Travis Lilley
Thinning Simulation

At the end of the second Exercise set, James claimed that thinning a Monte Carlo chain was unnecessary and in fact increased the variance of estimates. He asked us to run some simulations to test this claim ourselves. I ran several simulations using the Gibbs sampler from Exercise 2. My results confirm what James said: the posterior means for the β_i seem to have higher variance when we thin. The more dramatically we thin, the more this variance increases. R code is included at the end.

Simulation 1
Number of chains: 200
Chain length: 10000
Thin Ratio: 1/2

	$\hat{V}(\hat{\beta}_{0,\text{post}})$	$\hat{V}(\hat{\beta}_{1,\text{post}})$
Thin	$7.708 \cdot 10^{-9}$	$3.967 \cdot 10^{-6}$
No thin	$7.759 \cdot 10^{-9}$	$3.945 \cdot 10^{-6}$

Simulation 2
Number of chains: 200
Chain length: 10000
Thin Ratio: 1/5

	$\hat{V}(\hat{\beta}_{0,\text{post}})$	$\hat{V}(\hat{\beta}_{1,\text{post}})$
Thin	$6.947 \cdot 10^{-9}$	$4.056 \cdot 10^{-6}$
No thin	$6.566 \cdot 10^{-9}$	$3.558 \cdot 10^{-6}$

Simulation 3
Number of chains: 200
Chain length: 10000
Thin Ratio: 1/10

	$\hat{V}(\hat{\beta}_{0,\text{post}})$	$\hat{V}(\hat{\beta}_{1,\text{post}})$
Thin	$8.252 \cdot 10^{-9}$	$4.955 \cdot 10^{-6}$
No thin	$6.250 \cdot 10^{-9}$	$3.113 \cdot 10^{-6}$

Simulation 4
Number of chains: 500
Chain length: 10000
Thin Ratio: 1/2

	$\hat{V}(\hat{\beta}_{0,\text{post}})$	$\hat{V}(\hat{\beta}_{1,\text{post}})$
Thin	$6.489 \cdot 10^{-9}$	$3.585 \cdot 10^{-6}$
No thin	$6.397 \cdot 10^{-9}$	$3.548 \cdot 10^{-6}$

R Code

```
# SDS 383D: Statistical Modeling II
# Exercise 2
# GDP Growth Gibbs Sampler
# Thinning simulation to see if thinning reduces variance

# Add libraries
library(mvtnorm)

# Load data
data <- read.csv("C:\\Users\\trlil\\Google Drive\\UT Austin\\SDS 383D Statistical
Modeling II\\Homework\\Exercises 2\\gdpgrowth.csv",
                    header = T)

# Response
y <- data$GR6096

# Design matrix
X <- cbind(1, data$DEF60)

# Initialize prior parameters
d <- 1/100
eta <- 1/100
m <- c(0, 0)
K <- diag(c(0.001, 0.001))
h <- 1/100

# Number of Gibbs samples
G <- 10^5

# Number of Gibbs iterations
P <- 100

# Number of obs
n <- length(y)

# Objects to store Gibbs samples

# Lambda
L <- matrix(NA, nrow = G, n)

# Omega
w <- rep(0, G)

# Beta
B <- matrix(0, nrow = G, 2)

# Posterior mean of thinned betas
thin <- matrix(NA, nrow = 2, P)

# Posterior mean of non-thin betas
no_thin <- matrix(NA, nrow = 2, P)

# Initialize Gibbs
L[1, ] <- rep(1, n)
w[1] <- 1
B[1, ] <- c(0, 0)

# Run Gibbs M times
for(p in 1:P){
       for(i in 2:G){

              # Update location of sampler
              if(i %% (G/100) == 0){
                     print(i)
              }

              # Update L
              for(j in 1:n){
                     L[i, j] <- rgamma(1, h/2 + 1/2,
```

```
                                        w[i-1]/2*(y[j]-X[j,] %*% B[i-1,])^2 + h/2)
        }

        # Diagonal matrix Lambda
        L2 <- diag(L[i, ])

        # Update w
        w[i] <- rgamma(1, n/2 + d/2,
                        (1/2)*(eta + t(y) %*% L2 %*% y + t(m) %*% K %*% m -
                        t(t(X) %*% L2 %*% y + K %*% m) %*%
                        solve(t(X) %*% L2 %*% X + K) %*%
                        (t(X) %*% L2 %*% y + K %*% m)))

        # Update B
        B[i, ] <- rmvnorm(1, solve(t(X) %*% L2 %*% X + K) %*%
                              (t(X) %*% L2 %*% y + K %*% m),
                              solve(w[i] * t(X) %*% L2 %*% X + w[i] * K))
    }

    # Posterior mean of beta, no thin
    no_thin[1, p] <- mean(B[, 1])
    no_thin[2, p] <- mean(B[, 2])

    # Thin
    B_thin <- B[seq(1, G, by = 2), ]

    # Posterior mean of beta, thin
    thin[1, p] <- mean(B_thin[, 1])
    thin[2, p] <- mean(B_thin[, 2])
}
```