STATS 782 Assignment 2

Francis Tang, ID 240887036

Due: 10 April 2019

1. This question works with a data set on sodium intake. We can read it into R with the following code:

```
sodium <- read.table("sodium.txt", header=TRUE)
```

The Instructor is a nutrition advisor and Supplement is a nutritional supplement.

Extract just the first observation for each combination of Instructor and Supplement and create a matrix of the result.

```
extractedSodium <- sodium[!duplicated(sodium[ c("Instructor", "Supplement")]), ]
sodiumMat = matrix(extractedSodium$Sodium, nrow = 3, ncol = 4, byrow = TRUE)
dimnames(sodiumMat) = list(c("Brendon Small", "Coach McGuirk", "Melissa Robins"),
                           c("A", "B", "C", "D"))
sodiumMat
```

```
##                   A    B    C    D
## Brendon Small  1200 1150 1250 1300
## Coach McGuirk  1100 1250 1225 1200
## Melissa Robins  900 1150 1125 1100
```

Use apply and sweep to fit a model:

```
r = sodiumMat
(mu = mean(r))
```

```
## [1] 1162.5
```

```
r = r - mu

(alpha = apply(r, 1, mean))
```

```
##  Brendon Small  Coach McGuirk Melissa Robins
##          62.50          31.25         -93.75
```

```
r = sweep(r, 1, alpha)

(beta = apply(r, 2, mean))
```

```
##         A         B         C         D
## -95.83333  20.83333  37.50000  37.50000
```

```
r = sweep(r, 2, beta)
r
```

```
##                         A         B      C      D
## Brendon Small   70.833333 -95.83333 -12.50  37.50
## Coach McGuirk    2.083333  35.41667  -6.25 -31.25
## Melissa Robins -72.916667  60.41667  18.75  -6.25
```

Now we need to consider that whether this looks like an appropriate model or not. After swept out the column effects using sweep(), now we need to compute the row and column means of the residuals to check if all the effects have been swept out of the residuals.

```r
round(apply(r, 1, mean), 3)
```

```
##  Brendon Small   Coach McGuirk Melissa Robins
##              0               0              0
```

```r
round(apply(r, 2, mean), 4)
```

```
## A B C D
## 0 0 0 0
```

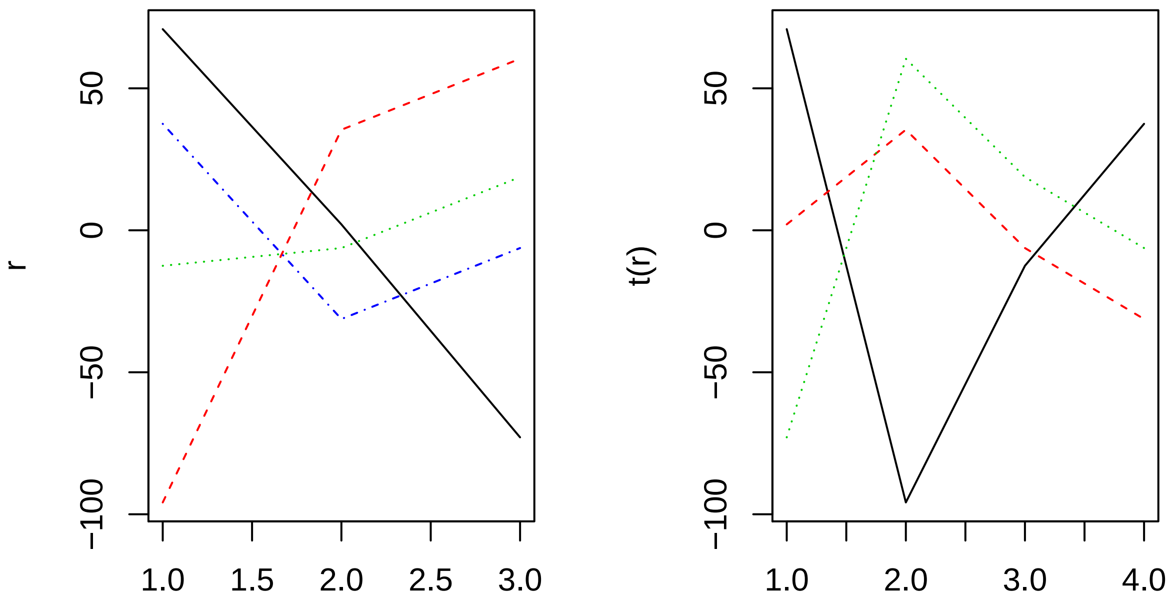
These are zero.

Another important indicator we need to consider is interactions. Now we are going to plot the residuals. Normally, as residuals are the results after getting rid of all the model patterns, which means the plot of residuals should not show any kind of patterns.

```r
r[order(alpha), order(beta)]
```

```
##                          A         B      C      D
## Melissa Robins -72.916667  60.41667  18.75  -6.25
## Coach McGuirk    2.083333  35.41667  -6.25 -31.25
## Brendon Small   70.833333 -95.83333 -12.50  37.50
```

```r
par(mfrow = c(1,2))
matplot(r, type = "l")
matplot(t(r), type = "l")
```



Unfortunately, the residual plots above show patterns that indicate that a model does not fit as well as it might. There is a slight suggestion of an interaction effect in these residuals.

2. This question works with a set of plant weights, measured under two experimental conditions.

```r
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
## Control = standard conditions
## Treatment = nutrient rich
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
```

```r
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
```

```r
segma = sd(weight)
```

Write an R function called like that calculates this likelihood, given a set of data x and a mean mu (Hint: the R function dnorm evaluates the Normal probability density given x, mu, and sigma).

```r
like <- function(x, mu)
  prod(dnorm(x, mu, segma, log = FALSE))

like(weight, 0)
```

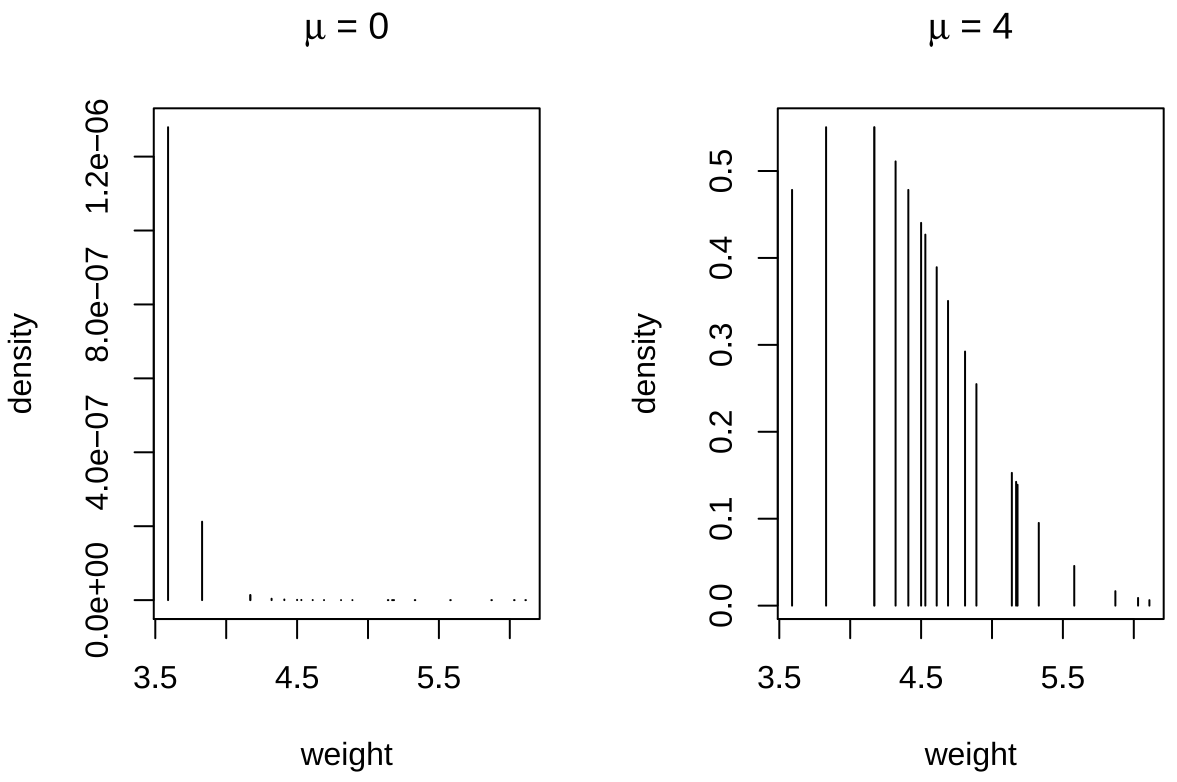
```
## [1] 1.359239e-215
```

```r
like(weight, 4)
```

```
## [1] 4.592718e-16
```

Draw a plot of the probability densities for each weight value for both mu = 0 and mu = 4 (and sigma equal to the sample standard deviation).

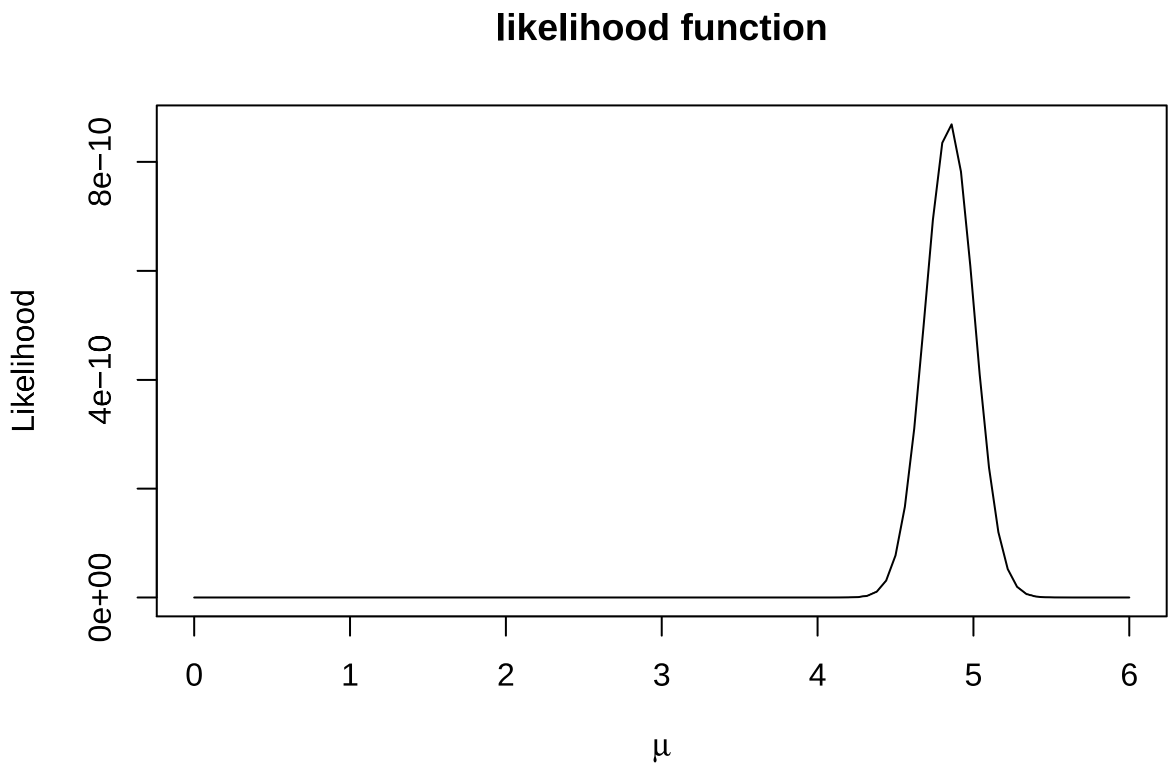
```r
par(mfrow = c(1,2))
density = dnorm(weight, 0, segma, log = FALSE)
plot(weight, density, type = 'h', main = expression(paste(paste(mu," = 0"))))
density = dnorm(weight, 4, segma, log = FALSE)
plot(weight, density, type = 'h', main = expression(paste(paste(mu," = 4"))))
```



Draw a plot of the likelihood function for mu varying from 0 to 6.

```r
fx <- function(mu) like(weight, mu)
Likelihood <- Vectorize(fx)
```

```r
plot(Likelihood, main = "likelihood function",
     xlab = expression(mu), ylab = "Likelihood", 0, 6)
```



Write a function loglike to calculate the log-likelihood (Hint: the dnorm function has an argument log).

```r
loglike <- function(x, mu)
  sum(dnorm(x, mu, segma, log = TRUE))

loglike(weight, 0)
```

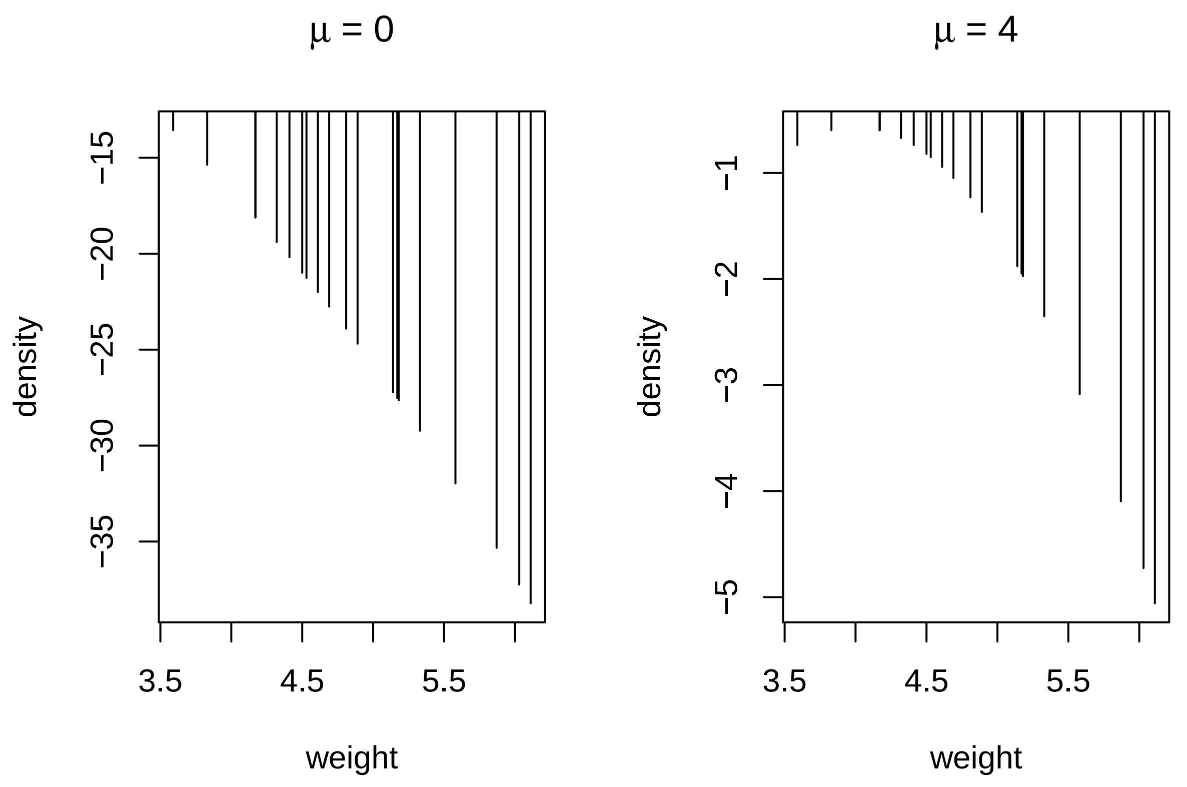
```
## [1] -494.7489
```

```r
loglike(weight, 4)
```

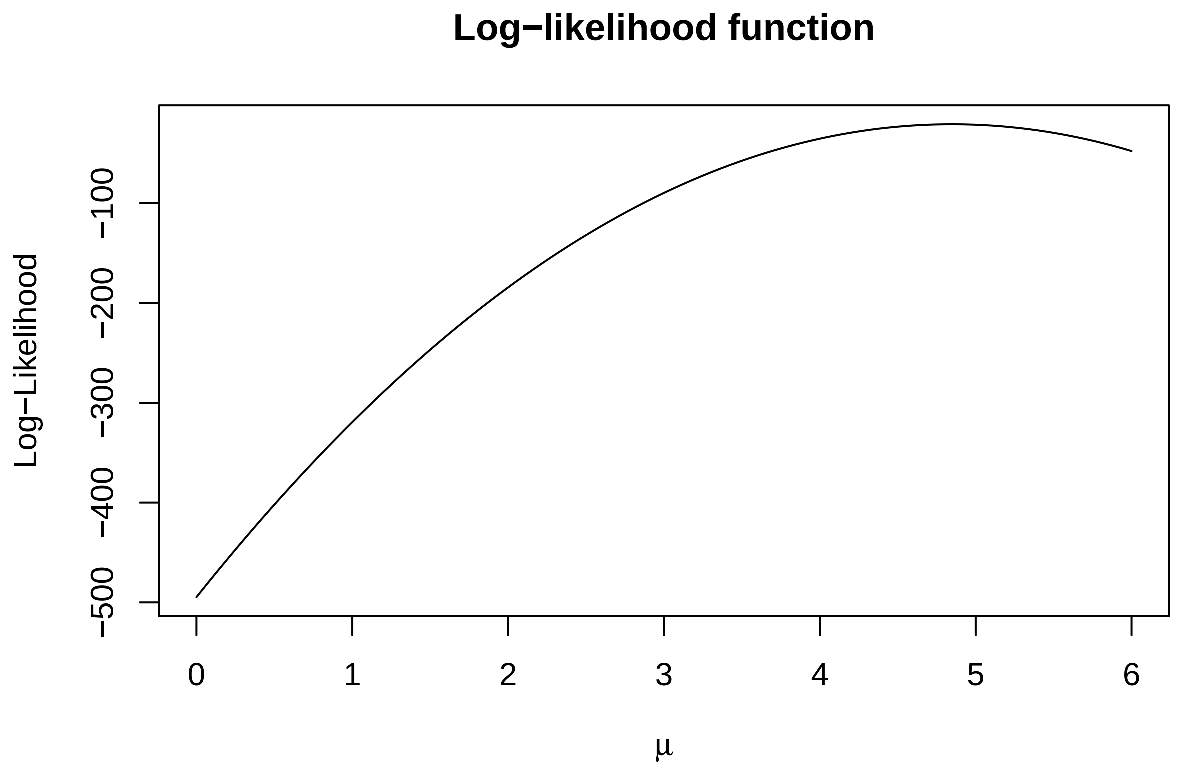
```
## [1] -35.31689
```

Plot log probability densities for the weight data given mu = 0 and mu = 4 and plot the log-likelihood curve for mu between 0 and 6.

```r
par(mfrow = c(1,2))
density = dnorm(weight, 0, segma, log = TRUE)
plot(weight, density, type = 'h',
     main = expression(paste(paste(mu," = 0"))))
density = dnorm(weight, 4, segma, log = TRUE)
plot(weight, density, type = 'h',
     main = expression(paste(paste(mu," = 4"))))
```



```r
fx <- function(mu) loglike(weight, mu)
Likelihood <- Vectorize(fx)
plot(Likelihood, main = "Log-likelihood function",
     xlab = expression(mu), ylab = "Log-Likelihood", 0, 6)
```



Use the optimise function to find the maximum likelihood estimate of mu (find the value of mu that maximises the log-likelihood function).

```
loglike1 <- function(mu)
  sum(dnorm(weight, mu, segma, log = TRUE))

z = optimise(loglike1, lower = 0, upper = 6, maximum = TRUE, tol = 1e-10)
muMLE <- z$maximum
muMLE
```

```
## [1] 4.8465
```

```
mean(weight)
```

```
## [1] 4.8465
```

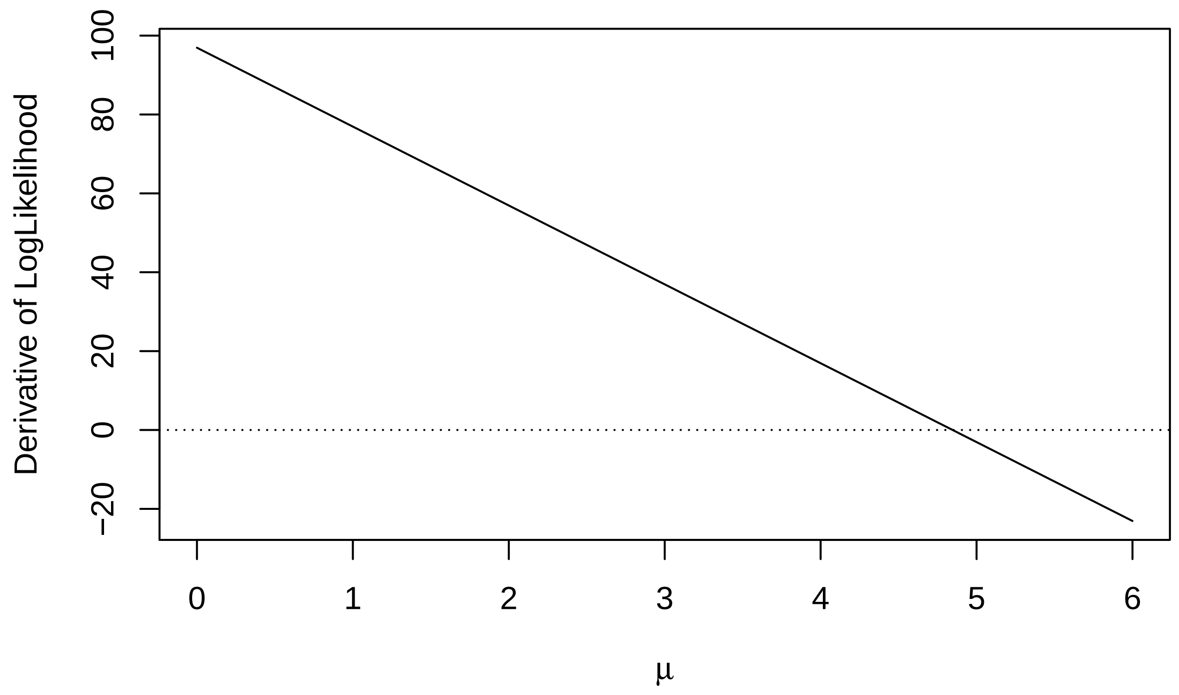
Write a function dllike that calculates this first derivative and use uniroot to find where this function is zero (a plot of the function is shown below). This should produce the same answer as above.

```
derivative <- function(mu)
  dist = sum(weight - mu)

dllike <- function(mu)
  res = sapply(mu, derivative)
```

```
uniroot(dllike, lower = 0, upper = 6, tol = 1e-10)$root
```

```
## [1] 4.8465
```

```
curve(dllike, from = 0, to = 6,
      xlab = expression(mu), ylab = "Derivative of LogLikelihood")
abline(h = 0, lty ="dotted")
```



3. This question also works with the set of plant weights and we will still assume that all weights are i.i.d. Normal(mu, sigma).

Write a function loglike2 to evaluate the log-likelihood, plot probability densities values for the weight data for both mu = 0; sigma = 1 and mu = 4; sigma = 1, and plot the log-likelihood function for mu between 0 and 6, with sigma = 1 and with sigma = .5.

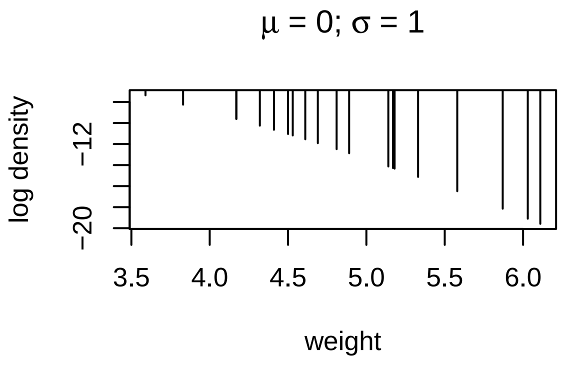
```
loglike2 <- function(x, mu, segma)
  sum(dnorm(x, mu, segma, log = TRUE))

loglike2(weight, 0, 1)
```

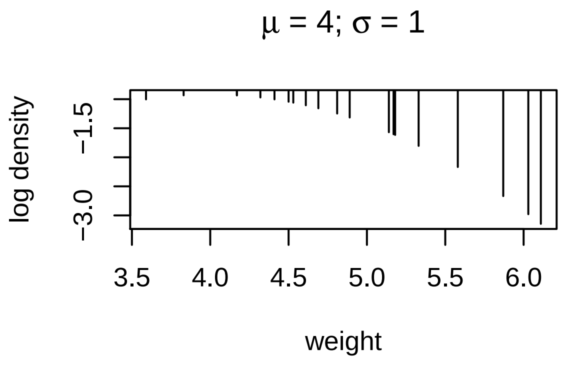
```
## [1] -257.9731
```

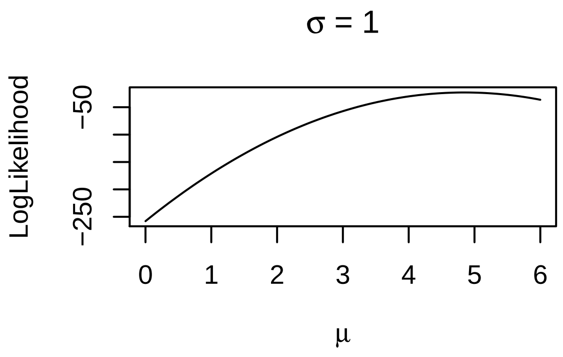
```
loglike2(weight, 4, 1)
```

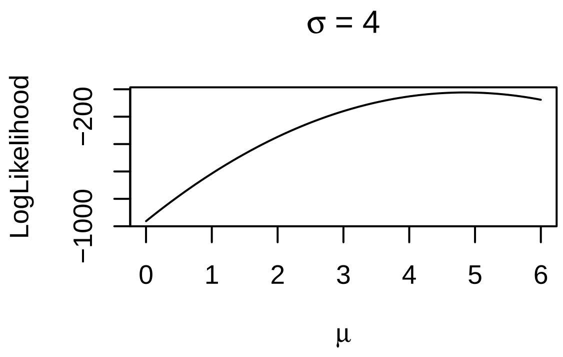
```
## [1] -30.25312
```

```
par(mfrow = c(2,2))

density = dnorm(weight, 0, 1, log = TRUE)
plot(weight, density, type = 'h',
     main = expression(paste(paste(mu," = 0; "),paste(sigma," = 1"))),
     ylab = "log density")

density = dnorm(weight, 4, 1, log = TRUE)
plot(weight, density, type = 'h',
     main = expression(paste(paste(mu," = 4; "),paste(sigma," = 1"))),
     ylab = "log density")

fx <- function(mu) loglike2(weight, mu, 1)
Likelihood <- Vectorize(fx)
plot(Likelihood, 0, 6, main = expression(paste(paste(sigma," = 1"))),
     xlab = expression(mu), ylab = "LogLikelihood")

fx <- function(mu) loglike2(weight, mu, 0.5)
Likelihood <- Vectorize(fx)
plot(Likelihood, 0, 6, main = expression(paste(paste(sigma," = 4"))),
     xlab = expression(mu), ylab = "LogLikelihood")
```









Use the optim function to find the maximum likelihood estimates for mu and sigma. These should correspond to the sample mean and (almost) the sample standard deviation.

First, we need to find good start points using outer():

```r
theta1 = seq(0, 6, by=0.01)
theta2 = seq(0.1, 2, by=0.01)

llmax = -Inf

for(t1 in theta1){
  for(t2 in theta2){
    lltheta = loglike2(weight, t1, t2)
    if (lltheta > llmax){
      llmax = lltheta
      start = c(t1,t2)
    }
  }
}

start
```

```
## [1] 4.85 0.69
```

```r
lloglike <- function(theta)
  sum(dnorm(weight, theta[1], theta[2], log = TRUE))
```

Optimisation:

```r
z2 = optim(c(4.85,0.69), lloglike, control=list(fnscale=-1),
method="BFGS")
```

```
muSigmaMLE <- z2$par
muSigmaMLE
```

```
## [1] 4.8464996 0.6862024
```

Our muSigmaMLE gives a very close answer to the sample mean and SD below, which indicates that our optimisation was appropriate.

```
mean(weight)
```

```
## [1] 4.8465
```

```
sd(weight)
```

```
## [1] 0.7040281
```

```
sd(weight)*sqrt((length(weight) - 1)/length(weight))
```

```
## [1] 0.6862017
```

4. This question also works with the set of plant weights, but now we will allow there to bea separate mean for the treatment and control groups.

Find the maximum likelihood estimates for beta0, beta1, and sigma.

```
loglike3 <- function(x, gp, b0, b1, segma)
  sum(dnorm(weight, b0 + gp * b1, segma, log = TRUE))

gp <- as.numeric(group) - 1

loglike3(weight, gp, 0, 1, 1)
```

```
## [1] -216.3631
```

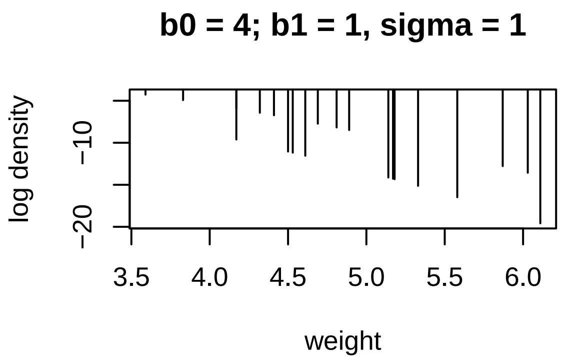
```
loglike3(weight, gp, 4, 1, 1)
```

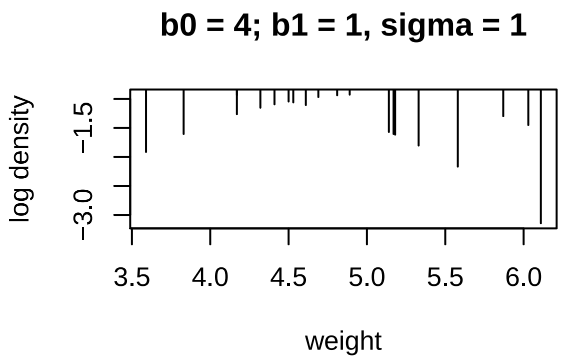
```
## [1] -28.64312
```

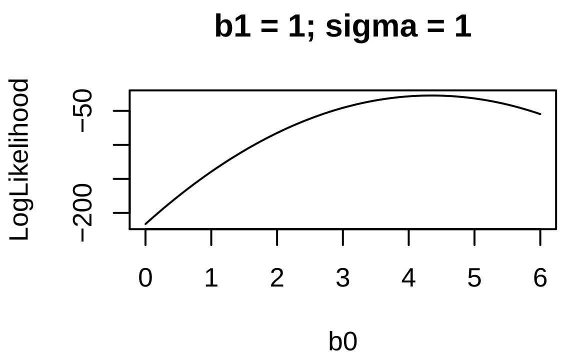
```
loglike4 <- function(x, b0, b1, segma)
  sum(dnorm(weight, b0 + gp * b1, segma, log = TRUE))
```

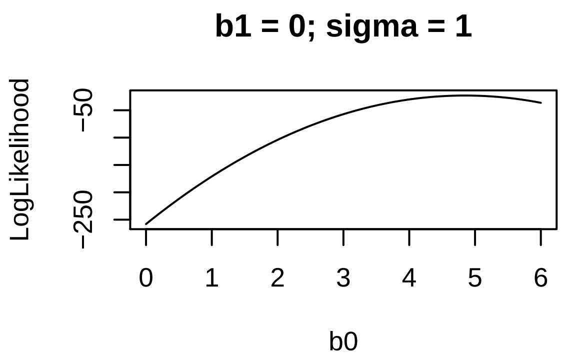
```
par(mfrow = c(2,2))
log_density = dnorm(weight, 0 + gp * 1, 1, log = TRUE)
plot(weight, log_density, type = 'h',
     main = "b0 = 4; b1 = 1, sigma = 1",
     ylab = "log density")

log_density = dnorm(weight, 4 + gp * 1, 1, log = TRUE)
plot(weight, log_density, type = 'h',
     main = "b0 = 4; b1 = 1, sigma = 1", ylab = "log density")

fx <- function(b0) loglike3(weight, gp, b0, 1, 1)
Likelihood <- Vectorize(fx)
plot(Likelihood, 0, 6, main = "b1 = 1; sigma = 1",
     xlab = "b0", ylab = "LogLikelihood")

fx <- function(b0) loglike3(weight, gp, b0, 0, 1)
Likelihood <- Vectorize(fx)
plot(Likelihood, 0, 6, main = "b1 = 0; sigma = 1",
```

```r
     xlab = "b0", ylab = "LogLikelihood")
```









To find the best start point, we use the following for loops:

```r
theta1 = seq(0, 6, by = 0.01)
theta2 = seq(-1, -0.01, by = 0.01)
theta3 = seq(0.01, 1, by = 0.01)

llmax = -Inf

for(t1 in theta1){
  for(t2 in theta2){
    for (t3 in theta3){
      lltheta = loglike4(weight, t1, t2, t3)
      if (lltheta > llmax){
        llmax = lltheta
        start = c(t1,t2,t3)
      }
    }
  }
}

start
```

```
## [1]  5.03 -0.37  0.66
```

Substitude (5.03, -0.37, 0.66) as start points for b0, b1 and sigma into optim():

```r
loglike5 <- function(theta)
  sum(dnorm(weight, theta[1] + gp * theta[2], theta[3], log = TRUE))
z3 = optim(c(5.03, -0.37, 0.66), loglike5, control=list(fnscale=-1), method="BFGS")
```

```r
params <- z3$par
params
```

```
## [1]  5.0320006 -0.3710020  0.6606541
```

The corresponding answer from lm is shown below, which is very close to our solution.

```r
lm.D9 <- lm(weight ~ group)
coef(lm.D9)
```

```
## (Intercept)    groupTrt
##       5.032      -0.371
```