HW6

Jaime Romero Florez

2023-12-04

Question 1

```
# Load data
file <- "http://www2.stat.duke.edu/~pdh10/FCBS/Exercises/azdiabetes.dat"
data <- read.table(file, header=T)

# Split by diabetes status
y.D <- data[data$diabetes=="Yes", 1:7]
y.N <- data[data$diabetes=="No", 1:7]

# Set up priors
ybar <- colMeans(y.D) # sample means
Lambda0 <- cov(y.D) # sample covariance
nu.0 <- 9
S.0 <- Lambda0
p <- dim(y.D)[2]
n <- dim(y.D)[1]
Cov.y <- cov(y.D)


# Gibbs sampling for diabetics
S <- 10000
mu.0 <- ybar;  Sigma <- S.0;  # Initial values

# Calculations that will be used repeatedly
Lambda0.inv <- solve(Lambda0)
Lam.inv.mu.0 <- Lambda0.inv %*% mu.0
nu.n <- nu.0 + n

# Now generate the Markov chain!  mu and Sigma
muD.chain <- matrix(NA, S, p)
SigmaD.chain <- matrix(NA, S, p^2)

for(s in 1:S)
{
 n.Sigma.inv <- n * solve(Sigma)
 Lambda.n <- solve( Lambda0.inv + n.Sigma.inv )
 mu.n <- Lambda.n %*% (Lam.inv.mu.0 + n.Sigma.inv %*% ybar)
 mu <- rmvnorm(1, mu.n, Lambda.n)[1,]
 S.n <- S.0 + (n-1)*Cov.y + n * (ybar-mu) %*% t(ybar-mu)
```

```
 Sigma <- solve( rWishart(1, nu.n, solve(S.n))[,,1] )
 muD.chain[s,] <- mu
 SigmaD.chain[s,] <- Sigma
}

# Gibbs sampling for non-diabetics
ybar <- colMeans(y.N) # sample means
Lambda0 <- cov(y.N) # sample covariance
nu.0 <- 9
S.0 <- Lambda0
p <- dim(y.N)[2]
n <- dim(y.N)[1]
Cov.y <- cov(y.N)

# Gibbs sampling for diabetics
S <- 10000
mu.0 <- ybar;  Sigma <- S.0;  # Initial values

# Calculations that will be used repeatedly
Lambda0.inv <- solve(Lambda0)
Lam.inv.mu.0 <- Lambda0.inv %*% mu.0
nu.n <- nu.0 + n

# Now generate the Markov chain!  mu and Sigma
muN.chain <- matrix(NA, S, p)
SigmaN.chain <- matrix(NA, S, p^2)

for(s in 1:S)
{
 n.Sigma.inv <- n * solve(Sigma)
 Lambda.n <- solve( Lambda0.inv + n.Sigma.inv )
 mu.n <- Lambda.n %*% (Lam.inv.mu.0 + n.Sigma.inv %*% ybar)
 mu <- rmvnorm(1, mu.n, Lambda.n)[1,]
 S.n <- S.0 + (n-1)*Cov.y + n * (ybar-mu) %*% t(ybar-mu)
 Sigma <- solve( rWishart(1, nu.n, solve(S.n))[,,1] )
 muN.chain[s,] <- mu
 SigmaN.chain[s,] <- Sigma
}


# (a) Compare marginal posteriors
par(mfrow=c(2,4))
for(j in 1:7){
  den.D = density(muD.chain[,j], adjust=2)
  den.N = density(muN.chain[,j], adjust=2)
  plot(NA, xlim=range(c(den.D$x, den.N$x)),
       ylim=c(0, max(c(den.D$y, den.N$y))),
       main=paste("Variable", names(y.D)[j]),
       xlab="Mean", ylab="Density")
  lines(den.D, col="red", lwd=2)
  lines(den.N, col="blue", lwd=2, lty=2)
}
legend("topright", inset=.05, lty=1:2, col=c("red","blue"),
```

```
       legend=c("Diabetics","Non-diabetics"), lwd=2)

# Compute Pr(muD > muN)
apply(muD.chain > muN.chain, 2, mean)
```

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

```
# (b) Compare variance matrices
vars <- colnames(data)[1:7]

SigmaD.hat <- matrix(apply(SigmaD.chain, 2, mean), p, p)
rownames(SigmaD.hat) <- vars;  colnames(SigmaD.hat) <- vars;
round(SigmaD.hat, 2)
```

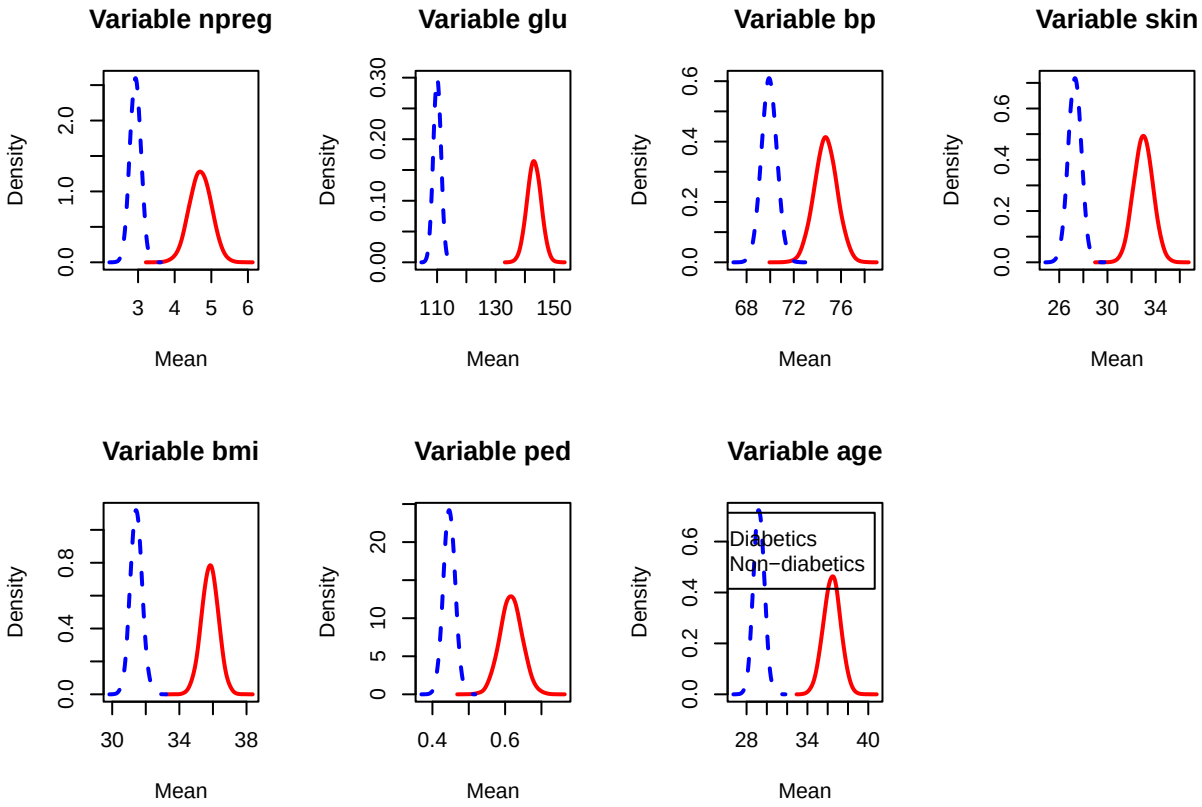
```
##       npreg    glu     bp   skin    bmi   ped    age
## npreg 15.37  -9.82   6.07  -4.13  -4.66 -0.10  23.53
## glu   -9.82 978.30  33.41  31.28  10.36  0.24  35.06
## bp     6.07  33.41 156.71  12.49  18.04 -0.18  36.28
## skin  -4.13  31.28  12.49 107.80  35.51  0.54  -7.38
## bmi   -4.66  10.36  18.04  35.51  43.75  0.39 -13.78
## ped   -0.10   0.24  -0.18   0.54   0.39  0.16  -0.15
## age   23.53  35.06  36.28  -7.38 -13.78 -0.15 117.41
```

```
SigmaN.hat <- matrix(apply(SigmaN.chain, 2, mean), p, p)
rownames(SigmaN.hat) <- vars;  colnames(SigmaN.hat) <- vars;
round(SigmaN.hat, 2)
```

```
##       npreg    glu     bp   skin   bmi   ped   age
## npreg  7.76   4.61   6.65   3.68  0.01 -0.04 18.28
## glu    4.61 589.43  56.28  32.50 25.51  0.66 43.02
## bp     6.65  56.28 141.74  28.62 23.06 -0.13 39.49
## skin   3.68  32.50  28.62 101.57 44.29  0.06 17.50
## bmi    0.01  25.51  23.06  44.29 42.85  0.10  4.49
## ped   -0.04   0.66  -0.13   0.06  0.10  0.09  0.07
## age   18.28  43.02  39.49  17.50  4.49  0.07 97.99
```

```
# (c) Compute predictive probability means are higher
y.tilde.D <- matrix(NA, S, p)
y.tilde.N <- matrix(NA, S, p)

for (s in 1:S) {
  y.tilde.D[s,] <- mvrnorm(1, muD.chain[s,], SigmaD.hat)
  y.tilde.N[s,] <- mvrnorm(1, muN.chain[s,], SigmaN.hat)
}

## now, computing Y.tilde.d.j > Y.tilde.n.j | Y

#print results
for (i in 1:7) {
  cat("For ", vars[i], ": ", mean(y.tilde.D[,i] > y.tilde.N[,i]), "\n")
}
```

```
## For  npreg :  0.6523
## For  glu :  0.8044
## For  bp :  0.6194
## For  skin :  0.65
## For  bmi :  0.6729
## For  ped :  0.6314
## For  age :  0.6904
```



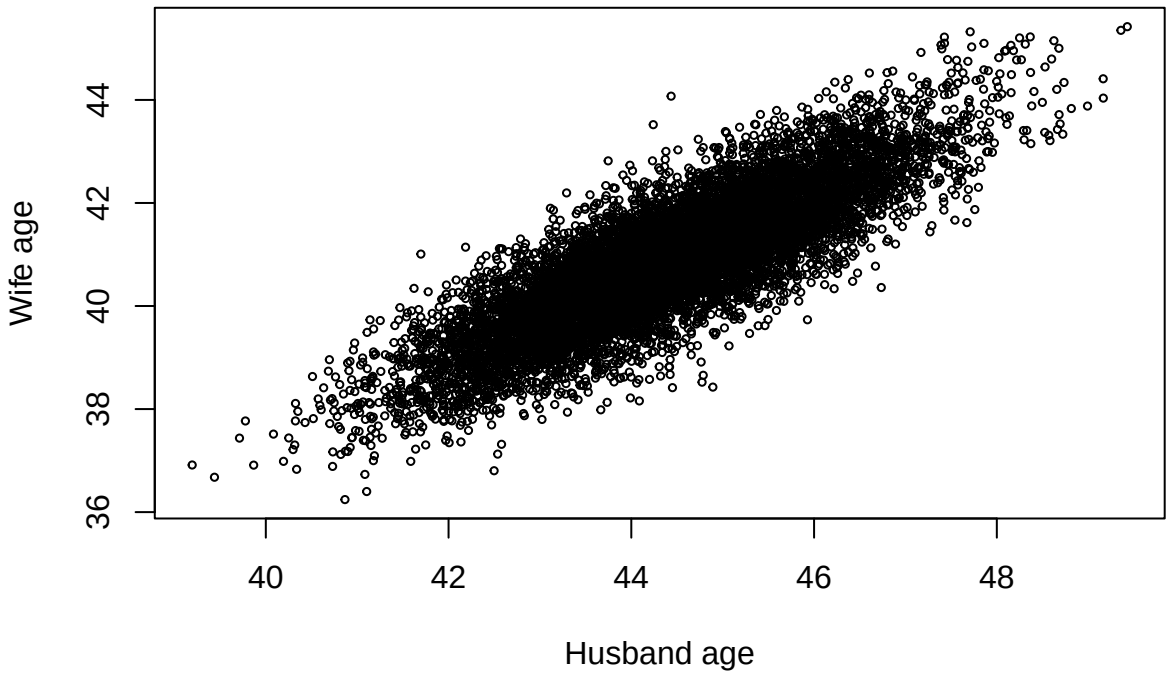
The probabilities are not unexpected when looking at the plots. For all variables, the distributions of diabetics to non-diabetics differs significantly, with diabetics having a higher average for all variables. Thus, the probability that diabetics have a higher variable observation than non-diabetics should be high. In this case, as we are looking at the posterior predictive model using our MCMC results, the probabilities are lower than for the in-sample observation, given the extra level of uncertainty from the predictive nature of the PPD.

Question 2

```
file <- "http://www2.stat.duke.edu/~pdh10/FCBS/Exercises/agehw.dat"
Data <- read.table(file=file, header=T);  rm(file);
```

a)

```
# Set up priors
ybar <- colMeans(Data) # sample means
Cov.y <- cov(Data) # sample covariance
nu.0 <- 3
S.0 <- 1000 * matrix(c(1,0,0,1), nrow = 2, ncol = 2)
Lambda0 <- 10e5 * matrix(c(1,0,0,1), nrow = 2, ncol = 2)
p <- dim(Data)[2]
n <- dim(Data)[1]


# Gibbs sampling for diabetics
S <- 10000
mu.0 <- c(0, 0);  Sigma <- S.0;  # Initial values

# Calculations that will be used repeatedly
Lambda0.inv <- solve(Lambda0)
Lam.inv.mu.0 <- Lambda0.inv %*% mu.0
nu.n <- nu.0 + n

# Now generate the Markov chain!  mu and Sigma
mu.chain <- matrix(NA, S, p)
Sigma.chain <- matrix(NA, S, p^2)

for(s in 1:S)
{
 n.Sigma.inv <- n * solve(Sigma)
 Lambda.n <- solve( Lambda0.inv + n.Sigma.inv )
 mu.n <- Lambda.n %*% (Lam.inv.mu.0 + n.Sigma.inv %*% ybar)
 mu <- rmvnorm(1, mu.n, Lambda.n)[1,]
 S.n <- S.0 + (n-1)*Cov.y + n * (ybar-mu) %*% t(ybar-mu)
 Sigma <- solve( rWishart(1, nu.n, solve(S.n))[,,1] )
 mu.chain[s,] <- mu
 Sigma.chain[s,] <- Sigma
}

plot(mu.chain, xlab="Husband age", ylab="Wife age", cex=.5)
```




```
# table

probs <- c(.025, .25, .5, .75, .975)
Quants <- apply(mu.chain, 2, quantile, probs=probs)
Quants <- t(Quants); rownames(Quants) <- c("Husband", "Wife")
round(Quants, 2)
```

```
##          2.5%   25%   50%   75% 97.5%
## Husband 41.77 43.49 44.38 45.35 47.10
## Wife    38.35 40.01 40.86 41.74 43.39
```

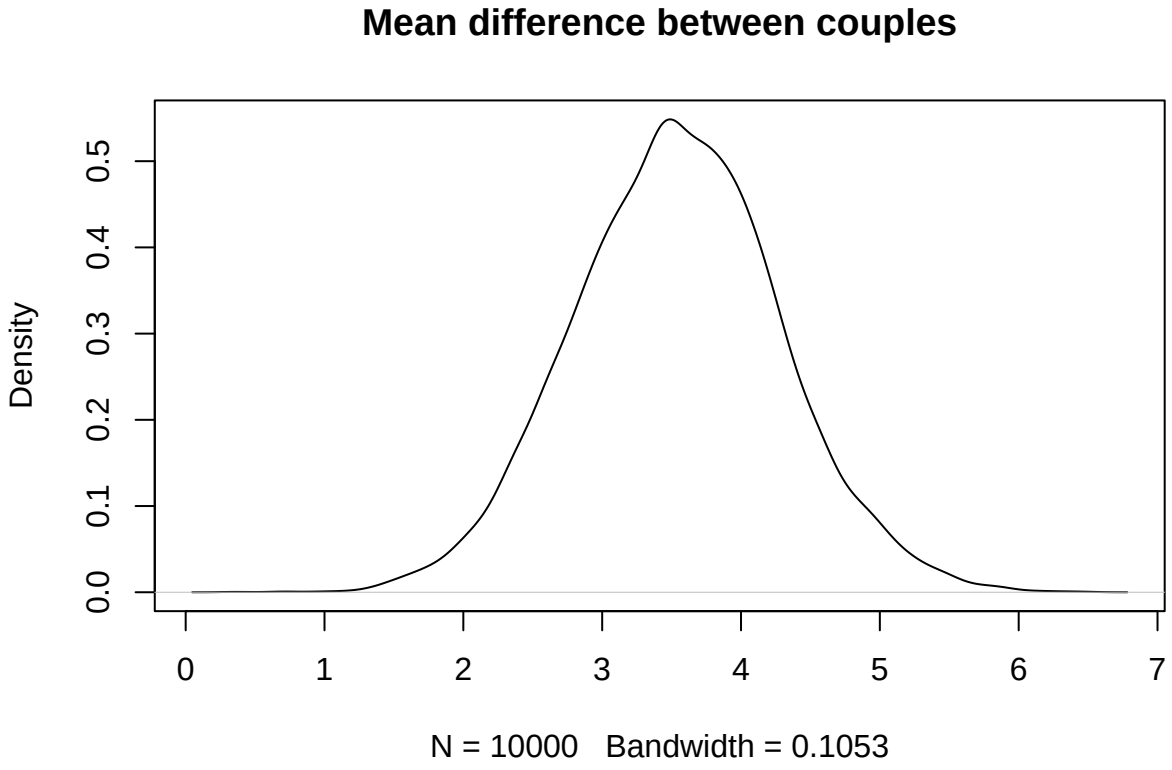
b)

```
rho.chain <- Sigma.chain[,2]/sqrt(Sigma.chain[,1]*Sigma.chain[,4])
Quants <- quantile(rho.chain, probs = probs)
Quants <- t(Quants)
rownames(Quants) <- c("rho")
round(Quants, 2)
```

```
##     2.5%  25%  50%  75% 97.5%
## rho 0.79 0.83 0.85 0.87   0.9
```

c)

```
diff.chain <- mu.chain[,1] - mu.chain[,2]

plot(density(diff.chain), main = "Mean difference between couples")
```




```
## Pr(mu_h > mu_w | y)
mean(mu.chain[,1] > mu.chain[,2])
```

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

The probability is very close to 1. This can be seen from the kernel density of the differences.

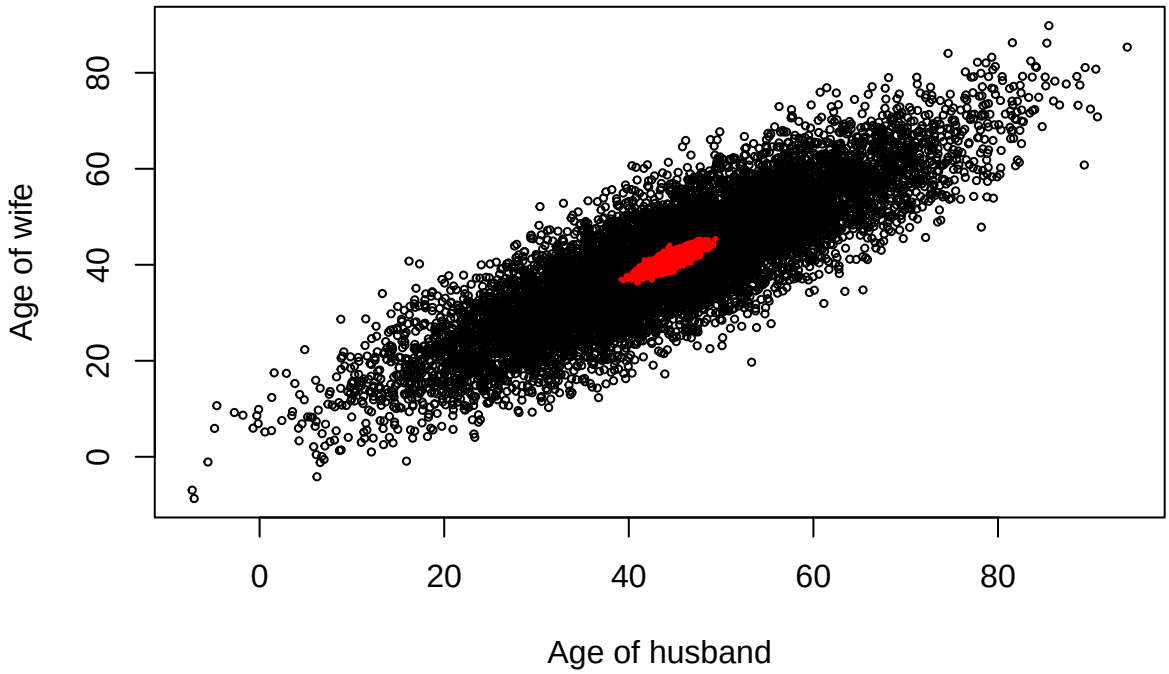
```
vars <- colnames(Data)
# first I will compute the Sigma.hat
Sigma.hat <- matrix(apply(Sigma.chain, 2, mean), p, p)
rownames(Sigma.hat) <- vars;  colnames(Sigma.hat) <- vars;
round(Sigma.hat, 2)
```

```
##        ageh   agew
## ageh 195.26 157.14
## agew 157.14 173.60
```

```
# now, I will find the PPD
y.tilde <- matrix(NA, S, p)
for (s in 1:S) {
  y.tilde[s,] <- mvrnorm(1, mu.chain[s,], Sigma.hat)
}

# Pr(Y.tilde.h > Y.ilde.w | y)
mean(y.tilde[,1] > y.tilde[,2])
```

```
## [1] 0.685
```

```r
# plot
plot(y.tilde, xlab="Age of husband", ylab="Age of wife", cex=.5)
points(mu.chain, pch=19, cex=.25, col="red")
```



Now, the probability is closer to 70% This is because of the added variability when constructing a predictive distribution. In part c, our data informed the distribution and the results were valid within the sample. If we create a predictive model, we are inherently introducing a higher variance to the model, which makes our inference slightly less certain. That is why our probability went from nearly 100% to less than 70%.

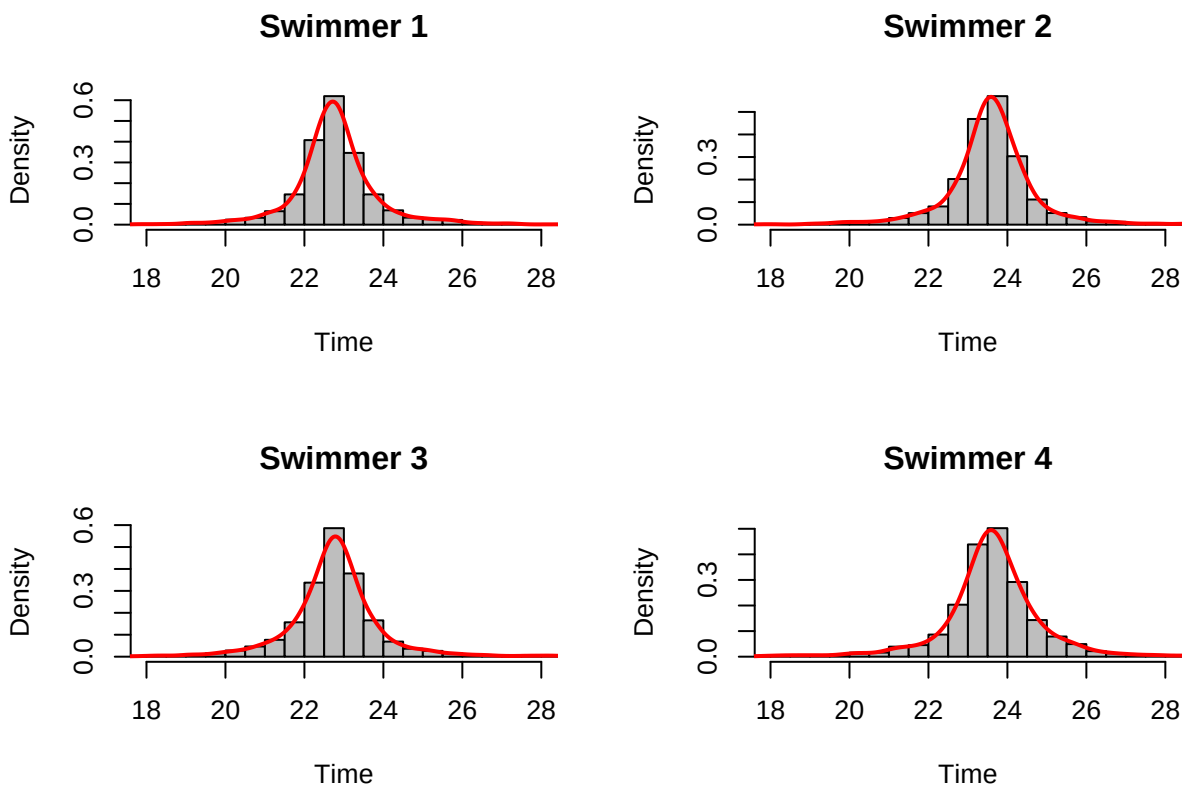
Question 3

```
file <- "http://www2.stat.duke.edu/~pdh10/FCBS/Exercises/swim.dat"
Data <- read.table(file=file, header=F)
rm(file)
```

```
# Bayesian analysis using semiconjugate prior
beta.0 <- c(23,0)
Sigma.0 <- matrix(c(1/4, 0, 0, 1/4), ncol = 2, nrow = 2)
nu.0 <- 1
sigma2.0 <- 1/4
p <- 6
n <- 1
S <- 5000

Sigma0.Inv <- solve(Sigma.0)  # invert once, not every time

# Starting values
beta <- beta.0;  sigma2 <- sigma2.0;

gibbs.regression <- function(y, X){
  # Generate the Markov chain
  beta.chain <- matrix(NA, S, 2)
  sigma2.chain <- rep(NA, S)

    for(s in 1:S) {
      ###
      # Update beta first
      ###
      V.beta <- solve( Sigma0.Inv + t(X) %*% X / sigma2 )
      m.beta <- V.beta %*% (Sigma0.Inv %*% beta.0 + t(X) %*% y / sigma2)
      beta <- rmvnorm(1, m.beta, V.beta)[1,]
      ###
      # Now update sigma2
      ###
      SSR <- sum( (y - X %*% beta)^2 )
      sigma2 <- (nu.0*sigma2.0 + SSR) / rchisq(1, nu.0 + n)
      ###
      # Now save updated values
      ###
      beta.chain[s,] <- beta;  sigma2.chain[s] <- sigma2;
    }

    results <- list(beta.chain = beta.chain, sigma2.chain = sigma2.chain)

    return(results)

}
```

a)

```
y1 <- t(Data[1,])
y2 <- t(Data[2,])
y3 <- t(Data[3,])
y4 <- t(Data[4,])

#create X matrix
X <- as.matrix(cbind(c(rep(1, 6)), c(1:6)))

# Use gibbs for all results
model1 <- gibbs.regression(y1, X)
model2 <- gibbs.regression(y2, X)
model3 <- gibbs.regression(y3, X)
model4 <- gibbs.regression(y4, X)

# posterior estimates for each model
estimates1 <- vector(length = S)
estimates2 <- vector(length = S)
estimates3 <- vector(length = S)
estimates4 <- vector(length = S)

for (s in 1:S) {
  estimates1[s] <- model1$beta.chain[s,1] + 7 * model1$beta.chain[s,2]
  estimates2[s] <- model2$beta.chain[s,1] + 7 * model2$beta.chain[s,2]
  estimates3[s] <- model3$beta.chain[s,1] + 7 * model3$beta.chain[s,2]
  estimates4[s] <- model4$beta.chain[s,1] + 7 * model4$beta.chain[s,2]
}


# Scatterplot

op <- par(mfrow=c(2,2))

hist(estimates1, freq=F, right=F, breaks=50, col="gray", xlim = c(18,28),
  xlab="Time", main="Swimmer 1")

lines(density(estimates1, adj=2), lwd=2, col="red")

hist(estimates2, freq=F, right=F, breaks=50, col="gray", xlim = c(18,28),
  xlab="Time", main="Swimmer 2")

lines(density(estimates2, adj=2), lwd=2, col="red")

hist(estimates3, freq=F, right=F, breaks=50, col="gray", xlim = c(18,28),
  xlab="Time", main="Swimmer 3")

lines(density(estimates3, adj=2), lwd=2, col="red")

hist(estimates4, freq=F, right=F, breaks=50, col="gray", xlim = c(18,28),
  xlab="Time", main="Swimmer 4")

lines(density(estimates4, adj=2), lwd=2, col="red")
```

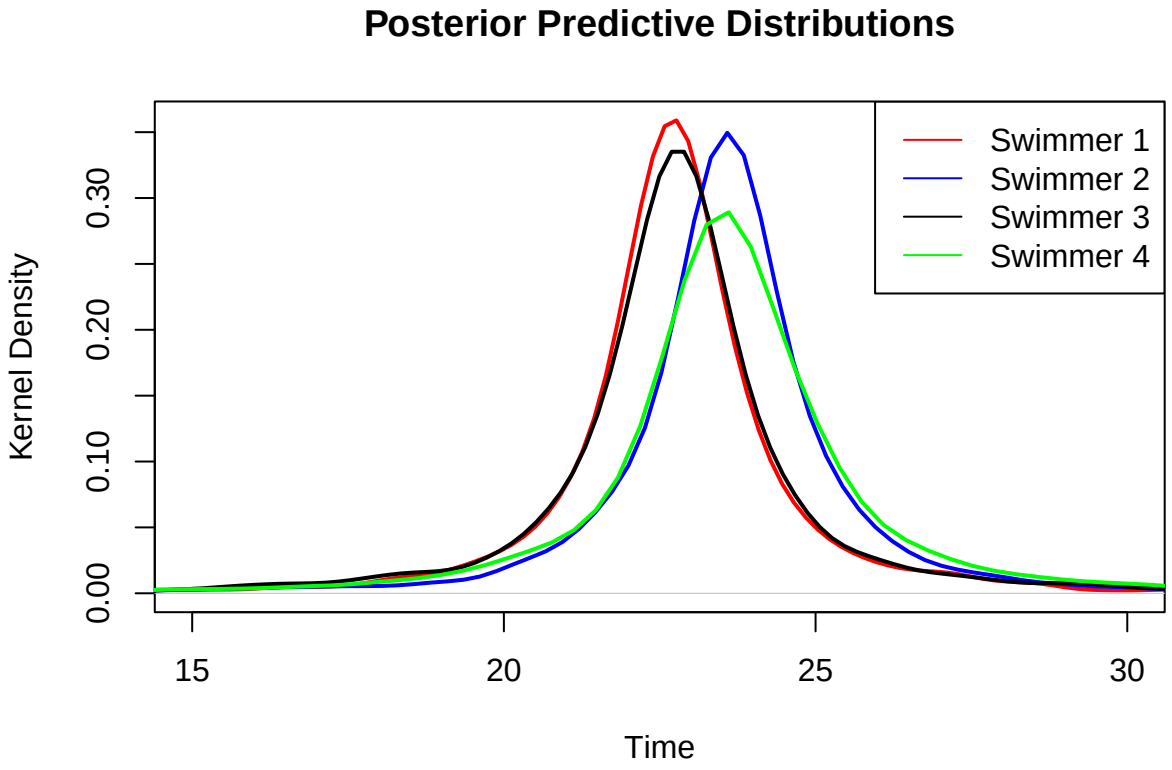



```
par(op)
```

b)

```
#PPD
x.sim <- c(1, 7)

y1.tilde <- rnorm(S, mean = as.vector(model1$beta.chain %*% x.sim),
                  sd = sqrt(model1$sigma2.chain))
y2.tilde <- rnorm(S, mean = as.vector(model2$beta.chain %*% x.sim),
                  sd = sqrt(model2$sigma2.chain))
y3.tilde <- rnorm(S, mean = as.vector(model3$beta.chain %*% x.sim),
                  sd = sqrt(model3$sigma2.chain))
y4.tilde <- rnorm(S, mean = as.vector(model4$beta.chain %*% x.sim),
                  sd = sqrt(model4$sigma2.chain))


plot(density(y1.tilde, adj=2), lwd=2, col="red", xlim = c(15,30),
     main = "Posterior Predictive Distributions",
     xlab = "Time", ylab = "Kernel Density")

lines(density(y2.tilde, adj=2), lwd=2, col="blue")

lines(density(y3.tilde, adj=2), lwd=2, col="black")

lines(density(y4.tilde, adj=2), lwd=2, col="green")
```

```
legend("topright", col = c("red", "blue", "black", "green"), lwd = 1,
       legend = c("Swimmer 1", "Swimmer 2", "Swimmer 3", "Swimmer 4"))
```




```
par(op)
```

c)

```
# Get all permutations
perms <- permutations(n = 4, r = 4, v = 1:4)

# Function to compute probability for a permutation
get_prob <- function(perm) {
  p <- mean(get(paste0("y", perm[1], ".tilde")) <
              get(paste0("y", perm[2], ".tilde")) &
              get(paste0("y", perm[2], ".tilde")) <
              get(paste0("y", perm[3], ".tilde")) &
              get(paste0("y", perm[3], ".tilde")) <
              get(paste0("y", perm[4], ".tilde")))
  return(p)
}

probabilities <- matrix(NA, nrow = 24, 1)
# Compute probabilities
for (i in 1:nrow(perms)) {
  probabilities[i,1] <- get_prob(perms[i,])
}
```

```r
# Put results in matrix
results <- cbind(perms, probabilities)

# Print sorted
print(results)
```

```
##       [,1] [,2] [,3] [,4]   [,5]
##  [1,]    1    2    3    4 0.0486
##  [2,]    1    2    4    3 0.0374
##  [3,]    1    3    2    4 0.0862
##  [4,]    1    3    4    2 0.0816
##  [5,]    1    4    2    3 0.0404
##  [6,]    1    4    3    2 0.0458
##  [7,]    2    1    3    4 0.0394
##  [8,]    2    1    4    3 0.0226
##  [9,]    2    3    1    4 0.0364
## [10,]    2    3    4    1 0.0258
## [11,]    2    4    1    3 0.0150
## [12,]    2    4    3    1 0.0128
## [13,]    3    1    2    4 0.0934
## [14,]    3    1    4    2 0.0804
## [15,]    3    2    1    4 0.0442
## [16,]    3    2    4    1 0.0328
## [17,]    3    4    1    2 0.0446
## [18,]    3    4    2    1 0.0384
## [19,]    4    1    2    3 0.0294
## [20,]    4    1    3    2 0.0418
## [21,]    4    2    1    3 0.0204
## [22,]    4    2    3    1 0.0152
## [23,]    4    3    1    2 0.0416
## [24,]    4    3    2    1 0.0258
```

```r
# now, solving for 1
results1 <- results[,-c(2:4)]
df <- as.data.frame(results1)
# Rename columns
result <- aggregate(df[, 2], by = list(df[, 1]), FUN = sum)
names(result) <- c("Swimmer A", "Prob")
result
```

```
##   Swimmer A   Prob
## 1         1 0.3400
## 2         2 0.1520
## 3         3 0.3338
## 4         4 0.1742
```

According to this, the coach should choose Swimmer 1 by a slight margin over Swimmer 3.

```r
# now, solving for 2
results2 <- results[,-c(3:4)]

for (i in 1:24) {
```

```
  values <- c(results2[i,1], results2[i,2])
  results2[i, 1:2] <- sort(values)
}

df <- as.data.frame(results2)
# Rename columns
result <- aggregate(df[, 3], by = list(df[, 1], df[, 2]), FUN = sum)
names(result) <- c("Swimmer A", "Swimmer B", "Prob")
result
```

```
##   Swimmer A Swimmer B   Prob
## 1         1         2 0.1480
## 2         1         3 0.3416
## 3         2         3 0.1392
## 4         1         4 0.1574
## 5         2         4 0.0634
## 6         3         4 0.1504
```

The best pair of swimmers is Swimmer 1 and 3, as expected.

```
# now, solving for 3
results3 <- results[,-c(4)]

for (i in 1:24) {
  values <- c(results3[i,1], results3[i,2], results3[i,3])
  results3[i, 1:3] <- sort(values)
}

df3 <- as.data.frame(results3)
# Rename columns
result <- aggregate(df3[, 4], by = list(df3[, 1], df3[, 2], df3[,3]),
                    FUN = sum)
names(result) <- c("Swimmer A", "Swimmer B", "Swimmer C", "Prob")
result
```

```
##   Swimmer A Swimmer B Swimmer C   Prob
## 1         1         2         3 0.3482
## 2         1         2         4 0.1652
## 3         1         3         4 0.3358
## 4         2         3         4 0.1508
```

For a group of 3, the coach should choose Swimmers 1,2,3. 1 and 3 make sense, since we saw before that they are by far the best combination. The question was then who would be the third swimmer. In this case, Swimmer 2 took a slight advantage over 4. Thus, the coach should choose Swimmers 1,2, and 3.