

Assignment Covering Chapter 4 of *Bayesian Computation with R*

Chris Hayduk

April 15, 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) 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))

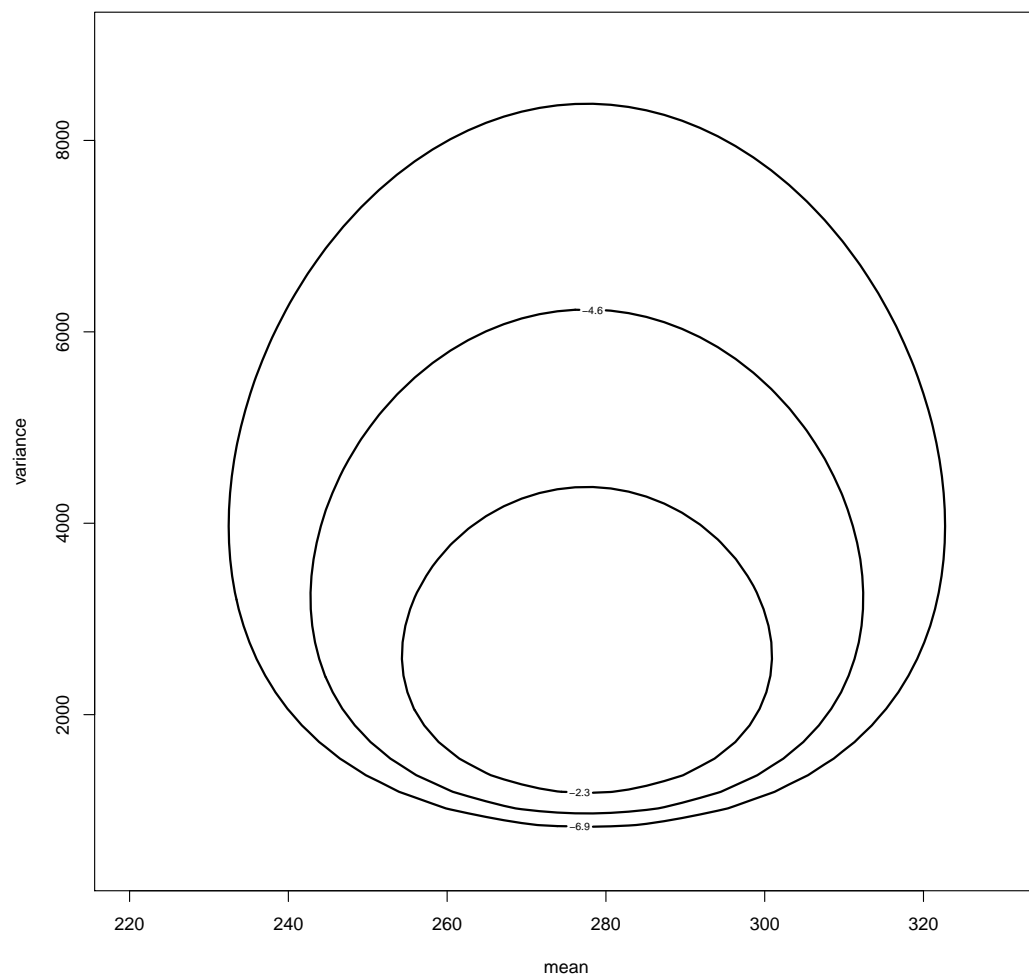
  log_density <- log(posterior_density)
  #print(log_density)
  return(log_density)
}

#Example run

data("marathontimes")
attach(marathontimes)

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

```



```

(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)

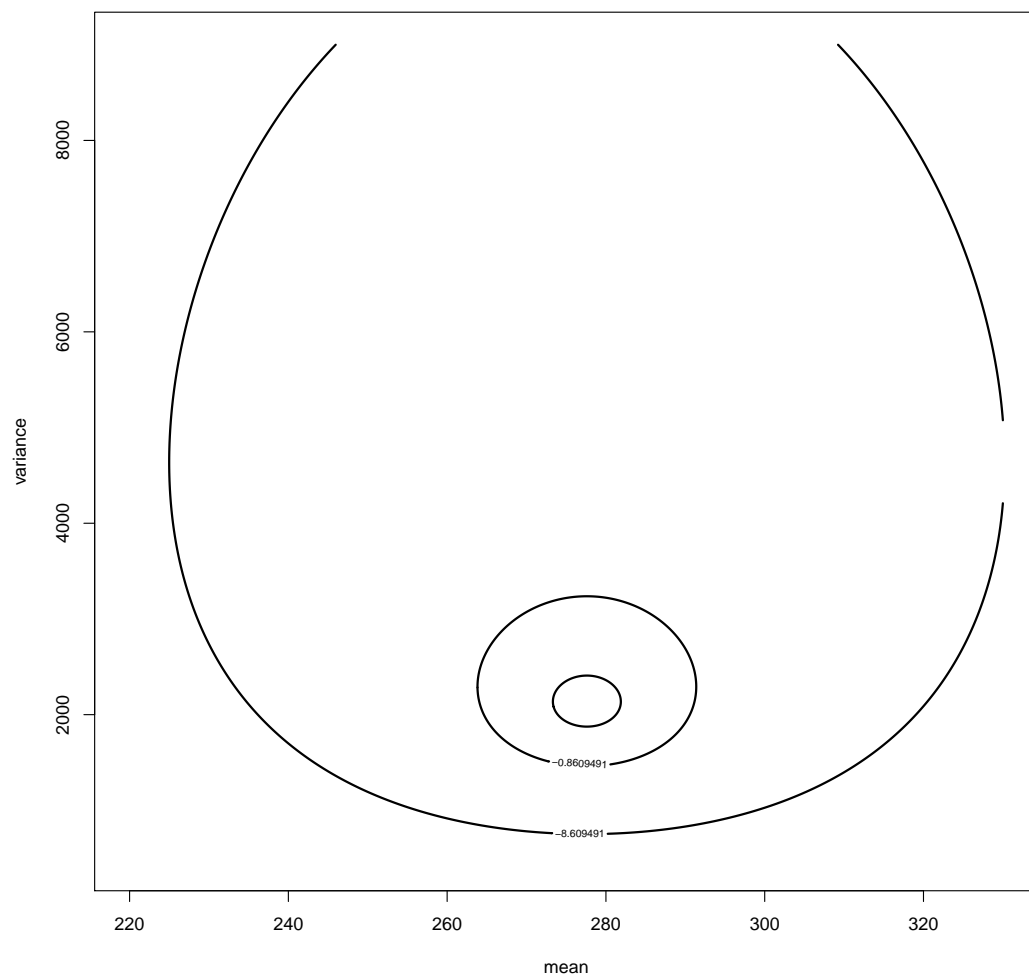
  contours <- c()

  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")

```



```

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

  n <- length(data)

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

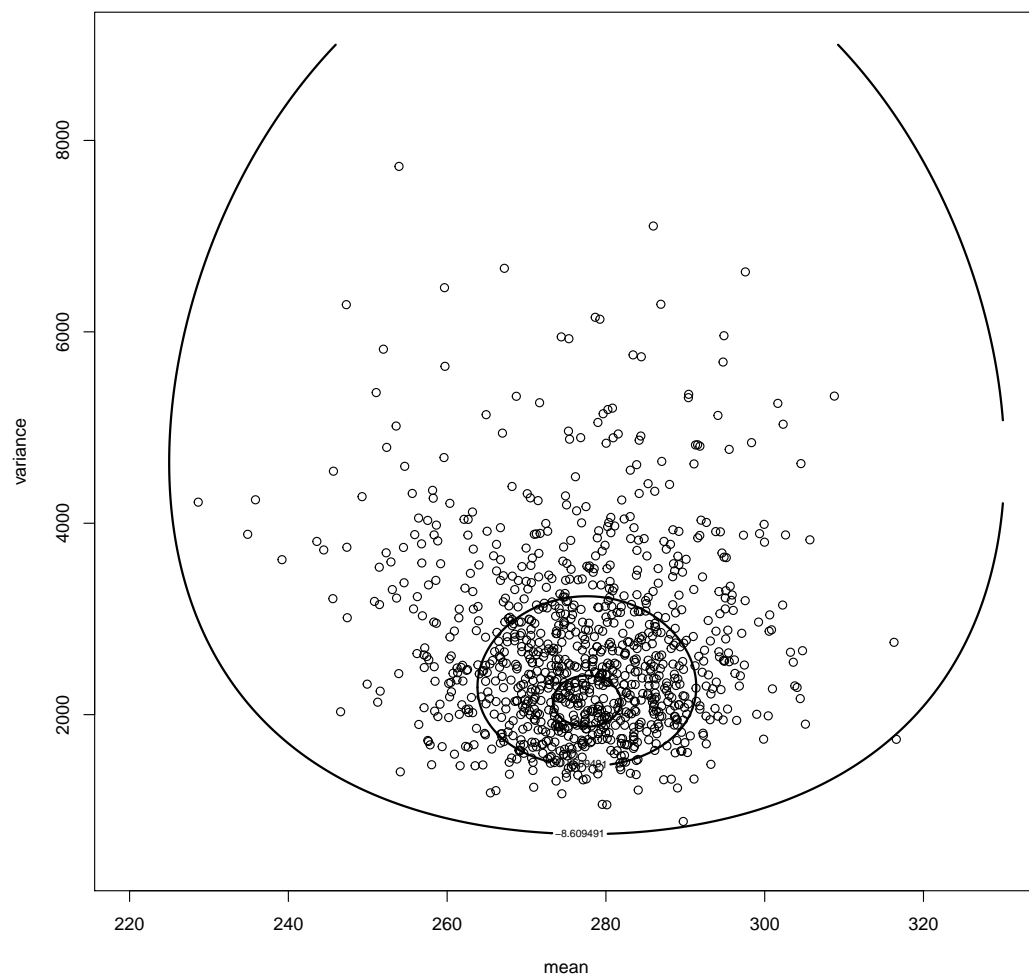
  mu <- rnorm(m, mean = mean(data), sd = sqrt(sigma2)/sqrt(n))

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

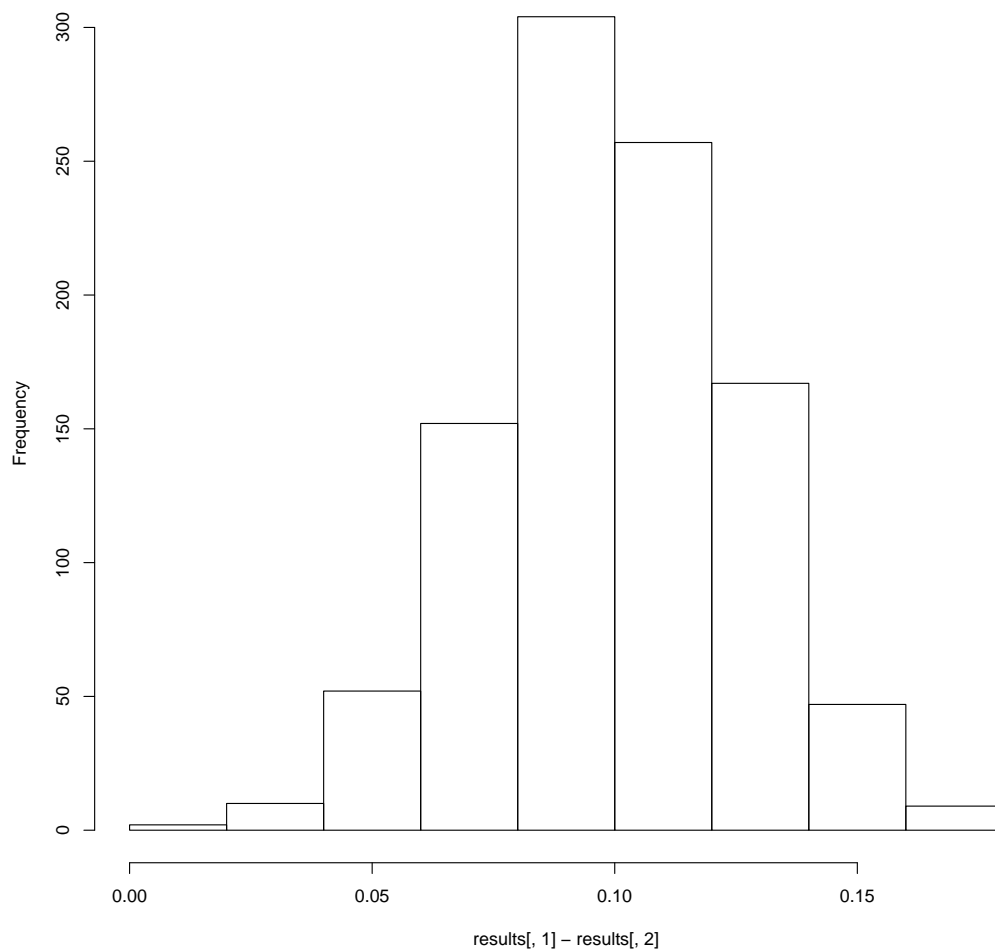
  return(results)
}

#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)

```



```
(d) 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) 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))
}

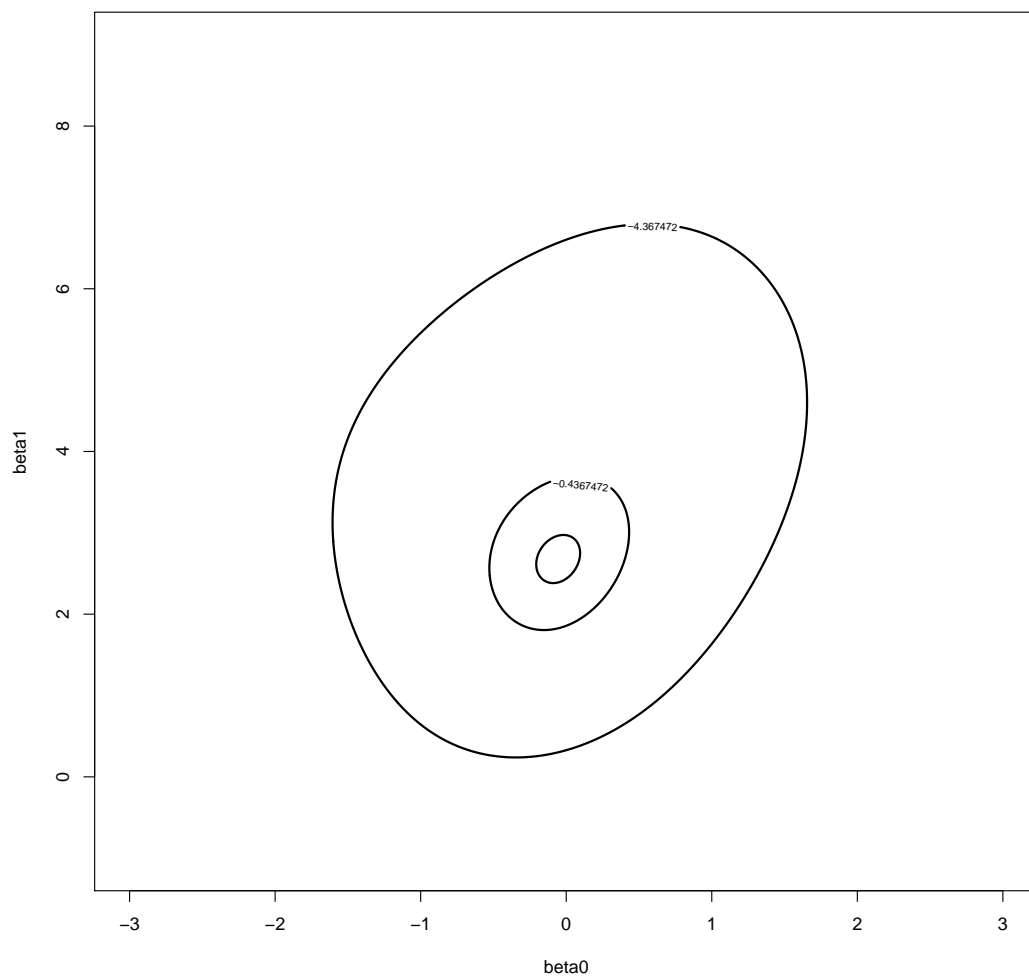
#Example run
x <- c(-0.86, -0.3, -0.05, 0.73)
n <- c(5, 5, 5, 5)
y <- c(0, 1, 3, 5)
data <- cbind(x, n, y)

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")

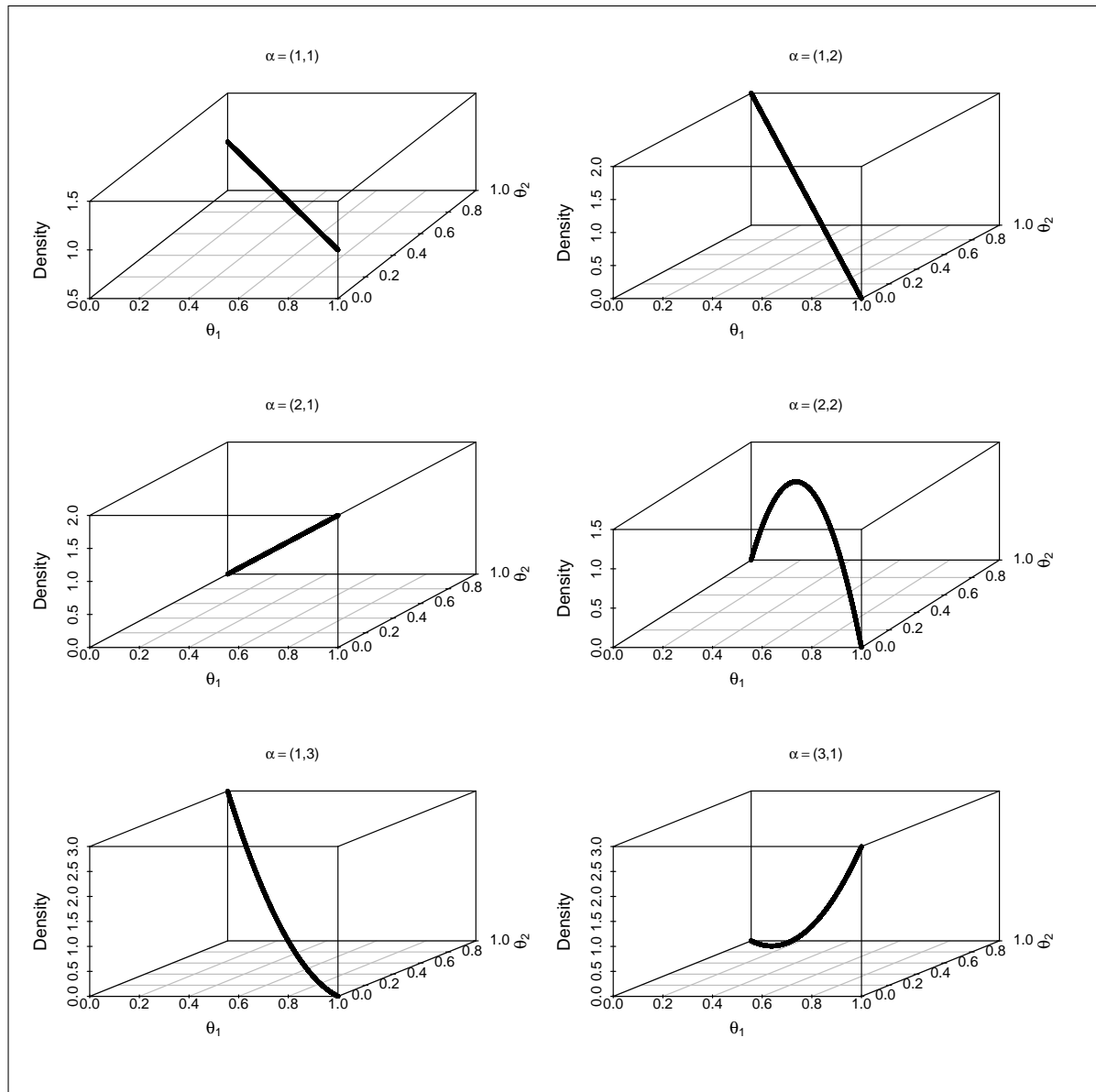
```



```
(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. #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%
## 7.633139 8.230009

sigma <- sqrt(sigma_2)

quantile(sigma, c(0.05, 0.95))

##          5%          95%
## 1.334067 2.306614

#c)
p_75 <- mu + 0.674*sigma

mean(p_75)

## [1] 9.111376

sd(p_75)

## [1] 0.2723855

```

2.

3.

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: