

1. Exercise 7.1 (pg 238) Jeffreys' prior: For the multivariate normal

model, Jeffreys' rule for generating a prior distribution on (θ, Σ) gives $p_J(\theta, \Sigma) \propto |\Sigma|^{-(p+2)/2}$.

a) Explain why the function p_J cannot actually be a probability density for (θ, Σ) .

Since the density is uniform with respect to θ , the integral over the support of this function is infinite and cannot be 1.

b) Let $p_J(\theta, \Sigma | y_1, \dots, y_n)$ be the probability density that is proportional

to $p_J(\theta, \Sigma) \times p(y_1, \dots, y_n | \theta, \Sigma)$. Obtain the form of $p_J(\theta, \Sigma | y_1, \dots, y_n)$, $p_J(\theta | \Sigma, y_1, \dots, y_n)$ and $p_J(\Sigma | y_1, \dots, y_n)$.

$$p_J(\theta, \Sigma | \mathbf{y}_1, \dots, \mathbf{y}_n) \propto p(\theta, \Sigma) \times p(\mathbf{y}_1, \dots, \mathbf{y}_n | \theta, \Sigma) \quad (1)$$

$$\propto \left[|\Sigma|^{-(p+2)/2} \right] \times \left[|\Sigma|^{-n/2} \exp(-\text{tr}(\mathbf{S}_\theta \Sigma^{-1})) \right] \quad (2)$$

$$\propto |\Sigma|^{-(p+n+2)/2} \exp(-\text{tr}(\mathbf{S}_\theta \Sigma^{-1})/2) \quad (3)$$

To obtain the full conditionals of a parameter, we treat the other parameters as constant, so

$$p_J(\theta | \mathbf{y}_1, \dots, \mathbf{y}_n, \Sigma) \propto \exp(-\text{tr}(\mathbf{S}_\theta \Sigma^{-1})/2) \quad (4)$$

$$= \exp\left(-\sum_{i=1}^n (\mathbf{y}_i - \theta)^T \Sigma^{-1} (\mathbf{y}_i - \theta)/2\right) \quad \text{Expand back} \quad (5)$$

$$= \exp(-n(\bar{\mathbf{y}} - \theta)^T \Sigma^{-1} (\bar{\mathbf{y}} - \theta)/2) \quad (6)$$

$$= \text{dnormal}(\bar{\mathbf{y}}, \Sigma/n) \quad (7)$$

and

$$p_J(\Sigma | \mathbf{y}_1, \dots, \mathbf{y}_n, \theta) \propto |\Sigma|^{-(p+n+2)/2} \exp(-\text{tr}(\mathbf{S}_\theta \Sigma^{-1})/2) \quad (8)$$

$$\propto \text{dinverse-wishart}(n+1, \mathbf{S}_\theta^{-1}) \quad (9)$$

2. Exercise 7.2 (pg 238) Unit information prior

Letting $\Psi = \Sigma^{-1}$, show that a unit information prior for (θ, Ψ) is given by $\theta | \Psi \sim \text{multivariate normal}(\bar{\mathbf{y}}, \Psi^{-1})$ and $\Psi \sim \text{Wishart}(p+1, S^{-1})$, where $S = \sum (y_i - \bar{y})(y_i - \bar{y})^T / n$. This can be done by mimicking the procedure outlined in Exercise 5.6 as follows:

a) Reparameterize the multivariate normal model in terms of the precision matrix

$\Psi = \Sigma^{-1}$. Write out the resulting log likelihood, and find a probability density $p_U(\theta, \Psi) = p_U(\theta | \Psi) p_U(\Psi)$ such that $\log p(\theta, \Psi) = l(\theta, \Psi | \mathbf{Y}) / n + c$, where c does not depend on θ or Ψ .

Hint: Write $(y_i - \theta)$ as $(y_i - \bar{y} + \bar{y} - \theta)$, and note that $\sum a_i^T \mathbf{B} a_i$ can be written as $\text{tr}(AB)$, where $\mathbf{A} = \sum a_i a_i^T$.

b) Let $p_U(\Sigma)$ be the inverse-Wishart density induced by $p_U(\Psi)$.

Obtain a density $p_U(\theta, \Sigma | y_1, \dots, y_n) \propto p_U(\theta | \Sigma) p_U(\Sigma) p(y_1, \dots, y_n | \theta, \Sigma)$. Can this be interpreted as a posterior distribution for θ and Σ ?

3. Exercise 7.4 (pg 239) Marriage data

The file `agehw.dat` contains data on the ages of 100 married couples sampled from the U.S. population.

```
agehw = as.matrix(read.table(url('http://www.stat.washington.edu/people/pdhoff/Book/Data/hwdata/agehw.d
colnames(agehw) = agehw[1, ]
agehw = agehw[-1, ]
agehw = matrix(as.numeric(agehw), nrow = 100)
```

```
#Store the agehw.dat files in the same folder as this Rmd file
Y.marr <- read.table("agehw.dat", sep=" ", header=T)
```

a) Before you look at the data, use your own knowledge to formulate a semiconjugate prior distribution for

$\theta = (\theta_h, \theta_w)^T$ and Σ , where θ_h, θ_w are mean husband and wife ages, and Σ is the covariance matrix.

With typical intuitions about life expectancy and age of marriage, I suspect that the ages of most of the married couples will fall between 25 and 80. There may be slight age differences among men and women, but nothing that I'm confident enough to encode in my prior. Thus I will set $\mu_0 = ((25+80)/2, (25+80)/2) = (52.5, 52.5)^T$

I have no choice but to pick a semiconjugate prior distribution in this case, but it does seem intuitive that there are less married couples at ages 25 and 80 than there are married couples around age 50, thus justifying a bell curve centered around 52.5 with variance $13.75^2 \approx 189$ such that approximately 95% of my prior falls within the range (25, 80).

I also have reason to believe the ages of the couples are quite tightly correlated, so knowing the above variance, I will aim for a prior correlation of 0.75. Solving the correlation equation gives

$$0.75 = \frac{\sigma_{1,2}}{189} \quad (10)$$

$$\implies \sigma_{1,2} = 141.75 \quad (11)$$

So I will set

$$\Lambda_0 = \begin{bmatrix} 189 & 141.75 \\ 141.75 & 189 \end{bmatrix}$$

Like previous problems, for the variance, I will set $\mathbf{S}_0^{-1} = \Lambda_0$ and $\nu_0 = p + 2 = 4$. Note that this only loosely centers our covariance matrix prior on Λ_0 , so I am being a bit conservative in terms of my belief in the variance and the correlation between the ages.

```
Y = agehw
p = ncol(agehw)
n = nrow(agehw)
ybar = colMeans(agehw)
```

```
mu0 = rep(52.5, p)
lambda0 = s0 = rbind(c(189, 141.75), c(141.75, 189))
# nu0 = p + 2
nu0 = p + 2 + 10
```

b) Generate a prior predictive dataset of size $n = 100$,

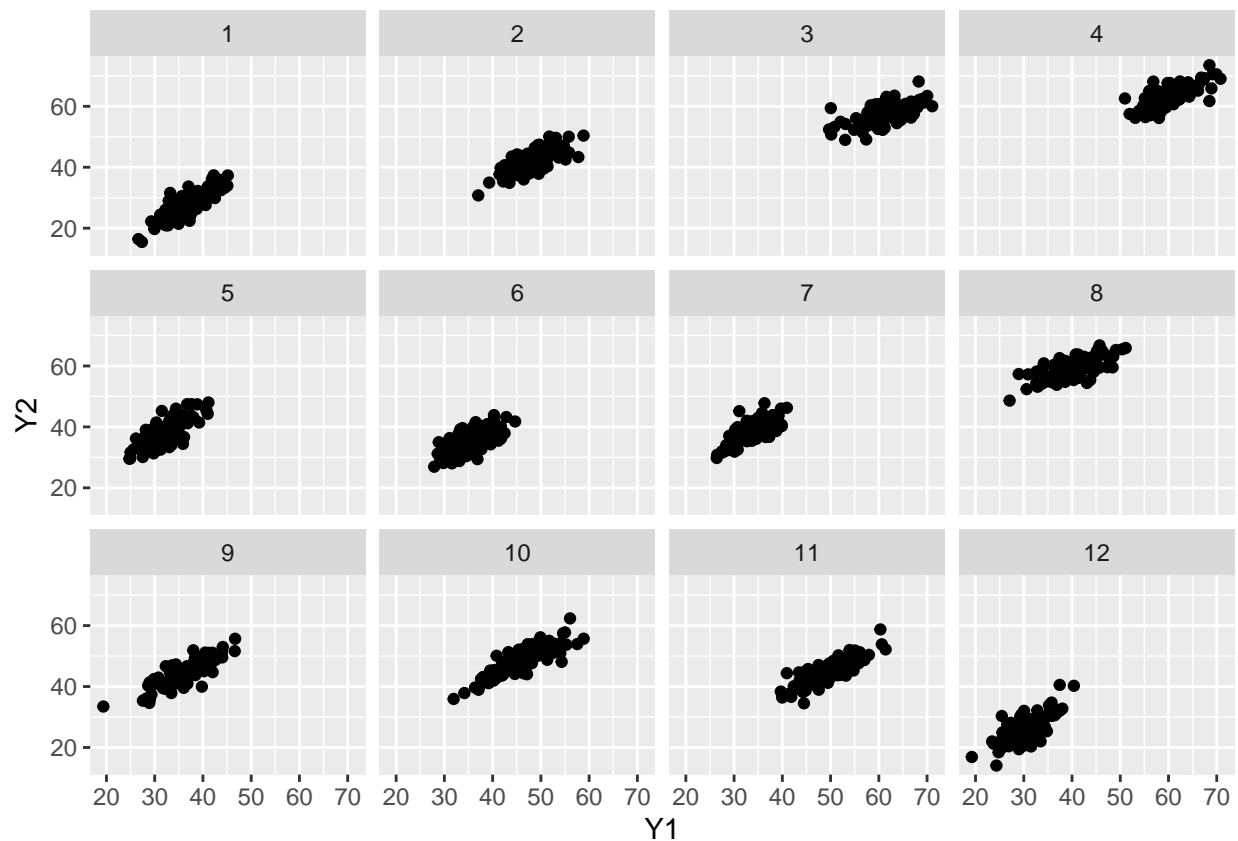
by sampling (θ, Σ) from your prior distribution and then simulating $Y_1, \dots, Y_n \sim \text{i.i.d. multivariate normal}(\theta, \Sigma)$. Generate several such datasets, make bivariate scatterplots for each dataset, and make sure they roughly represent your prior beliefs about what such a dataset would actually look like. If your prior predictive datasets do not conform to your beliefs, go back to part a) and formulate a new prior. Report the prior that you eventually decide upon, and provide scatterplots for at least three prior predictive datasets.

The wording of the question is interesting - I assume I'm supposed to sample a fixed θ, Σ and from there sample 100 points all with the same parameters. If I were to do this myself, I feel like I would sample a new data point for each sample of $\theta, \Sigma \dots$

In fact, because of that wording, I originally set $\nu_0 = p + 2 = 4$ to loosely center my prior. But given that this variance will often produce uncorrelated prior predictive datasets, I'm increasing ν_0 a bit...

After increasing ν_0 , I'm fairly comfortable with what these posterior predictive datasets look like.

```
N = 100
S = 12
Y_preds = lapply(1:S, function(s) {
  # Sample THETA according to prior
  theta = mvrnorm(n = 1, mu0, lambda0)
  sigma = inv(rWishart(1, nu0, inv(s0))[, , 1])
  Y_s = mvrnorm(n = 100, theta, sigma)
  data.frame(Y1 = Y_s[, 1], Y2 = Y_s[, 2], dataset = s)
})
Y_comb = do.call(rbind, Y_preds)
ggplot(Y_comb, aes(x = Y1, y = Y2)) +
  geom_point() +
  facet_wrap(~ dataset)
```



c) Using your prior distribution and the 100 values in the dataset,

obtain an MCMC approximation to $p(\theta, \Sigma | y_1, \dots, y_{100})$. Plot the joint posterior distribution of θ_h and θ_w , and also the marginal posterior density of the correlation between Y_h and Y_w , the ages of a husband and wife. Obtain 95% posterior confidence intervals for θ_h , θ_w and the correlation coefficient.

```
S = 10000
# Reuse this since we'll need to specify different priors
do_mcmc = function(Y, mu0, lambda0, s0, nu0) {
  ybar = colMeans(Y)
  p = ncol(Y)
  n = nrow(Y)
  THETA = matrix(nrow = S, ncol = p)
  SIGMA = array(dim = c(p, p, S))

  # Start with sigma sample
  sigma = cov(Y)

  # Gibbs sampling

  # Also, inv = solve to make it more readable
  inv = solve

  for (s in 1:S) {
    # Update theta
    lambdan = inv(inv(lambda0) + n * inv(sigma))
```

```

mun = lambda0 %*% (inv(lambda0) %*% mu0 + n * inv(sigma) %*% ybar)
theta = mvrnorm(n = 1, mun, lambda0)

# Update sigma
resid = t(Y) - c(theta)
stheta = resid %*% t(resid)
sn = s0 + stheta
sigma = inv(rWishart(1, nu0 + n, inv(sn))[, , 1])

THETA[s, ] = theta
SIGMA[, , s] = sigma
}

list(theta = THETA, sigma = SIGMA)
}
my_prior_mcmc = do_mcmc(agehw, mu0, lambda0, s0, nu0)
THETA = my_prior_mcmc$theta
SIGMA = my_prior_mcmc$sigma
# For reuse later
print_quantiles = function(THETA, SIGMA) {
  print("Husband")
  print(quantile(THETA[, 1], probs = c(0.025, 0.5, 0.975))) # Husband
  print("Wife")
  print(quantile(THETA[, 2], probs = c(0.025, 0.5, 0.975))) # Wife
  cors = apply(SIGMA, MARGIN = 3, FUN = function(covmat) {
    covmat[1, 2] / (sqrt(covmat[1, 1] * covmat[2, 2]))
  })
  print("Correlation")
  print(quantile(cors, probs = c(0.025, 0.5, 0.975)))
}
print_quantiles(THETA, SIGMA)

```

```

## [1] "Husband"
##      2.5%      50%      97.5%
## 42.00140 44.53102 47.05096
## [1] "Wife"
##      2.5%      50%      97.5%
## 38.59336 40.99697 43.37593
## [1] "Correlation"
##      2.5%      50%      97.5%
## 0.8614762 0.9027481 0.9316868

```

d) Obtain 95% posterior confidence intervals for θ_h , θ_w and the correlation coefficient using the following prior distributions:

I haven't done 7.2, but doing Jeffreys' prior and a "diffuse prior" below will still be helpful to see what effect prior information has.

i. Jeffreys' prior, described in Exercise 7.1;

```

THETA = matrix(nrow = S, ncol = p)
SIGMA = array(dim = c(p, p, S))
# Start with sigma sample
sigma = cov(Y)
# Gibbs sampling
# Also, inv = solve to make it more readable
inv = solve
for (s in 1:S) {
  # Update theta
  theta = mvrnorm(n = 1, ybar, sigma / n)

  # Update sigma
  resid = t(Y) - c(theta)
  stheta = resid %*% t(resid)
  sigma = inv(rWishart(1, n + 1, inv(stheta))[, , 1])

  THETA[s, ] = theta
  SIGMA[, , s] = sigma
}
print_quantiles(THETA, SIGMA)

```

```

## [1] "Husband"
##      2.5%      50%      97.5%
## 41.67567 44.38854 47.07351
## [1] "Wife"
##      2.5%      50%      97.5%
## 38.32512 40.87268 43.39238
## [1] "Correlation"
##      2.5%      50%      97.5%
## 0.8625790 0.9045666 0.9349914

```

ii. the unit information prior, described in Exercise 7.2;

Unit information prior (skipping)

iii. a “diffuse prior” with $\mu_0 = \mathbf{0}$, $\Lambda_0 = 10^5 \times \mathbf{I}$, $\mathbf{S}_0 = 1000 \times \mathbf{I}$ and $\nu_0 = 3$.

```

mu0 = rep(0, p)
lambda0 = 10^5 * diag(p)
s0 = 1000 * diag(p)
nu0 = 3
diffuse_mcmc = do_mcmc(agehw, mu0, lambda0, s0, nu0)
print_quantiles(diffuse_mcmc$theta, diffuse_mcmc$sigma)

```

```

## [1] "Husband"
##      2.5%      50%      97.5%
## 41.67345 44.40646 47.09832

```

```
## [1] "Wife"
##      2.5%      50%      97.5%
## 38.26279 40.87471 43.41812
## [1] "Correlation"
##      2.5%      50%      97.5%
## 0.7944943 0.8556862 0.8996030
```

e) Compare the confidence intervals from d) to those obtained in c).

Discuss whether or not you think that your prior information is helpful in estimating θ and Σ , or if you think one of the alternatives in d) is preferable. What about if the sample size were much smaller, say $n = 25$?

It doesn't really seem like the prior information matters because the sample size is so large. Regardless of whether the prior is informative, the quantiles and correlations end up quite similar. Maybe the diffuse prior is slightly different, but it's not a big difference.

If we have a smaller sample size, this may be different. Let's see what happens if I truncate the dataset to the first 25 variables and rerun with my informative prior and the diffuse prior:

- My prior

```
mu0 = rep(52.5, p)
lambda0 = s0 = rbind(c(189, 141.75), c(141.75, 189))
# nu0 = p + 2
nu0 = p + 2 + 10
my_prior_mcmc_short = do_mcmc(agehw[1:25, ], mu0, lambda0, s0, nu0)
print_quantiles(my_prior_mcmc_short$theta, my_prior_mcmc_short$sigma)
```

```
## [1] "Husband"
##      2.5%      50%      97.5%
## 40.95775 45.43896 49.86462
## [1] "Wife"
##      2.5%      50%      97.5%
## 38.45520 43.10250 47.65666
## [1] "Correlation"
##      2.5%      50%      97.5%
## 0.8402358 0.9132493 0.9536770
```

- Diffuse prior

```
mu0 = rep(0, p)
lambda0 = 10^5 * diag(p)
s0 = 1000 * diag(p)
nu0 = 3
diffuse_mcmc_short = do_mcmc(agehw[1:25, ], mu0, lambda0, s0, nu0)
print_quantiles(diffuse_mcmc_short$theta, diffuse_mcmc_short$sigma)
```

```
## [1] "Husband"
##      2.5%      50%      97.5%
## 39.25530 45.20976 51.14856
## [1] "Wife"
##      2.5%      50%      97.5%
```

```
## 36.80945 42.81444 49.04489
## [1] "Correlation"
##      2.5%      50%      97.5%
## 0.5385955 0.7612246 0.8832484
```

In this case, the effect of the prior on correlation especially is easily observed. The correlation for the diffuse prior is quite low, as it is being dragged towards nothing.

4. Exercise 9.1 (pg 242) Extrapolation

The file swim.dat contains data on the amount of time, in seconds, it takes each of four high school swimmers to swim 50 yards. Each swimmer has six times, taken on a biweekly basis.

```
#Store the agehw.dat files in the same folder as this Rmd file
Ywide.swim <- read.table("swim.dat") #one column per swimmer
Y.swim <- reshape2::melt(Ywide.swim)[-1]
# No id variables; using all as measure variables
Y.swim <- cbind(Y.swim, rep(seq(1,12,by=2),each=4), rep(1:4,times=6))
colnames(Y.swim) <- c("time", "week", "swimmer")
```

a) Perform the following data analysis for each swimmer separately:

- i. Fit a linear regression model of swimming time as the response and week as the explanatory variable. To formulate your prior, use the information that competitive times for this age group generally range from 22 to 24 seconds.
- ii. For each swimmer j , obtain a posterior predictive distribution for Y_j^* , their time if they were to swim two weeks from the last recorded time.

```
swim = read.table(url('http://www.stat.washington.edu/people/pdhoff/Book/Data/hwdata/swim.dat'))
```

To specify our prior, we let the prior expectation of our y -intercept to be 23, and we let the prior expectation of the effect of training week by week to be 0, so $\beta = (23, 0)^T$.

We expect no covariance with the β coefficients, but we do have uncertainty about our initial β estimates. Specifically, to let 95% of our uncertainty of the y -intercept to fall in $[22, 24]$, we let $\Sigma_{0(1,1)} = 1/4$ (so that ± 2 standard deviations is ± 1). We also expect that training has a relatively mild effect on time, so we let $\Sigma_{0(2,2)} = 0.1$ which is just an arbitrarily chosen small variance.

For our expectation of the variability of measurements, let's similarly set $\sigma_0^2 = 1/4$ and only lightly center this prior with $\nu_0 = 1$.

```
library(MASS) # mvrnorm
library(dplyr)
S = 5000
X = cbind(rep(1, 6), seq(1, 11, by = 2))
n = dim(X)[1]
p = dim(X)[2]
# Prior
beta0 = c(23, 0)
sigma0 = rbind(c(0.25, 0), c(0, 0.1))
nu0 = 1
```



```

s20 = 0.25
set.seed(1)
inv = solve
# For each swimmer, run linear regression gibbs sampling and obtain a posterior
# predictive distribution
swim_pred = apply(swim, MARGIN = 1, function(y) {

  # Store samples
  BETA = matrix(nrow = S, ncol = length(beta0))
  SIGMA = numeric(S)

  # Starting values - just use prior values?
  beta = c(23, 0)
  s2 = 0.7^2

  # Gibbs sampling algorithm from 9.2.1
  for (s in 1:S) {
    # 1a) Compute V and m
    V = inv(inv(sigma0) + (t(X) %*% X) / s2)
    m = V %*% (inv(sigma0) %*% beta0 + (t(X) %*% y) / s2)

    # 1b) sample beta
    beta = mvrnorm(1, m, V)

    # 2a) Compute SSR(beta) (specific formula from 9.1)
    ssr = (t(y) %*% y) - (2 * t(beta) %*% t(X) %*% y) + (t(beta) %*% t(X) %*% X %*% beta)

    # 2b) sample s2
    s2 = 1 / rgamma(1, (nu0 + n) / 2, (nu0 * s20 + ssr) / 2)
    BETA[s, ] = beta
    SIGMA[s] = s2
  }

  # Now sample posterior predictive - two weeks later
  xpred = c(1, 13)
  YPRED = rnorm(S, BETA %*% xpred, sqrt(SIGMA))

  YPRED
})

```

b) The coach of the team has to decide which of the four swimmers will compete

in a swimming meet in two weeks. Using your predictive distributions, compute $Pr(Y_j^* = \max\{Y_1^*, \dots, Y_4^*\} | Y)$ for each swimmer j , and based on this make a recommendation to the coach.

```

fastest_times = apply(swim_pred, MARGIN = 1, FUN = which.min)
table(fastest_times) / length(fastest_times)

```

```

## fastest_times
##      1      2      3      4
## 0.6524 0.0134 0.3060 0.0282

```

We notice that with our posterior predictive dataset, swimmer 1 is the fastest about 65% of the time by week 13, so we recommend that swimmer 1 race.

5. Exercise 9.3 (pg 243) Crime

The file `crime.dat` contains crime rates and data on 15 explanatory variables for 47 U.S. states, in which both the crime rates and the explanatory variables have been centered and scaled to have variance 1. A description of the variables can be obtained by typing `library(MASS); ?UScrime` in R.

```
data("UScrime", package="MASS")
namvars <- names(UScrime)
```

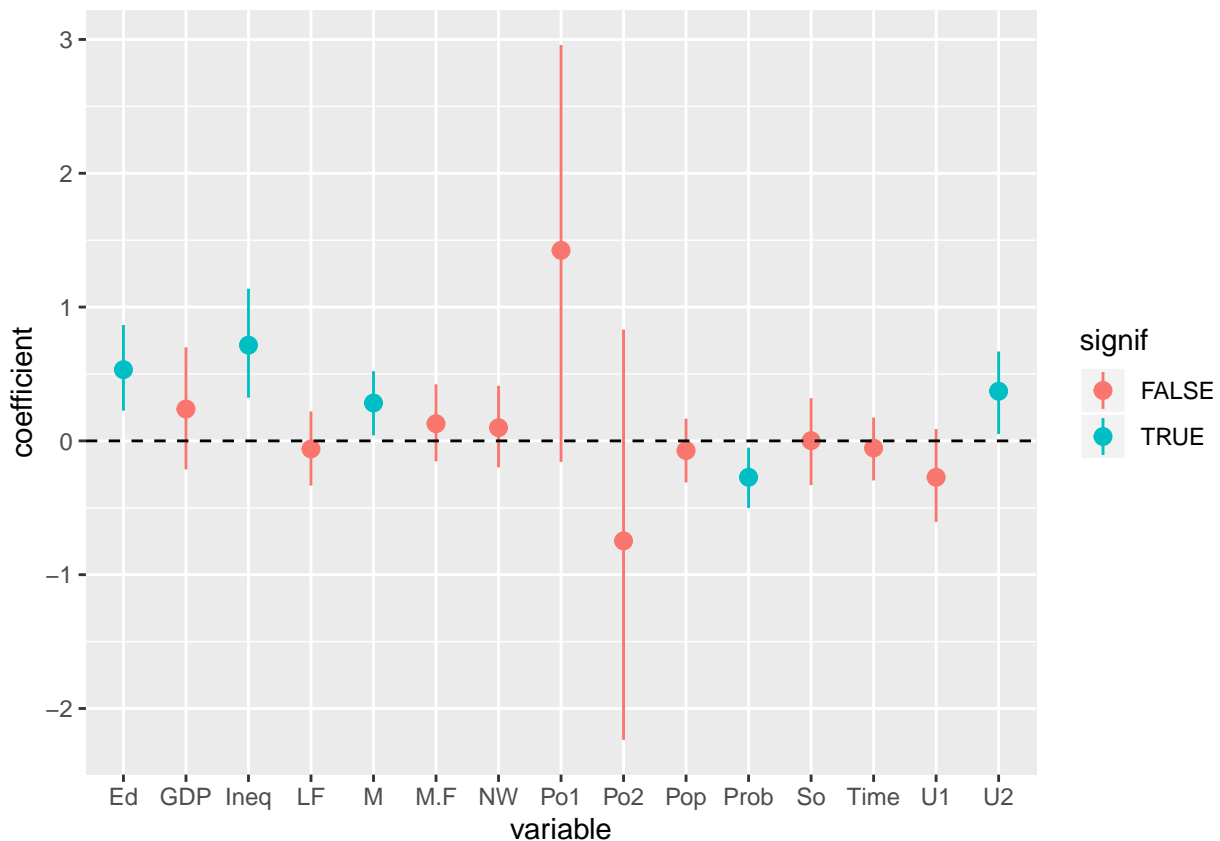
```
crime = read.table(url('http://www.stat.washington.edu/people/pdhoff/Book/Data/hwdata/crime.dat'), head=
```

a) Fit a regression model

$y = X\beta + \epsilon$ using the g-prior with $g = n$, $\nu_0 = 2$ and $\sigma_0^2 = 1$. Obtain marginal posterior means and 95% confidence intervals for β , and compare to the least squares estimates. Describe the relationships between crime and the explanatory variables. Which variables seem strongly predictive of crime rates?

```
y = crime$y
X = crime %>% select(-y) %>% as.matrix
n = dim(X)[1]
p = dim(X)[2]
g = n
nu0 = 2
s20 = 1
S = 1000
Hg = (g / (g + 1)) * X %*% inv(t(X) %*% X) %*% t(X)
SSRg = t(y) %*% (diag(1, nrow = n) - Hg) %*% y
s2 = 1 / rgamma(S, (nu0 + n) / 2, (nu0 * s20 + SSRg) / 2)
Vb = g * inv(t(X) %*% X) / (g + 1)
Eb = Vb %*% t(X) %*% y
E = matrix(rnorm(S * p, 0, sqrt(s2)), S, p)
beta = t(t(E %*% chol(Vb)) + c(Eb))
```

```
library(tidyr)
signif = apply(beta, MARGIN = 2, FUN = quantile, probs = c(0.025, 0.5, 0.975)) %>%
  apply(MARGIN = 2, FUN = function(y) !(y[1] < 0 && 0 < y[3]))
beta_df = as.data.frame(beta) %>%
  gather(key = 'variable', val = 'coefficient') %>%
  mutate(signif = signif[variable])
ggplot(beta_df, aes(x = variable, y = coefficient, color = signif)) +
  stat_summary(fun.y = mean, fun.ymin = function(y) quantile(y, probs = c(0.025)), fun.ymax = function(y) quantile(y, probs = c(0.975)), geom_hline(yintercept = 0, lty = 2))
```



Looks like Ed (mean years of schooling), Ineq (Income inequality), M (percentage of males aged 14-24), Prob (probability of imprisonment), and U2 (unemployment rate of urban males 35-39).

b) Lets see how well regression models can predict crime rates based on the

X-variables. Randomly divide the crime roughly in half, into a training set $\{y_{tr}, X_{tr}\}$ and a test set $\{y_{te}, X_{te}\}$

```
y = crime$y
X = crime %>% select(-y) %>% as.matrix
set.seed(1) # Reproducible!
train_i = sample.int(length(y), size = round(length(y) / 2), replace = FALSE)
ytr = y[train_i]
Xtr = X[train_i, ]
yte = y[-train_i]
Xte = X[-train_i, ]
```

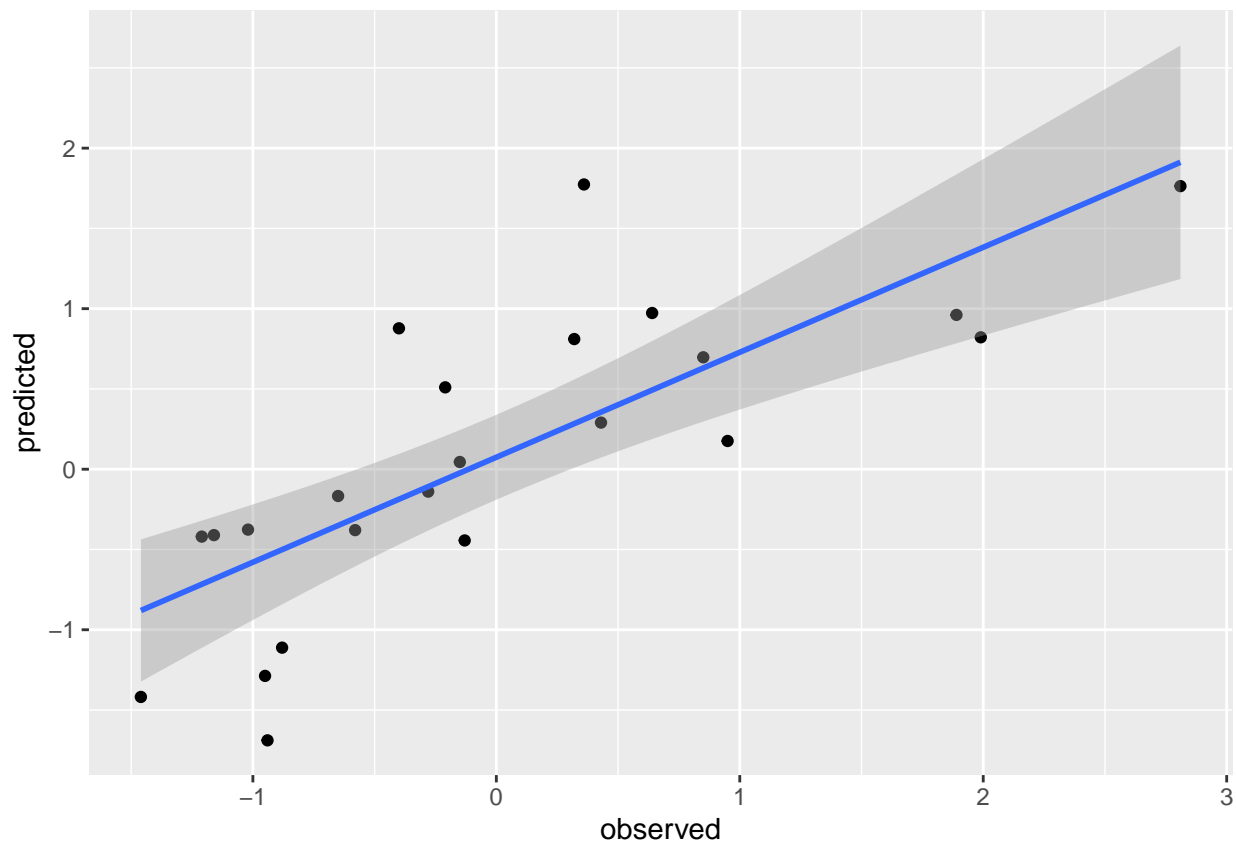
i. Using only the training set, obtain least squares regression coefficients $\hat{\beta}_{ols}$.

Obtain predicted values for the test data by computing $\hat{y}_{ols} = X_{te}\hat{\beta}_{ols}$. Plot \hat{y}_{ols} versus y_{te} and compute the prediction error $\frac{1}{n_{te}} \sum (y_{i,te} - \hat{y}_{i,ols})^2$.

```
# From 9.1
beta_ols = inv(t(Xtr) %*% Xtr) %*% t(Xtr) %*% ytr
beta_ols
```

```
##           [,1]
## M      0.19419790
## So      0.20624856
## Ed      0.64334998
## Po1     0.30590821
## Po2     0.44725441
## LF     -0.09198965
## M.F     0.03537926
## Pop     0.08440462
## NW     -0.01227990
## U1     -0.03278782
## U2      0.15494093
## GDP     0.10455812
## Ineq    0.76691166
## Prob   -0.25823362
## Time    0.05938070
```

```
y_ols = Xte %*% beta_ols
ols_df = data.frame(
  observed = yte,
  predicted = y_ols
)
ggplot(ols_df, aes(x = observed, y = predicted)) +
  geom_point() +
  geom_smooth(method = 'lm')
```



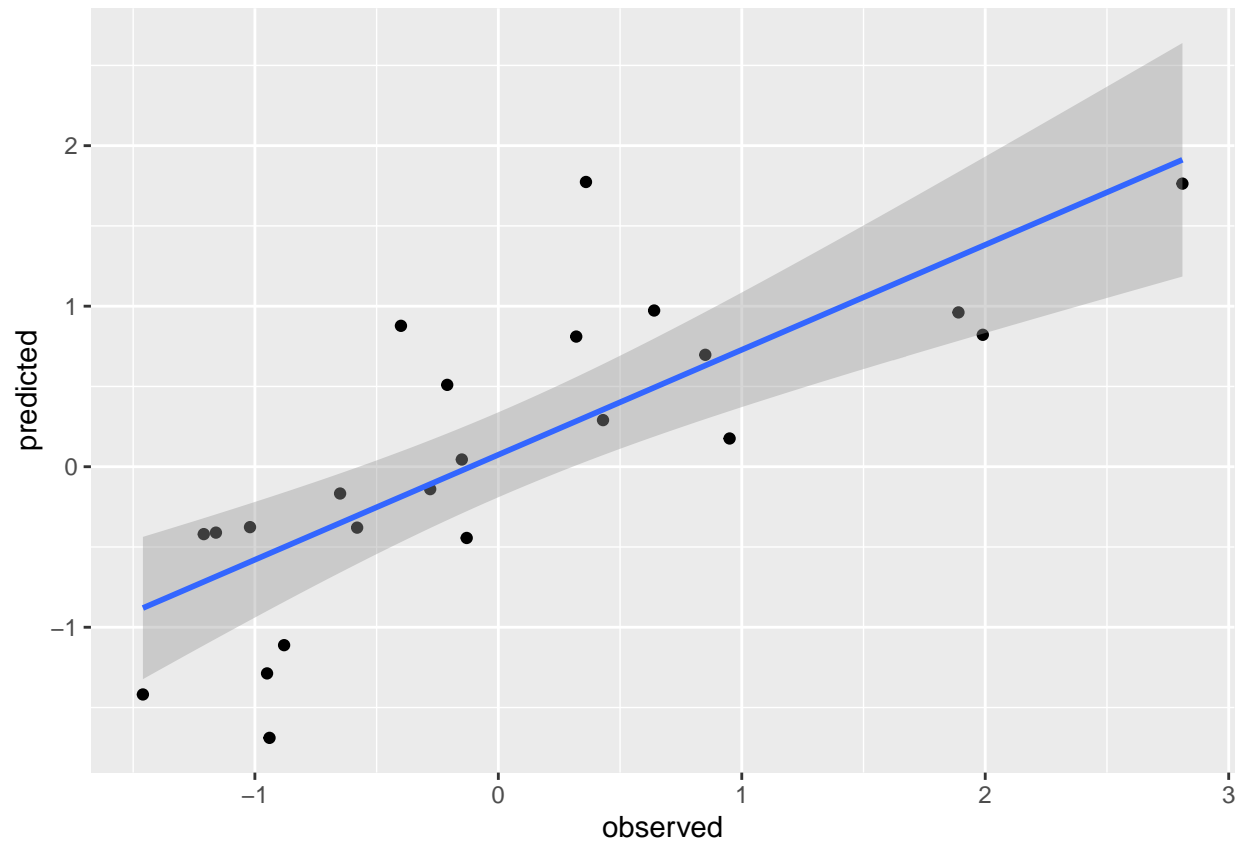
```
pred_error = sum((yte - y_ols)^2) / length(yte)
pred_error
```

```
## [1] 0.4880959
```

ii. Now obtain the posterior mean $\beta_{Bayes} = E[\beta|y_{tr}]$ using the g-prior described above and the training data only.

Obtain predictions for the test set $\hat{y}_{Bayes} = \mathbf{X}_{test}\hat{\beta}_{Bayes}$. Plot versus the test data, compute the prediction error, and compare to the OLS prediction error. Explain the results.

```
y = ytr
X = Xtr
n = dim(X)[1]
p = dim(X)[2]
g = n
nu0 = 2
s20 = 1
S = 1000
Hg = (g / (g + 1)) * X %*% inv(t(X) %*% X) %*% t(X)
SSRg = t(y) %*% (diag(1, nrow = n) - Hg) %*% y
s2 = 1 / rgamma(S, (nu0 + n) / 2, (nu0 * s20 + SSRg) / 2)
Vb = g * inv(t(X) %*% X) / (g + 1)
Eb = Vb %*% t(X) %*% y
E = matrix(rnorm(S * p, 0, sqrt(s2)), S, p)
beta = t(t(E %*% chol(Vb)) + c(Eb))
beta_bayes = as.matrix(colMeans(beta))
y_bayes = Xte %*% beta_bayes
bayes_df = data.frame(
  observed = yte,
  predicted = y_bayes
)
ggplot(ols_df, aes(x = observed, y = predicted)) +
  geom_point() +
  geom_smooth(method = 'lm')
```



```
pred_error = sum((yte - y_bayes)^2) / length(yte)
pred_error
```

```
## [1] 0.4910634
```

At least when the seed is 1, there doesn't appear to be a major difference between the prediction errors.

c) Repeat the procedures in b) many times with different randomly generated

test and training sets. Compute the average prediction error for both the OLS and Bayesian methods.

```
# Unnecessary code repeating here. Clean this up later w/ functions
N = 100
set.seed(1)
pred_errors = t(sapply(1:N, function(i) {
  y = crime$y
  X = crime %>% select(-y) %>% as.matrix
  train_i = sample.int(length(y), size = round(length(y) / 2), replace = FALSE)
  ytr = y[train_i]
  Xtr = X[train_i, ]
  yte = y[-train_i]
  Xte = X[-train_i, ]

  # OLS
  beta_ols = inv(t(Xtr) %*% Xtr) %*% t(Xtr) %*% ytr
```

```

beta_ols

y_ols = Xte %>% beta_ols

pred_error_ols = sum((yte - y_ols)^2) / length(yte)

# Bayes
y = ytr
X = Xtr

n = dim(X)[1]
p = dim(X)[2]

g = n
nu0 = 2
s20 = 1

S = 1000

Hg = (g / (g + 1)) * X %>% inv(t(X) %>% X) %>% t(X)
SSRg = t(y) %>% (diag(1, nrow = n) - Hg) %>% y

s2 = 1 / rgamma(S, (nu0 + n) / 2, (nu0 * s20 + SSRg) / 2)
Vb = g * inv(t(X) %>% X) / (g + 1)
Eb = Vb %>% t(X) %>% y

E = matrix(rnorm(S * p, 0, sqrt(s2)), S, p)
beta = t(t(E %>% chol(Vb)) + c(Eb))

beta_bayes = as.matrix(colMeans(beta))

y_bayes = Xte %>% beta_bayes

pred_error_bayes = sum((yte - y_bayes)^2) / length(yte)
c(pred_error_ols, pred_error_bayes)
})) %>% as.data.frame
colnames(pred_errors) = c('ols', 'bayes')

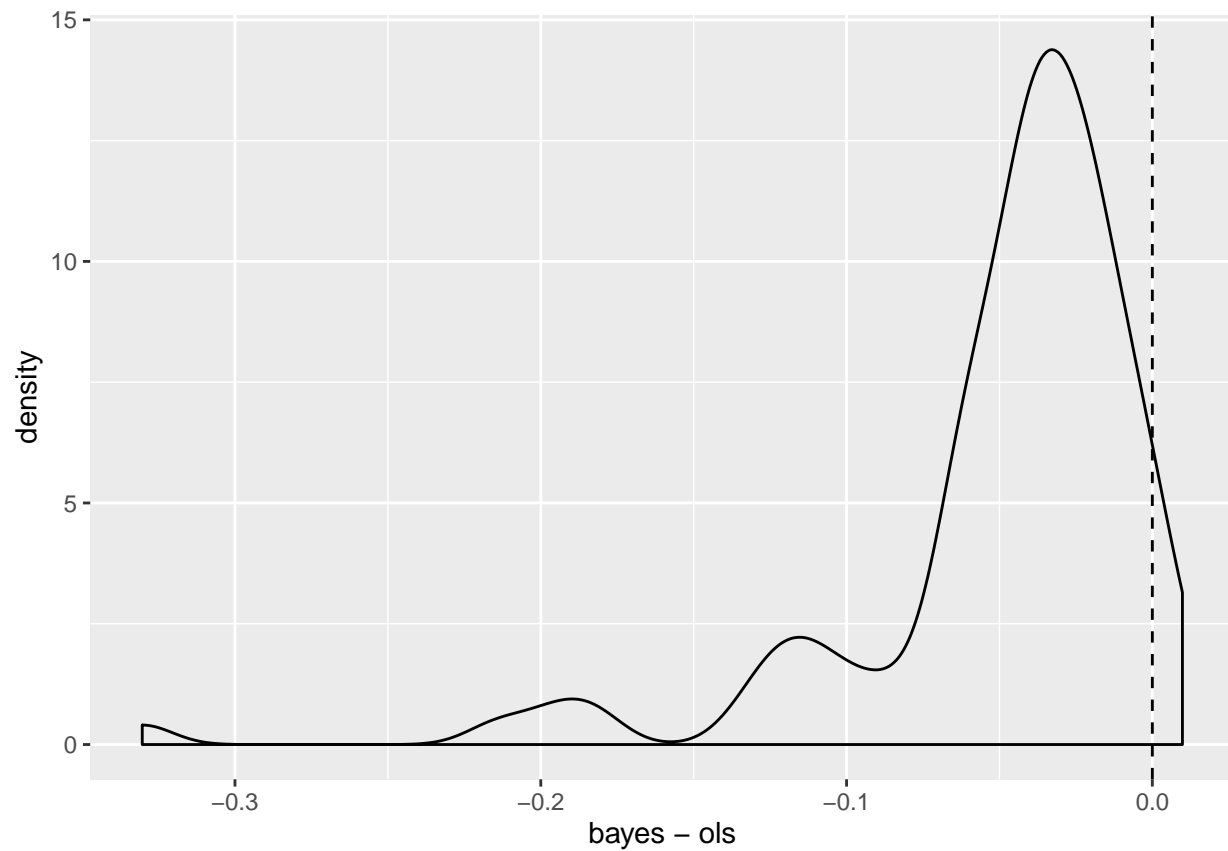
```

Here's a plot of the density of $\text{err}_{\text{Bayes}} - \text{err}_{\text{ols}}$. If this is less than 0, then the Bayes estimator did better than the OLS estimator:

```

pred_diff = pred_errors %>% transmute(`bayes - ols` = bayes - ols)
ggplot(pred_diff, aes(x = `bayes - ols`)) +
  geom_density() +
  geom_vline(xintercept = 0, lty = 2)

```



Lastly,

```
mean(pred_errors$bayes < pred_errors$ols)
```

```
## [1] 0.96
```

For N samples, 96% of the time, the predictive error using the Bayes estimators is less than the predictive error using the OLS estimators. Nice!