```
source('knitrOpts.R')
rm(list = objects())
options(digits = 7)
```

STATS 782 **Assignment 2** **2019**

Note:

- Submit a hard copy with a signed, QR-coded coversheet to the SRC and a soft copy of your R code (as Assignment 2) on Canvas. **Ideally**, the soft copy is a .Rmd file (or similar) and the hard copy is produced by "knitting" the .Rmd file (or similar).
- Include everything in the hard copy: R code (tidied up), outputs (including error/warning messages), and your explanations (if any).
- Print some intermediate results to show how your code works step by step, if not obvious.
- Comment your code wherever appropriate, e.g., for functions, blocks of code, and key variables.

1. [10 marks]

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)
```

... and here are the first few rows of data ...

```
> head(sodium)
     Instructor Supplement Sodium
1 Brendon Small          A   1200
2 Brendon Small          A   1400
3 Brendon Small          A   1350
4 Brendon Small          A    950
5 Brendon Small          A   1400
6 Brendon Small          B   1150
```

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.

```
> sodiumSubset <- seq(1, by=5, length.out=12)
> sodiumMat <- matrix(sodium$Sodium[sodiumSubset], ncol=4, byrow=TRUE)
> dimnames(sodiumMat) <- list(Instructor=unique(sodium$Instructor),
                              Supplement=unique(sodium$Supplement))
```

```
> sodiumMat
                Supplement
Instructor           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 of the form ...

$$y_{ij} = \mu + \alpha_i + \beta_j + \epsilon_{ij}$$

... to these data.

```
> r <- sodiumMat
> mu <- mean(sodiumMat)
> r <- r - mu
> r <- sweep(r, 1, apply(r, 1, mean))
> r <- sweep(r, 2, apply(r, 2, mean))
> r
                Supplement
Instructor                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
```

Does it look like this is an appropriate model ?

2. [10 marks]

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

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

We will first assume that all weights are i.i.d. Normal(μ, σ).

We will further assume that σ is the sample standard deviation.

```
> sigma <- sd(weight)
```

```
> sigma
[1] 0.7040281
```

We are going to estimate the mean, μ, for the plant weights using Maximum Likelihood.

The likelihood function is

$$\prod_{i=1}^{n} f(x_i; \mu)$$

where $f(x_i; \mu)$ is the Normal probability density function

$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

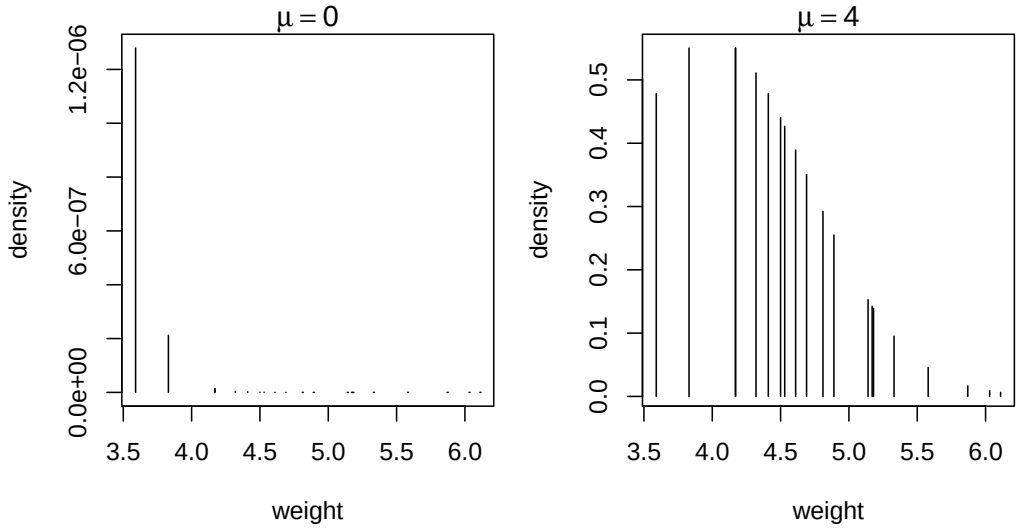
Write an R function called `like` that calculates this likelihood, given a set of data `x` and a mean `mu` (**Hint:** the R function `dnorm` evaluates the Normal probability density given `x`, `mu`, and `sigma`).

```
> like <- function(y, mu) {
      prod(dnorm(y, mu, sigma))
  }
```

```
> like(weight, 0)
[1] 1.359239e-215
> like(weight, 4)
[1] 4.592718e-16
```

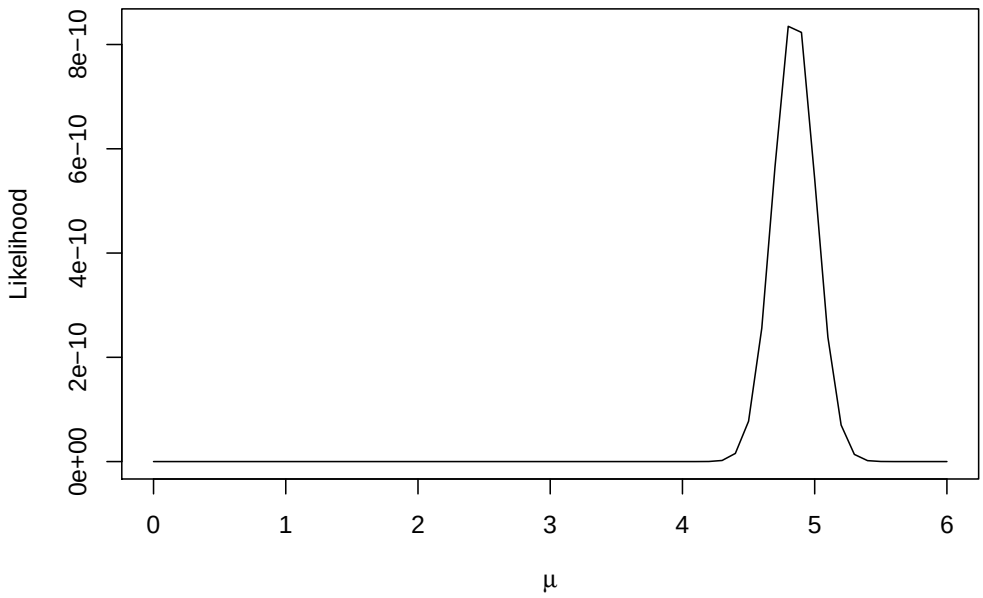
Draw a plot of the probability densities for each weight value for both $\mu = 0$ and $\mu = 4$ (and σ equal to the sample standard deviation).

```
> opar <- par(mfrow=c(1, 2), mar=c(4, 4, 1, 1))
> plot(weight, dnorm(weight, 0, sigma), type="h",
      main=expression(mu == 0), ylab="density")
> plot(weight, dnorm(weight, 4, sigma), type="h",
      main=expression(mu == 4), ylab="density")
> par(opar)
```



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

```
> mus <- seq(0, 6, .1)
> likes <- sapply(mus, function(mu) like(weight, mu))
> plot(mus, likes, type="l", xlab=expression(mu), ylab="Likelihood")
```



The log-likelihood is

$$\sum_{i=1}^{n} \log(f(x_i; \mu))$$

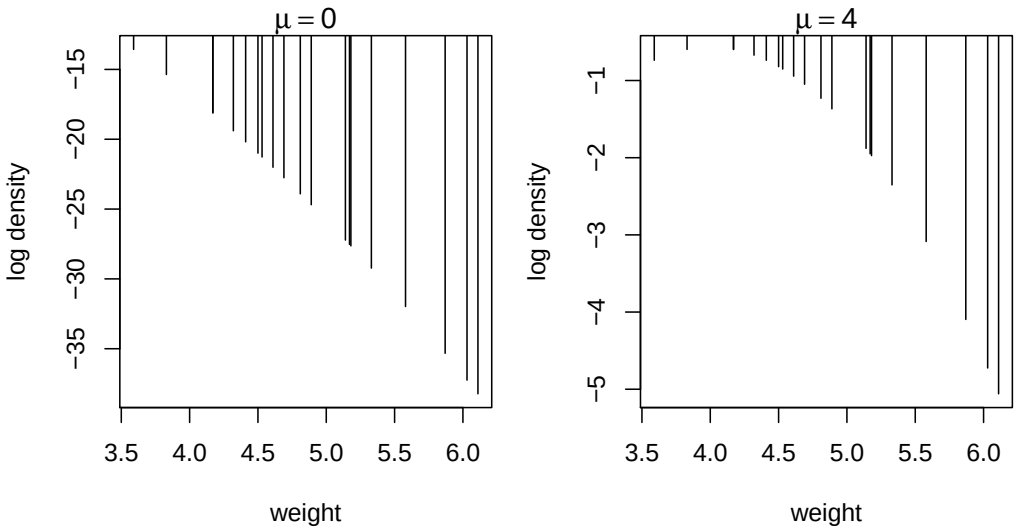
Write a function `loglike` to calculate the log-likelihood (**Hint:** the `dnorm` function has an argument `log`).

```
> loglike <- function(y, mu) {
      sum(dnorm(y, mu, sigma, log=TRUE))
  }
```

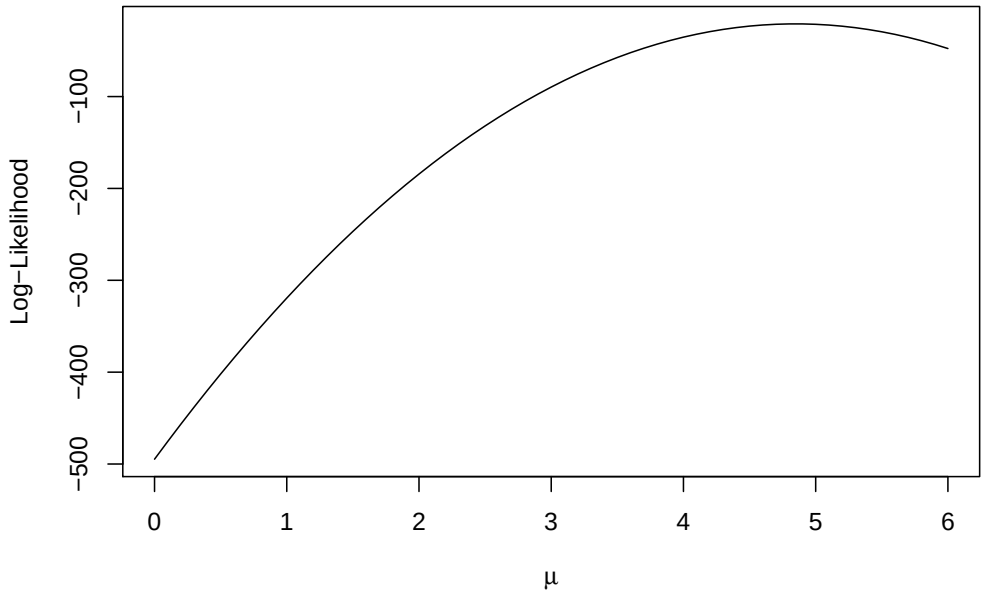
```
> loglike(weight, 0)
[1] -494.7489
> loglike(weight, 4)
[1] -35.31689
```

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

```
> opar <- par(mfrow=c(1, 2), mar=c(4, 4, 1, 1))
> plot(weight, dnorm(weight, 0, sigma, log=TRUE), type="h",
       main=expression(mu == 0), ylab="log density")
> plot(weight, dnorm(weight, 4, sigma, log=TRUE), type="h",
       main=expression(mu == 4), ylab="log density")
> par(opar)
```




```
> mus <- seq(0, 6, .1)
> loglikes <- sapply(mus, function(mu) loglike(weight, mu))
> plot(mus, loglikes, type="l", xlab=expression(mu), ylab="Log-Likelihood")
```



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

```
> llike <- function(data) {
      function(mu) {
          loglike(data, mu)
      }
  }
> muMLE <- optimise(llike(weight), c(0, 10), maximum=TRUE)$maximum
```

```
> muMLE
[1] 4.8465
```

This should equal the sample mean.

```
> mean(weight)
[1] 4.8465
```

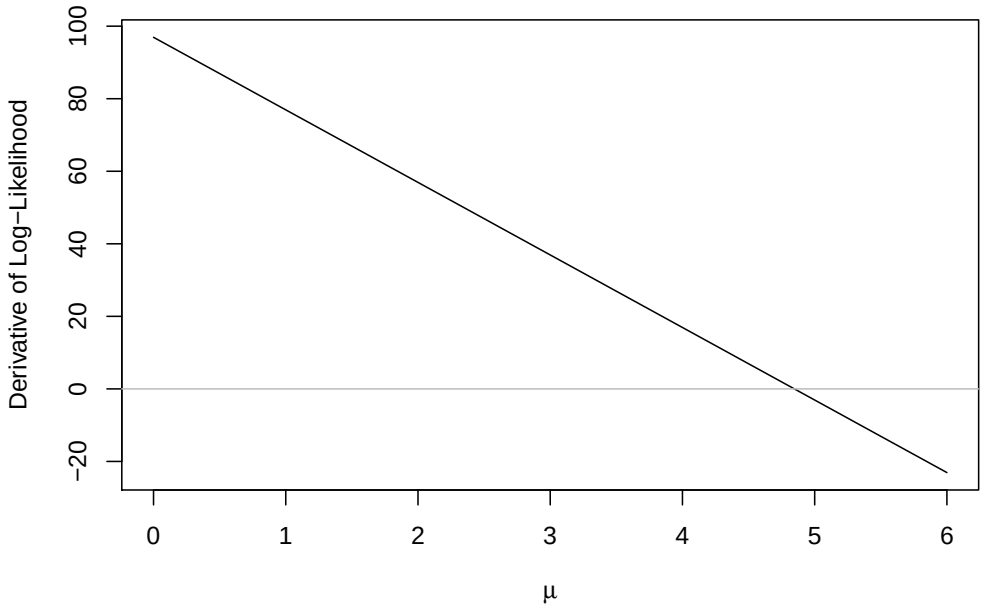
NOTE: using maximum likelihood estimation is NOT how we would normally estimate this parameter (or the parameters in the next two questions), but it can be a useful exercise to help understand how maximum likelihood works.

The first derivative of the log-likelihood function (w.r.t. μ, assuming σ known constant, and data x fixed) is

$$\text{constant} * \sum_{i=1}^{n} x_i - \mu$$

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.

```
> dllike <- function(data) {
      function(mu) {
          sum(data - mu)
      }
  }
> mus <- seq(0, 6, .1)
> dllikes <- sapply(mus, dllike(weight))
> plot(mus, dllikes, type="l", xlab=expression(mu),
       ylab="Derivative of Log-Likelihood")
> abline(h=0, col="grey")
> uniroot(dllike(weight), c(0, 10))$root
[1] 4.8465
```



3. [10 marks]

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

However, we will now estimate both μ and σ using maximum likelihood.

The log-likelihood function is now

$$\sum_{i=1}^{n} \log(f(x_i; \mu, \sigma))$$

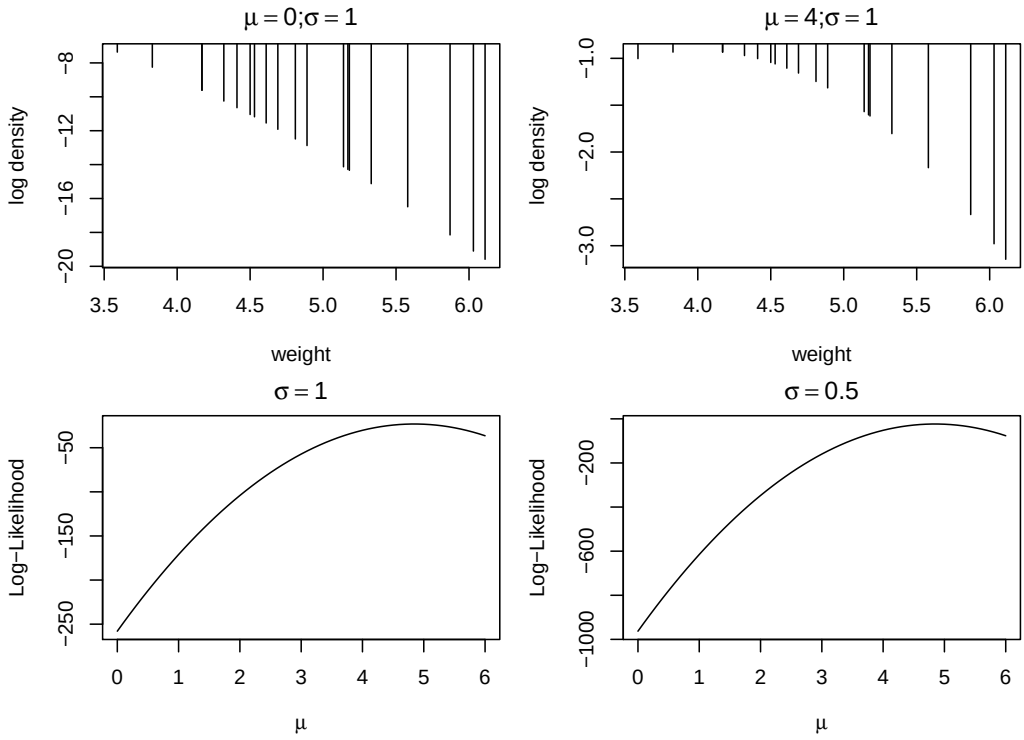
Write a function `loglike2` to evaluate the log-likelihood, plot probability densities values for the weight data for both $\mu = 0; \sigma = 1$ and $\mu = 4, \sigma = 1$, and plot the log-likelihood function for μ between 0 and 6, with $\sigma = 1$ and with $\sigma = .5$.

```
> loglike2 <- function(y, mu, sigma) {
      sum(dnorm(y, mu, sigma, log=TRUE))
  }
```

```
> loglike2(weight, 0, 1)
[1] -257.9731
> loglike2(weight, 4, 1)
[1] -30.25312
```

```
> opar <- par(mfrow=c(2, 2), mar=c(4, 4, 2, 1))
> plot(weight, dnorm(weight, 0, 1, log=TRUE), type="h",
       main=expression(paste(mu == 0, ";", sigma == 1)), ylab="log density")
```

```
> plot(weight, dnorm(weight, 4, 1, log=TRUE), type="h",
      main=expression(paste(mu == 4, ";", sigma == 1)), ylab="log density")
> mus <- seq(0, 6, .1)
> loglikes <- sapply(mus, function(mu) loglike2(weight, mu, 1))
> plot(mus, loglikes, type="l", xlab=expression(mu), ylab="Log-Likelihood",
      main=expression(sigma == 1))
> loglikes <- sapply(mus, function(mu) loglike2(weight, mu, .5))
> plot(mus, loglikes, type="l", xlab=expression(mu), ylab="Log-Likelihood",
      main=expression(sigma == 0.5))
```




```
> sigmas <- seq(.5, 2, .1)
> theta <- expand.grid(mus, sigmas)
> loglikes <- apply(theta, 1, function(theta) loglike2(weight, theta[1], theta[2]))
> par(mfrow=c(1, 1))
> persp(mus, sigmas, matrix(loglikes, nrow=length(mus)), theta=-10, phi=30,
       xlab=expression(mu), ylab=expression(sigma), zlab="Log-Likelihood",
       ticktype="detailed")
```

Use the `optim` function to find the maximum likelihood estimates for μ and σ.

```
> llike2 <- function(data) {
      function(theta) {
          loglike2(data, theta[1], theta[2])
      }
  }
> suppressWarnings(
  muSigmaMLE <- optim(c(0, 1), llike2(weight), control=list(fnscale=-1))$par
  )
```

These should correspond to the sample mean and (almost) the sample standard deviation.

```
> muSigmaMLE
[1] 4.846455 0.686466
```

```
> mean(weight)
[1] 4.8465
> sd(weight)
[1] 0.7040281
> sd(weight)*sqrt((length(weight) - 1)/length(weight))
[1] 0.6862017
```

4. [20 marks]

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

The log-likelihood function now looks like this

$$\sum_{i=1}^{n} \log(f(x_i; \beta_0 + g * \beta_1, \sigma)$$

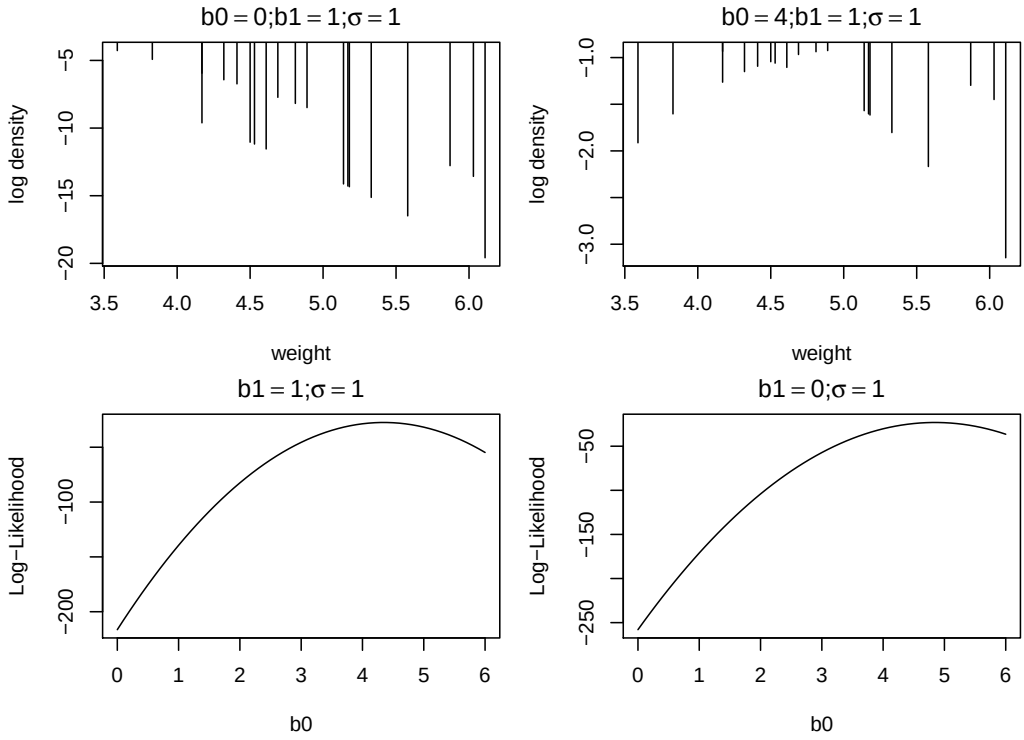
where g is 0 for control weights and 1 for treatment weights.

Find the maximum likelihood estimates for β_0, β_1, and σ.

```
> loglike3 <- function(y, g, b0, b1, sigma) {
      sum(dnorm(y, b0 + g*b1, sigma, 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
```

```
> par(mfrow=c(2, 2), mar=c(4, 4, 2, 1))
> plot(weight, dnorm(weight, gp, 1, log=TRUE), type="h",
       main=expression(paste(b0 == 0, ";", b1 == 1, ";", sigma == 1)),
       ylab="log density")
> plot(weight, dnorm(weight, 4 + gp, 1, log=TRUE), type="h",
       main=expression(paste(b0 == 4, ";", b1 == 1, ";", sigma == 1)),
       ylab="log density")
> b0s <- seq(0, 6, .1)
> loglikes <- sapply(b0s, function(b0) loglike3(weight, gp, b0, 1, 1))
> plot(b0s, loglikes, type="l", xlab=expression(b0), ylab="Log-Likelihood",
       main=expression(paste(b1 == 1, ";", sigma == 1)))
> loglikes <- sapply(b0s, function(b0) loglike3(weight, gp, b0, 0, 1))
> plot(b0s, loglikes, type="l", xlab=expression(b0), ylab="Log-Likelihood",
       main=expression(paste(b1 == 0, ";", sigma == 1)))
```




```
> b1s <- seq(-3, 3, .1)
> theta <- expand.grid(b0s, b1s)
> loglikes <- apply(theta, 1,
                    function(theta) loglike3(weight, gp, theta[1], theta[2], 1))
> par(mfrow=c(1, 1))
> persp(b0s, b1s, matrix(loglikes, nrow=length(b0s)), theta=-20, phi=20,
        xlab=expression(b0), ylab=expression(b1), zlab="Log-Likelihood",
        ticktype="detailed")
```

```
> llike3 <- function(data, gp) {
      function(theta) {
          loglike3(data, gp, theta[1], theta[2], theta[3])
      }
  }
> solution <- optim(c(0, 1, 1), llike3(weight, gp), control=list(fnscale=-1),
                    hessian=TRUE)
> params <- solution$par
```

```
> params
[1]  5.0319929 -0.3709581  0.6604996
```

The corresponding answer from `lm` is shown below.

```
> lm.D9 <- lm(weight ~ group)
> coef(lm.D9)
(Intercept)    groupTrt
      5.032      -0.371
```