BST249 HW3

Sijia Huo

2/24/2022

1.

(a)

The i.i.d normal model does make sense here. First, it's ok to assume that the weekly snowfall are independent with each other because the temporal dependence of the snowfall is not that strong across weeks and is even weaker across months and years. The strength of the temporal dependence can be strong within each week, but as the number of timepoints increase, the strength decreases significantly.

Denote the snowfall of each week across the year as a random variable X_i with differerent mean μ_i. The differences in μ_i is caused by the yearly time trend of the snowfall (there's more snow in winter and no snow during summer). Assume Lyapunov's condition can be satisifed. Then the annual snowfall, which is the summation of the weekly ones, converges in distribution to a normal one based on the Lyapunov CLT. Though the support of the normal distribution is from negative infinity to the positive infinity whereas the annual snowfall can not be negative, since the expectation of the annual snowfall is far away from 0 for both cities and the probability of zero annual snowfall is extremely small for both cities, the difference in support doesn't really matter here.

Finally, though weakly correlated, the annual snowfall can be treated as indepedent with each other. In addition, there's no clear trend for the annual snowfall despite the fact of global warming. Therefore,the iid assumption holds here.

(b)

```
# import data

aomori <- c(188.6, 244.9, 255.9, 329.1, 244.5, 167.7, 288.2, 208.3, 311.4, 273.2, 395.3, 353.5, 183.9,

valdez <- c(351.0, 379.3, 196.1, 312.3, 301.4, 240.6, 257.6, 304.5, 296.0, 338.8, 299.9, 384.7, 353.5,
```

Set priors as follows:

Aomori: $(\boldsymbol{\mu}_A, \boldsymbol{\lambda}_A) \sim \text{NormalGamma}(m_1, c_1, a_1, b_1)$ Valdez: $(\boldsymbol{\mu}_V, \boldsymbol{\lambda}_V) \sim \text{NormalGamma}(m_2, c_2, a_2, b_2)$

Set $c_1 = c_2 = 1$ and $a_1 = a_2 = 1/2$ because we're pretty uncertain about the actual mean and variance for both cities, so the prior certainty is equivalent to info in 1 datapoints.

Based on the data provieded, we get the average annual snowfall acorss two countries as 301.6859. Since we're unsure about whether the mean annual snowfall in one city is higher than another, we can set $m_1 = m_2 = 302$.

```
c = rep(1,2)
a = rep(1/2,2)
m = rep(round(mean(c(aomori,valdez)),0),2) # m1, m2
print(m)
```

```
## [1] 302 302
```

Similarly, we can set the $b_1 = b_2 = \hat{\sigma}^2 * a$ where $a = 1/2$ and $\hat{\sigma}^2$ is the sample variance across two city.

```r
b = rep(round(sd(c(aomori,valdez))**2*1/2,0),2)
print(b) # b1, b2
```

```
## [1] 3253 3253
```

Then the rest of the simulation goes as follows

```r
# function to calculate the normal-gamma posterior from the prior
pos_normal_gamma = function(x,m,c,a,b){
  n = length(x)
  A = a + n/2
  C = c + n
  M = ((c*m) + sum(x))/(c+n)
  B = b + 1/2*(c*(m^2) - C*(M^2) + sum(x^2))
  return (c(M,C,A,B))
}

# function to sample n observations from normal-gamma distribution
rnormalgamma = function(n,m,c,a,b){
  lambda_sample = rgamma(n,a,b)
  mu_sample = m + rnorm(n)*(c*lambda_sample)^(-1/2)
  return(mu_sample)
}

# main
n = 10000 # simulation size

aomori_pos = pos_normal_gamma(aomori,m[1],c[1],a[1],b[1]) # posterior for Aomori
mu_A = do.call(rnormalgamma, as.list(c(n,aomori_pos))) # sampling

valdez_pos = pos_normal_gamma(valdez,m[2],c[2],a[2],b[2]) # posterior for Valdez
mu_V = do.call(rnormalgamma, as.list(c(n,valdez_pos))) # sampling

p_A_V  = sum(mu_A < mu_V)/n # counting
print(p_A_V)
```

```
## [1] 0.9909
```

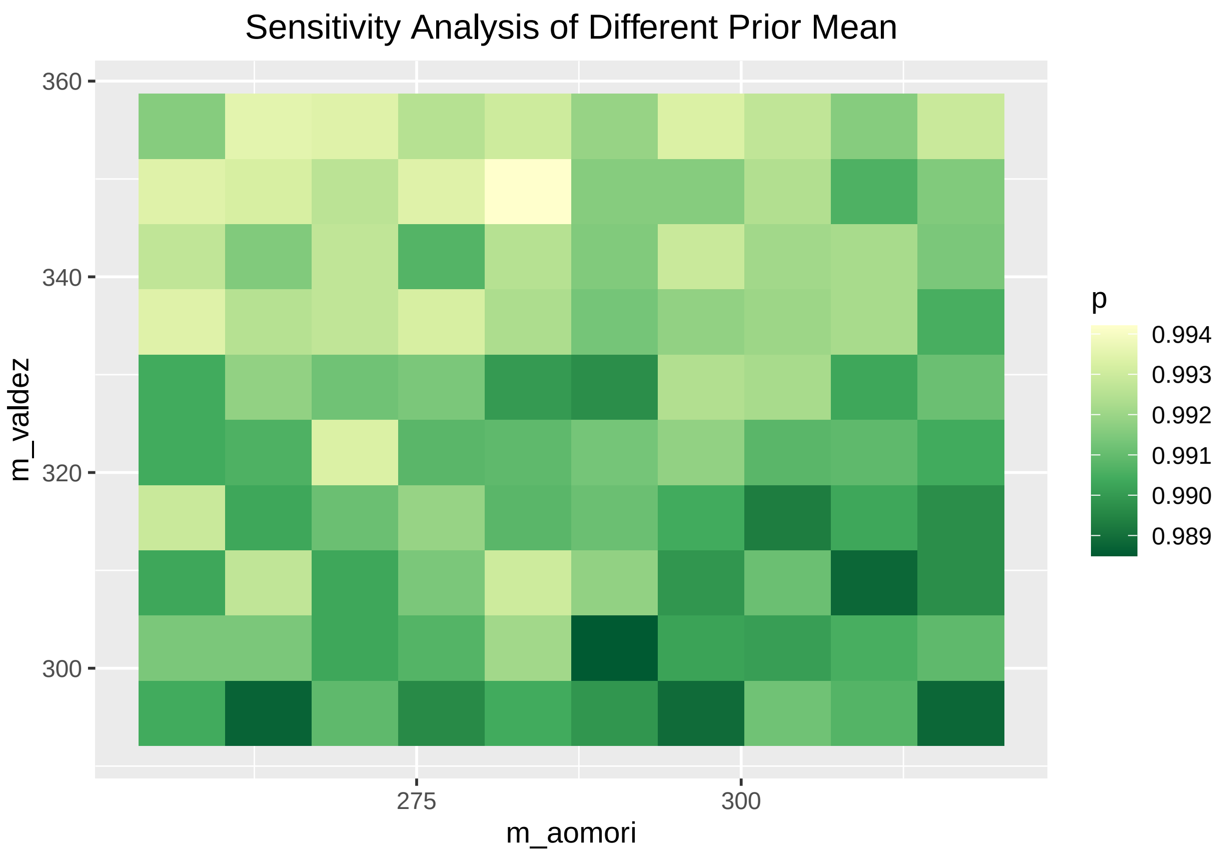
Based on our calculation, we have

$\mathbb{P}\left(\mu_V > \mu_A \mid x^A_{1:n_A}, x^V_{1:n_V}\right) \approx \frac{1}{10000}\sum_{i=1}^{N=10000} \mathrm{I}\left(\mu_V^{(i)} > \mu_A^{(i)}\right) \approx 0.99 > 0.95$

Therefore, we conclude that these data seem to support the hypothesis that the mean annual snowfall in Valdez is higher than that in Aomor.

(c)

```r
# reset the priors
b = c(sd(aomori)**2*1/2,sd(valdez)**2*1/2) # b based on sample variance
m_a = seq(mean(aomori)-30, mean(aomori)+30,length.out = 10)
m_v = seq(mean(valdez)-30, mean(valdez)+30,length.out = 10)
p_matrix  = matrix(0,100,3)

# main
for (i in 1:10){
  for (j in 1:10){

    aomori_pos = pos_normal_gamma(aomori,m_a[i],c[1],a[1],b[1]) # posterior for Aomori
    mu_A = do.call(rnormalgamma, as.list(c(n,aomori_pos))) # sampling
    valdez_pos = pos_normal_gamma(valdez,m_v[j],c[2],a[2],b[2]) # posterior for Valdez
    mu_V = do.call(rnormalgamma, as.list(c(n,valdez_pos))) # sampling
    p_matrix[10*(i-1)+j,]  = c(m_a[i], m_v[j], sum(mu_A < mu_V)/n) # counting
  }
}

# ggplot
library(ggplot2)
p_df = as.data.frame(p_matrix)
colnames(p_df) = c("m_aomori","m_valdez","p")
ggplot(p_df, aes(m_aomori, m_valdez, fill= p)) +
  geom_tile() + scale_fill_distiller(palette = "YlGn") +
  ggtitle("Sensitivity Analysis of Different Prior Mean")+
  theme(plot.title = element_text(hjust = 0.5))
```



Based on our observation, the change of the posterior probability of $\mathbb{P}\left(\mu_V > \mu_A \mid x^A_{1:n_A}, x^V_{1:n_V}\right)$ is extremely small with different m_1 and m_2 values. Therefore, our conclusion made in 1(b) is not sensitive and pretty robust.

2.

(a)

Based on the fomula given in the lecture, we have

$$\boldsymbol{\sigma^2} \mid y_{1:n} \sim \text{InvGamma}\left(\frac{1}{2}\left(\nu_0+n\right), \frac{1}{2}\left(\nu_0\sigma_0^2+\text{SSR}_g\right)\right)$$
$$\boldsymbol{\beta} \mid \sigma^2, y_{1:n} \sim \mathcal{N}\left(\frac{g}{g+1}\left(X^{\mathrm{T}}X\right)^{-1}X^{\mathrm{T}}y, \frac{g}{g+1}\sigma^2\left(X^{\mathrm{T}}X\right)^{-1}\right)$$

where

$$\text{SSR}_g = y^{\mathrm{T}}y - \frac{g}{g+1}y^{\mathrm{T}}X\left(X^{\mathrm{T}}X\right)^{-1}X^{\mathrm{T}}y$$

Set $g=n$, $\nu_0=1$ and $\sigma_0^2=\hat{\sigma}^2_{\text{MLE}}=\frac{1}{N}\sum_{i=1}^N\left(y_i-x_i\widehat{\beta}\right)^2$. Since we have the closed form, we can generate i.i.d. samples directly from the posterior.

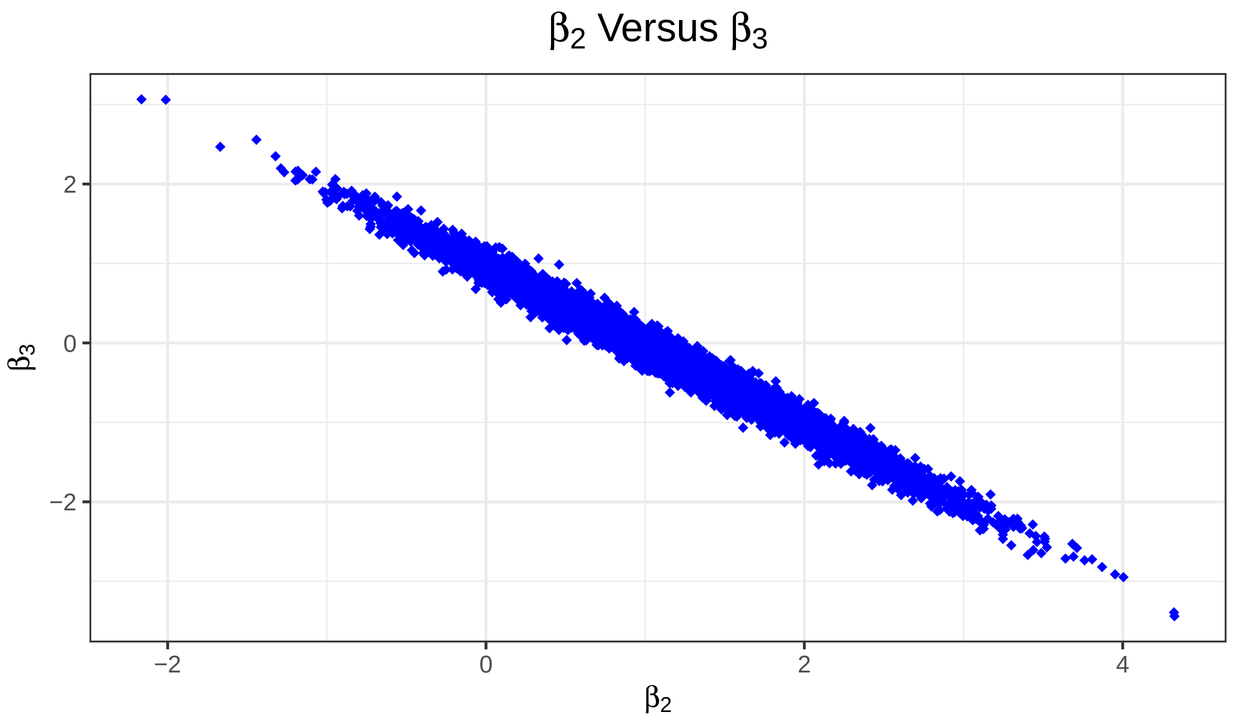
```
# import dataset
data = read.delim("~/Documents/Spring2022/BST249/Homework/HW3/homework-3-data.tsv")
y = as.matrix(data$y)
X = as.matrix(data[,2:4])

# a function to generate n MVN samples with mean vector mu_vec, scale matrix scale_mat and a vector of

mvn_generator = function(mu_vec, scale_mat, sig_vec, n){
  dim = length(mu_vec)
  normal_scale_sig = matrix(rnorm(n*dim,0, sig_vec),n,dim) # each row is a beta vector, scale using std
  normal_scale = normal_scale_sig %*% chol(scale_mat) # use Choleski decomposition to scale
  return (t(normal_scale) + c(mu_vec)) # each column is a sampled beta vector, n columns in total
}

# initialization
nu0 = 1
g = n = length(y)
beta_mle = solve(t(X)%*%X)%*%t(X)%*%y
sigma2_mle = sigma2_0 = (t(y - X%*%beta_mle) %*% (y - X%*%beta_mle))/ n
SSR_g = t(y)%*%y - g/(g+1)*(t(y)%*%X%*%beta_mle)

# set up sampling parameters

n_sim = 10000 # simulation size
inv_gamma_a = (nu0+n)/2 # sigma
inv_gamma_b = (nu0*sigma2_0 + SSR_g)/2 # sigma
beta_mu = g/(g+1)*beta_mle # beta
beta_scale_mat = g/(g+1)*solve(t(X)%*%X) # beta

# main: sample sigma and beta

sig_vec1 = (rgamma(n_sim,inv_gamma_a,inv_gamma_b))^(-1/2) #sample vector of sigma (standard deviation)
beta_vec1 = mvn_generator(beta_mu,beta_scale_mat,sig_vec1,n_sim) # sample beta from posterior

# report Monte Carlo approximations
print(mean(sig_vec1^2)) # expectation of sigma^2
```

```
## [1] 1.228081
```

```
print(rowMeans(beta_vec1)) # expectation of beta
```

```
##          x1          x2          x3
##  0.4155664  1.1546708 -0.1907255
```

```
beta3_1 = beta_vec1[3,]

# plot
gg_df_a = as.data.frame(t(beta_vec1))
colnames(gg_df_a) = c("beta1","beta2","beta3")
ggplot(gg_df_a, aes(x=beta2, y=beta3)) +
  geom_point(shape=18, color="blue") +
  ggtitle(expression(paste(beta[2], " Versus ", beta[3]))) +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5,face="bold", size=15)) +
  xlab(expression(beta[2])) + ylab(expression(beta[3])) +  theme(aspect.ratio=3/6)
```



Based on the scatterplot, the correlation between β_2 and β_3 is extremely strong. This correlation is quantified by the term $g\sigma^2\left(X^{\mathrm{T}}X\right)^{-1}$. More specifically, since the correlation between X_2 and X_3 is very strong, the corresponding term in the matrix $\left(X^{\mathrm{T}}X\right)^{-1}$ has a large magnitude.

```
cor(X[,2],X[,3])
```

```
## [1] 0.9893682
```

(b)

The conditional posterior distribution of β remains as

$$\boldsymbol{\beta} \mid \sigma^2, y_{1:n} \sim \mathcal{N}\left(\frac{g}{g+1}\left(X^{\mathrm{T}}X\right)^{-1}X^{\mathrm{T}}y, \frac{g}{g+1}\sigma^2\left(X^{\mathrm{T}}X\right)^{-1}\right)$$

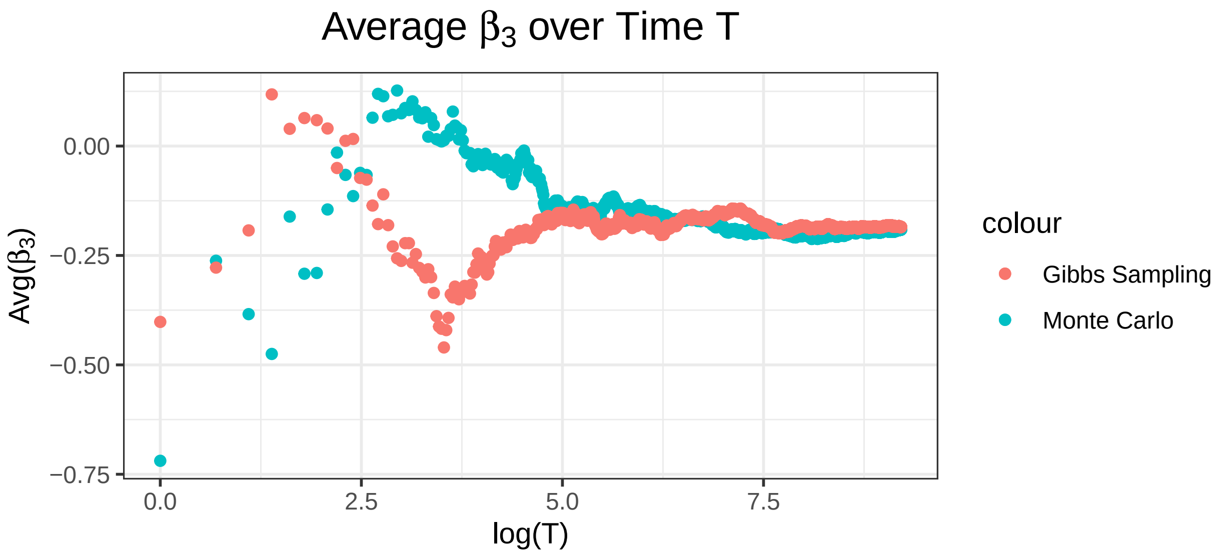
Derive the conditional posterior distribution of σ^2 as

$$
\begin{aligned}
p(\boldsymbol{\sigma^2} \mid \boldsymbol{\beta}, y_{1:n}) &\propto p(y_{1:n} \mid \beta, \sigma^2) \times p(\beta \mid \sigma^2) \times p(\sigma^2) \\
&\propto \left(\sigma^2\right)^{-\frac{n}{2}} e^{-\frac{(Y-X\beta)^T(Y-X\beta)}{2\sigma^2}} \times \left(\sigma^2\right)^{-\frac{1}{2}\nu_0 - 1} e^{-\frac{\nu_0\sigma_0^2/2}{\sigma^2}} \times \frac{\exp\left(-\frac{1}{2g\sigma^2}\beta^{\mathrm{T}}X^T X\beta\right)}{\sqrt{|g\sigma^2\left(X^{\mathrm{T}}X\right)^{-1}|}} \\
&\propto \exp(-\frac{1}{\sigma^2}\cdot\frac{1}{2}\{(Y-X\beta)^T(Y-X\beta) + \nu_0\sigma_0^2 + \frac{1}{g}\beta^{\mathrm{T}}X^T X\beta\}) \times (\sigma^2)^{-\frac{1}{2}(n+\nu_0)-1} \times \frac{1}{\sqrt{(\sigma^2)^3|g\left(X^{\mathrm{T}}X\right)^{-1}|}} \\
&\propto \exp(-\frac{1}{\sigma^2}\cdot\frac{1}{2}\{(Y-X\beta)^T(Y-X\beta) + \nu_0\sigma_0^2 + \frac{1}{g}\beta^{\mathrm{T}}X^T X\beta\}) \times (\sigma^2)^{-\frac{1}{2}(n+\nu_0)-1-\frac{3}{2}} \\
\boldsymbol{\sigma^2} \mid \boldsymbol{\beta}, y_{1:n} &\sim \mathrm{InvGamma}\left(\frac{1}{2}\left(\nu_0+n+3\right), \frac{1}{2}\left((Y-X\beta)^T(Y-X\beta) + \nu_0\sigma_0^2 + \frac{1}{g}\beta^{\mathrm{T}}X^T X\beta\right)\right)
\end{aligned}
$$

```r
# sigma sampler to sample one sigma from beta and y
inv_gamma_a_gibbs = (nu0+n+3)/2 # a parameter for gibbs

sigma_sampler = function(beta_pre){
  y_hat =  X%*%beta_pre
  inv_gamma_b_gibbs = (t(y - y_hat) %*% (y - y_hat) + nu0*sigma2_mle + 1/g*(t(y_hat)%*%y_hat))/2
  return(1/rgamma(1,inv_gamma_a_gibbs,inv_gamma_b_gibbs)) #sample sigma
}

# Gibbs sampler (n iterations), write out an array of simulated sigma and a matrix of simulated beta
gibbs_sampler_n = function(beta_start,n_sim){
  #set.seed(123)
  dim = length(beta_start)
  ret_sigma_vec = rep(0,n_sim) # vector to hold sigma samples
  ret_beta_mat = matrix(0,nrow = dim, ncol = n_sim+1) # each column is a sampled beta vector, n+1 colum
  ret_beta_mat[,1] = beta_start # starting point
  for (i in 1:n_sim){
    ret_sigma_vec[i] = sigma_sampler(ret_beta_mat[,i])
    ret_beta_mat[,i+1] = mvn_generator(beta_mu,beta_scale_mat,sqrt(ret_sigma_vec[i]),1) # sample beta
  }
  return(list(ret_sigma_vec,ret_beta_mat[,-1])) # remove the starting point of beta
}

# main
gibbs_sampling_ret = gibbs_sampler_n(beta_mle,n_sim+1000)
beta3_2 = gibbs_sampling_ret[[2]][3,1001:(n_sim+1000)] # extract all results of beta3, exclude burn-in

# ggplot

library("dplyr")
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
gg_df_b = data.frame(cbind(log(c(1:n_sim)),cummean(beta3_1),cummean(beta3_2)))
colnames(gg_df_b) = c("T","Monte_Carlo","Gibbs_Sampling")

ggplot(gg_df_b, aes(T)) +
  geom_point(aes(y = Monte_Carlo, colour = "Monte Carlo")) +
  geom_point(aes(y = Gibbs_Sampling, colour = "Gibbs Sampling")) +
  ggtitle(expression(paste("Average ", beta[3], " over Time T"))) +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5,face="bold", size=15)) +
  xlab("log(T)") + ylab(expression(paste("Avg(", beta[3],")"))) +  theme(aspect.ratio=3/6)
```



Based on the visualization above, we conclude that both samplers converge after about $T = e^5 \approx 150$ iterations and the rate of convergence is somewhat similar across methods.

(c)

I would expect this to do worse. Based on our scatterplot in 2a, there is a strong correlation between β_2 and β_3. As a result, if we sample β_2 conditioning on β_3 or sample β_3 conditioning on β_2, we may stuck at a specific region due to the high correlation between the adjacent samples but never move out, ending up with a seemingly good mixing (sample average quickly converge) but actually very poor one (never reach other regions of the sampling distribution).

(d)

Given the proposal distribution $\beta_{\text{prop}} \mid \beta \sim \mathcal{N}(\beta, 0.25I)$, we have the acceptance ratio $\alpha\left(\beta, \beta_{prop}\right) = \frac{\pi(\beta_{prop})q(\beta|\beta_{prop})}{\pi(\beta)q(\beta_{prop}|\beta)}$. Due to the symmetric property of the normal distribution, we can simplify the acceptance ratio as $\alpha\left(\beta, \beta_{prop}\right) = \frac{p(\boldsymbol{\beta_{prop}}|\sigma^2, y_{1:n})}{p(\boldsymbol{\beta}|\sigma^2, y_{1:n})}$ where $\boldsymbol{\beta} \mid \sigma^2, y_{1:n} \sim \mathcal{N}\left(\frac{g}{g+1}\left(X^{\mathrm{T}}X\right)^{-1}X^{\mathrm{T}}y, \frac{g}{g+1}\sigma^2\left(X^{\mathrm{T}}X\right)^{-1}\right)$

```r
# a MH sampler for beta

mh_beta_sampler = function(beta_pre, sigma_pre){
  beta_prop = mvn_generator(beta_pre, diag(3), sqrt(0.25), 1) # generate proposed beta
  a_ratio_num = exp(-1/2*(t(beta_prop-beta_mu)%*%solve(sigma_pre*beta_scale_mat)%*%(beta_prop-beta_mu))
  a_ratio_denom = exp(-1/2*(t(beta_pre-beta_mu)%*%solve(sigma_pre*beta_scale_mat)%*%(beta_pre-beta_mu))
  a_ratio = a_ratio_num/a_ratio_denom # acceptance ratio
  u = runif(1,0,1)
  if (u < a_ratio){
    return (beta_prop)
  }
  return (beta_pre)
}

# gibbs + mh sampler for n samples

gibbs_mh_sampler = function(beta_start,n_sim,setseed = FALSE){
  if(setseed){
    set.seed(12345)
  }
  dim = length(beta_start)
  ret_sigma_vec = rep(0,n_sim)
  ret_beta_mat = matrix(0,nrow = dim, ncol = n_sim+1) # each column is a sampled beta vector, n columns
  ret_beta_mat[,1] = beta_start
  for (i in 1:n_sim){
    ret_sigma_vec[i] = sigma_sampler(ret_beta_mat[,i])
    ret_beta_mat[,i+1] = mh_beta_sampler(ret_beta_mat[,i],ret_sigma_vec[i])
  }
  return(list(ret_sigma_vec,ret_beta_mat[,-1]))
}

# main

gibbs_mh_ret = gibbs_mh_sampler(beta_mle,n_sim+1000,TRUE)
beta3_3 = gibbs_mh_ret[[2]][3,1001:(n_sim+1000)] # extract all results of beta3, exclude the burn in 10
print(rowMeans(gibbs_mh_ret[[2]][,1001:(n_sim+1000)])) #sanity check
```

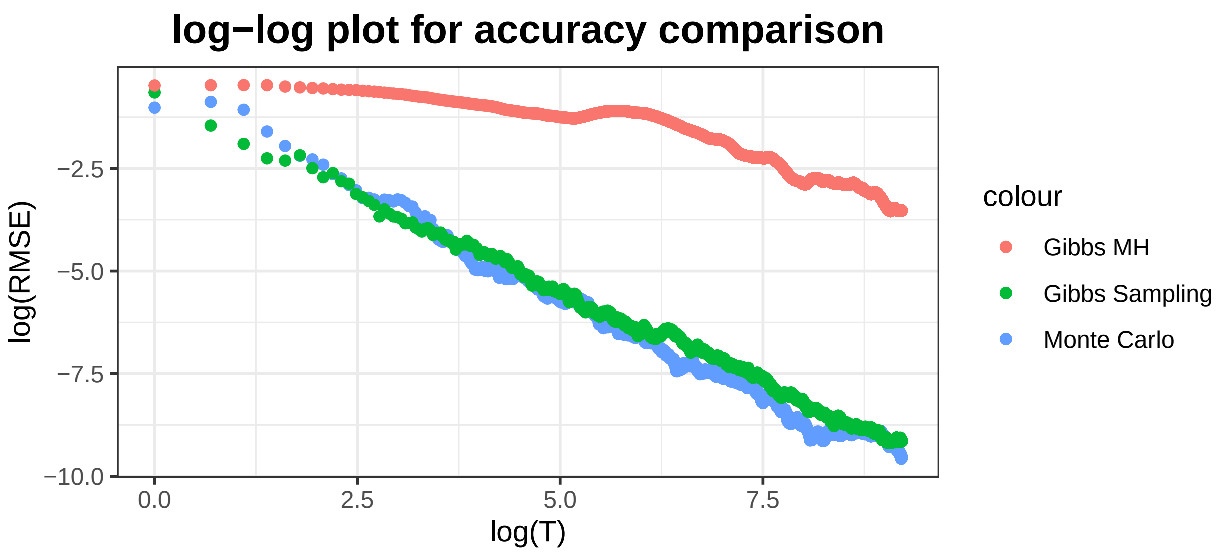
```
## [1]  0.4178391  1.1801475 -0.2137911
```

The approximate $E(\beta|y_{1:n}) = [0.4178391, 1.1801475, -0.2137911]^T$ using the seed we specified. This number is not too far away from our MLE estimator, which means that our sampling procedure makes sense.

(e)

```r
# simulate 25*10^4 samples for each method

# MC
beta3_1e = replicate(25,
                     {
                       sig_vec1 = (rgamma(n_sim,inv_gamma_a,inv_gamma_b))^(-1/2) #sample sigma
                       beta_vec1 = mvn_generator(beta_mu,beta_scale_mat,sig_vec1,n_sim) # sample beta f
                       beta_vec1[3,] # extract beta3
                     })

# Gibbs

beta3_2e = replicate(25,
                     {
                       gibbs_sampling_ret = gibbs_sampler_n(beta_mle,n_sim+1000)
                       gibbs_sampling_ret[[2]][3,1001:(n_sim+1000)] # extract all results of beta3
                     })
```

```r
# Gibbs + MH

beta3_3e = replicate(25,
                     {
                       gibbs_mh_ret = gibbs_mh_sampler(beta_mle,n_sim+1000)
                       gibbs_mh_ret[[2]][3,1001:(n_sim+1000)] # extract all results of beta3, exclude t
                     })


# A function to calculate RMSE(a,T), take 10^4 by 25 matrix as input
b3 = -0.1901

RMSE = function(beta_matrix){
  RMSE_r_matrix = apply(beta_matrix, 2,function(x) {(cummean(x)-b3)^2}) # cumulative mean difference ov
  return (rowMeans(RMSE_r_matrix)) #average over r replicates
}

# ggplot, take log for all measurements

gg_df_e = data.frame(cbind(log(c(1:n_sim)),log(RMSE(beta3_1e)),log(RMSE(beta3_2e)),log(RMSE(beta3_3e)))
colnames(gg_df_e) = c("T","Monte_Carlo","Gibbs_Sampling", "Gibbs_MH")

ggplot(gg_df_e, aes(T)) +
  geom_point(aes(y = Monte_Carlo, colour = "Monte Carlo")) +
  geom_point(aes(y = Gibbs_Sampling, colour = "Gibbs Sampling")) +
  geom_point(aes(y = Gibbs_MH, colour = "Gibbs MH")) +
  ggtitle("log-log plot for accuracy comparison") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5,face="bold", size=15)) +
  xlab("log(T)") + ylab("log(RMSE)") +  theme(aspect.ratio=3/6)
```



Based on the visualization, the RMSE decreases over time for all three samplers, indicating the procedure of convergency (as expected). However, the rate of convergence is much faster for Gibbs sampler and Monte Carlo sampler compared to the Gibb_MH one. It is also as expected because the Gibbs sampling is a special case of Metropolis–Hastings but always accept the proposed value. Therefore, Gibbs sampler coverge faster whereas MH sampler rejects some proposed values and converge slower.

The RMSE after 10000 iterations is extremely small for Gibbs and MC, indicating that the sample means of these two samplers have already converged to the "accurate MC approximation". Therefore, the results we get from these two samplers are pretty good, as expected.