Assignment Covering Chapter 4 of *Bayesian Computation with R*

Chris Hayduk

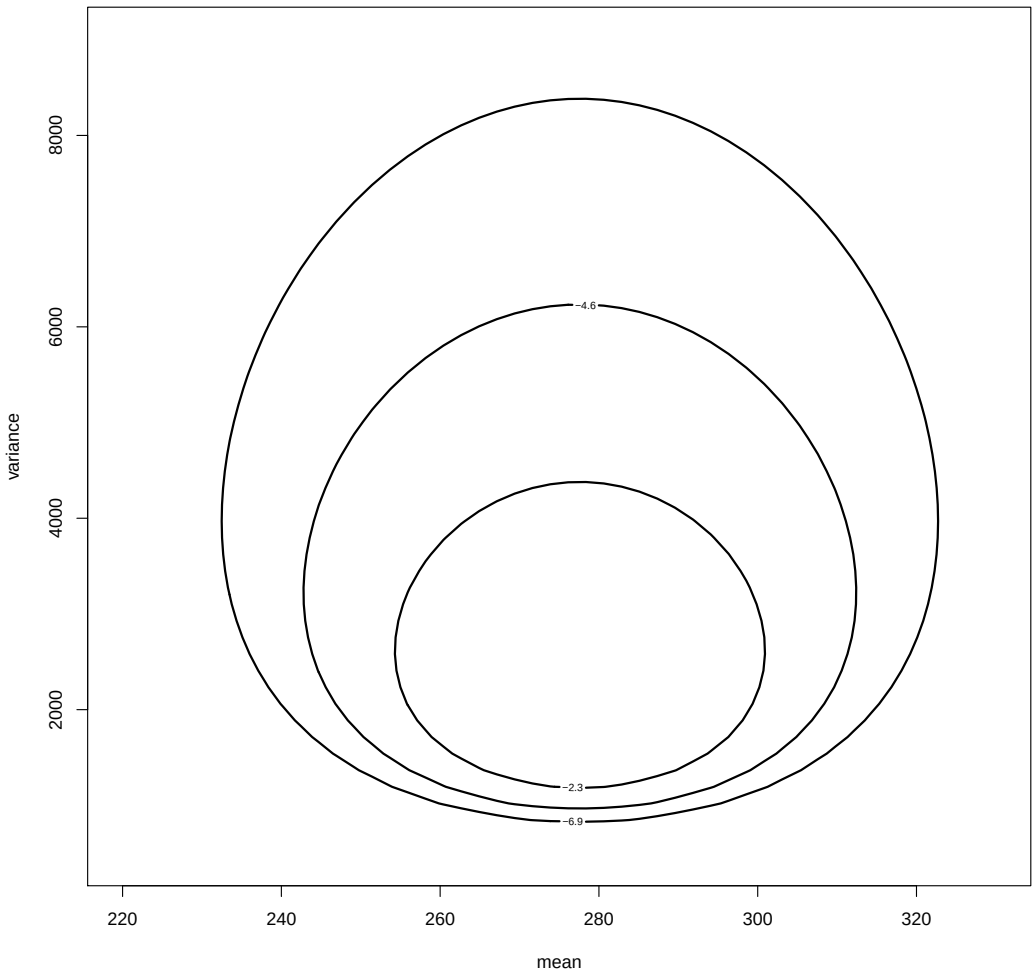
April 29, 2019

1. **Chapter 4: Multiparameter Models** Write your own code for the following:
   (a) `normchi2post()` on pg. 64
   (b) `mycontour()` on pg. 64
   (c) `normpostsim()` on pg. 64
   (d) `rdirichlet()` on pg. 66
   (e) `beta.select()` on pg. 71
   (f) `logisticpost()` on pg. 72
   (g) `howardprior()` on pg. 77

Solution:

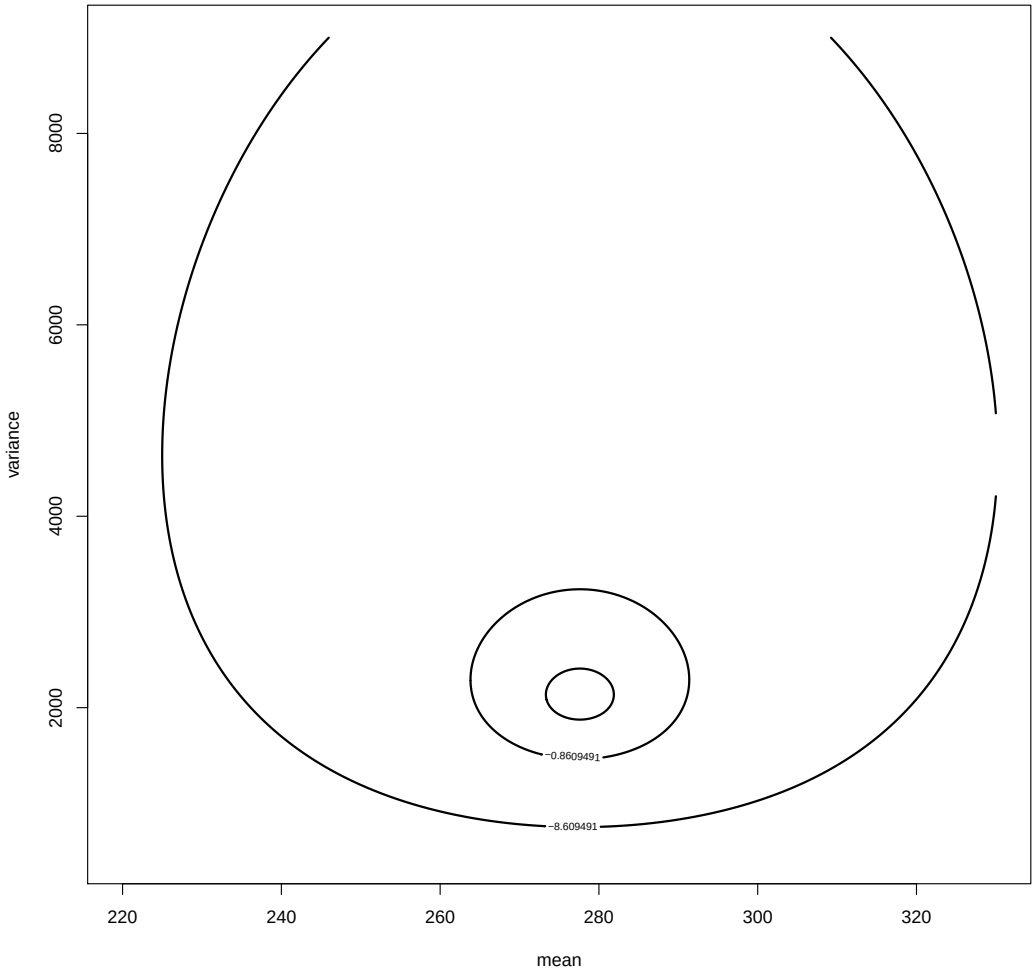
(a)

```r
normchi2post <- function(theta, data){
  mean <- theta[1]
  variance <- theta[2]

  y_bar <- mean(data)

  S <- sum((data-y_bar)^2)
  n <- length(data)

  posterior_density <- 1/((variance^(n/2+1))) *
    exp(-1/(2*variance)*(S+n*(mean-y_bar)^2))

  log_density <- log(posterior_density)
  #print(log_density)
  return(log_density)
}

#Example run

data("marathontimes")
attach(marathontimes)

d <- LearnBayes::mycontour(normchi2post, c(220, 330, 500, 9000), time, xlab="mean",
               ylab="variance")
```


variance
8000
6000
4000
2000
−4.6
−2.3
−6.9
220
240
260
280
300
320
mean

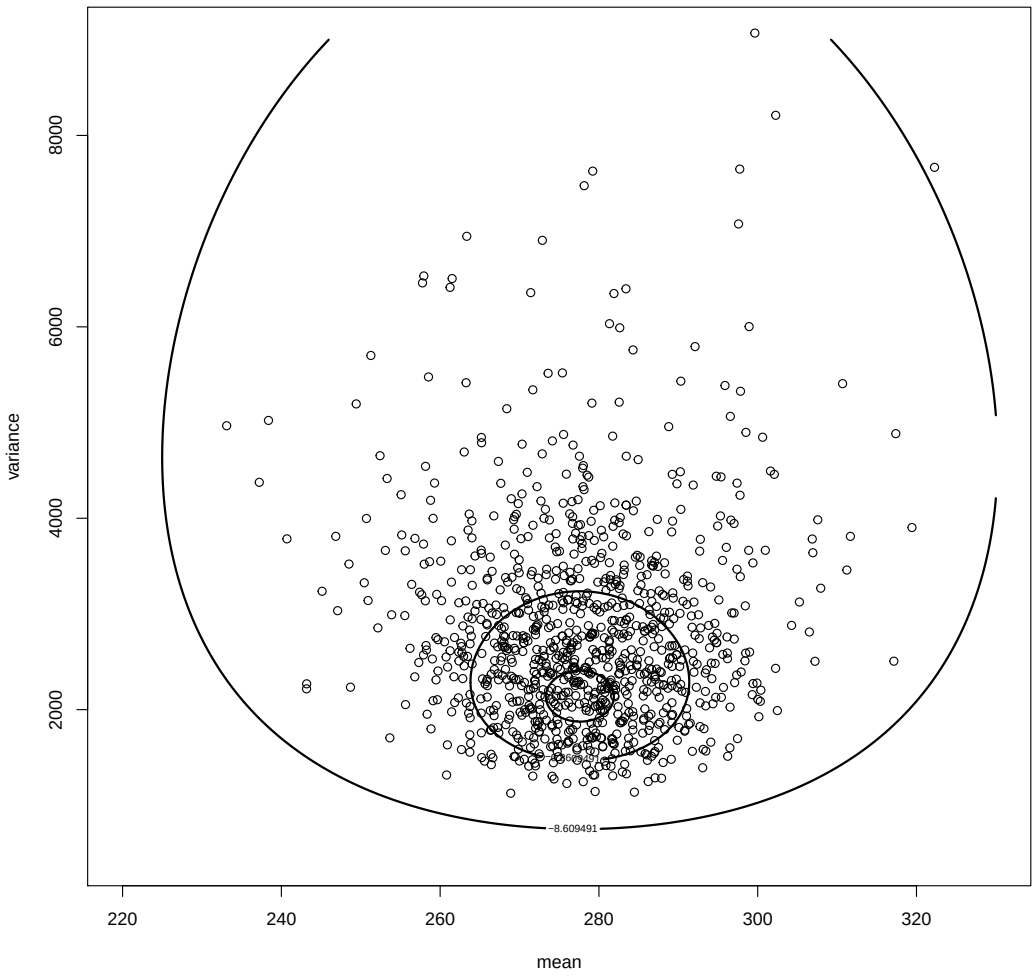
(b)

```
mycontour <- function (logf, limits, data, num_points, levels, ...)
{
    LOGF = function(theta, data) {
        if (is.matrix(theta) == TRUE) {
            val = matrix(0, c(dim(theta)[1], 1))
            for (j in 1:dim(theta)[1]) val[j] = logf(theta[j,
                ], data)
        }
        else val = logf(theta, data)
        return(val)
    }
    ng = num_points
    x0 = seq(limits[1], limits[2], len = ng)
    y0 = seq(limits[3], limits[4], len = ng)
    X = outer(x0, rep(1, ng))
    Y = outer(rep(1, ng), y0)
    n2 = ng^2
    Z = LOGF(cbind(X[1:n2], Y[1:n2]), data)

    #max_z <- optimize(LOGF,
    #interval = c(c(limits[1], limits[2]),
    #c(limits[3], limits[4])), data, maximum = TRUE)
    max_z = max(Z)

    Z = Z - max_z
    Z = matrix(Z, c(ng, ng))

    min_z <- min(Z)

    contours <- c()

    for(i in 1:length(levels)){
      contours <- c(contours, (levels[i]*min_z))
    }

    contour(x0, y0, Z, levels = contours, lwd = 2,
        ...)
}

d <- mycontour(normchi2post, c(220, 330, 500, 9000),
               time, num_points = 1000,
               levels = c(0.1, 0.01, 0.001),
               xlab="mean", ylab="variance")
```


variance
8000
6000
4000
2000
220
240
260
280
300
320
mean
−0.8609491
−8.609491

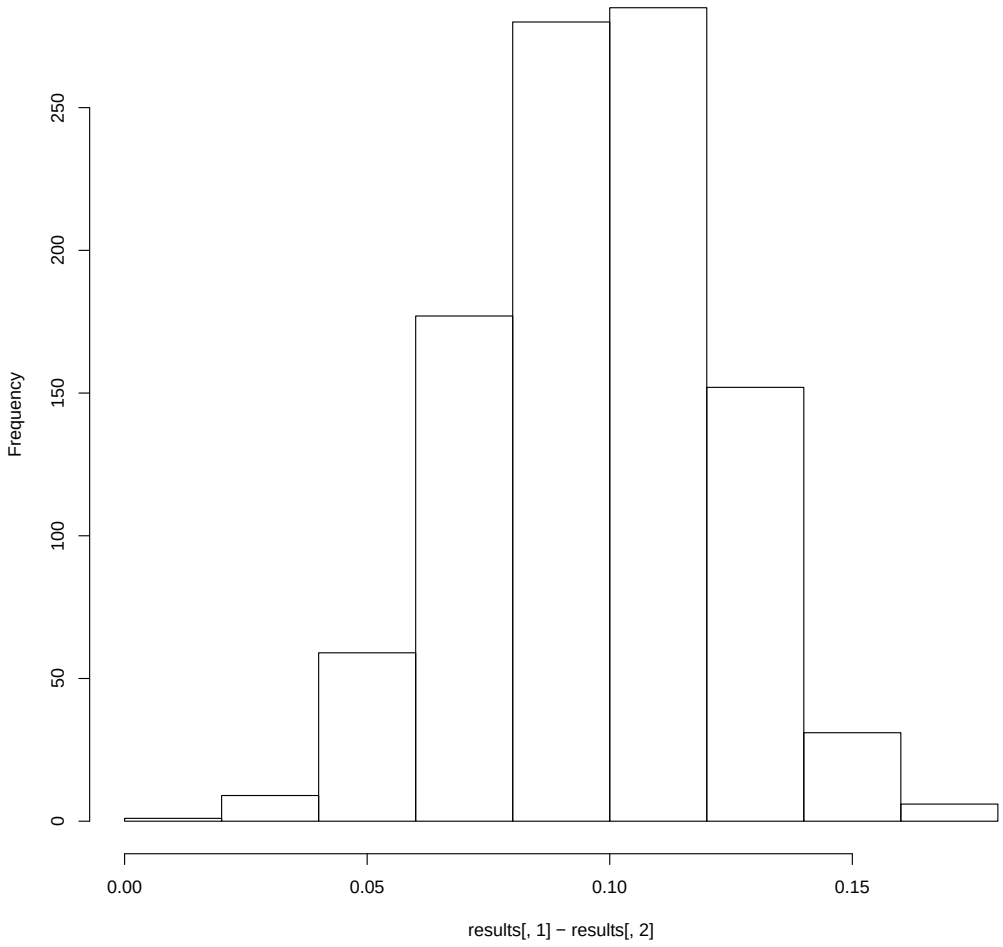
(c)

```r
normpostsim <- function(data, m){
  S <- sum((data-mean(data))^2)

  n <- length(data)

  sigma2 <- S/rchisq(m, n-1)

  mu <- rnorm(m, mean = mean(data), sd = sqrt(sigma2)/sqrt(n))

  results <- data.frame(mu = mu, sigma2 = sigma2)

  return(results)
}

#Example run
results <- normpostsim(time, 1000)
d <- mycontour(normchi2post, c(220, 330, 500, 9000), time,
               num_points = 1000, levels <- c(0.1, 0.01, 0.001),
               xlab="mean", ylab="variance")
points(results$mu, results$sigma2)
```


variance
8000
6000
4000
2000
220
240
260
280
300
320
mean
−8.609491

(d)

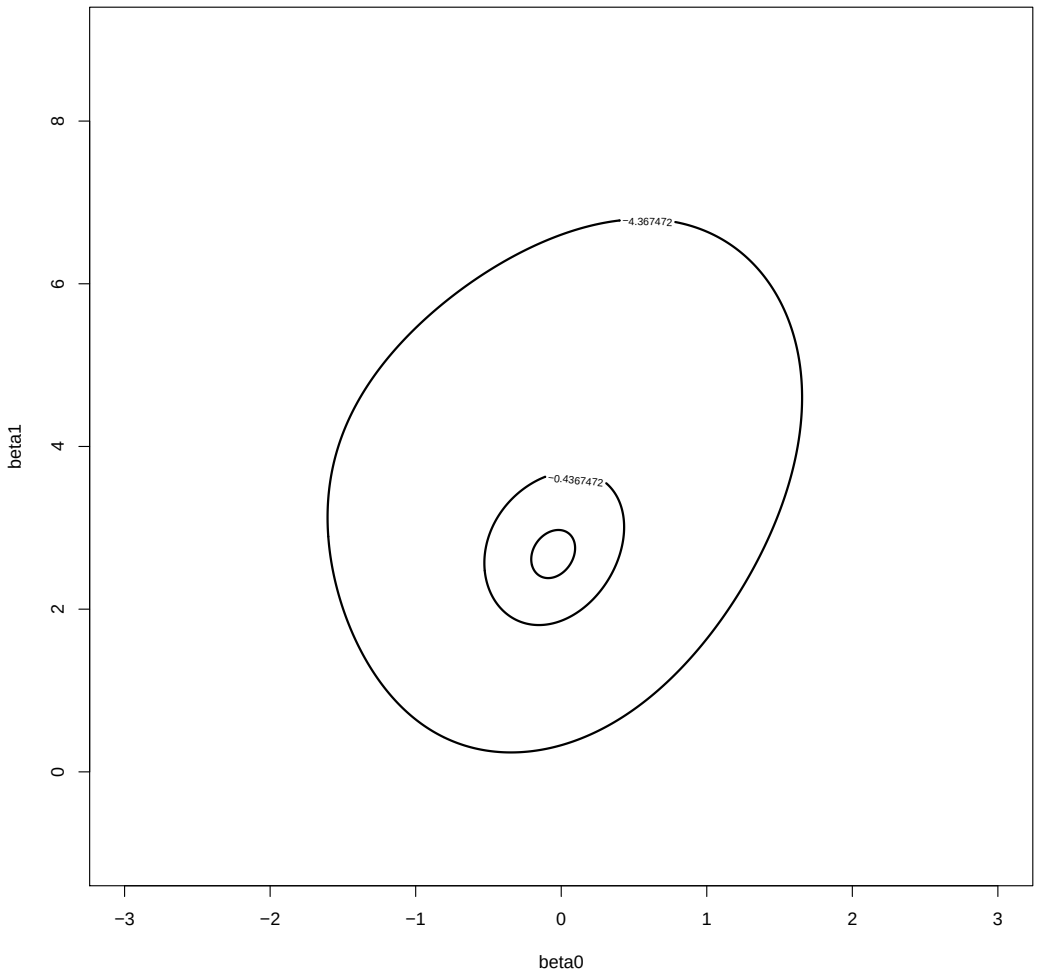
```r
rdirichlet <- function(m, alpha){
  theta <- matrix(nrow = m, ncol = length(alpha))

  for(i in 1:length(alpha)){
    theta[,i] <- rgamma(m, alpha[i], 1)
  }

  for(i in 1:nrow(theta)){
    T = sum(theta[i,])

    theta[i,] <- theta[i,]/T
  }

  return(theta)
}

results <- rdirichlet(1000, c(728,584,138))

hist(results[,1] - results[,2], main = "")
```



(e)

```
beta.select <- function(quantile1, quantile2){

}

#Example run
beta.select(list(p=0.5, x=0.2), list(p=0.9, x=0.5))
## NULL
```

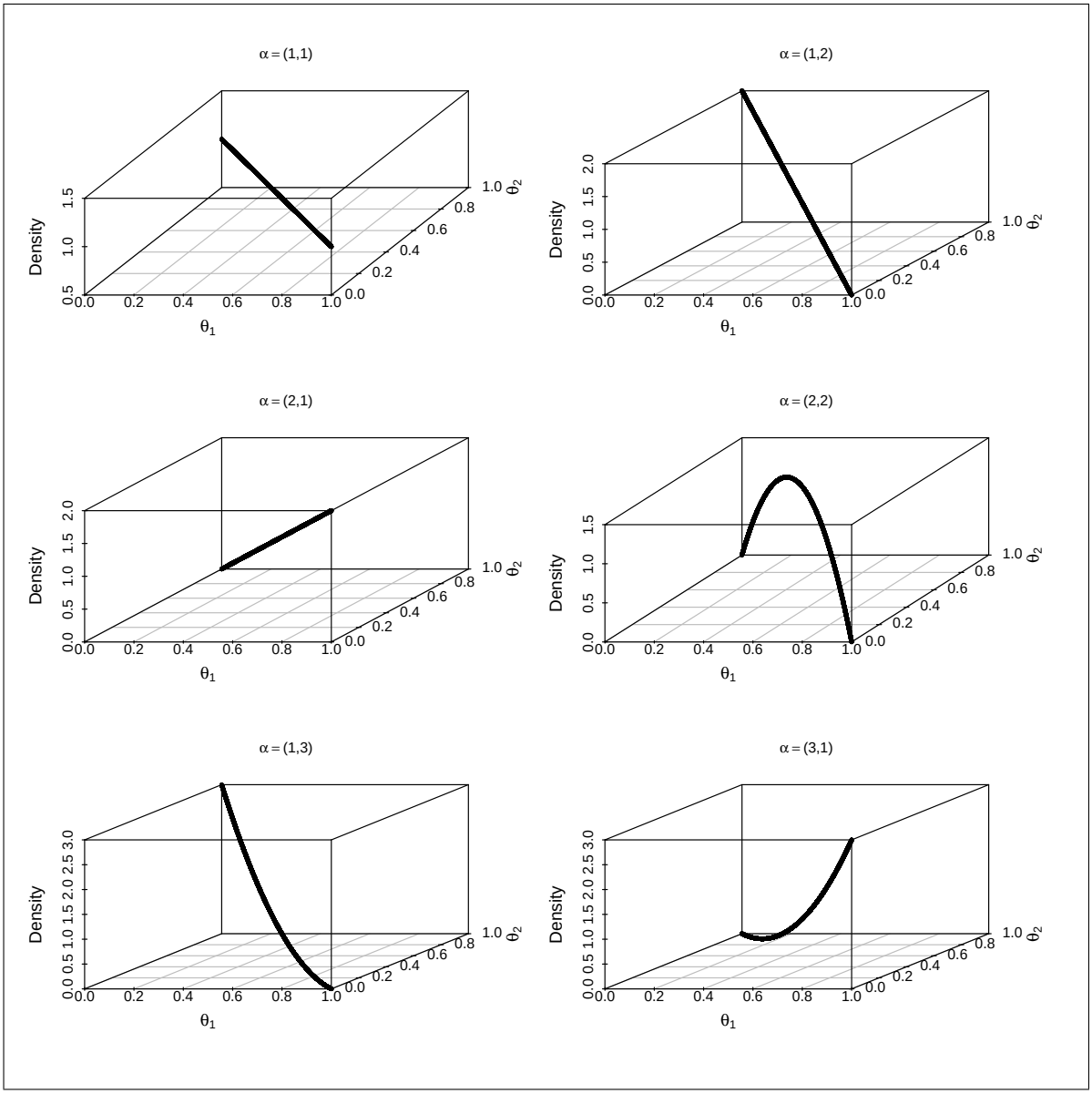
(f)
```r
logisticpost <- function(beta, data){
  post <- 1

  for(i in 1:nrow(data)){
    p <- exp(beta[1] + beta[2]*data[i, 1])

    p <- p/(1+exp(beta[1] + beta[2]*data[i, 1]))

    y <- data[i, 3]
    n <- data[i, 2]

    post <- post * (p^y * (1-p)^(n-y))
  }

  return(log(post))
}

#Example run
x <- c(-0.86, -0.3, -0.05, 0.73)
n <- c(5, 5, 5, 5)
y <- c(0, 1, 3, 5)
data <- cbind(x, n, y)

prior <- rbind(c(-0.7, 4.68, 1.12),
               c(0.6, 2.10, 0.74))

data.new <- rbind(data,prior)

mycontour(logisticpost, c(-3,3,-1,9), data.new,
          num_points = 1000, levels <- c(0.1, 0.01, 0.001),
          xlab="beta0", ylab="beta1")
```



(g)
```
howardprior <- function(){

}
```

2. **Chapter 4: Multiparameter Models** Pg. 66 introduces the Dirichlet distribution. Draw pdfs of this distribution for various parameter values using the functions in the R package. Find some examples on the internet of where the Dirichlet distribution is used in statistics.

Solution:



3. Do all CH. 4 exercises.

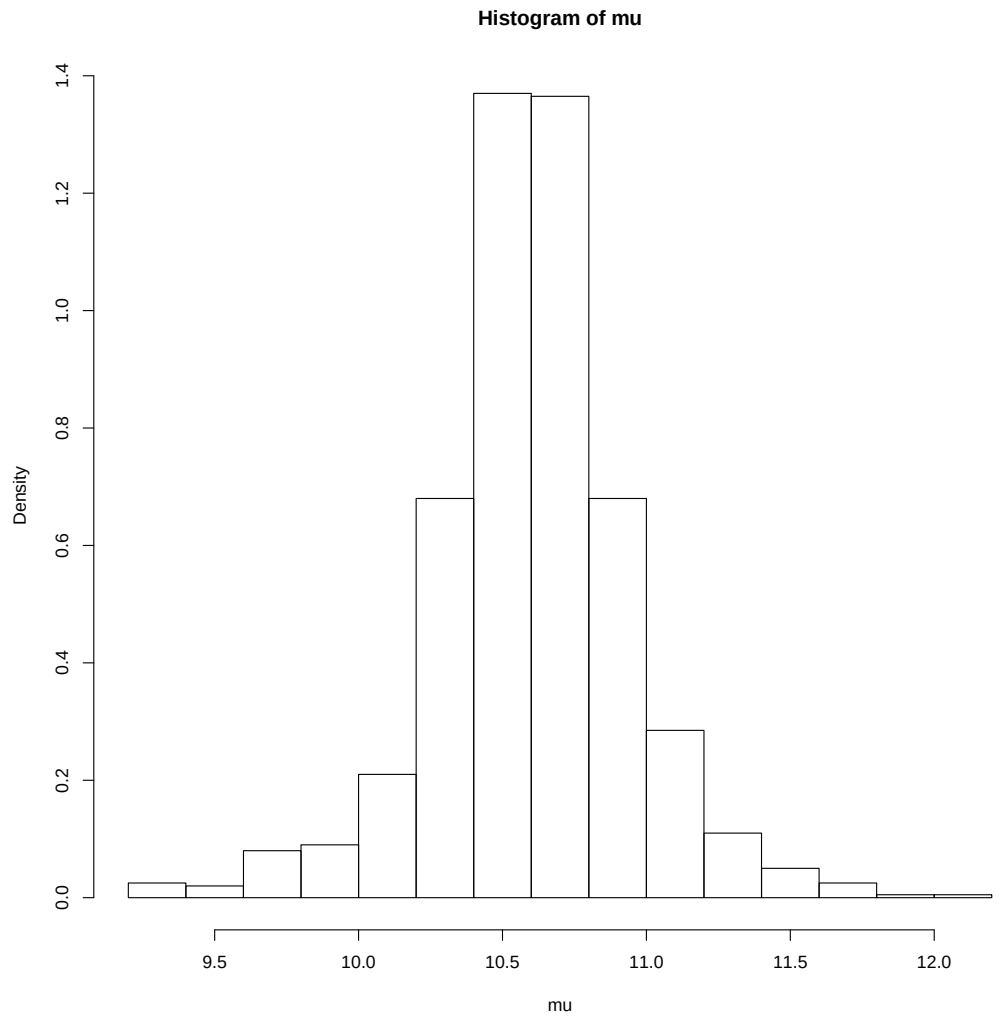
Solution:

1.
```r
#a)
data <- c(9.0, 8.5, 7.0, 8.5, 6.0, 12.5, 6.0, 9.0, 8.5, 7.5,
          8.0, 6.0, 9.0, 8.0, 7.0, 10.0, 9.0, 7.5, 5.0, 6.5)
n <- length(data)
y_bar <- mean(data)

s_2 <- sum((data-y_bar)^2)*(1/(n-1))

sigma_2 <- rinvchisq(1000, n-1, s_2)

mu <- rnorm(1000, y_bar, sigma_2/n)

#b)
quantile(mu, c(0.05, 0.95))

##       5%      95%
## 7.639289 8.192113

sigma <- sqrt(sigma_2)

quantile(sigma, c(0.05, 0.95))

##       5%      95%
## 1.359014 2.303797

#c)
p_75 <- mu + 0.674*sigma

mean(p_75)

## [1] 9.119036

sd(p_75)

## [1] 0.2585771
```

2.
```r
#c)
males <- c(120, 107, 110, 116, 114, 111, 113, 117, 114, 112)

females <- c(110, 111, 107, 108, 110, 105, 107, 106, 111, 111)

S1 <- var(males)

S2 <- var(females)

n <- length(males)

sigma2_males <- S1/rchisq(1000,n-1)
sigma2_females <- S2/rchisq(1000,n-1)

mu_males <- rnorm(1000, mean=mean(males), sd = sqrt(sigma2_males)/sqrt(n))
mu_females <- rnorm(1000, mean=mean(females), sd = sqrt(sigma2_females)/sqrt(n))

mu_diff <- mu_males - mu_females

sum(mu_diff > 0)/length(mu_diff)

## [1] 1
```

3.
```r
#b)
```

4.
```r
#a)
weights <- c(10, 11, 12, 11, 9)

S <- var(weights)

n <- length(weights)

sigma2 <- S/rchisq(1000, n-1)

mu <- rnorm(1000, mean = mean(weights), sd = sqrt(sigma2)/sqrt(n))

hist(mu, freq = FALSE, breaks = 15)
```


Histogram of mu
Density
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
9.5
10.0
10.5
11.0
11.5
12.0
mu

```
#b)

#c)

#d)
```

5.

6.

7.

8.
```
#a)
quantile1 <- list(p=0.25, x=0.15)
quantile2 <- list(p=0.75, x=0.35)

beta1 <- beta.select(quantile1, quantile2)

print(beta1)

## NULL

quantile1 <- list(p=0.25, x=0.75)
quantile2 <- list(p=0.75, x=0.95)

beta2 <- beta.select(quantile1, quantile2)

print(beta2)

## NULL

#b)
data <- cbind(c(16, 18, 20, 22, 24, 26, 28), c(0, 0, 6, 12, 7, 9, 3), c(2, 7, 14, 26, 13, 14
#prior <- c(beta1, beta2)
#data.new <- rbind(data, prior)
mycontour(logisticpost, c(-3, 3, -1, 9), data, xlab="beta0",
          ylab = "beta1")

## Error in mycontour(logisticpost, c(-3, 3, -1, 9), data, xlab = "beta0", :  argument
"num_points" is missing, with no default

#c)

#d)
```

4. **CH4, Q8 extra:** fit model using a frequentist logistic regression model; figure out how to make the equivalent confidence interval for part (d). (Could use bootstrap, probably BC_a and its approximation ABC, try package ***bootBCa***; include a histogram of your BC_a bootstrap resampled probability values and normal quantile plot of those values. Let number of bootstrap samples be $B = 9,999$.)

Solution:

```r
act <- c(rep(16, times = 2), rep(18, times = 7), rep(20, times = 14), rep(22, times = 26),
         rep(24, times = 13), rep(26, times = 14), rep(28, times = 3))

success <- c(rep(0, times = 9), rep(1, times = 6), rep(0, times = 14-6),
             rep(1, times = 12), rep(0, times = 26-12),
             rep(1, times = 7), rep(0, times = 13-7),
             rep(1, times = 9), rep(0, times = 14-9),
             rep(1, times = 3))

data <- data.frame(act = act, success = success)

model <- glm(success ~ act, family="binomial", data = data)

summary(model)
```

```
##
## Call:
## glm(formula = success ~ act, family = "binomial", data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6008  -1.0638  -0.6339   1.0363   1.5701
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.03444    2.23042  -3.154  0.00161 **
## act          0.30732    0.09839   3.123  0.00179 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 109.201  on 78  degrees of freedom
## Residual deviance:  97.413  on 77  degrees of freedom
## AIC: 101.41
##
## Number of Fisher Scoring iterations: 4
```

```r
new.data <- data.frame(act = c(20))
predict(model, newdata = new.data, type="response")
```

```
##         1
## 0.2915336
```