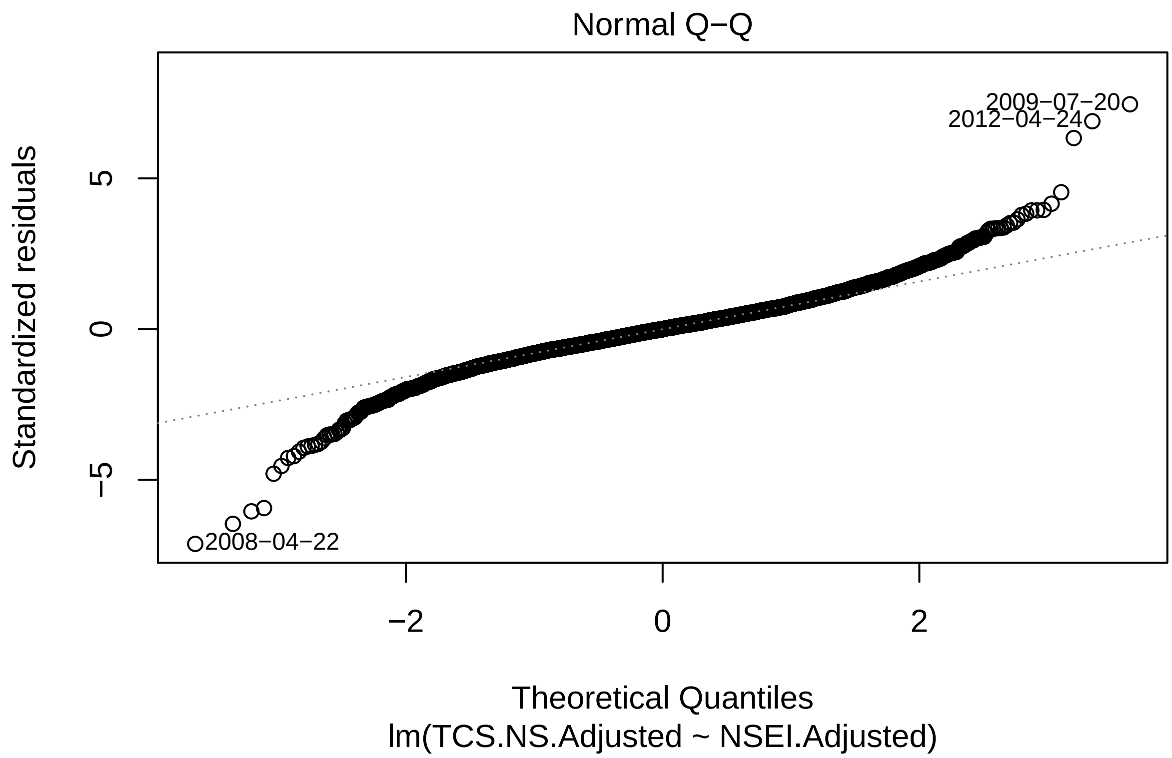
```
## The following objects are masked from retrn (pos = 3):
##
##     NSEI.Adjusted, TCS.NS.Adjusted
```

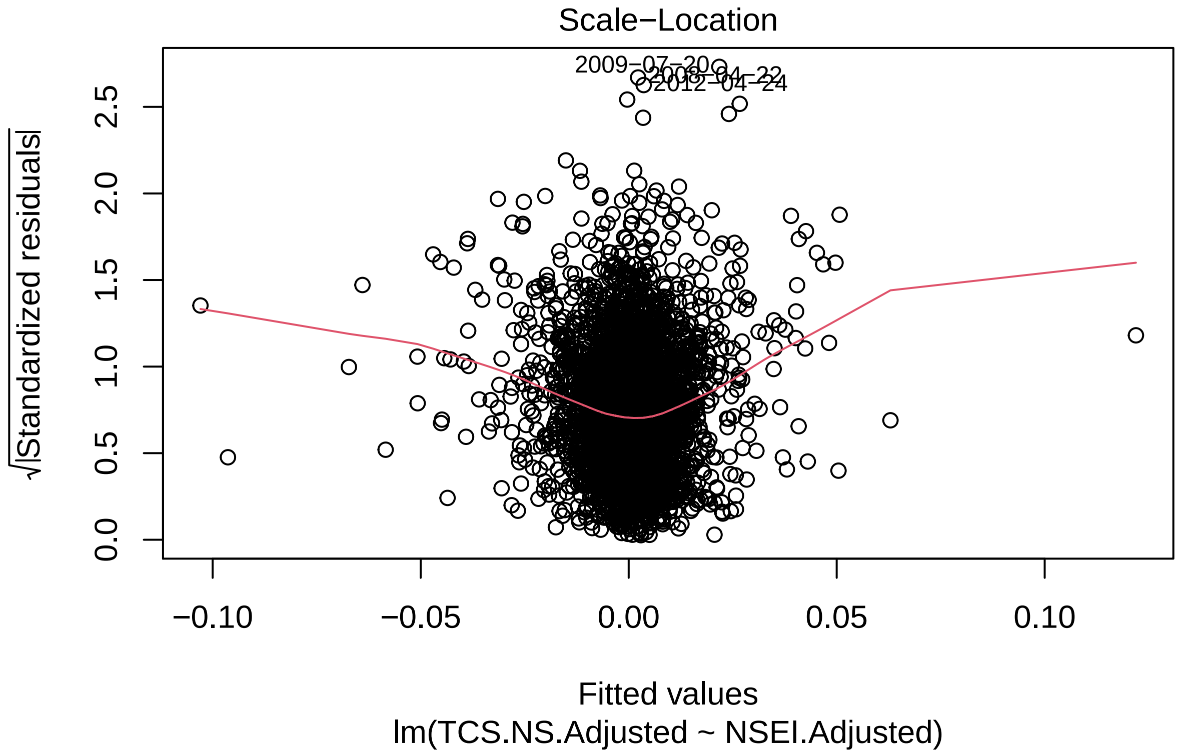
```r
# storing values in 2X1 matrix for solving of system of
# equations:
y <- array(0, 2)
y[1] <- mean(TCS.NS.Adjusted)
y[2] <- mean(C)
B <- matrix(y, ncol = 1)
par_solution <- solve(A, B)

# Now estimating through OLS using lm function
model_ols <- lm(TCS.NS.Adjusted ~ NSEI.Adjusted, retrn)
summary(model_ols)
```

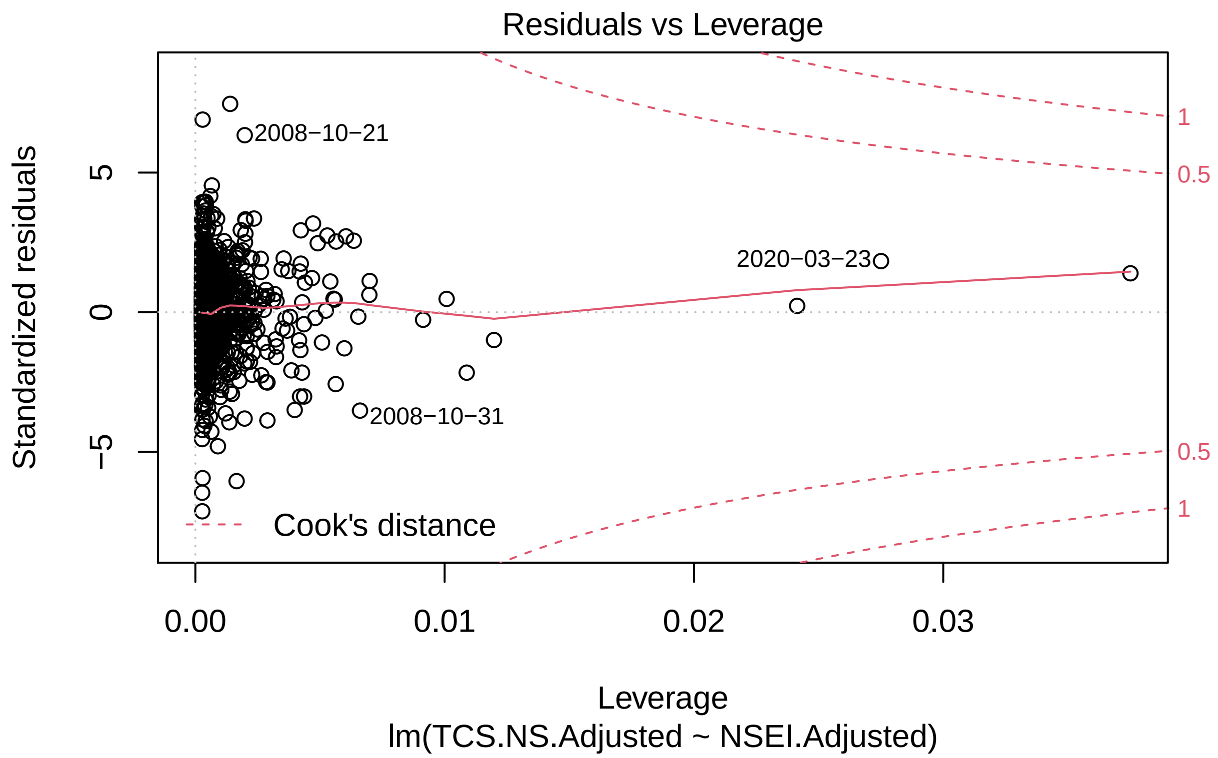
```
##
## Call:
## lm(formula = TCS.NS.Adjusted ~ NSEI.Adjusted, data = retrn)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.115338 -0.008756 -0.000086  0.008537  0.120641
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.0004617  0.0002668    1.73   0.0836 .
## NSEI.Adjusted 0.7436611  0.0191618   38.81   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01618 on 3681 degrees of freedom
## Multiple R-squared:  0.2904, Adjusted R-squared:  0.2902
## F-statistic:  1506 on 1 and 3681 DF,  p-value: < 2.2e-16
```

```r
matrix_coeff <- summary(model_ols)$coefficients
plot(model_ols)
```


Residuals vs Fitted
2009−07−20
2012−04−24
2008−04−22
Residuals
0.10
0.00
−0.10
−0.10
−0.05
0.00
0.05
0.10
Fitted values
lm(TCS.NS.Adjusted ~ NSEI.Adjusted)


Normal Q−Q
2009−07−20
2012−04−24
2008−04−22
Standardized residuals
5
0
−5
−2
0
2
Theoretical Quantiles
lm(TCS.NS.Adjusted ~ NSEI.Adjusted)


Scale−Location
2009−07−20
2008−04−22
2012−04−24
2.5
2.0
1.5
1.0
0.5
0.0
√|Standardized residuals|
−0.10
−0.05
0.00
0.05
0.10
Fitted values
lm(TCS.NS.Adjusted ~ NSEI.Adjusted)




```
x <- var(model_ols$residuals)^0.5

# estimating sigma^2
sigma_2 <- mean((TCS.NS.Adjusted - par_solution[1, 1] - par_solution[2,
    1] * NSEI.Adjusted) * (TCS.NS.Adjusted - par_solution[1,
    1] - par_solution[2, 1] * NSEI.Adjusted))

# unbiased estimator of sigma_2
usigma_2 <- sigma_2 * (length(TCS.NS.Adjusted)/(length(TCS.NS.Adjusted) -
    1))
usigma <- usigma_2^0.5

Parameter_List <- data.frame(Parameters = c("alpha", "beta",
    "sigma"), MoM = c(par_solution[1, 1], par_solution[2, 1],
    usigma_2^0.5), OLS = c(model_ols$coefficients[1], model_ols$coefficients[2],
    var(model_ols$residuals)^0.5))
```

```
# The data frame consisting of all the estimated parameters
Parameter_List
```

```
##                 Parameters          MoM          OLS
## (Intercept)          alpha 0.0004616529 0.0004616529
## NSEI.Adjusted         beta 0.7436610936 0.7436610936
##                      sigma 0.0161826166 0.0161826166
```

Part 3

Parameters	Method of Moments	OLS
α	4.6165292×10^{-4}	4.6165292×10^{-4}
β	0.7436611	0.7436611
σ	0.0161826	0.0161826

```
n_ft <- c(log(18200) - log(18000))
n_tcs <- predict(model_ols, newdata = data.frame(NSEI.Adjusted = n_ft))
New_value <- exp(n_tcs + log(3200))
```

Part 4

The predicted price of TCS is 3227.8936249

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.