Homework 2

Jieqi Tu

9/28/2020

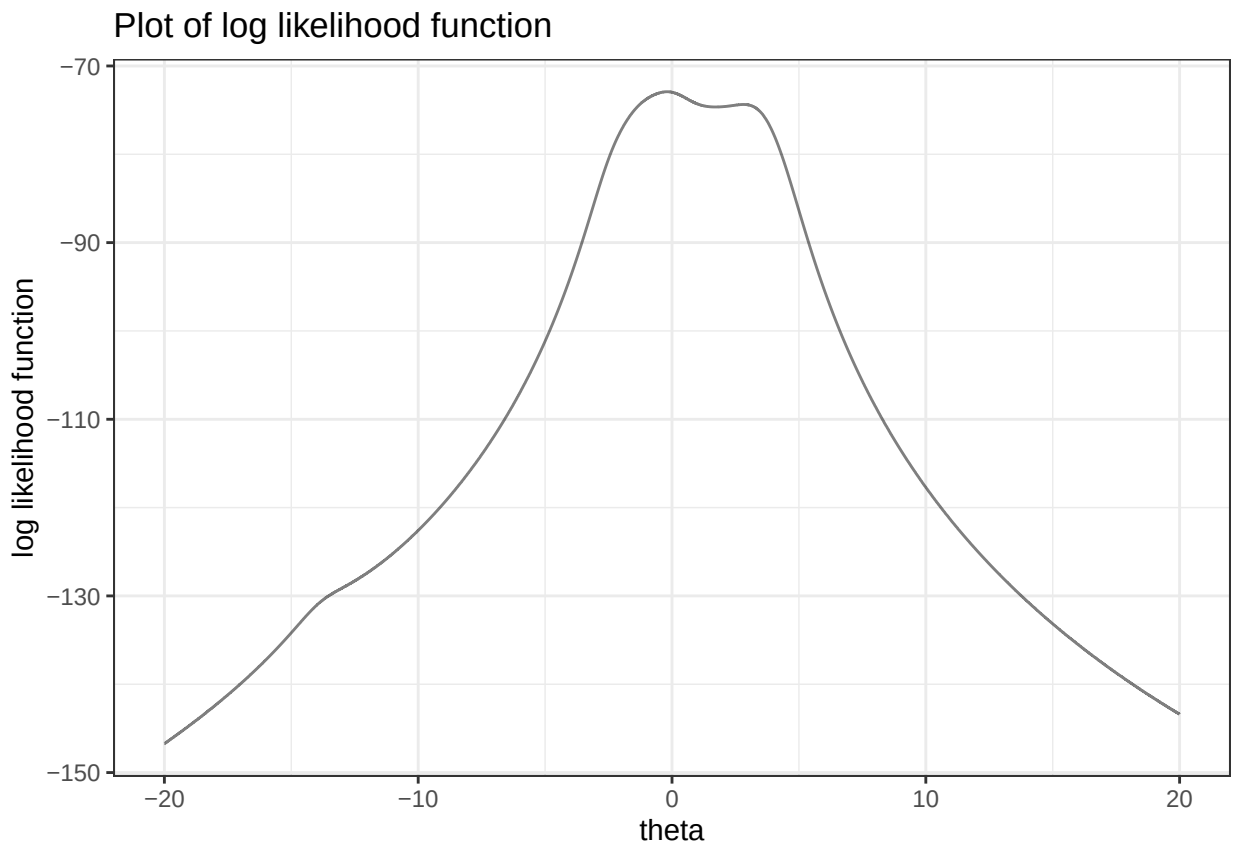
Question 1

(a)

Plot the log likelihood function.

```
# Import data
x_value = c(1.77, -0.23, 2.76, 3.80, 3.47,
            56.75, -1.34, 4.24, -2.44, 3.29,
            3.71, -2.40, 4.53, -0.07, -1.05,
            -13.87, -2.53, -1.75, 0.27, 43.21)

# Define the log likelihood function
log_likelihood = function(x, theta) {
  l = -20*log(pi) - sum(log(1+(x-theta)^2))
  return(l)
}

# Define the first derivative function of the log likelihood function
l_derivative_1 = function(x, theta) {
  l_1 = 2*sum((x-theta)/(1+(x-theta)^2))
  return(l_1)
}

# Define the second derivative function of the log likelihood function
l_derivative_2 = function(x, theta) {
  l_2 = 2*sum(((x-theta)^2-1)/(1+(x-theta)^2)^2)
  return(l_2)
}

# Graph the log likelihood function
theta_range = seq(-20, 20, 0.001)
n_loop = length(theta_range)
log_likelihood_value = numeric(n_loop)
for (i in 1:n_loop) {
  log_likelihood_value[i] = log_likelihood(x_value, theta_range[i])
}
result_l = cbind(theta_range, log_likelihood_value) %>% as.data.frame()
result_l %>%
  ggplot(aes(x = theta_range, y = log_likelihood_value)) + geom_line(alpha = 0.5) +
  theme_bw() + labs(
    x = "theta",
```

```
    y = "log likelihood function",
    title = "Plot of log likelihood function"
  )
```



Find the MLE for theta using Newton-Raphson method.

```
n_iteration = 1000
theta_iteration = numeric(n_iteration)
theta_iteration[1] = mean(x_value)
# Mean of the data as the starting point
for (i in 2:n_iteration) {
  theta_iteration[i] = theta_iteration[i-1] - l_derivative_1(x_value, theta_iteration[i-1])/l_derivative
}
tail(theta_iteration)
```

```
## [1] 54.87662 54.87662 54.87662 54.87662 54.87662 54.87662
```

```
# Starting point = -11
theta_iteration[1] = -11
for (i in 2:n_iteration) {
  theta_iteration[i] = theta_iteration[i-1] - l_derivative_1(x_value, theta_iteration[i-1])/l_derivative
}
tail(theta_iteration)
```

```
## [1] NaN NaN NaN NaN NaN NaN
```

```r
# Starting point = -1
theta_iteration[1] = -1
for (i in 2:n_iteration) {
  theta_iteration[i] = theta_iteration[i-1] - l_derivative_1(x_value, theta_iteration[i-1])/l_derivative
}
tail(theta_iteration)
```

```
## [1] -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866
```

```r
# Starting point = 0
theta_iteration[1] = 0
for (i in 2:n_iteration) {
  theta_iteration[i] = theta_iteration[i-1] - l_derivative_1(x_value, theta_iteration[i-1])/l_derivative
}
tail(theta_iteration)
```

```
## [1] -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866
```

```r
# Starting point = 1.5
theta_iteration[1] = 1.5
for (i in 2:n_iteration) {
  theta_iteration[i] = theta_iteration[i-1] - l_derivative_1(x_value, theta_iteration[i-1])/l_derivative
}
tail(theta_iteration)
```

```
## [1] 1.713587 1.713587 1.713587 1.713587 1.713587 1.713587
```

```r
# Starting point = 4
theta_iteration[1] = 4
for (i in 2:n_iteration) {
  theta_iteration[i] = theta_iteration[i-1] - l_derivative_1(x_value, theta_iteration[i-1])/l_derivative
}
tail(theta_iteration)
```

```
## [1] 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472
```

```r
# Starting point = 4.7
theta_iteration[1] = 4.7
for (i in 2:n_iteration) {
  theta_iteration[i] = theta_iteration[i-1] - l_derivative_1(x_value, theta_iteration[i-1])/l_derivative
}
tail(theta_iteration)
```

```
## [1] -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866
```

```r
# Starting point = 7
theta_iteration[1] = 7
for (i in 2:n_iteration) {
  theta_iteration[i] = theta_iteration[i-1] - l_derivative_1(x_value, theta_iteration[i-1])/l_derivative
}
tail(theta_iteration)
```

```
## [1] 41.04085 41.04085 41.04085 41.04085 41.04085 41.04085
```

```
# Starting point = 8
theta_iteration[1] = 8
for (i in 2:n_iteration) {
  theta_iteration[i] = theta_iteration[i-1] - l_derivative_1(x_value, theta_iteration[i-1])/l_derivative
}
tail(theta_iteration)
```

```
## [1] NaN NaN NaN NaN NaN NaN
```

```
# Starting point = 38
theta_iteration[1] = 38
for (i in 2:n_iteration) {
  theta_iteration[i] = theta_iteration[i-1] - l_derivative_1(x_value, theta_iteration[i-1])/l_derivative
}
tail(theta_iteration)
```

```
## [1] 42.79538 42.79538 42.79538 42.79538 42.79538 42.79538
```

Discussion: the true theta would be around 0 and a little bit less than 0 (from the plot). So from our results, starting point -1 and 0 did a great job. Newton-Raphson method is very sensitive to the starting points. The mean of data is not a good starting point. We should set the starting point close to what we guess from the plot.

(b)

Use Bisection method with starting points -1 and 1.

```
# Starting points = -1 and 1
L = -1
U = 1
n_iteration = 100000
theta_iteration = numeric(n_iteration)
for (i in 1:n_iteration) {
  theta_iteration[i] = (L + U)/2
  eval_1 = l_derivative_1(x_value, theta_iteration[i])
  eval_2 = l_derivative_1(x_value, L)
  eval_3 = eval_1 * eval_2
  if(eval_3 < 0) {
    U = theta_iteration[i]
  } else {L = theta_iteration[i]}
}

tail(theta_iteration)
```

```
## [1] -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866
```

```
# Starting points = 1 and 2
L = 1
U = 2
```

```
n_iteration = 100000
theta_iteration = numeric(n_iteration)
for (i in 1:n_iteration) {
  theta_iteration[i] = (L + U)/2
  eval_1 = l_derivative_1(x_value, theta_iteration[i])
  eval_2 = l_derivative_1(x_value, L)
  eval_3 = eval_1 * eval_2
  if(eval_3 < 0) {
    U = theta_iteration[i]
  } else {L = theta_iteration[i]}
}

tail(theta_iteration)
```

```
## [1] 1.713587 1.713587 1.713587 1.713587 1.713587 1.713587
```

```
# Starting points = 50 and 52
L = 50
U = 55
n_iteration = 100000
theta_iteration = numeric(n_iteration)
for (i in 1:n_iteration) {
  theta_iteration[i] = (L + U)/2
  eval_1 = l_derivative_1(x_value, theta_iteration[i])
  eval_2 = l_derivative_1(x_value, L)
  eval_3 = eval_1 * eval_2
  if(eval_3 < 0) {
    U = theta_iteration[i]
  } else {L = theta_iteration[i]}
}

tail(theta_iteration)
```

```
## [1] 54.87662 54.87662 54.87662 54.87662 54.87662 54.87662
```

Bisection method sometimes still fails to find the globlal maximum. Since it would be easier to get the local maximum near the starting points, sometimes it is possible to converge to a local maximum. Moreover, Bisection method might be slower than Newton-Raphson method.

(c)

Apply fixed-point iterations, starting from -1, with and without scaling.

```
# Define the g function
g_function = function(theta, alpha) {
  g_value = theta + alpha*l_derivative_1(x_value, theta)
  return(g_value)
}
# Without scaling (alpha = 1)
n_iteration = 2000000
theta_iteration = numeric(n_iteration)
theta_iteration[1] = -1
```

```
for (i in 2:n_iteration) {
  theta_iteration[i] = g_function(theta_iteration[i-1], 1)
}

tail(theta_iteration)
```

```
## [1] -0.8210256  0.6863265 -0.8584719  0.7297522 -0.7781577  0.6392481
```

```
# Alpha = 0.64
n_iteration = 100000
theta_iteration = numeric(n_iteration)
theta_iteration[1] = -1
for (i in 2:n_iteration) {
  theta_iteration[i] = g_function(theta_iteration[i-1], 0.64)
}

tail(theta_iteration)
```

```
## [1] -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866
```

```
# Alpha = 0.25
n_iteration = 100
theta_iteration = numeric(n_iteration)
theta_iteration[1] = -1
for (i in 2:n_iteration) {
  theta_iteration[i] = g_function(theta_iteration[i-1], 0.25)
}

tail(theta_iteration)
```

```
## [1] -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866
```

```
head(theta_iteration)
```

```
## [1] -1.0000000 -0.5196461 -0.2999846 -0.2198341 -0.1987656 -0.1937823
```

With the same starting point -1, the more shrinkage we have on the original function, the faster we can reach the convergence of theta.

```
# Alpha = 0.25, starting point = 0
n_iteration = 100
theta_iteration = numeric(n_iteration)
theta_iteration[1] = 0
for (i in 2:n_iteration) {
  theta_iteration[i] = g_function(theta_iteration[i-1], 0.25)
}

tail(theta_iteration)
```

```
## [1] -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866
```

```
theta_iteration
```

```
##   [1]  0.0000000 -0.1502570 -0.1829210 -0.1901521 -0.1917973 -0.1921743
##   [7] -0.1922608 -0.1922807 -0.1922853 -0.1922863 -0.1922865 -0.1922866
##  [13] -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866
##  [19] -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866
##  [25] -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866
##  [31] -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866
##  [37] -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866
##  [43] -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866
##  [49] -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866
##  [55] -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866
##  [61] -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866
##  [67] -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866
##  [73] -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866
##  [79] -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866
##  [85] -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866
##  [91] -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866 -0.1922866
##  [97] -0.1922866 -0.1922866 -0.1922866 -0.1922866
```

```
# Alpha = 0.25, starting point = 2
n_iteration = 100
theta_iteration = numeric(n_iteration)
theta_iteration[1] = 2
for (i in 2:n_iteration) {
  theta_iteration[i] = g_function(theta_iteration[i-1], 0.25)
}

tail(theta_iteration)
```

```
## [1] 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472
```

```
theta_iteration
```

```
##   [1] 2.000000 2.053837 2.116650 2.189510 2.273055 2.366643 2.466962 2.566709
##   [9] 2.655287 2.723247 2.767730 2.792994 2.805933 2.812156 2.815051 2.816375
##  [17] 2.816976 2.817248 2.817371 2.817427 2.817452 2.817463 2.817468 2.817470
##  [25] 2.817471 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472
##  [33] 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472
##  [41] 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472
##  [49] 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472
##  [57] 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472
##  [65] 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472
##  [73] 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472
##  [81] 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472
##  [89] 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472 2.817472
##  [97] 2.817472 2.817472 2.817472 2.817472
```

Having the same scaling, convergence could be totally different with different starting point. So the choice of starting point is very important, and should be close to the global maximum.

(d)

Apply the secant method to estimate theta.

```
# Define the secant function
secant_function = function(theta1, theta2) {
  f_xi = l_derivative_1(x_value, theta2)
  f_xi_1 = l_derivative_1(x_value, theta1)
  theta_new = theta2 - f_xi*(theta2-theta1)/(f_xi-f_xi_1)
  return(theta_new)
}

# Starting points = -2 and -1
n_iteration = 10
theta_iteration = numeric(n_iteration)
theta_iteration[1] = -2
theta_iteration[2] = -1
for (i in 3:n_iteration) {
  theta_iteration[i] = secant_function(theta_iteration[i-2], theta_iteration[i-1])
}
tail(theta_iteration)
```

```
## [1] -0.1984022 -0.1923655 -0.1922865 -0.1922866 -0.1922866 -0.1922866
```

```
# Starting points = -3 and 3
n_iteration = 10
theta_iteration = numeric(n_iteration)
theta_iteration[1] = -3
theta_iteration[2] = 3
for (i in 3:n_iteration) {
  theta_iteration[i] = secant_function(theta_iteration[i-2], theta_iteration[i-1])
}
tail(theta_iteration)
```

```
## [1] 2.826083 2.817013 2.817466 2.817472 2.817472 2.817472
```

```
# Starting points = 50 and 55
n_iteration = 10
theta_iteration = numeric(n_iteration)
theta_iteration[1] = 50
theta_iteration[2] = 55
for (i in 3:n_iteration) {
  theta_iteration[i] = secant_function(theta_iteration[i-2], theta_iteration[i-1])
}
tail(theta_iteration)
```

```
## [1] 54.87662 54.87662 54.87662 54.87662 54.87662      NaN
```

We get to the true convergence when starting points = -2 and -1. When the starting points are -3 and 3, or 50 and 55, we got the local maximum. So the starting points should be chosen carefully.

(e)

Compare the speed and stability of these methods.

- Newton-Raphson method is usually the fatest method than others, but it is also very sensitive to the starting points.
- Fixed point method is the most stable one but it is slower, the scaling can ajust the speed to convergence.
- Bisection method is also stable and slower.
- Secant is also fast. However, after reaching convergence, the theta might becomes NaN, due to the zero in denominator.

Question 2

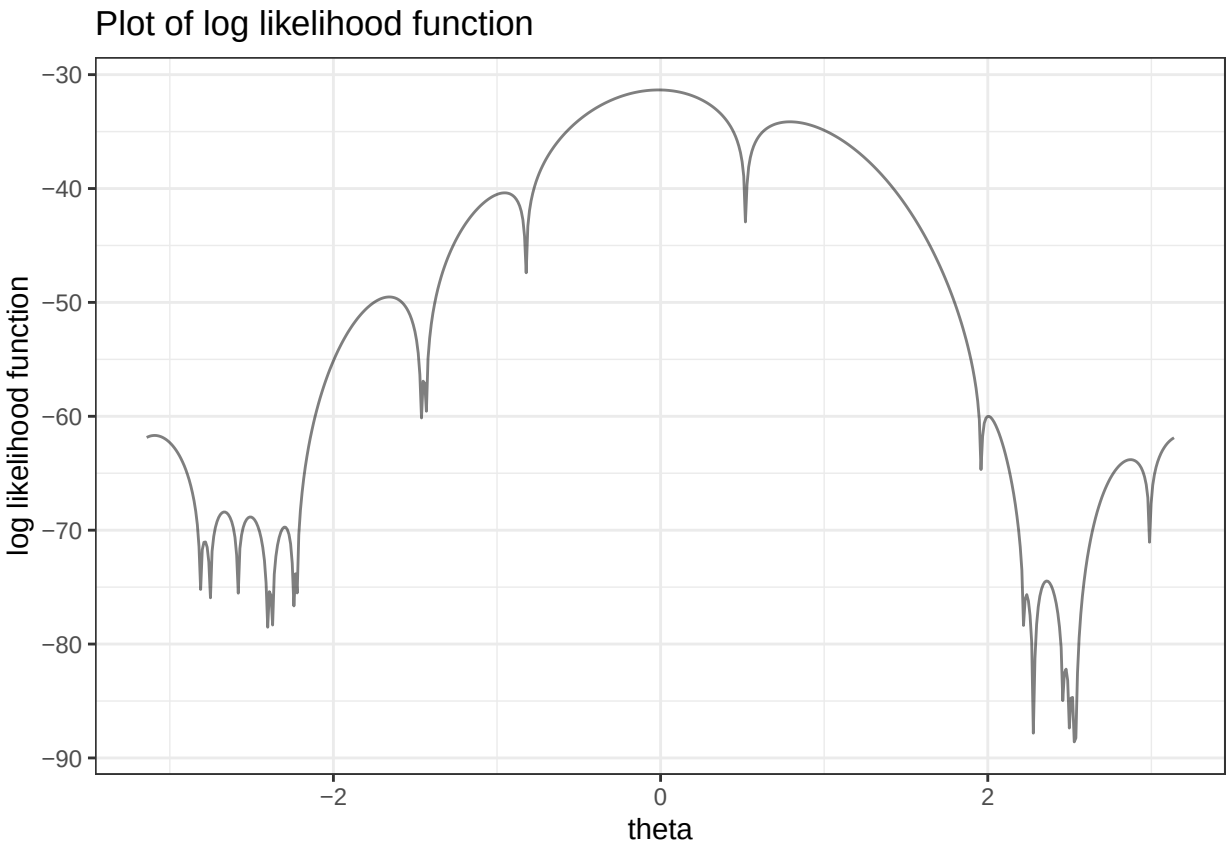
```
# Import data
x_observed = c(3.91, 4.85, 2.28, 4.06, 3.70, 4.04, 5.46, 3.53, 2.28, 1.96, 2.53, 3.88, 2.22, 3.47, 4.82
2.54, 0.52, 2.50)

# Define the log likelihood function
l_function = function(x, theta) {
  l_value = -20*log(2*pi) + sum(log(1-cos(x-theta)))
  return(l_value)
}

# Define the 1st derivative of the log likelihood function
l_derivative_1 = function(x, theta) {
  l_der_value = -sum(sin(x-theta)/(1-cos(x-theta)))
  return(l_der_value)
}

# Define the 2nd derivative of the log likelihood function
l_derivative_2 = function(x, theta) {
  l_second_der = -sum(1/(1-cos(x-theta)))
  return(l_second_der)
}
```

(a)

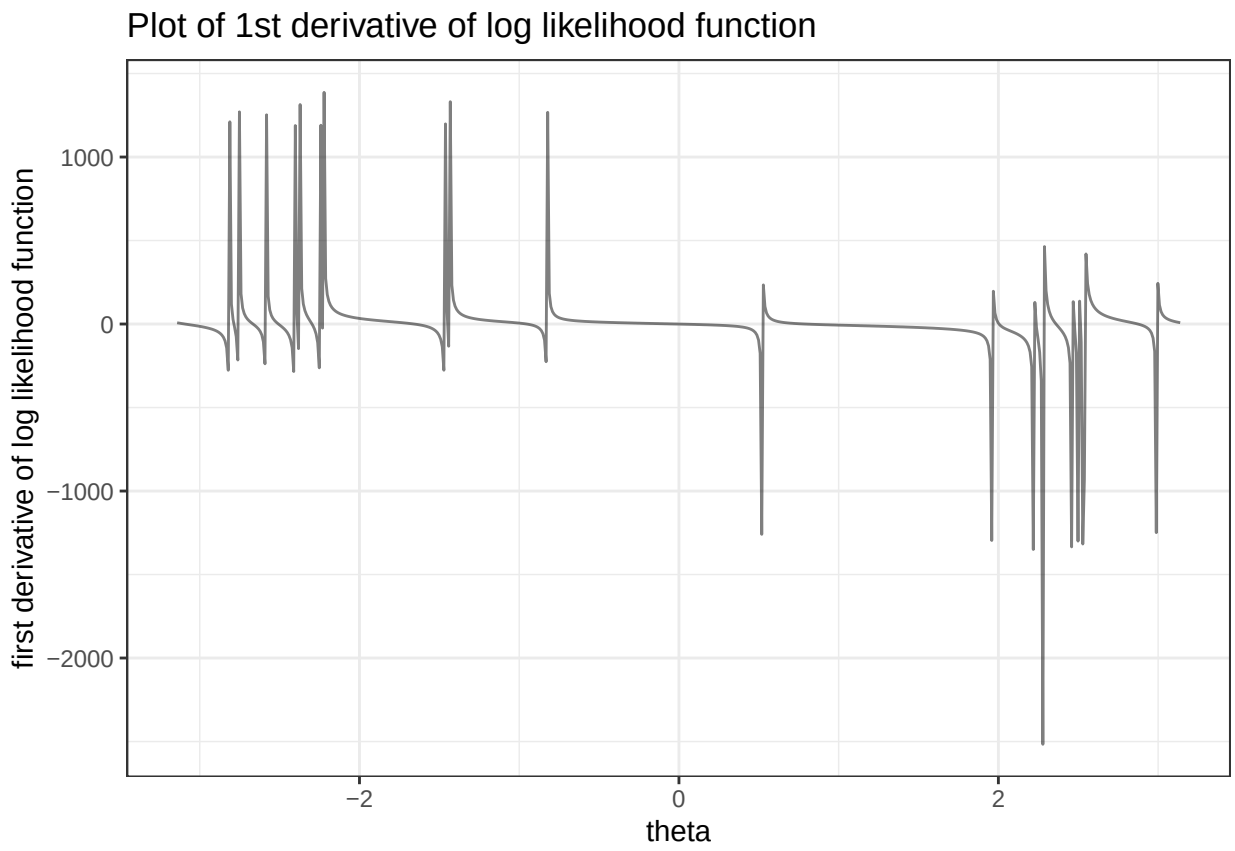
Plot the log likelihood function and the 1st derivative of the log likelihood function between $-\pi$ and π.

```
x_range = seq(-pi, pi, 0.01)
l_value = numeric(length(x_range))
l_derivative = numeric(length(x_range))
for (i in 1:length(x_range)) {
  l_value[i] = l_function(x_observed, x_range[i])
  l_derivative[i] = l_derivative_1(x_observed, x_range[i])
}
result = cbind(x_range, l_value, l_derivative) %>% as.data.frame()

# Plot the log likelihood function
result %>%
  ggplot(aes(x = x_range, y = l_value)) + geom_line(alpha= 0.5) +
```

```
  theme_bw() + labs(
    x = "theta",
    y = "log likelihood function",
    title = "Plot of log likelihood function"
  )
```




```
# Plot the derivative of log likelihood function
result %>%
  ggplot(aes(x = x_range, y = l_derivative)) + geom_line(alpha= 0.5) +
  theme_bw() + labs(
    x = "theta",
    y = "first derivative of log likelihood function",
    title = "Plot of 1st derivative of log likelihood function"
  )
```



(b)

Find the method-of-moments estimator of θ. The first theoretical moment about the origin is: $E(X) = \int_{-\infty}^{\infty} xf(x)dx = \int_0^{2\pi} x(1 - cos(x - \theta))/2\pi dx = \frac{1}{2\pi}(sin(\theta) + 2\pi^2)$. Therefore, we have $sin\theta = E(X) - \pi$. So the method-of-moments estimator would be $\hat{\theta} = \arcsin((E(X) - \pi))$.

```
# MME of theta
theta_mme = asin((mean(x_observed)-pi));theta_mme
```

```
## [1] 0.05844061
```

(c)

Find the MLE for θ using the Newton-Raphson method.

```
n_iteration = 100
theta_iteration = numeric(n_iteration)
theta_iteration[1] = theta_mme
# MME of theta as the starting point
for (i in 2:n_iteration) {
  theta_iteration[i] = theta_iteration[i-1] - l_derivative_1(x_observed, theta_iteration[i-1])/l_deriva
}
tail(theta_iteration)
```

```
## [1] -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972
```

```r
# Starting point = -2.7
theta_iteration = numeric(n_iteration)
theta_iteration[1] = -2.7
for (i in 2:n_iteration) {
  theta_iteration[i] = theta_iteration[i-1] - l_derivative_1(x_observed, theta_iteration[i-1])/l_deriva
}
tail(theta_iteration)
```

```
## [1] -2.6667 -2.6667 -2.6667 -2.6667 -2.6667 -2.6667
```

```r
# Starting point = 2.7
theta_iteration = numeric(n_iteration)
theta_iteration[1] = 2.7
for (i in 2:n_iteration) {
  theta_iteration[i] = theta_iteration[i-1] - l_derivative_1(x_observed, theta_iteration[i-1])/l_deriva
}
tail(theta_iteration)
```

```
## [1] 2.873095 2.873095 2.873095 2.873095 2.873095 2.873095
```

When the starting value is MME, it converges to -0.011972. When the starting value is -2.7, it converges to -2.6667. When the starting value is 2.7, it converges to 2.873095.

(d)

Repeat part (c) using 200 equally spaced starting values between $-\pi$ and π.

```r
# Create starting values
start_values = seq(-pi, pi, length = 200)

# Newton-Raphson method for each starting values
convergence = numeric(200)
n_iteration = 100
theta_iteration = numeric(n_iteration)
for (j in 1:200) {
  # Set starting value
  theta_iteration[1] = start_values[j]
  # Find the convergence theta value
  for (i in 2:n_iteration) {
    theta_iteration[i] = theta_iteration[i-1] - l_derivative_1(x_observed, theta_iteration[i-1])/l_deri
  }
  convergence[j] = theta_iteration[100]
}

result_list = cbind(start_values, convergence)
result_list %>% knitr::kable()
```

start_values	convergence
-3.1415927	-3.0930917
-3.1100189	-3.0930917
-3.0784451	-3.0930917

start_values	convergence
-3.0468713	-3.0930917
-3.0152975	-3.0930917
-2.9837237	-3.0930917
-2.9521499	-3.0930917
-2.9205761	-3.0930917
-2.8890023	-3.0930917
-2.8574285	-3.0930917
-2.8258547	-3.0930917
-2.7942809	-2.7861668
-2.7627071	-2.7861668
-2.7311333	-2.6666999
-2.6995595	-2.6666999
-2.6679857	-2.6666999
-2.6364119	-2.6666999
-2.6048381	-2.6666999
-2.5732643	-2.5076132
-2.5416905	-2.5076132
-2.5101167	-2.5076132
-2.4785429	-2.5076132
-2.4469692	-2.5076132
-2.4153954	-2.5076132
-2.3838216	-2.3882005
-2.3522478	-2.2972562
-2.3206740	-2.2972562
-2.2891002	-2.2972562
-2.2575264	-2.2972562
-2.2259526	-2.2321673
-2.1943788	-1.6582832
-2.1628050	-1.6582832
-2.1312312	-1.6582832
-2.0996574	-1.6582832
-2.0680836	-1.6582832
-2.0365098	-1.6582832
-2.0049360	-1.6582832
-1.9733622	-1.6582832
-1.9417884	-1.6582832
-1.9102146	-1.6582832
-1.8786408	-1.6582832
-1.8470670	-1.6582832
-1.8154932	-1.6582832
-1.7839194	-1.6582832
-1.7523457	-1.6582832
-1.7207719	-1.6582832
-1.6891981	-1.6582832
-1.6576243	-1.6582832
-1.6260505	-1.6582832
-1.5944767	-1.6582832
-1.5629029	-1.6582832
-1.5313291	-1.6582832
-1.4997553	-1.6582832
-1.4681815	-1.6582832
-1.4366077	-1.4474788

start_values	convergence
-1.4050339	-0.9533363
-1.3734601	-0.9533363
-1.3418863	-0.9533363
-1.3103125	-0.9533363
-1.2787387	-0.9533363
-1.2471649	-0.9533363
-1.2155911	-0.9533363
-1.1840173	-0.9533363
-1.1524435	-0.9533363
-1.1208697	-0.9533363
-1.0892959	-0.9533363
-1.0577221	-0.9533363
-1.0261484	-0.9533363
-0.9945746	-0.9533363
-0.9630008	-0.9533363
-0.9314270	-0.9533363
-0.8998532	-0.9533363
-0.8682794	-0.9533363
-0.8367056	-0.9533363
-0.8051318	-0.0119720
-0.7735580	-0.0119720
-0.7419842	-0.0119720
-0.7104104	-0.0119720
-0.6788366	-0.0119720
-0.6472628	-0.0119720
-0.6156890	-0.0119720
-0.5841152	-0.0119720
-0.5525414	-0.0119720
-0.5209676	-0.0119720
-0.4893938	-0.0119720
-0.4578200	-0.0119720
-0.4262462	-0.0119720
-0.3946724	-0.0119720
-0.3630986	-0.0119720
-0.3315249	-0.0119720
-0.2999511	-0.0119720
-0.2683773	-0.0119720
-0.2368035	-0.0119720
-0.2052297	-0.0119720
-0.1736559	-0.0119720
-0.1420821	-0.0119720
-0.1105083	-0.0119720
-0.0789345	-0.0119720
-0.0473607	-0.0119720
-0.0157869	-0.0119720
0.0157869	-0.0119720
0.0473607	-0.0119720
0.0789345	-0.0119720
0.1105083	-0.0119720
0.1420821	-0.0119720
0.1736559	-0.0119720
0.2052297	-0.0119720

start_values	convergence
0.2368035	-0.0119720
0.2683773	-0.0119720
0.2999511	-0.0119720
0.3315249	-0.0119720
0.3630986	-0.0119720
0.3946724	-0.0119720
0.4262462	-0.0119720
0.4578200	-0.0119720
0.4893938	-0.0119720
0.5209676	0.7906013
0.5525414	0.7906013
0.5841152	0.7906013
0.6156890	0.7906013
0.6472628	0.7906013
0.6788366	0.7906013
0.7104104	0.7906013
0.7419842	0.7906013
0.7735580	0.7906013
0.8051318	0.7906013
0.8367056	0.7906013
0.8682794	0.7906013
0.8998532	0.7906013
0.9314270	0.7906013
0.9630008	0.7906013
0.9945746	0.7906013
1.0261484	0.7906013
1.0577221	0.7906013
1.0892959	0.7906013
1.1208697	0.7906013
1.1524435	0.7906013
1.1840173	0.7906013
1.2155911	0.7906013
1.2471649	0.7906013
1.2787387	0.7906013
1.3103125	0.7906013
1.3418863	0.7906013
1.3734601	0.7906013
1.4050339	0.7906013
1.4366077	0.7906013
1.4681815	0.7906013
1.4997553	0.7906013
1.5313291	0.7906013
1.5629029	0.7906013
1.5944767	0.7906013
1.6260505	0.7906013
1.6576243	0.7906013
1.6891981	0.7906013
1.7207719	0.7906013
1.7523457	0.7906013
1.7839194	0.7906013
1.8154932	0.7906013
1.8470670	0.7906013

start_values	convergence
1.8786408	0.7906013
1.9102146	0.7906013
1.9417884	0.7906013
1.9733622	2.0036449
2.0049360	2.0036449
2.0365098	2.0036449
2.0680836	2.0036449
2.0996574	2.0036449
2.1312312	2.0036449
2.1628050	2.0036449
2.1943788	2.0036449
2.2259526	2.2362194
2.2575264	2.2362194
2.2891002	2.3607182
2.3206740	2.3607182
2.3522478	2.3607182
2.3838216	2.3607182
2.4153954	2.3607182
2.4469692	2.3607182
2.4785429	2.4753736
2.5101167	2.5135932
2.5416905	2.8730945
2.5732643	2.8730945
2.6048381	2.8730945
2.6364119	2.8730945
2.6679857	2.8730945
2.6995595	2.8730945
2.7311333	2.8730945
2.7627071	2.8730945
2.7942809	2.8730945
2.8258547	2.8730945
2.8574285	2.8730945
2.8890023	2.8730945
2.9205761	2.8730945
2.9521499	2.8730945
2.9837237	2.8730945
3.0152975	3.1900936
3.0468713	3.1900936
3.0784451	3.1900936
3.1100189	3.1900936
3.1415927	3.1900936

Discussion: Different starting values have different convergence of theta. However, only starting values from -0.8051318 to 0.4893938 lead to the true maximum.

(e)

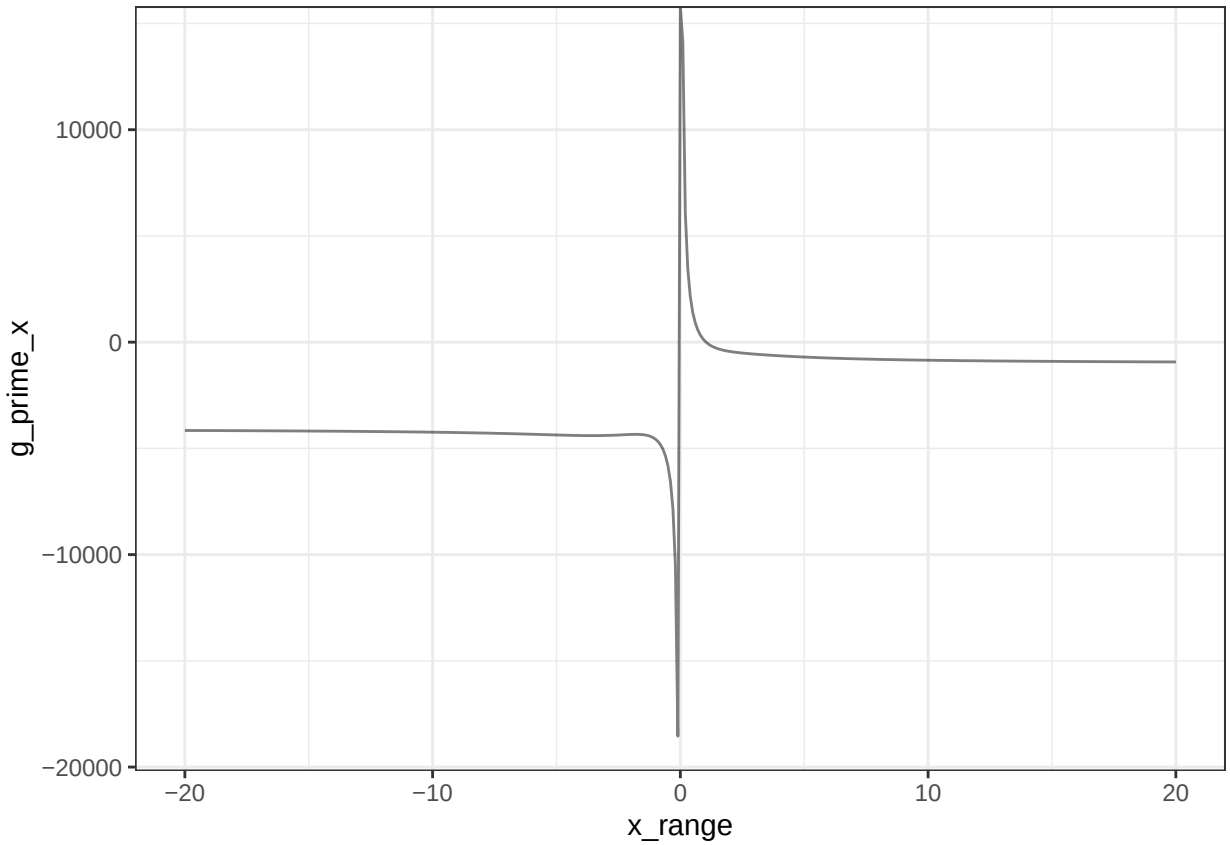
Find two starting values, as nearly equal as you can, for which the Newton–Raphson method converges to two different solutions. From part (d), we could know that, we can split the range of $[-\pi, \pi]$ as much as possible. We could assign the space as small as possible.

Question 3

(a)

Use fixed point iteration with $G_1(x)$.

```
# Define g'(x)
g_x_derivative = function(epsilon, x) {
  g_x_value = 1628/x - 1013 - 3062*(1-epsilon)*exp(-x)/(epsilon+(1-epsilon)*exp(-x))
  return(g_x_value)
}

# Define G1(x)
G1_x = function(epsilon, x) {
  G1_value = 1628*(epsilon+(1-epsilon)*exp(-x))/(3062*(1-epsilon)*exp(-x)+1013*(epsilon+(1-epsilon)*exp
  return(G1_value)
}

# Plot the g'(x)
x_range = seq(-20, 20, 0.1)
g_prime_x = numeric(length(x_range))
for (i in 1:length(x_range)) {
  g_prime_x[i] = g_x_derivative(0.61489, x_range[i])
}
result = cbind(x_range, g_prime_x) %>% as.data.frame()
result %>%
  ggplot(aes(x = x_range, y = g_prime_x)) + geom_line(alpha = 0.5) +
  theme_bw()
```




```
# Fixed point iteration with G1(x)
n_iteration = 20
x_list = numeric(n_iteration)
x_list = 1 # Starting value is 1
for (i in 2:n_iteration) {
  x_list[i] = G1_x(0.61489, x_list[i-1])
}
g_prime_x=g_x_derivative(0.61489, x_list)
result = cbind(x_list, g_prime_x) %>% as.data.frame()
result %>% knitr::kable()
```

x_list	g_prime_x
1.000000	41.6104171
1.026230	12.1233052
1.034132	3.6135223
1.036512	1.0843050
1.037228	0.3260186
1.037443	0.0980832
1.037508	0.0295138
1.037528	0.0088814
1.037534	0.0026726
1.037535	0.0008043
1.037536	0.0002420
1.037536	0.0000728
1.037536	0.0000219

x_list	g_prime_x
1.037536	0.0000066
1.037536	0.0000020
1.037536	0.0000006
1.037536	0.0000002
1.037536	0.0000001
1.037536	0.0000000
1.037536	0.0000000

Using fixed point iteration, $G_1(x)$ converges to 1.037536 with starting value 1.

(b)

Demonstrate that $G_2(x) = x + g'(x)$ fails to converge.

```
# Define G2(x)
G2_x = function(epsilon, x) {
  G2_value = x + 1628/x - 1013 - 3062*(1-epsilon)*exp(-x)/(epsilon+(1-epsilon)*exp(-x))
  return(G2_value)
}

# Fixed point iteration with G2(x)
n_iteration = 5
x_list = numeric(n_iteration)
x_list[1] = 1 # Starting value = 1
for (i in 2:n_iteration) {
  x_list[i] = G2_x(0.61489, x_list[i-1])
}
g_prime_x=g_x_derivative(0.61489, x_list)
result = cbind(x_list, g_prime_x) %>% as.data.frame()
result %>% knitr::kable()
```

x_list	g_prime_x
1.00000	41.61042
42.61042	-974.79338
-932.18296	NaN
NaN	NaN
NaN	NaN

The x values become NaN after a few iterations. So it is not convergent to some value with starting point 1.

(c)

Try $G_3(x) = x + \alpha g'(x), \alpha = 1/1000$.

```
# Define G3(x)
G3_x = function(epsilon, x, alpha) {
  G3_value = x + alpha*(1628/x - 1013 - 3062*(1-epsilon)*exp(-x)/(epsilon+(1-epsilon)*exp(-x)))
  return(G3_value)
}
```

```r
# Fixed point iteration with scaling
n_iteration = 10
x_list = numeric(n_iteration)
x_list[1] = 1
for (i in 2:n_iteration) {
  x_list[i] = G3_x(0.61489, x_list[i-1], 0.001)
}
g_prime_x=g_x_derivative(0.61489, x_list)
result = cbind(x_list, g_prime_x) %>% as.data.frame()
result %>% knitr::kable()
```

x_list	g_prime_x
1.000000	41.6104171
1.041610	-4.2858584
1.037325	0.2236357
1.037548	-0.0128569
1.037535	0.0007356
1.037536	-0.0000421
1.037536	0.0000024
1.037536	-0.0000001
1.037536	0.0000000
1.037536	0.0000000

After scaling with $\alpha = 0.001$, it converges to 1.037536.

(d)

Newton-Raphson method using starting value 3.

```r
# Define the second derivative of g(x)
g_x_derivative_2 = function(epsilon, x) {
  second_deri_value = 3062*(1-epsilon)*epsilon*exp(-x)/(epsilon+(1-epsilon)*exp(-x))^2-1628/x^2
  return(second_deri_value)
}

# Newton-Raphson iteration with starting value 3
n_iteration = 5
x_list = numeric(n_iteration)
x_list[1] = 3
for (i in 2:n_iteration) {
  x_list[i] = x_list[i-1] - g_x_derivative(0.61489, x_list[i-1])/g_x_derivative_2(0.61489, x_list[i-1])
}
g_prime_x=g_x_derivative(0.61489, x_list)
result = cbind(x_list, g_prime_x) %>% as.data.frame()
result %>% knitr::kable()
```

x_list	g_prime_x
3.000000	-562.9254
-3.179426	-4396.2872
243.487291	-1006.3138

x_list	g_prime_x
-36402.939062	NaN
NaN	NaN

(e)

Use Newton-Raphson method with starting value = 1.5.

```
n_iteration = 10
x_list = numeric(n_iteration)
x_list[1] = 1.5 # Starting value = 1.5
for (i in 2:n_iteration) {
  x_list[i] = x_list[i-1] - g_x_derivative(0.61489, x_list[i-1])/g_x_derivative_2(0.61489, x_list[i-1])
}
g_prime_x=g_x_derivative(0.61489, x_list)
result = cbind(x_list, g_prime_x) %>% as.data.frame()
result %>% knitr::kable()
```

x_list	g_prime_x
1.5000000	-303.1079151
0.7309801	504.7674436
0.9327450	126.9932930
1.0244014	14.1159457
1.0373228	0.2254716
1.0375360	0.0000597
1.0375360	0.0000000
1.0375360	0.0000000
1.0375360	0.0000000
1.0375360	0.0000000

(f)

Use secant method with starting value 1.5 and 1.49.

```
# Define the secant function
secant_function = function(x_1, x_2) {
  f_xi = g_x_derivative(0.61489, x_2)
  f_xi_1 = g_x_derivative(0.61489, x_1)
  x_new = x_2 - f_xi*(x_2-x_1)/(f_xi-f_xi_1)
  return(x_new)
}

# Secant method with starting value = 1.5 and 1.49
n_iteration = 10
x_list = numeric(n_iteration)
x_list[1] = 1.5
x_list[2] = 1.49
for (i in 3:n_iteration) {
  x_list[i] = secant_function(x_list[i-2], x_list[i-1])
}
g_prime_x=g_x_derivative(0.61489, x_list)
```

```
result = cbind(x_list, g_prime_x) %>% as.data.frame()
result %>% knitr::kable()
```

x_list	g_prime_x
1.5000000	-303.1079151
1.4900000	-299.1303221
0.7379615	487.4956992
1.2040223	-145.1702726
1.0970809	-58.5768213
1.0247396	13.7467313
1.0384897	-1.0070132
1.0375512	-0.0159851
1.0375360	0.0000189
1.0375360	0.0000000

(g)

Use Muller's method. Choose your third starting value as 1.48.

```
# Initialization
n_iteration = 10
x_list = numeric(n_iteration)
x_list[1] = 1.5
x_list[2] = 1.49
x_list[3] = 1.48

# Define Muller's function
muller_function = function(x1,x2,x3) {
  f_i_2 = g_x_derivative(0.61489, x1)
  f_i_1 = g_x_derivative(0.61489, x2)
  f_i = g_x_derivative(0.61489, x3)
  dd = (f_i - 2*f_i_1 + f_i_2)/(x3 - x1)
  s = f_i - f_i_1 + (x3 - x1)/dd
  q = (x3-x2)/(x2-x1)
  A = q*f_i - q*(1+q)*f_i_1 + q^2*f_i_2
  B = (2*q+1)*f_i-(1+q)^2*f_i_1+q^2*f_i_2
  C = (1+q)*f_i
  x4 = x3-(x3-x2)*2*C/(B+sign(s)*sqrt(B^2-4*A*C))
  return(x4)
}

# Iterations
for (i in 4:n_iteration) {
  x_list[i] = muller_function(x_list[i-3], x_list[i-2], x_list[i-1])
}
g_prime_x=g_x_derivative(0.61489, x_list)
result = cbind(x_list, g_prime_x) %>% as.data.frame()
result %>% knitr::kable()
```

x_list	g_prime_x
1.5000000	-303.1079151
1.4900000	-299.1303221
1.4800000	-295.0792741
0.9814546	63.6802061
1.0459032	-8.7548905
1.0377325	-0.2076829
1.0375362	-0.0001350
1.0375360	0.0000000
1.0375360	0.0000000
1.0375360	0.0000000

(h)

Use bisection method. The starting values would be 1 and 2.

```
L = 1
U = 2
n_iteration = 25
x_list = numeric(n_iteration)
for (i in 1:n_iteration) {
  x_list[i] = (L + U)/2
  eval_1 = g_x_derivative(0.61489, x_list[i])
  eval_2 = g_x_derivative(0.61489, L)
  eval_3 = eval_1 * eval_2
  if(eval_3 < 0) {
    U = x_list[i]
  } else {L = x_list[i]}
}

g_prime_x=g_x_derivative(0.61489, x_list)
result = cbind(x_list, g_prime_x) %>% as.data.frame()
result %>% knitr::kable()
```

x_list	g_prime_x
1.500000	-303.1079151
1.250000	-176.4526784
1.125000	-83.2880692
1.062500	-25.5957503
1.031250	6.6979962
1.046875	-9.7600619
1.039062	-1.6107709
1.035156	2.5234289
1.037109	0.4513145
1.038086	-0.5809779
1.037598	-0.0651446
1.037354	0.1930067
1.037476	0.0639115
1.037537	-0.0006214
1.037506	0.0316438
1.037521	0.0155109
1.037529	0.0074446

x_list	g_prime_x
1.037533	0.0034116
1.037535	0.0013951
1.037536	0.0003868
1.037536	-0.0001173
1.037536	0.0001347
1.037536	0.0000087
1.037536	-0.0000543
1.037536	-0.0000228

(i)

Use secant-bracket method. The starting values are 1 and 2.

```
n_iteration = 10
x_list = numeric(n_iteration)
L = 1 # x0
U = 2 # x1
for (i in 1:n_iteration) {
  x_list[i] = (L*g_x_derivative(0.61489, U)-U*g_x_derivative(0.61489, L))/(g_x_derivative(0.61489, U) -
  eval1 = g_x_derivative(0.61489, x_list[i])
  eval2 = g_x_derivative(0.61489, L)
  eval3 = eval1*eval2
  if (eval3<0) {
    U = x_list[i]
  } else {
    L = x_list[i]
  }
}
g_prime_x=g_x_derivative(0.61489, x_list)
result = cbind(x_list, g_prime_x) %>% as.data.frame()
result %>% knitr::kable()
```

x_list	g_prime_x
1.086712	-48.9740702
1.039832	-2.4199380
1.037642	-0.1123734
1.037541	-0.0052028
1.037536	-0.0002408
1.037536	-0.0000111
1.037536	-0.0000005
1.037536	0.0000000
1.037536	0.0000000
1.037536	0.0000000

(j)

Use Illinois method. Starting values = 1 and 2

```r
n_iteration = 10
x_list = numeric(n_iteration)
L = 1 # x0
U = 2 # x1
x_list[1] = U
FL = g_x_derivative(0.61489, L)
FU = g_x_derivative(0.61489, U)
for (i in 2:n_iteration) {
  x_list[i] = (L*FU - U*FL)/(FU-FL)
  eval = g_x_derivative(0.61489, x_list[i])*FL

  if(eval < 0) {
    U = x_list[i]
    FU = g_x_derivative(0.61489, U)
    if (FU*g_x_derivative(0.61489, x_list[i-1])>0) {
      FL = FL/2
    }
  } else {
    L = x_list[i]
    FL = g_x_derivative(0.61489, L)
    if (FL*g_x_derivative(0.61489, x_list[i-1])>0) {
      FU = FU/2
    }
  }
}
g_prime_x=g_x_derivative(0.61489, x_list)
result = cbind(x_list, g_prime_x) %>% as.data.frame()
result %>% knitr::kable()
```

x_list	g_prime_x
2.000000	-438.2595841
1.086712	-48.9740702
1.025854	12.5322208
1.038254	-0.7582987
1.037546	-0.0109875
1.037526	0.0106508
1.037536	-0.0000001
1.037536	0.0000000
1.037536	0.0000000
1.037536	0.0000000

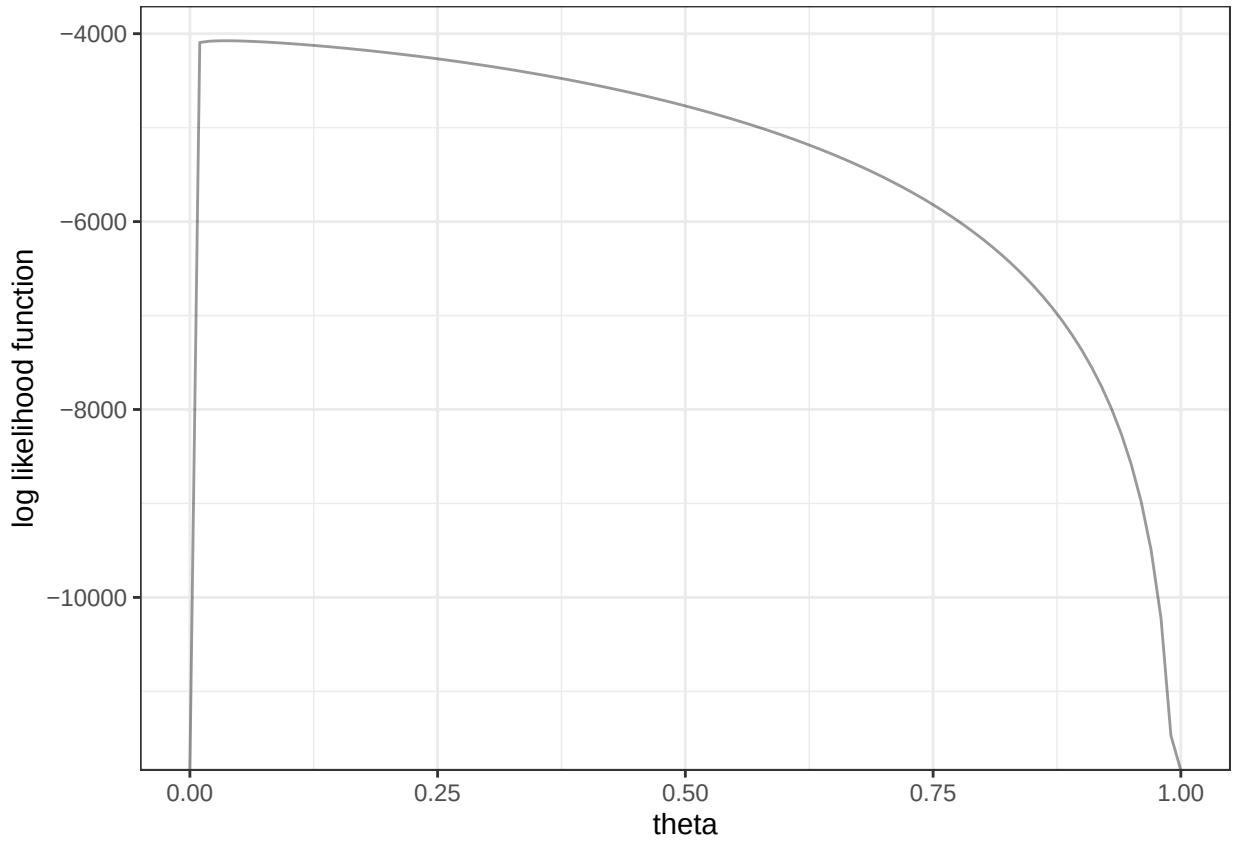
Question 4

(a)

```r
# Import data
m = c(1997, 906, 904, 32)
sample_size = sum(m); sample_size
```

```
## [1] 3839
```

```r
# Define the log likelihood function
log_L = function(theta) {
  log_L_value = m[1]*log((2+theta)/4)+(m[2]+m[3])*log((1-theta)/4)+m[4]*log(theta/4)
  return(log_L_value)
}

theta_range = seq(0, 1, 0.01)
log_L_plot_value = numeric(length(theta_range))
for (i in 1:length(theta_range)) {
  log_L_plot_value[i] = log_L(theta_range[i])
}
plot_variables = cbind(theta_range, log_L_plot_value) %>% as.data.frame()
# Plot the log likelihood function of theta
plot_variables %>%
  ggplot(aes(x = theta_range, y = log_L_plot_value)) + geom_line(alpha = 0.4) +
  theme_bw() + labs(
    x = "theta",
    y = "log likelihood function"
  )
```



From the plot, we could see that the global maximum is a bit greater than 0.

Use Newton-Raphsom method to estimate theta using unbiased estimate of theta.

```r
# Define the 1st derivative of log likelihood function
l_derivative_1 = function(theta) {
```

```
  derivative1 = 4*(m[1]/(2+theta) -(m[2]+m[3])/(1-theta) + m[4]/theta)
  return(derivative1)
}

# Define the 2nd derivative of log likelihood function
l_derivative_2 = function(theta) {
  derivative2 = 4*(-m[1]/(2+theta)^2-(m[2]+m[3])/(1-theta)^2-m[4]/theta^2)
  return(derivative2)
}

# Define the Fisher function
l_derivative_2_Fisher = function(theta) {
  m_expected = sample_size * c((2+theta)/4, (1-theta)/4, (1-theta)/4, theta/4)
  fisher = 4*(m_expected[1]/(2+theta)^2+(m_expected[2]+m_expected[3])/(1-theta)^2+m_expected[4]/theta^2)
  return(fisher)
}

n_iteration = 7
theta_list = numeric(n_iteration)
theta_list[1] = 0.0570461 # Starting value = 0.0570461
for (i in 2:n_iteration) {
  theta_list[i] = theta_list[i-1] - l_derivative_1(theta_list[i-1])/l_derivative_2(theta_list[i-1])
}
score_values = l_derivative_1(theta_list)
obs_Hessian = -l_derivative_2(theta_list)
exp_Hessian = l_derivative_2_Fisher(theta_list)
result = cbind(theta_list, score_values, obs_Hessian, exp_Hessian) %>% as.data.frame()
result %>% knitr::kable()
```

theta_list	score_values	obs_Hessian	exp_Hessian
0.0570461	-1550.9624374	49363.36	2420666
0.0256268	1507.8248540	204476.90	11917197
0.0330009	320.7743643	127208.22	7193932
0.0355225	20.9939854	111149.54	6211454
0.0357114	0.1010834	110082.04	6146125
0.0357123	0.0000024	110076.89	6145809
0.0357123	0.0000000	110076.89	6145809

```
# Calculate the standard error
var_NR = (exp_Hessian[n_iteration])^(-1)
std_error_NR = sqrt(var_NR);std_error_NR
```

```
## [1] 0.0004033764
```

Use Fisher scoring to estimate theta with the same starting value.

```
n_iteration = 10
theta_list = numeric(n_iteration)
theta_list[1] = 0.0570461 # Starting value = 0.0570461
for (i in 2:n_iteration) {
  theta_list[i] = theta_list[i-1] + l_derivative_1(theta_list[i-1])/l_derivative_2_Fisher(theta_list[i-
```

```
}
score_values = l_derivative_1(theta_list)
obs_Hessian = -l_derivative_2(theta_list)
exp_Hessian = l_derivative_2_Fisher(theta_list)
result = cbind(theta_list, score_values, obs_Hessian, exp_Hessian) %>% as.data.frame()
result %>% knitr::kable()
```

theta_list	score_values	obs_Hessian	exp_Hessian
0.0570461	-1550.9624374	49363.36	2420956
0.0369833	-135.5306571	103315.49	5732565
0.0357908	-8.6288050	109637.99	6119590
0.0357172	-0.5354540	110049.62	6144785
0.0357126	-0.0331734	110075.20	6146350
0.0357123	-0.0020550	110076.79	6146447
0.0357123	-0.0001273	110076.88	6146453
0.0357123	-0.0000079	110076.89	6146454
0.0357123	-0.0000005	110076.89	6146454
0.0357123	0.0000000	110076.89	6146454

```
# Calculate the standard error
var_fisher = (exp_Hessian[n_iteration])^(-1)
std_error_fisher = sqrt(var_fisher);std_error_fisher
```

```
## [1] 0.0004033552
```

(b)

Use Newton-Raphson method using starting value 0.5.

```
n_iteration = 10
theta_list = numeric(n_iteration)
theta_list[1] = 0.5 # Starting value = 0.5
for (i in 2:n_iteration) {
  theta_list[i] = theta_list[i-1] - l_derivative_1(theta_list[i-1])/l_derivative_2(theta_list[i-1])
}
score_values = l_derivative_1(theta_list)
obs_Hessian = -l_derivative_2(theta_list)
exp_Hessian = l_derivative_2_Fisher(theta_list)
result = cbind(theta_list, score_values, obs_Hessian, exp_Hessian) %>% as.data.frame()
result %>% knitr::kable()
```

theta_list	score_values	obs_Hessian	exp_Hessian
0.5000000	-1.102880e+04	30750.080	91719.94
0.1413408	-3.795764e+03	17969.043	369477.87
-0.0698983	-4.459588e+03	34667.669	1431732.00
-0.1985365	-2.251245e+03	10748.841	189292.00
-0.4079773	-4.383551e+02	7572.821	53221.26
-0.4658626	-6.995785e+00	7353.164	43217.97
-0.4668140	-1.219100e-03	7350.606	43083.84
-0.4668142	0.000000e+00	7350.606	43083.82

theta_list	score_values	obs_Hessian	exp_Hessian
-0.4668142	0.000000e+00	7350.606	43083.82
-0.4668142	0.000000e+00	7350.606	43083.82

It converges to a incorrect root, as the range for theta is from 0 to 1.

Use Fisher scoring method using starting value 0.5.

```
n_iteration = 11
theta_list = numeric(n_iteration)
theta_list[1] = 0.5 # Starting value = 0.5
for (i in 2:n_iteration) {
  theta_list[i] = theta_list[i-1] + l_derivative_1(theta_list[i-1])/l_derivative_2_Fisher(theta_list[i-
}
score_values = l_derivative_1(theta_list)
obs_Hessian = -l_derivative_2(theta_list)
exp_Hessian = l_derivative_2_Fisher(theta_list)
result = cbind(theta_list, score_values, obs_Hessian, exp_Hessian) %>% as.data.frame()
result %>% knitr::kable()
```

theta_list	score_values	obs_Hessian	exp_Hessian
0.5000000	-1.102880e+04	30750.08	95568.52
0.0511201	-1.231683e+03	58920.72	3010347.23
0.0366401	-9.977588e+01	105071.79	5840708.16
0.0357697	-6.306360e+00	109756.00	6127443.58
0.0357159	-3.911544e-01	110056.97	6145864.79
0.0357125	-2.423280e-02	110075.66	6147008.54
0.0357123	-1.501200e-03	110076.81	6147079.40
0.0357123	-9.300000e-05	110076.89	6147083.79
0.0357123	-5.800000e-06	110076.89	6147084.07
0.0357123	-4.000000e-07	110076.89	6147084.08
0.0357123	0.000000e+00	110076.89	6147084.08

Fisher scoring method can still converge to the correct root.

Question 5

Use Newton-Raphson method to find the estimated parameter.

```
# Import data
j = 1:4
y = c(76, 132, 15, 29)
n = c(20959, 12937, 6826, 4191)
x = matrix(c(
  1, 0, 0,
  1, 0, 1,
  1, 1, 0,
  1, 1, 1
), 4, 3, byrow = T)
```

```r
x = t(x) # Get the transpose of x

# Define the log likelihood function
log_L = function(beta) {
  l_beta = apply((y-n)*beta%*%x, 1, sum)-apply(log(1+exp(-beta%*%x)), 1, sum)
  return(l_beta)
}

# Define the 1st derivative of the log likelihood function
l_derivative_1 = function(beta) {
  p = 1/(1+exp(-beta%*%x))
  derivative1 = -(n*p-y)%*%t(x)
  return(derivative1)
}

l_derivative_2 = function(beta) {
  p = 1/(1+exp(-beta%*%x))
  derivative2 = -sum(n*p*(1-p))*x%*%t(x)
  return(derivative2)
}

n_iteration = 120
beta_list = matrix(0, n_iteration, 3)
beta_list[1, ] = c(-5, 0, 0) # set starting value = 5, 0, 0
for (i in 2:n_iteration) {
  beta_list[i,] = beta_list[i-1,] - l_derivative_1(beta_list[i-1,])%*%solve(l_derivative_2(beta_list[i-
  print(l_derivative_1(beta_list[i-1,]))
}
```

```
##           [,1]      [,2]     [,3]
## [1,] -48.59601 -29.73514 46.36485
##           [,1]      [,2]     [,3]
## [1,] -31.91467 -24.68092 36.60017
##           [,1]      [,2]     [,3]
## [1,] -24.24481 -21.26898 27.25883
##         [,1]      [,2]     [,3]
## [1,] -20.692 -18.80557 19.26021
##           [,1]      [,2]     [,3]
## [1,] -18.78684 -16.88365 12.93021
##           [,1]      [,2]     [,3]
## [1,] -17.41939 -15.28256 8.192931
##           [,1]      [,2]     [,3]
## [1,] -16.16511 -13.88796 4.787599
##           [,1]      [,2]     [,3]
## [1,] -14.90645 -12.64151 2.413327
##           [,1]     [,2]      [,3]
## [1,] -13.64319 -11.5128 0.7996979
##           [,1]      [,2]      [,3]
## [1,] -12.40755 -10.48485 -0.269898
##           [,1]      [,2]       [,3]
## [1,] -11.23116 -9.546907 -0.9582354
##           [,1]      [,2]      [,3]
## [1,] -10.13537 -8.691046 -1.383057
```

```
##           [,1]     [,2]      [,3]
## [1,] -9.130881 -7.91061 -1.627582
##           [,1]      [,2]      [,3]
## [1,] -8.220184 -7.199576 -1.749831
##           [,1]      [,2]      [,3]
## [1,] -7.400435 -6.552304 -1.789952
##           [,1]      [,2]      [,3]
## [1,] -6.665732 -5.963475 -1.775636
##           [,1]      [,2]      [,3]
## [1,] -6.008731 -5.428075 -1.725966
##           [,1]      [,2]     [,3]
## [1,] -5.421688 -4.941415 -1.65409
##          [,1]      [,2]      [,3]
## [1,] -4.89706 -4.499136 -1.569073
##           [,1]      [,2]     [,3]
## [1,] -4.427829 -4.097213 -1.47715
##           [,1]      [,2]      [,3]
## [1,] -4.007642 -3.731948 -1.382595
##           [,1]      [,2]      [,3]
## [1,] -3.630838 -3.399958 -1.288311
##          [,1]      [,2]      [,3]
## [1,] -3.29243 -3.098161 -1.196238
##           [,1]     [,2]      [,3]
## [1,] -2.988044 -2.82375 -1.107637
##           [,1]      [,2]      [,3]
## [1,] -2.713856 -2.574181 -1.023292
##           [,1]      [,2]       [,3]
## [1,] -2.466525 -2.347145 -0.9436457
##           [,1]      [,2]       [,3]
## [1,] -2.243129 -2.140552 -0.8689044
##           [,1]     [,2]       [,3]
## [1,] -2.041109 -1.95251 -0.7991074
##           [,1]      [,2]       [,3]
## [1,] -1.858217 -1.781305 -0.7341794
##           [,1]      [,2]      [,3]
## [1,] -1.692477 -1.625387 -0.673968
##           [,1]      [,2]       [,3]
## [1,] -1.542144 -1.483355 -0.6182703
##           [,1]      [,2]       [,3]
## [1,] -1.405674 -1.353939 -0.5668534
##           [,1]      [,2]       [,3]
## [1,] -1.281697 -1.235988 -0.5194675
##           [,1]      [,2]       [,3]
## [1,] -1.168995 -1.128463 -0.4758567
##           [,1]     [,2]       [,3]
## [1,] -1.066482 -1.03042 -0.4357657
##            [,1]      [,2]       [,3]
## [1,] -0.9731859 -0.941004 -0.3989451
##            [,1]       [,2]       [,3]
## [1,] -0.8882381 -0.8594396 -0.3651543
##            [,1]       [,2]      [,3]
## [1,] -0.8108581 -0.7850237 -0.334164
##            [,1]       [,2]       [,3]
## [1,] -0.7403446 -0.7171178 -0.3057573
```

```
##           [,1]       [,2]       [,3]
## [1,] -0.676066 -0.6551424 -0.2797304
##           [,1]       [,2]       [,3]
## [1,] -0.6174528 -0.5985708 -0.2558927
##           [,1]       [,2]       [,3]
## [1,] -0.5639907 -0.5469246 -0.2340667
##           [,1]       [,2]       [,3]
## [1,] -0.5152149 -0.4997687 -0.2140877
##           [,1]       [,2]       [,3]
## [1,] -0.4707044 -0.4567073 -0.1958031
##           [,1]       [,2]       [,3]
## [1,] -0.4300782 -0.4173805 -0.1790722
##           [,1]       [,2]       [,3]
## [1,] -0.3929904 -0.3814606 -0.1637651
##           [,1]       [,2]       [,3]
## [1,] -0.3591273 -0.3486492 -0.1497622
##           [,1]       [,2]       [,3]
## [1,] -0.3282039 -0.3186747 -0.1369536
##           [,1]       [,2]       [,3]
## [1,] -0.2999613 -0.2912893 -0.1252384
##           [,1]       [,2]       [,3]
## [1,] -0.2741639 -0.2662676 -0.1145239
##           [,1]       [,2]       [,3]
## [1,] -0.2505976 -0.2434039 -0.1047251
##           [,1]       [,2]      [,3]
## [1,] -0.2290671 -0.2225107 -0.095764
##           [,1]       [,2]       [,3]
## [1,] -0.209395 -0.2034171 -0.08756934
##           [,1]       [,2]       [,3]
## [1,] -0.1914192 -0.185967 -0.08007571
##           [,1]       [,2]       [,3]
## [1,] -0.1749925 -0.1700181 -0.07322327
##           [,1]       [,2]       [,3]
## [1,] -0.1599803 -0.1554407 -0.06695723
##           [,1]       [,2]       [,3]
## [1,] -0.1462599 -0.1421162 -0.06122747
##           [,1]       [,2]       [,3]
## [1,] -0.1337196 -0.1299364 -0.05598811
##           [,1]       [,2]       [,3]
## [1,] -0.1222572 -0.1188026 -0.05119722
##           [,1]       [,2]       [,3]
## [1,] -0.1117797 -0.1086246 -0.0468164
##           [,1]       [,2]       [,3]
## [1,] -0.102202 -0.09932006 -0.04281056
##             [,1]       [,2]       [,3]
## [1,] -0.09344661 -0.0908138 -0.03914759
##             [,1]       [,2]       [,3]
## [1,] -0.08544256 -0.08303711 -0.03579815
##             [,1]       [,2]       [,3]
## [1,] -0.07812519 -0.07592725 -0.03273539
##           [,1]       [,2]       [,3]
## [1,] -0.0714354 -0.0694269 -0.02993476
##             [,1]       [,2]       [,3]
## [1,] -0.06531922 -0.06348368 -0.02737381
```

```
##             [,1]        [,2]        [,3]
## [1,] -0.05972733 -0.05804974 -0.02503204
##             [,1]        [,2]        [,3]
## [1,] -0.05461469 -0.05308137 -0.02289066
##             [,1]        [,2]        [,3]
## [1,] -0.04994013 -0.04853859 -0.02093253
##             [,1]       [,2]        [,3]
## [1,] -0.04566604 -0.0443849 -0.01914195
##             [,1]        [,2]        [,3]
## [1,] -0.04175805 -0.04058691 -0.01750459
##             [,1]        [,2]        [,3]
## [1,] -0.03818475 -0.03711414 -0.01600732
##             [,1]        [,2]        [,3]
## [1,] -0.03491745 -0.03393868 -0.01463815
##             [,1]        [,2]        [,3]
## [1,] -0.03192988 -0.03103507 -0.01338612
##             [,1]     [,2]        [,3]
## [1,] -0.02919809 -0.02838 -0.01224121
##             [,1]        [,2]        [,3]
## [1,] -0.02670014 -0.02595217 -0.01119424
##           [,1]        [,2]        [,3]
## [1,] -0.024416 -0.02373213 -0.01023683
##             [,1]        [,2]         [,3]
## [1,] -0.02232735 -0.02170208 -0.009361328
##             [,1]        [,2]         [,3]
## [1,] -0.02041744 -0.01984573 -0.008560712
##             [,1]        [,2]         [,3]
## [1,] -0.01867097 -0.01814823 -0.007828579
##             [,1]        [,2]        [,3]
## [1,] -0.01707394 -0.01659596 -0.00715907
##             [,1]       [,2]         [,3]
## [1,] -0.01561355 -0.0151765 -0.006546825
##             [,1]        [,2]         [,3]
## [1,] -0.01427811 -0.01387848 -0.005986947
##             [,1]       [,2]         [,3]
## [1,] -0.01305692 -0.0126915 -0.005474955
##             [,1]        [,2]         [,3]
## [1,] -0.01194021 -0.01160606 -0.005006752
##             [,1]        [,2]         [,3]
## [1,] -0.01091902 -0.01061347 -0.004578593
##              [,1]         [,2]         [,3]
## [1,] -0.009985184 -0.009705787 -0.004187052
##             [,1]         [,2]         [,3]
## [1,] -0.00913123 -0.008875741 -0.003828996
##              [,1]         [,2]         [,3]
## [1,] -0.008350319 -0.008116692 -0.003501563
##              [,1]         [,2]         [,3]
## [1,] -0.007636203 -0.007422565 -0.003202131
##              [,1]         [,2]         [,3]
## [1,] -0.006983165 -0.006787806 -0.002928307
##              [,1]         [,2]       [,3]
## [1,] -0.006385982 -0.006207336 -0.0026779
##              [,1]         [,2]         [,3]
## [1,] -0.005839874 -0.005676511 -0.002448907
```

```
##              [,1]         [,2]         [,3]
## [1,] -0.005340473 -0.005191084 -0.002239497
##              [,1]         [,2]         [,3]
## [1,] -0.004883782 -0.004747172 -0.002047994
##              [,1]         [,2]         [,3]
## [1,] -0.004466148 -0.004341224 -0.001872868
##              [,1]         [,2]         [,3]
## [1,] -0.004084231 -0.003969992 -0.001712718
##              [,1]         [,2]         [,3]
## [1,] -0.003734976 -0.003630508 -0.001566263
##              [,1]         [,2]         [,3]
## [1,] -0.003415588 -0.003320055 -0.001432332
##              [,1]         [,2]         [,3]
## [1,] -0.003123514 -0.003036152 -0.001309854
##              [,1]         [,2]         [,3]
## [1,] -0.002856417 -0.002776527 -0.001197849
##              [,1]         [,2]         [,3]
## [1,] -0.002612161 -0.002539103 -0.001095421
##              [,1]         [,2]         [,3]
## [1,] -0.002388793 -0.002321983 -0.001001753
##              [,1]        [,2]         [,3]
## [1,] -0.002184526 -0.00212343 -0.000916094
##              [,1]         [,2]          [,3]
## [1,] -0.001997726 -0.001941855 -0.0008377599
##              [,1]         [,2]          [,3]
## [1,] -0.001826901 -0.001775808 -0.0007661242
##              [,1]         [,2]          [,3]
## [1,] -0.001670683 -0.001623959 -0.0007006141
##              [,1]         [,2]          [,3]
## [1,] -0.001527824 -0.001485096 -0.0006407057
##              [,1]         [,2]          [,3]
## [1,] -0.001397181 -0.001358107 -0.0005859201
##              [,1]         [,2]          [,3]
## [1,] -0.001277709 -0.001241976 -0.0005358192
##              [,1]         [,2]          [,3]
## [1,] -0.001168453 -0.001135777 -0.0004900023
##             [,1]         [,2]          [,3]
## [1,] -0.00106854 -0.001038658 -0.0004481032
##               [,1]          [,2]          [,3]
## [1,] -0.0009771711 -0.0009498439 -0.0004097869
##               [,1]          [,2]          [,3]
## [1,] -0.0008936147 -0.0008686244 -0.0003747469
##               [,1]          [,2]          [,3]
## [1,] -0.0008172032 -0.0007943499 -0.0003427032
##               [,1]          [,2]          [,3]
## [1,] -0.0007473257 -0.0007264265 -0.0003133994
##               [,1]          [,2]          [,3]
## [1,] -0.0006834233 -0.0006643113 -0.0002866014
##               [,1]          [,2]          [,3]
## [1,] -0.0006249851 -0.0006075074 -0.0002620948
##               [,1]          [,2]          [,3]
## [1,] -0.0005715439 -0.0005555608 -0.0002396837
```

```r
# Estimated beta
beta_est = round(beta_list, 2); beta_est
```

```
##          [,1]  [,2] [,3]
##   [1,] -5.00  0.00 0.00
##   [2,] -5.15 -0.02 0.24
##   [3,] -5.26 -0.05 0.42
##   [4,] -5.33 -0.08 0.57
##   [5,] -5.39 -0.11 0.68
##   [6,] -5.44 -0.14 0.76
##   [7,] -5.47 -0.17 0.82
##   [8,] -5.50 -0.19 0.87
##   [9,] -5.52 -0.21 0.91
##  [10,] -5.54 -0.23 0.94
##  [11,] -5.56 -0.24 0.96
##  [12,] -5.57 -0.26 0.98
##  [13,] -5.58 -0.27 0.99
##  [14,] -5.59 -0.28 1.00
##  [15,] -5.59 -0.30 1.01
##  [16,] -5.60 -0.31 1.02
##  [17,] -5.60 -0.32 1.02
##  [18,] -5.61 -0.33 1.03
##  [19,] -5.61 -0.33 1.03
##  [20,] -5.61 -0.34 1.04
##  [21,] -5.61 -0.35 1.04
##  [22,] -5.62 -0.36 1.04
##  [23,] -5.62 -0.36 1.04
##  [24,] -5.62 -0.37 1.05
##  [25,] -5.62 -0.37 1.05
##  [26,] -5.62 -0.38 1.05
##  [27,] -5.62 -0.38 1.05
##  [28,] -5.62 -0.39 1.05
##  [29,] -5.62 -0.39 1.05
##  [30,] -5.62 -0.39 1.05
##  [31,] -5.62 -0.40 1.05
##  [32,] -5.62 -0.40 1.05
##  [33,] -5.62 -0.40 1.06
##  [34,] -5.62 -0.41 1.06
##  [35,] -5.63 -0.41 1.06
##  [36,] -5.63 -0.41 1.06
##  [37,] -5.63 -0.41 1.06
##  [38,] -5.63 -0.41 1.06
##  [39,] -5.63 -0.41 1.06
##  [40,] -5.63 -0.42 1.06
##  [41,] -5.63 -0.42 1.06
##  [42,] -5.63 -0.42 1.06
##  [43,] -5.63 -0.42 1.06
##  [44,] -5.63 -0.42 1.06
##  [45,] -5.63 -0.42 1.06
##  [46,] -5.63 -0.42 1.06
##  [47,] -5.63 -0.42 1.06
##  [48,] -5.63 -0.42 1.06
##  [49,] -5.63 -0.42 1.06
```

```
##  [50,] -5.63 -0.42 1.06
##  [51,] -5.63 -0.43 1.06
##  [52,] -5.63 -0.43 1.06
##  [53,] -5.63 -0.43 1.06
##  [54,] -5.63 -0.43 1.06
##  [55,] -5.63 -0.43 1.06
##  [56,] -5.63 -0.43 1.06
##  [57,] -5.63 -0.43 1.06
##  [58,] -5.63 -0.43 1.06
##  [59,] -5.63 -0.43 1.06
##  [60,] -5.63 -0.43 1.06
##  [61,] -5.63 -0.43 1.06
##  [62,] -5.63 -0.43 1.06
##  [63,] -5.63 -0.43 1.06
##  [64,] -5.63 -0.43 1.06
##  [65,] -5.63 -0.43 1.06
##  [66,] -5.63 -0.43 1.06
##  [67,] -5.63 -0.43 1.06
##  [68,] -5.63 -0.43 1.06
##  [69,] -5.63 -0.43 1.06
##  [70,] -5.63 -0.43 1.06
##  [71,] -5.63 -0.43 1.06
##  [72,] -5.63 -0.43 1.06
##  [73,] -5.63 -0.43 1.06
##  [74,] -5.63 -0.43 1.06
##  [75,] -5.63 -0.43 1.06
##  [76,] -5.63 -0.43 1.06
##  [77,] -5.63 -0.43 1.06
##  [78,] -5.63 -0.43 1.06
##  [79,] -5.63 -0.43 1.06
##  [80,] -5.63 -0.43 1.06
##  [81,] -5.63 -0.43 1.06
##  [82,] -5.63 -0.43 1.06
##  [83,] -5.63 -0.43 1.06
##  [84,] -5.63 -0.43 1.06
##  [85,] -5.63 -0.43 1.06
##  [86,] -5.63 -0.43 1.06
##  [87,] -5.63 -0.43 1.06
##  [88,] -5.63 -0.43 1.06
##  [89,] -5.63 -0.43 1.06
##  [90,] -5.63 -0.43 1.06
##  [91,] -5.63 -0.43 1.06
##  [92,] -5.63 -0.43 1.06
##  [93,] -5.63 -0.43 1.06
##  [94,] -5.63 -0.43 1.06
##  [95,] -5.63 -0.43 1.06
##  [96,] -5.63 -0.43 1.06
##  [97,] -5.63 -0.43 1.06
##  [98,] -5.63 -0.43 1.06
##  [99,] -5.63 -0.43 1.06
## [100,] -5.63 -0.43 1.06
## [101,] -5.63 -0.43 1.06
## [102,] -5.63 -0.43 1.06
## [103,] -5.63 -0.43 1.06
```

```
## [104,] -5.63 -0.43 1.06
## [105,] -5.63 -0.43 1.06
## [106,] -5.63 -0.43 1.06
## [107,] -5.63 -0.43 1.06
## [108,] -5.63 -0.43 1.06
## [109,] -5.63 -0.43 1.06
## [110,] -5.63 -0.43 1.06
## [111,] -5.63 -0.43 1.06
## [112,] -5.63 -0.43 1.06
## [113,] -5.63 -0.43 1.06
## [114,] -5.63 -0.43 1.06
## [115,] -5.63 -0.43 1.06
## [116,] -5.63 -0.43 1.06
## [117,] -5.63 -0.43 1.06
## [118,] -5.63 -0.43 1.06
## [119,] -5.63 -0.43 1.06
## [120,] -5.63 -0.43 1.06
```

```r
# Score
for (i in 1:n_iteration) {
  print(l_derivative_1(beta_list[i,]))
}
```

```
##           [,1]      [,2]     [,3]
## [1,] -48.59601 -29.73514 46.36485
##           [,1]      [,2]     [,3]
## [1,] -31.91467 -24.68092 36.60017
##           [,1]      [,2]     [,3]
## [1,] -24.24481 -21.26898 27.25883
##         [,1]      [,2]     [,3]
## [1,] -20.692 -18.80557 19.26021
##           [,1]      [,2]     [,3]
## [1,] -18.78684 -16.88365 12.93021
##           [,1]      [,2]     [,3]
## [1,] -17.41939 -15.28256 8.192931
##           [,1]      [,2]     [,3]
## [1,] -16.16511 -13.88796 4.787599
##           [,1]      [,2]     [,3]
## [1,] -14.90645 -12.64151 2.413327
##           [,1]     [,2]      [,3]
## [1,] -13.64319 -11.5128 0.7996979
##           [,1]      [,2]      [,3]
## [1,] -12.40755 -10.48485 -0.269898
##           [,1]      [,2]       [,3]
## [1,] -11.23116 -9.546907 -0.9582354
##           [,1]      [,2]      [,3]
## [1,] -10.13537 -8.691046 -1.383057
##           [,1]     [,2]      [,3]
## [1,] -9.130881 -7.91061 -1.627582
##           [,1]      [,2]      [,3]
## [1,] -8.220184 -7.199576 -1.749831
##           [,1]      [,2]      [,3]
## [1,] -7.400435 -6.552304 -1.789952
##           [,1]      [,2]      [,3]
```

```
## [1,] -6.665732 -5.963475 -1.775636
##           [,1]      [,2]      [,3]
## [1,] -6.008731 -5.428075 -1.725966
##           [,1]      [,2]     [,3]
## [1,] -5.421688 -4.941415 -1.65409
##          [,1]      [,2]      [,3]
## [1,] -4.89706 -4.499136 -1.569073
##           [,1]      [,2]     [,3]
## [1,] -4.427829 -4.097213 -1.47715
##           [,1]      [,2]      [,3]
## [1,] -4.007642 -3.731948 -1.382595
##           [,1]      [,2]      [,3]
## [1,] -3.630838 -3.399958 -1.288311
##          [,1]      [,2]      [,3]
## [1,] -3.29243 -3.098161 -1.196238
##           [,1]     [,2]      [,3]
## [1,] -2.988044 -2.82375 -1.107637
##           [,1]      [,2]      [,3]
## [1,] -2.713856 -2.574181 -1.023292
##           [,1]      [,2]       [,3]
## [1,] -2.466525 -2.347145 -0.9436457
##           [,1]      [,2]       [,3]
## [1,] -2.243129 -2.140552 -0.8689044
##           [,1]     [,2]       [,3]
## [1,] -2.041109 -1.95251 -0.7991074
##           [,1]      [,2]       [,3]
## [1,] -1.858217 -1.781305 -0.7341794
##           [,1]      [,2]      [,3]
## [1,] -1.692477 -1.625387 -0.673968
##           [,1]      [,2]       [,3]
## [1,] -1.542144 -1.483355 -0.6182703
##           [,1]      [,2]       [,3]
## [1,] -1.405674 -1.353939 -0.5668534
##           [,1]      [,2]       [,3]
## [1,] -1.281697 -1.235988 -0.5194675
##           [,1]      [,2]       [,3]
## [1,] -1.168995 -1.128463 -0.4758567
##           [,1]     [,2]       [,3]
## [1,] -1.066482 -1.03042 -0.4357657
##           [,1]      [,2]       [,3]
## [1,] -0.9731859 -0.941004 -0.3989451
##           [,1]       [,2]       [,3]
## [1,] -0.8882381 -0.8594396 -0.3651543
##           [,1]       [,2]      [,3]
## [1,] -0.8108581 -0.7850237 -0.334164
##           [,1]       [,2]       [,3]
## [1,] -0.7403446 -0.7171178 -0.3057573
##           [,1]       [,2]       [,3]
## [1,] -0.676066 -0.6551424 -0.2797304
##           [,1]       [,2]       [,3]
## [1,] -0.6174528 -0.5985708 -0.2558927
##           [,1]       [,2]       [,3]
## [1,] -0.5639907 -0.5469246 -0.2340667
##           [,1]       [,2]       [,3]
```

```
## [1,] -0.5152149 -0.4997687 -0.2140877
##            [,1]       [,2]       [,3]
## [1,] -0.4707044 -0.4567073 -0.1958031
##            [,1]       [,2]       [,3]
## [1,] -0.4300782 -0.4173805 -0.1790722
##            [,1]       [,2]       [,3]
## [1,] -0.3929904 -0.3814606 -0.1637651
##            [,1]       [,2]       [,3]
## [1,] -0.3591273 -0.3486492 -0.1497622
##            [,1]       [,2]       [,3]
## [1,] -0.3282039 -0.3186747 -0.1369536
##            [,1]       [,2]       [,3]
## [1,] -0.2999613 -0.2912893 -0.1252384
##            [,1]       [,2]       [,3]
## [1,] -0.2741639 -0.2662676 -0.1145239
##            [,1]       [,2]       [,3]
## [1,] -0.2505976 -0.2434039 -0.1047251
##            [,1]       [,2]      [,3]
## [1,] -0.2290671 -0.2225107 -0.095764
##           [,1]       [,2]        [,3]
## [1,] -0.209395 -0.2034171 -0.08756934
##            [,1]      [,2]        [,3]
## [1,] -0.1914192 -0.185967 -0.08007571
##            [,1]       [,2]        [,3]
## [1,] -0.1749925 -0.1700181 -0.07322327
##            [,1]       [,2]        [,3]
## [1,] -0.1599803 -0.1554407 -0.06695723
##            [,1]       [,2]        [,3]
## [1,] -0.1462599 -0.1421162 -0.06122747
##            [,1]       [,2]        [,3]
## [1,] -0.1337196 -0.1299364 -0.05598811
##            [,1]       [,2]        [,3]
## [1,] -0.1222572 -0.1188026 -0.05119722
##            [,1]       [,2]       [,3]
## [1,] -0.1117797 -0.1086246 -0.0468164
##           [,1]        [,2]        [,3]
## [1,] -0.102202 -0.09932006 -0.04281056
##             [,1]       [,2]        [,3]
## [1,] -0.09344661 -0.0908138 -0.03914759
##             [,1]        [,2]        [,3]
## [1,] -0.08544256 -0.08303711 -0.03579815
##             [,1]        [,2]        [,3]
## [1,] -0.07812519 -0.07592725 -0.03273539
##            [,1]       [,2]        [,3]
## [1,] -0.0714354 -0.0694269 -0.02993476
##             [,1]        [,2]        [,3]
## [1,] -0.06531922 -0.06348368 -0.02737381
##             [,1]        [,2]        [,3]
## [1,] -0.05972733 -0.05804974 -0.02503204
##             [,1]        [,2]        [,3]
## [1,] -0.05461469 -0.05308137 -0.02289066
##             [,1]        [,2]        [,3]
## [1,] -0.04994013 -0.04853859 -0.02093253
##             [,1]       [,2]        [,3]
```

```
## [1,] -0.04566604 -0.0443849 -0.01914195
##             [,1]        [,2]        [,3]
## [1,] -0.04175805 -0.04058691 -0.01750459
##             [,1]        [,2]        [,3]
## [1,] -0.03818475 -0.03711414 -0.01600732
##             [,1]        [,2]        [,3]
## [1,] -0.03491745 -0.03393868 -0.01463815
##             [,1]        [,2]        [,3]
## [1,] -0.03192988 -0.03103507 -0.01338612
##             [,1]     [,2]        [,3]
## [1,] -0.02919809 -0.02838 -0.01224121
##             [,1]        [,2]        [,3]
## [1,] -0.02670014 -0.02595217 -0.01119424
##           [,1]        [,2]        [,3]
## [1,] -0.024416 -0.02373213 -0.01023683
##             [,1]        [,2]         [,3]
## [1,] -0.02232735 -0.02170208 -0.009361328
##             [,1]        [,2]         [,3]
## [1,] -0.02041744 -0.01984573 -0.008560712
##             [,1]        [,2]         [,3]
## [1,] -0.01867097 -0.01814823 -0.007828579
##             [,1]        [,2]        [,3]
## [1,] -0.01707394 -0.01659596 -0.00715907
##             [,1]       [,2]         [,3]
## [1,] -0.01561355 -0.0151765 -0.006546825
##             [,1]        [,2]         [,3]
## [1,] -0.01427811 -0.01387848 -0.005986947
##             [,1]       [,2]         [,3]
## [1,] -0.01305692 -0.0126915 -0.005474955
##             [,1]        [,2]         [,3]
## [1,] -0.01194021 -0.01160606 -0.005006752
##             [,1]        [,2]         [,3]
## [1,] -0.01091902 -0.01061347 -0.004578593
##              [,1]         [,2]         [,3]
## [1,] -0.009985184 -0.009705787 -0.004187052
##             [,1]         [,2]         [,3]
## [1,] -0.00913123 -0.008875741 -0.003828996
##              [,1]         [,2]         [,3]
## [1,] -0.008350319 -0.008116692 -0.003501563
##              [,1]         [,2]         [,3]
## [1,] -0.007636203 -0.007422565 -0.003202131
##              [,1]         [,2]         [,3]
## [1,] -0.006983165 -0.006787806 -0.002928307
##              [,1]         [,2]       [,3]
## [1,] -0.006385982 -0.006207336 -0.0026779
##              [,1]         [,2]         [,3]
## [1,] -0.005839874 -0.005676511 -0.002448907
##              [,1]         [,2]         [,3]
## [1,] -0.005340473 -0.005191084 -0.002239497
##              [,1]         [,2]         [,3]
## [1,] -0.004883782 -0.004747172 -0.002047994
##              [,1]         [,2]         [,3]
## [1,] -0.004466148 -0.004341224 -0.001872868
##              [,1]         [,2]         [,3]
```

```
## [1,] -0.004084231 -0.003969992 -0.001712718
##              [,1]         [,2]         [,3]
## [1,] -0.003734976 -0.003630508 -0.001566263
##              [,1]         [,2]         [,3]
## [1,] -0.003415588 -0.003320055 -0.001432332
##              [,1]         [,2]         [,3]
## [1,] -0.003123514 -0.003036152 -0.001309854
##              [,1]         [,2]         [,3]
## [1,] -0.002856417 -0.002776527 -0.001197849
##              [,1]         [,2]         [,3]
## [1,] -0.002612161 -0.002539103 -0.001095421
##              [,1]         [,2]         [,3]
## [1,] -0.002388793 -0.002321983 -0.001001753
##              [,1]        [,2]         [,3]
## [1,] -0.002184526 -0.00212343 -0.000916094
##              [,1]         [,2]          [,3]
## [1,] -0.001997726 -0.001941855 -0.0008377599
##              [,1]         [,2]          [,3]
## [1,] -0.001826901 -0.001775808 -0.0007661242
##              [,1]         [,2]          [,3]
## [1,] -0.001670683 -0.001623959 -0.0007006141
##              [,1]         [,2]          [,3]
## [1,] -0.001527824 -0.001485096 -0.0006407057
##              [,1]         [,2]          [,3]
## [1,] -0.001397181 -0.001358107 -0.0005859201
##              [,1]         [,2]          [,3]
## [1,] -0.001277709 -0.001241976 -0.0005358192
##              [,1]         [,2]          [,3]
## [1,] -0.001168453 -0.001135777 -0.0004900023
##             [,1]         [,2]          [,3]
## [1,] -0.00106854 -0.001038658 -0.0004481032
##               [,1]          [,2]          [,3]
## [1,] -0.0009771711 -0.0009498439 -0.0004097869
##               [,1]          [,2]          [,3]
## [1,] -0.0008936147 -0.0008686244 -0.0003747469
##               [,1]          [,2]          [,3]
## [1,] -0.0008172032 -0.0007943499 -0.0003427032
##               [,1]          [,2]          [,3]
## [1,] -0.0007473257 -0.0007264265 -0.0003133994
##               [,1]          [,2]          [,3]
## [1,] -0.0006834233 -0.0006643113 -0.0002866014
##               [,1]          [,2]          [,3]
## [1,] -0.0006249851 -0.0006075074 -0.0002620948
##               [,1]          [,2]          [,3]
## [1,] -0.0005715439 -0.0005555608 -0.0002396837
##               [,1]         [,2]         [,3]
## [1,] -0.0005226725 -0.000508056 -0.000219189
```

```
# Hessian
for (i in 1:n_iteration) {
  print(-l_derivative_2(beta_list[i,]))
}
```

```
##           [,1]     [,2]     [,3]
```

```
## [1,] 1194.3367 597.1683 597.1683
## [2,]  597.1683 597.1683 298.5842
## [3,]  597.1683 298.5842 597.1683
##           [,1]     [,2]     [,3]
## [1,] 1128.3811 564.1905 564.1905
## [2,]  564.1905 564.1905 282.0953
## [3,]  564.1905 282.0953 564.1905
##           [,1]     [,2]     [,3]
## [1,] 1097.8755 548.9378 548.9378
## [2,]  548.9378 548.9378 274.4689
## [3,]  548.9378 274.4689 548.9378
##           [,1]     [,2]     [,3]
## [1,] 1083.5990 541.7995 541.7995
## [2,]  541.7995 541.7995 270.8997
## [3,]  541.7995 270.8997 541.7995
##           [,1]     [,2]     [,3]
## [1,] 1075.8430 537.9215 537.9215
## [2,]  537.9215 537.9215 268.9607
## [3,]  537.9215 268.9607 537.9215
##           [,1]     [,2]     [,3]
## [1,] 1070.2417 535.1208 535.1208
## [2,]  535.1208 535.1208 267.5604
## [3,]  535.1208 267.5604 535.1208
##           [,1]     [,2]     [,3]
## [1,] 1065.1227 532.5614 532.5614
## [2,]  532.5614 532.5614 266.2807
## [3,]  532.5614 266.2807 532.5614
##           [,1]     [,2]     [,3]
## [1,] 1060.0192 530.0096 530.0096
## [2,]  530.0096 530.0096 265.0048
## [3,]  530.0096 265.0048 530.0096
##           [,1]     [,2]     [,3]
## [1,] 1054.9253 527.4626 527.4626
## [2,]  527.4626 527.4626 263.7313
## [3,]  527.4626 263.7313 527.4626
##           [,1]     [,2]     [,3]
## [1,] 1049.9626 524.9813 524.9813
## [2,]  524.9813 524.9813 262.4907
## [3,]  524.9813 262.4907 524.9813
##           [,1]     [,2]     [,3]
## [1,] 1045.2513 522.6256 522.6256
## [2,]  522.6256 522.6256 261.3128
## [3,]  522.6256 261.3128 522.6256
##           [,1]     [,2]     [,3]
## [1,] 1040.8716 520.4358 520.4358
## [2,]  520.4358 520.4358 260.2179
## [3,]  520.4358 260.2179 520.4358
##           [,1]     [,2]     [,3]
## [1,] 1036.8628 518.4314 518.4314
## [2,]  518.4314 518.4314 259.2157
## [3,]  518.4314 259.2157 518.4314
##           [,1]     [,2]     [,3]
## [1,] 1033.2322 516.6161 516.6161
## [2,]  516.6161 516.6161 258.3081
```

```
## [3,]  516.6161 258.3081 516.6161
##           [,1]     [,2]     [,3]
## [1,] 1029.9670 514.9835 514.9835
## [2,]  514.9835 514.9835 257.4918
## [3,]  514.9835 257.4918 514.9835
##           [,1]     [,2]     [,3]
## [1,] 1027.0424 513.5212 513.5212
## [2,]  513.5212 513.5212 256.7606
## [3,]  513.5212 256.7606 513.5212
##           [,1]     [,2]     [,3]
## [1,] 1024.4285 512.2142 512.2142
## [2,]  512.2142 512.2142 256.1071
## [3,]  512.2142 256.1071 512.2142
##           [,1]     [,2]     [,3]
## [1,] 1022.0938 511.0469 511.0469
## [2,]  511.0469 511.0469 255.5235
## [3,]  511.0469 255.5235 511.0469
##          [,1]    [,2]    [,3]
## [1,] 1020.008 510.004 510.004
## [2,]  510.004 510.004 255.002
## [3,]  510.004 255.002 510.004
##           [,1]     [,2]     [,3]
## [1,] 1018.1431 509.0715 509.0715
## [2,]  509.0715 509.0715 254.5358
## [3,]  509.0715 254.5358 509.0715
##           [,1]     [,2]     [,3]
## [1,] 1016.4734 508.2367 508.2367
## [2,]  508.2367 508.2367 254.1183
## [3,]  508.2367 254.1183 508.2367
##           [,1]     [,2]     [,3]
## [1,] 1014.9764 507.4882 507.4882
## [2,]  507.4882 507.4882 253.7441
## [3,]  507.4882 253.7441 507.4882
##          [,1]    [,2]    [,3]
## [1,] 1013.632 506.816 506.816
## [2,]  506.816 506.816 253.408
## [3,]  506.816 253.408 506.816
##           [,1]     [,2]     [,3]
## [1,] 1012.4231 506.2116 506.2116
## [2,]  506.2116 506.2116 253.1058
## [3,]  506.2116 253.1058 506.2116
##           [,1]     [,2]     [,3]
## [1,] 1011.3342 505.6671 505.6671
## [2,]  505.6671 505.6671 252.8336
## [3,]  505.6671 252.8336 505.6671
##          [,1]    [,2]    [,3]
## [1,] 1010.352 505.176 505.176
## [2,]  505.176 505.176 252.588
## [3,]  505.176 252.588 505.176
##           [,1]     [,2]     [,3]
## [1,] 1009.4650 504.7325 504.7325
## [2,]  504.7325 504.7325 252.3663
## [3,]  504.7325 252.3663 504.7325
##           [,1]     [,2]     [,3]
```

```
## [1,] 1008.6629 504.3315 504.3315
## [2,]  504.3315 504.3315 252.1657
## [3,]  504.3315 252.1657 504.3315
##           [,1]     [,2]     [,3]
## [1,] 1007.9368 503.9684 503.9684
## [2,]  503.9684 503.9684 251.9842
## [3,]  503.9684 251.9842 503.9684
##           [,1]     [,2]     [,3]
## [1,] 1007.2788 503.6394 503.6394
## [2,]  503.6394 503.6394 251.8197
## [3,]  503.6394 251.8197 503.6394
##          [,1]     [,2]     [,3]
## [1,] 1006.682 503.3410 503.3410
## [2,]  503.341 503.3410 251.6705
## [3,]  503.341 251.6705 503.3410
##           [,1]     [,2]     [,3]
## [1,] 1006.1402 503.0701 503.0701
## [2,]  503.0701 503.0701 251.5351
## [3,]  503.0701 251.5351 503.0701
##           [,1]     [,2]     [,3]
## [1,] 1005.6481 502.8241 502.8241
## [2,]  502.8241 502.8241 251.4120
## [3,]  502.8241 251.4120 502.8241
##           [,1]     [,2]     [,3]
## [1,] 1005.2007 502.6004 502.6004
## [2,]  502.6004 502.6004 251.3002
## [3,]  502.6004 251.3002 502.6004
##           [,1]     [,2]     [,3]
## [1,] 1004.7938 502.3969 502.3969
## [2,]  502.3969 502.3969 251.1984
## [3,]  502.3969 251.1984 502.3969
##           [,1]     [,2]     [,3]
## [1,] 1004.4235 502.2117 502.2117
## [2,]  502.2117 502.2117 251.1059
## [3,]  502.2117 251.1059 502.2117
##           [,1]     [,2]     [,3]
## [1,] 1004.0863 502.0431 502.0431
## [2,]  502.0431 502.0431 251.0216
## [3,]  502.0431 251.0216 502.0431
##           [,1]     [,2]     [,3]
## [1,] 1003.7791 501.8896 501.8896
## [2,]  501.8896 501.8896 250.9448
## [3,]  501.8896 250.9448 501.8896
##           [,1]     [,2]     [,3]
## [1,] 1003.4992 501.7496 501.7496
## [2,]  501.7496 501.7496 250.8748
## [3,]  501.7496 250.8748 501.7496
##          [,1]    [,2]    [,3]
## [1,] 1003.244 501.622 501.622
## [2,]  501.622 501.622 250.811
## [3,]  501.622 250.811 501.622
##           [,1]     [,2]     [,3]
## [1,] 1003.0114 501.5057 501.5057
## [2,]  501.5057 501.5057 250.7529
```

```
## [3,]  501.5057 250.7529 501.5057
##           [,1]     [,2]     [,3]
## [1,] 1002.7992 501.3996 501.3996
## [2,]  501.3996 501.3996 250.6998
## [3,]  501.3996 250.6998 501.3996
##           [,1]     [,2]     [,3]
## [1,] 1002.6056 501.3028 501.3028
## [2,]  501.3028 501.3028 250.6514
## [3,]  501.3028 250.6514 501.3028
##           [,1]     [,2]     [,3]
## [1,] 1002.4289 501.2145 501.2145
## [2,]  501.2145 501.2145 250.6072
## [3,]  501.2145 250.6072 501.2145
##           [,1]     [,2]     [,3]
## [1,] 1002.2677 501.1338 501.1338
## [2,]  501.1338 501.1338 250.5669
## [3,]  501.1338 250.5669 501.1338
##           [,1]     [,2]     [,3]
## [1,] 1002.1204 501.0602 501.0602
## [2,]  501.0602 501.0602 250.5301
## [3,]  501.0602 250.5301 501.0602
##          [,1]     [,2]     [,3]
## [1,] 1001.986 500.9930 500.9930
## [2,]  500.993 500.9930 250.4965
## [3,]  500.993 250.4965 500.9930
##           [,1]     [,2]     [,3]
## [1,] 1001.8633 500.9316 500.9316
## [2,]  500.9316 500.9316 250.4658
## [3,]  500.9316 250.4658 500.9316
##           [,1]     [,2]     [,3]
## [1,] 1001.7512 500.8756 500.8756
## [2,]  500.8756 500.8756 250.4378
## [3,]  500.8756 250.4378 500.8756
##           [,1]     [,2]     [,3]
## [1,] 1001.6488 500.8244 500.8244
## [2,]  500.8244 500.8244 250.4122
## [3,]  500.8244 250.4122 500.8244
##           [,1]     [,2]     [,3]
## [1,] 1001.5552 500.7776 500.7776
## [2,]  500.7776 500.7776 250.3888
## [3,]  500.7776 250.3888 500.7776
##           [,1]     [,2]     [,3]
## [1,] 1001.4698 500.7349 500.7349
## [2,]  500.7349 500.7349 250.3674
## [3,]  500.7349 250.3674 500.7349
##           [,1]     [,2]     [,3]
## [1,] 1001.3917 500.6958 500.6958
## [2,]  500.6958 500.6958 250.3479
## [3,]  500.6958 250.3479 500.6958
##           [,1]     [,2]     [,3]
## [1,] 1001.3203 500.6602 500.6602
## [2,]  500.6602 500.6602 250.3301
## [3,]  500.6602 250.3301 500.6602
##           [,1]     [,2]     [,3]
```

```
## [1,] 1001.2551 500.6276 500.6276
## [2,]  500.6276 500.6276 250.3138
## [3,]  500.6276 250.3138 500.6276
##           [,1]     [,2]     [,3]
## [1,] 1001.1956 500.5978 500.5978
## [2,]  500.5978 500.5978 250.2989
## [3,]  500.5978 250.2989 500.5978
##           [,1]     [,2]     [,3]
## [1,] 1001.1411 500.5705 500.5705
## [2,]  500.5705 500.5705 250.2853
## [3,]  500.5705 250.2853 500.5705
##           [,1]     [,2]     [,3]
## [1,] 1001.0913 500.5457 500.5457
## [2,]  500.5457 500.5457 250.2728
## [3,]  500.5457 250.2728 500.5457
##           [,1]     [,2]     [,3]
## [1,] 1001.0458 500.5229 500.5229
## [2,]  500.5229 500.5229 250.2615
## [3,]  500.5229 250.2615 500.5229
##           [,1]     [,2]     [,3]
## [1,] 1001.0042 500.5021 500.5021
## [2,]  500.5021 500.5021 250.2511
## [3,]  500.5021 250.2511 500.5021
##           [,1]     [,2]     [,3]
## [1,] 1000.9662 500.4831 500.4831
## [2,]  500.4831 500.4831 250.2416
## [3,]  500.4831 250.2416 500.4831
##           [,1]     [,2]     [,3]
## [1,] 1000.9315 500.4657 500.4657
## [2,]  500.4657 500.4657 250.2329
## [3,]  500.4657 250.2329 500.4657
##           [,1]     [,2]     [,3]
## [1,] 1000.8997 500.4498 500.4498
## [2,]  500.4498 500.4498 250.2249
## [3,]  500.4498 250.2249 500.4498
##           [,1]     [,2]     [,3]
## [1,] 1000.8706 500.4353 500.4353
## [2,]  500.4353 500.4353 250.2177
## [3,]  500.4353 250.2177 500.4353
##          [,1]    [,2]    [,3]
## [1,] 1000.844 500.422 500.422
## [2,]  500.422 500.422 250.211
## [3,]  500.422 250.211 500.422
##           [,1]     [,2]     [,3]
## [1,] 1000.8198 500.4099 500.4099
## [2,]  500.4099 500.4099 250.2050
## [3,]  500.4099 250.2050 500.4099
##           [,1]     [,2]     [,3]
## [1,] 1000.7976 500.3988 500.3988
## [2,]  500.3988 500.3988 250.1994
## [3,]  500.3988 250.1994 500.3988
##           [,1]     [,2]     [,3]
## [1,] 1000.7773 500.3887 500.3887
## [2,]  500.3887 500.3887 250.1943
```

```
## [3,]  500.3887 250.1943 500.3887
##           [,1]     [,2]     [,3]
## [1,] 1000.7588 500.3794 500.3794
## [2,]  500.3794 500.3794 250.1897
## [3,]  500.3794 250.1897 500.3794
##           [,1]     [,2]     [,3]
## [1,] 1000.7418 500.3709 500.3709
## [2,]  500.3709 500.3709 250.1854
## [3,]  500.3709 250.1854 500.3709
##           [,1]     [,2]     [,3]
## [1,] 1000.7263 500.3631 500.3631
## [2,]  500.3631 500.3631 250.1816
## [3,]  500.3631 250.1816 500.3631
##           [,1]     [,2]     [,3]
## [1,] 1000.7121 500.3561 500.3561
## [2,]  500.3561 500.3561 250.1780
## [3,]  500.3561 250.1780 500.3561
##           [,1]     [,2]     [,3]
## [1,] 1000.6991 500.3496 500.3496
## [2,]  500.3496 500.3496 250.1748
## [3,]  500.3496 250.1748 500.3496
##           [,1]     [,2]     [,3]
## [1,] 1000.6873 500.3436 500.3436
## [2,]  500.3436 500.3436 250.1718
## [3,]  500.3436 250.1718 500.3436
##           [,1]     [,2]     [,3]
## [1,] 1000.6764 500.3382 500.3382
## [2,]  500.3382 500.3382 250.1691
## [3,]  500.3382 250.1691 500.3382
##           [,1]     [,2]     [,3]
## [1,] 1000.6665 500.3333 500.3333
## [2,]  500.3333 500.3333 250.1666
## [3,]  500.3333 250.1666 500.3333
##           [,1]     [,2]     [,3]
## [1,] 1000.6574 500.3287 500.3287
## [2,]  500.3287 500.3287 250.1644
## [3,]  500.3287 250.1644 500.3287
##           [,1]     [,2]     [,3]
## [1,] 1000.6492 500.3246 500.3246
## [2,]  500.3246 500.3246 250.1623
## [3,]  500.3246 250.1623 500.3246
##           [,1]     [,2]     [,3]
## [1,] 1000.6416 500.3208 500.3208
## [2,]  500.3208 500.3208 250.1604
## [3,]  500.3208 250.1604 500.3208
##           [,1]     [,2]     [,3]
## [1,] 1000.6346 500.3173 500.3173
## [2,]  500.3173 500.3173 250.1587
## [3,]  500.3173 250.1587 500.3173
##           [,1]     [,2]     [,3]
## [1,] 1000.6283 500.3142 500.3142
## [2,]  500.3142 500.3142 250.1571
## [3,]  500.3142 250.1571 500.3142
##           [,1]     [,2]     [,3]
```

```
## [1,] 1000.6225 500.3113 500.3113
## [2,]  500.3113 500.3113 250.1556
## [3,]  500.3113 250.1556 500.3113
##           [,1]     [,2]     [,3]
## [1,] 1000.6172 500.3086 500.3086
## [2,]  500.3086 500.3086 250.1543
## [3,]  500.3086 250.1543 500.3086
##           [,1]     [,2]     [,3]
## [1,] 1000.6124 500.3062 500.3062
## [2,]  500.3062 500.3062 250.1531
## [3,]  500.3062 250.1531 500.3062
##          [,1]    [,2]    [,3]
## [1,] 1000.608 500.304 500.304
## [2,]  500.304 500.304 250.152
## [3,]  500.304 250.152 500.304
##           [,1]     [,2]     [,3]
## [1,] 1000.6039 500.3019 500.3019
## [2,]  500.3019 500.3019 250.1510
## [3,]  500.3019 250.1510 500.3019
##           [,1]     [,2]     [,3]
## [1,] 1000.6002 500.3001 500.3001
## [2,]  500.3001 500.3001 250.1500
## [3,]  500.3001 250.1500 500.3001
##           [,1]     [,2]     [,3]
## [1,] 1000.5968 500.2984 500.2984
## [2,]  500.2984 500.2984 250.1492
## [3,]  500.2984 250.1492 500.2984
##           [,1]     [,2]     [,3]
## [1,] 1000.5937 500.2968 500.2968
## [2,]  500.2968 500.2968 250.1484
## [3,]  500.2968 250.1484 500.2968
##           [,1]     [,2]     [,3]
## [1,] 1000.5908 500.2954 500.2954
## [2,]  500.2954 500.2954 250.1477
## [3,]  500.2954 250.1477 500.2954
##           [,1]     [,2]     [,3]
## [1,] 1000.5883 500.2941 500.2941
## [2,]  500.2941 500.2941 250.1471
## [3,]  500.2941 250.1471 500.2941
##           [,1]     [,2]     [,3]
## [1,] 1000.5859 500.2929 500.2929
## [2,]  500.2929 500.2929 250.1465
## [3,]  500.2929 250.1465 500.2929
##           [,1]     [,2]     [,3]
## [1,] 1000.5837 500.2919 500.2919
## [2,]  500.2919 500.2919 250.1459
## [3,]  500.2919 250.1459 500.2919
##           [,1]     [,2]     [,3]
## [1,] 1000.5817 500.2909 500.2909
## [2,]  500.2909 500.2909 250.1454
## [3,]  500.2909 250.1454 500.2909
##         [,1]    [,2]    [,3]
## [1,] 1000.58 500.290 500.290
## [2,]  500.29 500.290 250.145
```

```
## [3,]  500.29 250.145 500.290
##           [,1]     [,2]     [,3]
## [1,] 1000.5783 500.2891 500.2891
## [2,]  500.2891 500.2891 250.1446
## [3,]  500.2891 250.1446 500.2891
##           [,1]     [,2]     [,3]
## [1,] 1000.5767 500.2884 500.2884
## [2,]  500.2884 500.2884 250.1442
## [3,]  500.2884 250.1442 500.2884
##           [,1]     [,2]     [,3]
## [1,] 1000.5754 500.2877 500.2877
## [2,]  500.2877 500.2877 250.1438
## [3,]  500.2877 250.1438 500.2877
##          [,1]     [,2]     [,3]
## [1,] 1000.574 500.2870 500.2870
## [2,]  500.287 500.2870 250.1435
## [3,]  500.287 250.1435 500.2870
##           [,1]     [,2]     [,3]
## [1,] 1000.5729 500.2865 500.2865
## [2,]  500.2865 500.2865 250.1432
## [3,]  500.2865 250.1432 500.2865
##           [,1]     [,2]     [,3]
## [1,] 1000.5719 500.2859 500.2859
## [2,]  500.2859 500.2859 250.1430
## [3,]  500.2859 250.1430 500.2859
##           [,1]     [,2]     [,3]
## [1,] 1000.5709 500.2855 500.2855
## [2,]  500.2855 500.2855 250.1427
## [3,]  500.2855 250.1427 500.2855
##          [,1]     [,2]     [,3]
## [1,] 1000.570 500.2850 500.2850
## [2,]  500.285 500.2850 250.1425
## [3,]  500.285 250.1425 500.2850
##           [,1]     [,2]     [,3]
## [1,] 1000.5692 500.2846 500.2846
## [2,]  500.2846 500.2846 250.1423
## [3,]  500.2846 250.1423 500.2846
##           [,1]     [,2]     [,3]
## [1,] 1000.5685 500.2842 500.2842
## [2,]  500.2842 500.2842 250.1421
## [3,]  500.2842 250.1421 500.2842
##           [,1]     [,2]     [,3]
## [1,] 1000.5678 500.2839 500.2839
## [2,]  500.2839 500.2839 250.1419
## [3,]  500.2839 250.1419 500.2839
##           [,1]     [,2]     [,3]
## [1,] 1000.5672 500.2836 500.2836
## [2,]  500.2836 500.2836 250.1418
## [3,]  500.2836 250.1418 500.2836
##           [,1]     [,2]     [,3]
## [1,] 1000.5666 500.2833 500.2833
## [2,]  500.2833 500.2833 250.1416
## [3,]  500.2833 250.1416 500.2833
##          [,1]     [,2]     [,3]
```

```
## [1,] 1000.566 500.2830 500.2830
## [2,]  500.283 500.2830 250.1415
## [3,]  500.283 250.1415 500.2830
##           [,1]     [,2]     [,3]
## [1,] 1000.5656 500.2828 500.2828
## [2,]  500.2828 500.2828 250.1414
## [3,]  500.2828 250.1414 500.2828
##           [,1]     [,2]     [,3]
## [1,] 1000.5652 500.2826 500.2826
## [2,]  500.2826 500.2826 250.1413
## [3,]  500.2826 250.1413 500.2826
##           [,1]     [,2]     [,3]
## [1,] 1000.5648 500.2824 500.2824
## [2,]  500.2824 500.2824 250.1412
## [3,]  500.2824 250.1412 500.2824
##           [,1]     [,2]     [,3]
## [1,] 1000.5644 500.2822 500.2822
## [2,]  500.2822 500.2822 250.1411
## [3,]  500.2822 250.1411 500.2822
##          [,1]    [,2]    [,3]
## [1,] 1000.564 500.282 500.282
## [2,]  500.282 500.282 250.141
## [3,]  500.282 250.141 500.282
##           [,1]     [,2]     [,3]
## [1,] 1000.5638 500.2819 500.2819
## [2,]  500.2819 500.2819 250.1409
## [3,]  500.2819 250.1409 500.2819
##           [,1]     [,2]     [,3]
## [1,] 1000.5635 500.2817 500.2817
## [2,]  500.2817 500.2817 250.1409
## [3,]  500.2817 250.1409 500.2817
##           [,1]     [,2]     [,3]
## [1,] 1000.5632 500.2816 500.2816
## [2,]  500.2816 500.2816 250.1408
## [3,]  500.2816 250.1408 500.2816
##           [,1]     [,2]     [,3]
## [1,] 1000.5630 500.2815 500.2815
## [2,]  500.2815 500.2815 250.1408
## [3,]  500.2815 250.1408 500.2815
##           [,1]     [,2]     [,3]
## [1,] 1000.5628 500.2814 500.2814
## [2,]  500.2814 500.2814 250.1407
## [3,]  500.2814 250.1407 500.2814
##           [,1]     [,2]     [,3]
## [1,] 1000.5626 500.2813 500.2813
## [2,]  500.2813 500.2813 250.1407
## [3,]  500.2813 250.1407 500.2813
```