Help File

Lab 1

Assignment 1 *Daniel Bernoulli*

Let $y_1, ..., y_n \mid \theta \sim \text{Bern}(\theta)$, and assume that you have obtained a sample with s = 13 successes in n = 50 trials. Assume a Beta(α_0, β_0) prior for θ and let $\alpha_0 = \beta_0 = 5$.

Task 1a

Question:Draw random numbers from the posterior $\theta \mid y \sim Beta(\alpha_0 + s, \beta_0 + f)$, where $y = (y_1, ..., y_n)$, and verify graphically that the posterior mean $E[\theta|y]$ and standard deviation $SD[\theta|y]$ converges to the true values as the number of random draws grows large. [Hint: use rbeta()].

The mean value and the standard deviation of the Beta distribution for θ are calculated by the below formulas:

$$\begin{aligned} E[\theta] &= \frac{a_0 + s}{a_0 + s + b_0 + f} \\ &= \frac{18}{60} \\ &= 0.3 \end{aligned}$$

$$\begin{aligned} Var[\theta] &= \frac{(a_0 + s)(b_0 + f)}{\Big((a_0 + s) + (b_0 + f)\Big)^2 \Big((a_0 + s) + (b_0 + f) + 1\Big)} \\ &= \frac{18 \cdot 42}{(18 + 42)^2 (18 + 42 + 1)} \\ &= 0.003442623 \end{aligned}$$

$$SD[\theta] = \sqrt{Var[\theta]} = 0.05867387$$

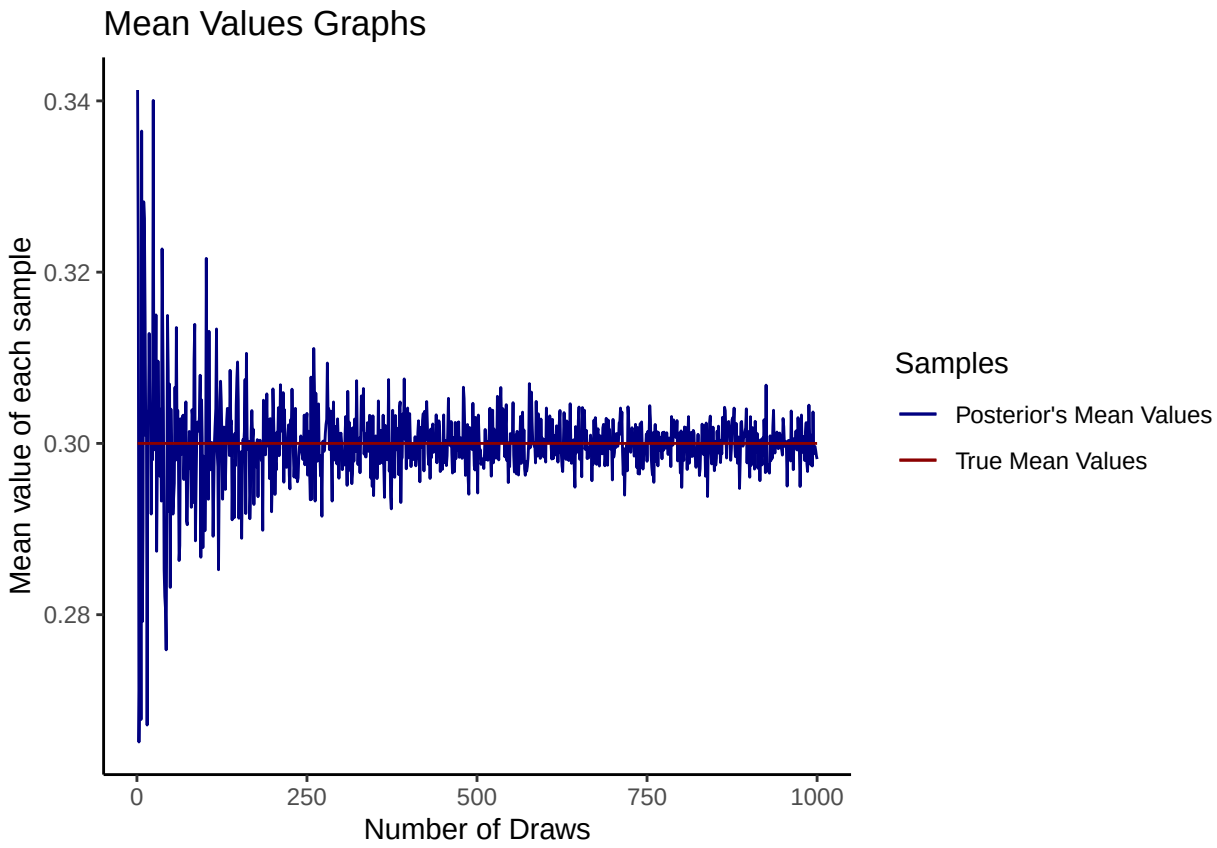
Where a_0 and b_0 are the arguments of the Beta prior, s is the number of successes and f is the number of failures.

```
# parameters
a0 <- 5
b0 <- 5
n <- 50
s <- 13
f <- n - s

# true mean
mean_true <- (a0 + s)/(a0 + s + b0 + f)
# true var
var_true <- ((a0 + s) * (b0 + f))/(((a0 + s + b0 + f)^2) * (a0 +
    s + b0 + f + 1))
# true sd
sd_true <- sqrt(var_true)
```

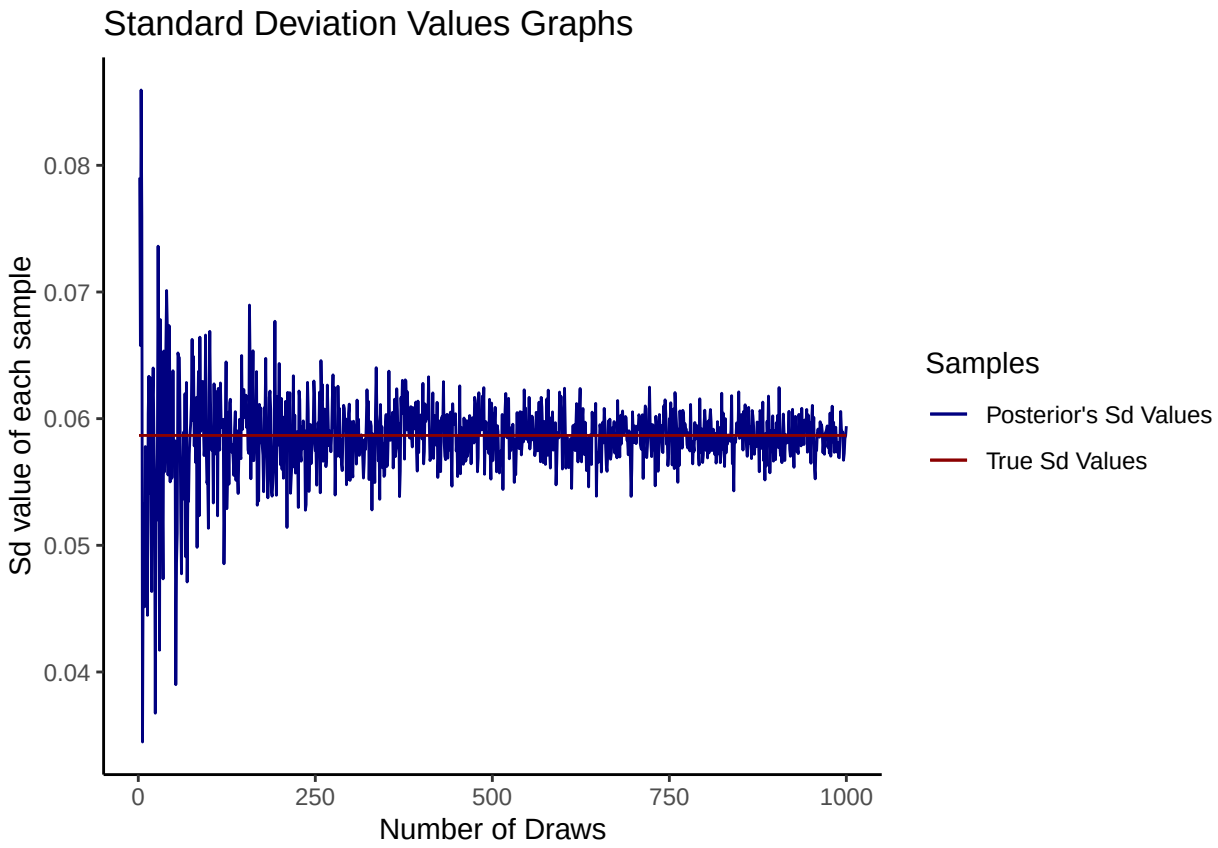
```r
set.seed(12345)

# calculate posterior's mean
mean_posterior = c()
for (i in 1:1000) {
    # rbeta generates random deviates
    mean_posterior[i] = mean(rbeta(n = i, shape1 = a0 + s, shape2 = b0 +
        f))
}

# calculate posterior's sd
sd_posterior = c()
for (i in 1:1000) {
    sd_posterior[i] = sd(rbeta(n = i, shape1 = a0 + s, shape2 = b0 +
        f))
}
```

The graphs represent the mean value and the standard deviation of the Beta distribution for θ and the mean value $E[\theta|y]$ (red line) and standard deviation $SD[\theta|y]$ from the posterior $\theta|y \sim Beta(a_0 + s, b_0 + f)$ (blue points), where $y = y_1, y_2, ..., y_n$.

```r
# data for plots
df_plot1 <- data.frame(draws = 1:1000, mean_true = mean_true,
    sd_true = sd_true, mean_posterior = mean_posterior, sd_posterior = sd_posterior)

library(ggplot2)

# plot of mean values
ggplot(df_plot1) + geom_line(aes(x = draws, y = mean_posterior,
    color = "nany")) + geom_line(aes(x = draws, y = mean_true,
    color = "red4")) + theme(legend.position = "right") + scale_color_manual(values = c("navy",
    "red4"), name = "Samples", labels = c("Posterior's Mean Values",
    "True Mean Values")) + ggtitle("Mean Values Graphs") + xlab("Number of Draws") +
    ylab("Mean value of each sample") + theme_classic()
```




```
# plot of sd values
ggplot(df_plot1) + geom_line(aes(x = draws, y = sd_posterior,
    color = "nany")) + geom_line(aes(x = draws, y = sd_true,
    color = "red4")) + theme(legend.position = "right") + scale_color_manual(values = c("navy",
    "red4"), name = "Samples", labels = c("Posterior's Sd Values",
    "True Sd Values")) + ggtitle("Standard Deviation Values Graphs") +
    xlab("Number of Draws") + ylab("Sd value of each sample") +
    theme_classic()
```



From the above plots, it could be seen that both posterior's mean and standard deviation values converge to the actual mean and standard deviation values, respectively. More specifically, between 0 and approximately 250 draws in both graphs, some of the posterior's values abstain from true values. However, after the 250 draws, the posterior's values start to converge more and more to the true ones in both graphs.

Task 1b

Question:Use simulation (nDraws = 10000) to compute the posterior probability $\Pr(\theta < 0.3|y)$ and compare with the exact value from the Beta posterior. [Hint: use pbeta()].

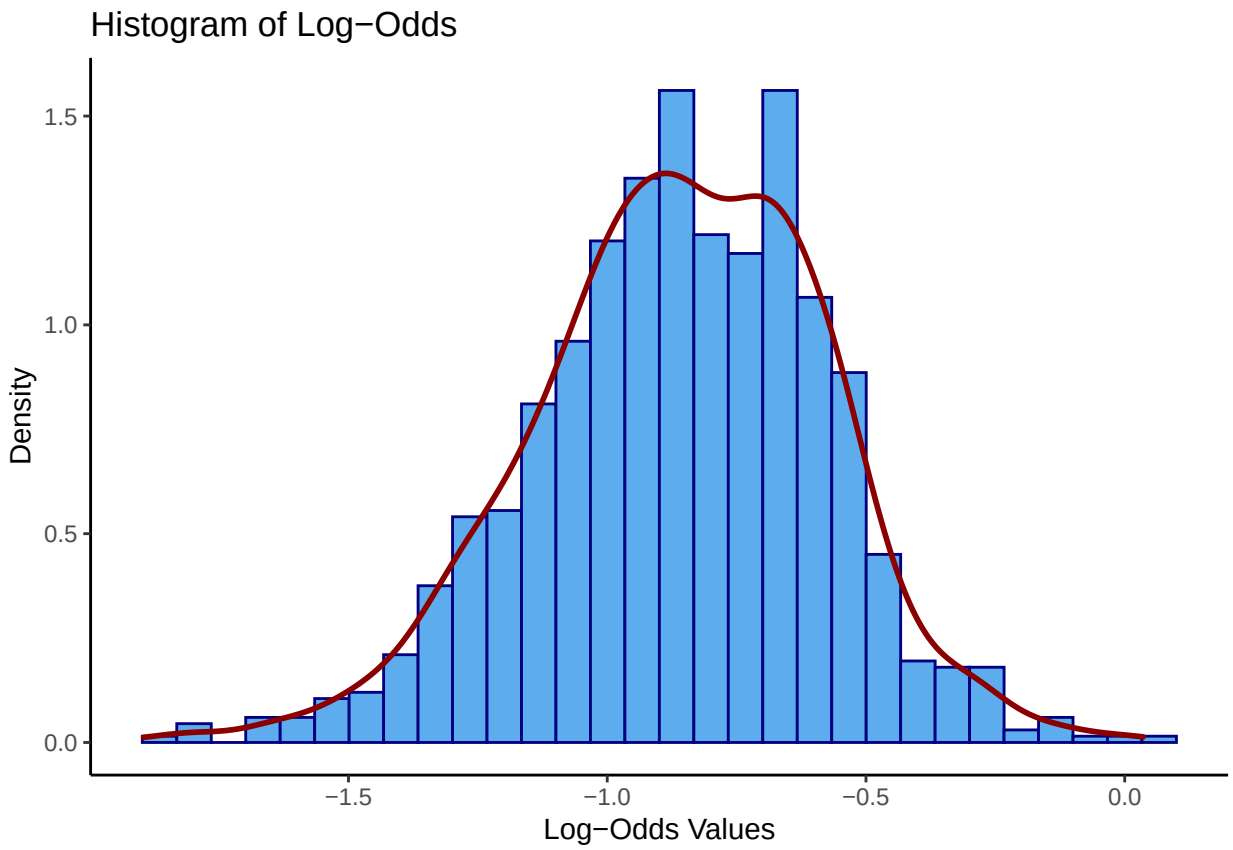
```
set.seed(12345)

# generates 1,000 random deviates.
posterior_sample <- rbeta(n = 1000, shape1 = a0 + s, shape2 = b0 +
    f)

# posterior probability
posterior_prob <- sum(posterior_sample < 0.3)/1000

# exact posterior prob pbeta the distribution function
exact_prob <- pbeta(q = 0.3, shape1 = a0 + s, shape2 = b0 + f)
```

The posterior probability $P(\theta < 0.3|y)$ equals 0.506, and the exact probability value from the Beta posterior is 0.5150226; thus, it could be assumed that both values are pretty similar.

Task 1c

Question:Simulate draws from the posterior distribution of the log-odds $\phi = log(\frac{\theta}{1-\theta})$ using simulated draws from the Beta posterior for θ and plot the posterior distribution of ϕ (nDraws = 10000). [Hint: hist() and density() can be utilized].

```
# log-odds
phi <- log(posterior_sample/(1 - posterior_sample))

# data for plot
df_plot2 <- data.frame(phi = phi)

# plot of log-odds
ggplot(df_plot2, aes(x = phi)) + geom_histogram(bins = 30, color = "navy",
    fill = "steelblue2", aes(y = ..density..)) + geom_density(colour = "red4",
    size = 1) + ggtitle("Histogram of Log-Odds") + xlab("Log-Odds Values") +
    ylab("Density") + theme_classic()
```



Assignment 2 *Log-normal distribution and the Gini coefficient.*

Assume that you have asked 9 randomly selected persons about their monthly in- come (in thousands Swedish Krona) and obtained the following nine observations: 33, 24, 48, 32, 55, 74, 23, 76 and 17. A common model for non-negative continuous variables is the log-normal distribution. The log-normal distribution $log\ \mathcal{N}(\mu, \sigma^2)$ has density function:

$$p(y \mid \mu, \sigma^2) = \frac{1}{y\sqrt{2\pi\sigma^2}} \exp\big[-\frac{1}{2\sigma^2}(\log\ y - \mu^2)\big]$$

where $y > 0, \mu > 0$ ND σ^2. The log-normal distribution is related to the normal distribution as follows: if $y \sim \log\ \mathcal{N}(\mu, \sigma^2)$ then $\log\ y \sim \mathcal{N}(\mu, \sigma^2)$. Let $y_1, ..., y_n \mid \mu, \sigma^2 \overset{\text{iid}}{\sim} log\mathcal{N}(\mu, \sigma^2)$, where $\mu = 3.5$ is assumed to be known but σ^2 is unknown with non-informative prior $p(\sigma^2) \propto \frac{1}{\sigma^2}$. The posterior for σ^2 is the $Inv\ \chi^2(n, \tau^2)$ distribution, where

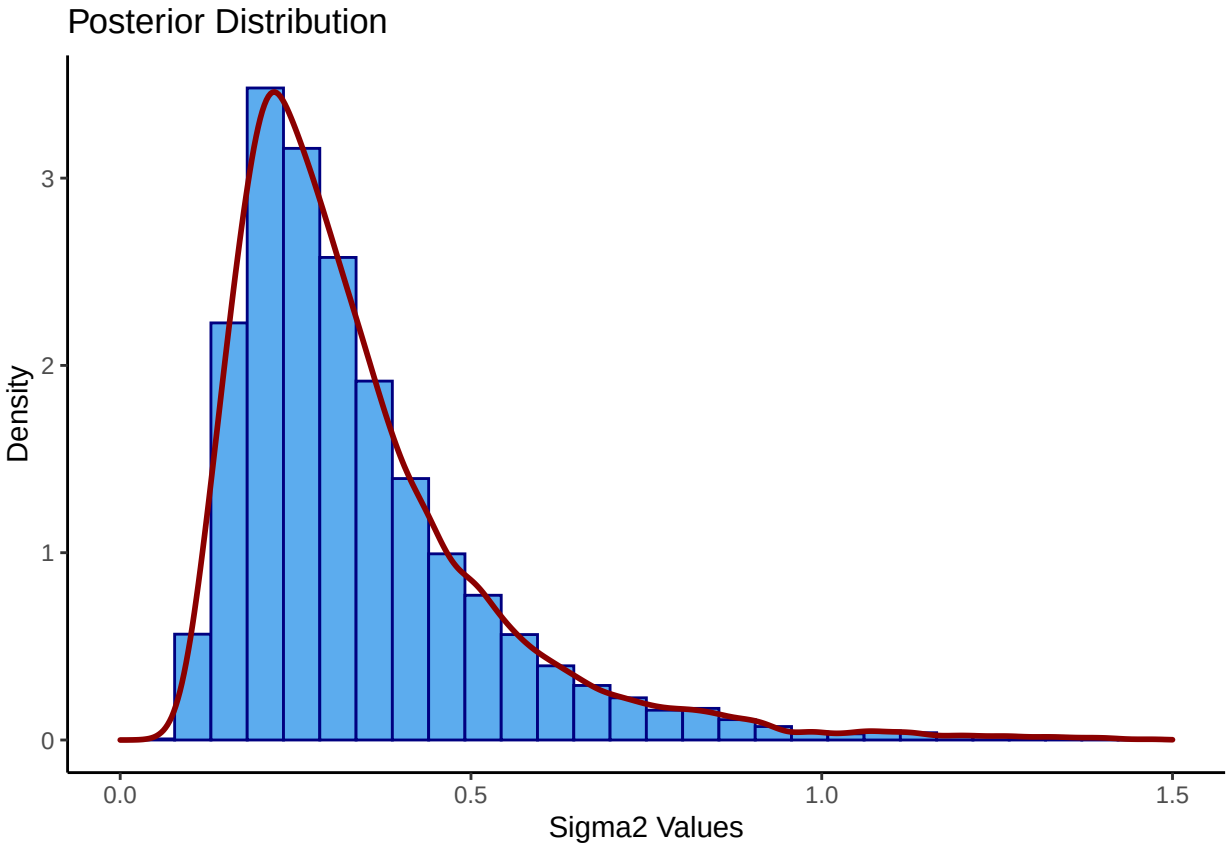
$$\tau^2 = \frac{\sum_{i=1}^{2}(\log\ y_i - \mu)^2}{n}$$

Task 2a

Question:Simulate 10, 000 draws from the posterior of σ^2 by assuming $\mu = 3.5$ and plot the posterior distribution.

```
# observations
obs <- c(33, 24, 48, 32, 55, 74, 23, 76, 17)
n <- length(obs)

# tau^2
tau_2 <- sum((log(obs) - 3.5)^2)/n

# generates 10,000 random deviates.
set.seed(12345)
sigma2 <- c()
for (i in 1:10000) {
    # simulate from chi squared distribution
    sigma2[i] <- 9 * tau_2/rchisq(1, n)
}
```

```
# data for plot
df_plot2 <- data.frame(sigma2 = sigma2)

# plot
ggplot(df_plot2, aes(x = sigma2)) + geom_histogram(bins = 30,
    color = "navy", fill = "steelblue2", aes(y = ..density..)) +
    geom_density(colour = "red4", size = 1) + scale_x_continuous(limits = c(0,
    1.5)) + ggtitle("Posterior Distribution") + xlab("Sigma2 Values") +
    ylab("Density") + theme_classic()
```



Task 2b

Question The most common measure of income inequality is the Gini coefficient, G, where $0 \leqslant G \leqslant 1$. G = 0 means a completely equal income distribution, whereas G = 1 means complete income inequality (see e.g. Wikipedia for more information about the Gini coefficient). It can be shown that $G = 2\Phi(\frac{\sigma}{\sqrt{2}}) - 1$ when incomes follow a log $\mathcal{N}(\mu, \sigma)$ distribution. $\Phi(z)$ is the cumulative distribution function (CDF) for the standard normal distribution with mean zero and unit variance. Use the posterior draws in a) to compute the posterior distribution of the Gini coefficient G for the current data set.

```
# Gini calculation
gini <- 2 * pnorm(sqrt(sigma2)/sqrt(2)) - 1
```

Task 2c

Question:Use the posterior draws from b) to compute a 95% equal tail credible interval for G. A 95% equal tail interval (a,b) cuts off 2.5% percent of the posterior probability mass to the left of a, and 2.5% to the right of b.

```
# producing sample quantiles corresponding to the
# probabilities
interval <- quantile(gini, probs = c(0.025, 0.975))

# table for the interval
df_intervals <- data.frame(lower_bound = interval[1], upper_bound = interval[2])
colnames(df_intervals) <- c("lower bound", "upper bound")
rownames(df_intervals) <- c("95% Equal Tail Credible Interval")
knitr::kable(df_intervals)
```

	lower bound	upper bound
95% Equal Tail Credible Interval	0.197551	0.4908131

The above table illustrates the 95% equal tail credible interval for the Gini coefficient.

Task 2d

Question 2d:Use the posterior draws from b) to compute a 95% Highest Posterior Density Interval (HPDI) for G. Compare the two intervals in (c) and (d). [Hint: do a kernel density estimate of the posterior of G using the density function in R with default settings, and use that kernel density estimate to compute the HPDI. Note that you need to order/sort the estimated density values to obtain the HPDI.].

```
#kernel density estimation
gini_density <- density(gini)

df_density <- data.frame(
  #the n coordinates of the points where the density is estimated
  "coord" = gini_density$x,
  #the estimated density values
  "estimated_vals" = gini_density$y)

#order/sort the estimated density values
df_density <- df_density[order(gini_density$y, decreasing = TRUE),]

#calculate the probs
df_density$probs <- cumsum(df_density$estimated_vals)/sum(df_density$estimated_vals)

#get the indexes
index <- which(df_density$probs <= 0.95)

#get the probs
hdps <- df_density[index,]

#low hdpi
low_hdpi <- min(hdps$coord)
#upper hdpi
upp_hdpi <- max(hdps$coord)

intervals <- data.frame("lower_bound" = c(interval[1],low_hdpi), "upper_bound" = c(interval[2],upp_hdpi
rownames(intervals) <- c("95% Equal Tail Credible Interval","95% Highest Posterior Density Interval")

knitr::kable(intervals)
```

	lower_bound	upper_bound
95% Equal Tail Credible Interval	0.1975510	0.4908131
95% Highest Posterior Density Interval	0.1810286	0.4621952

The above table illustrates the 95% equal tail credible interval and the 95% highest posterior density interval for the Gini coefficient. It could be assumed that both intervals have almost similar lower and upper bounds.

Assignment 3 *Bayesian inference for the concentration parameter in the von Mises distribution*

This exercise is concerned with directional data. The point is to show you that the posterior distribution for somewhat weird models can be obtained by plotting it over a grid of values. The data points are observed wind directions at a given location on 10 different days. The data are recorded in degrees: (285, 296, 314, 20, 299, 296, 40, 303, 326, 308), where North is located at zero degrees (see Figure 1 on the next page, where the angles are measured clockwise). To fit with Wikipedias description of probability distributions for circular data we convert the data into radians $-\pi \leqslant y \leqslant \pi$. The 10 observations in radians are: (1.83, 2.02, 2.33, -2.79, 2.07, 2.02, -2.44, 2.14, 2.54, 2.23). Assume that these data points conditional on (μ, κ) are independent observations from the following von Mises distribution:

$$p(y \mid \mu, \kappa) = \frac{\exp[\kappa \cdot \cos(y - \mu)]}{2\pi I_0(\kappa)}, -\pi \leqslant y \leqslant \pi$$

where $I_0(\kappa)$ is the modified Bessel function of the first kind of order zero. The parameter $\mu(-\pi \leqslant y \leqslant \pi)$ is the mean direction and $\kappa > 0$ is called the concentration parameter. Large κ gives a small variance around μ, and vice versa. Assume that μ is known to be 2.39. Let $\kappa \sim Exponential(\lambda = 1)$ a priori, where λ is the rate parameter of the exponential distribution (so that the mean is $1/\lambda$).

Task 3a

Question:Derive the expression for what the posterior $p(\kappa|y, \mu)$ is proportional to. Hence, derive the function $f(\kappa)$ such that $p(\kappa|y, \mu) \propto f(\kappa)$. Then, plot the posterior distribution of κ for the wind direction data over a fine grid of κ values. [Hint: you need to normalize the posterior distribution of κ so that it integrates to one.]

The posterior is given by the below expression:

$$p(\kappa|y, \mu) = \frac{p(y, \mu|\kappa) \cdot p(\kappa)}{\int_k p(y, \mu|\kappa) \cdot p(\kappa) d\kappa}$$

The numerator is consisted of the likelihood $p(y, \mu|\kappa)$ and the prior $p(\kappa)$. The denominator is the marginal likelihood, which is the normalising constant, ensuring that the posterior distribution of κ adds up to one.

The likelihood is given by the below formula:

$$\begin{aligned}
p(y, \mu|\kappa) &= \prod_{i=1}^{n} p(y|\mu, \kappa) \\
&= \prod_{i=1}^{n} \frac{\exp(\kappa \cdot \cos(y_i - \mu))}{2\pi \cdot I_o(\kappa)} \\
&= \Big(\frac{1}{2\pi \cdot I_o(\kappa)}\Big)^n \exp(\kappa \cdot \sum_{i=1}^{n} \cos(y_i - \mu)) \\
&= \frac{1}{(2\pi)^n} \cdot \frac{1}{I_o(\kappa)^n} \cdot \exp(\kappa \cdot \sum_{i=1}^{n} \cos(y_i - \mu)) \\
&\propto \frac{1}{I_o(\kappa)^n} \cdot \exp(\kappa \cdot \sum_{i=1}^{n} \cos(y_i - \mu))
\end{aligned}$$

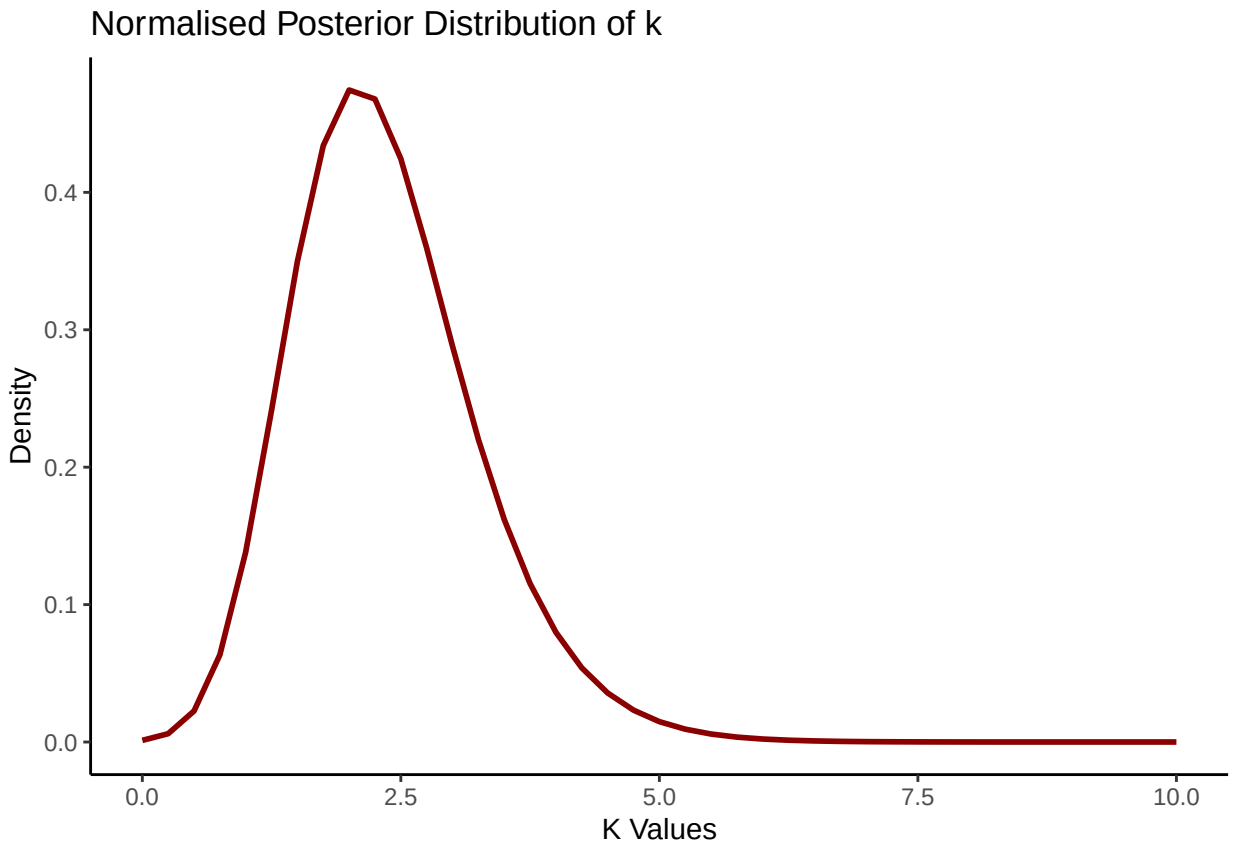
It is given that $\kappa \sim Exp(\lambda = 1)$; thus, it is only needed to calculate the probability density function of κ.

$$\begin{aligned}
p(\kappa) &= \lambda \cdot \exp(-\lambda x) \\
&= \exp(-\kappa)
\end{aligned}$$

Hence the numerator is given by the below expression:

$$p(\kappa|y,\mu) = \frac{1}{I_o(\kappa)^n} \cdot \exp(\kappa \cdot \sum_{i=1}^{n} \cos(y_i - \mu)) \cdot \exp(-\kappa)$$
$$= \frac{1}{I_o(\kappa)^n} \cdot \exp\left(\kappa \cdot \left(\sum_{i=1}^{n} \cos(y_i - \mu) - 1\right)\right)$$

```r
# data
degrees <- c(285, 296, 314, 20, 299, 296, 40, 303, 326, 308)
radians = c(-2.44, 2.14, 2.54, 1.83, 2.02, 2.33, -2.79, 2.23,
    2.07, 2.02)

# kappa values
k <- seq(0, 10, 0.25)

# posterior
posterior <- exp(k * (sum(cos(radians - 2.51)) - 1))/(besselI(x = k,
    nu = 0)^10)

# normalising constant/marginal likelihood
norm_constant <- 0.25 * sum(posterior)

# normalised posterior
norm_posterior <- posterior/norm_constant

# data for plotting
df_plot3 <- data.frame(k = k, posterior_vals = norm_posterior)

# plot
ggplot(df_plot3) + geom_line(aes(x = k, y = posterior_vals),
    colour = "red4", size = 1) + ggtitle("Normalised Posterior Distribution of k") +
    xlab("K Values") + ylab("Density") + theme_classic()
```



Task 3b

Question:Find the (approximate) posterior mode of κ from the information in a).

```
# get the index of the max value
max_index <- which.max(norm_posterior)

# get the max value
k_mode <- norm_posterior[max_index]
```

The approximate posterior mode of κ equals 0.04552635

Lab 2

Assignment 1 *Linear and polynomial regression*

The dataset *TempLambohov.txt* contains daily temperatures (in Celcius degrees) at at Lambohov, Linköping over the course of the year 2019. The response variable is temp and the covariate is

$$time = \frac{the\ number\ of\ days\ since\ beginning\ of\ year}{365}$$

A Bayesian analysis of the following quadratic regression model is to be performed:

$$temp = \beta_0 + \beta_1 \cdot\ time + \beta_2 \cdot\ time^2 + \varepsilon \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$$

```
# reading data
temperatures <- read.table("TempLambohov.txt", header = TRUE)
```

Task 1a

Question: Use the conjugate prior for the linear regression model. The prior hyperparameters μ_0, Ω_0, u_0 and σ_0^2 shall be set to sensible values.. Start with $\mu_0 = (-10, 100, -100)^T$,$\Omega_0 = 0.02 \cdot I_3$, $u_0 = 3$, $\sigma_0^2 = 2$. Check if this prior agrees with your prior opinions by simulating draws from the joint prior of all parameters and for every draw compute the regression curve. This gives a collection of regression curves; one for each draw from the prior. Does the collection of curves look reasonable? If not, change the prior hyperparameters until the collection of prior regression curves agrees with your prior beliefs about the regression curve. [Hint: the R package *mvtnorm* will be handy. And use your $Inv - \chi^2$ simulator from Lab 1.]

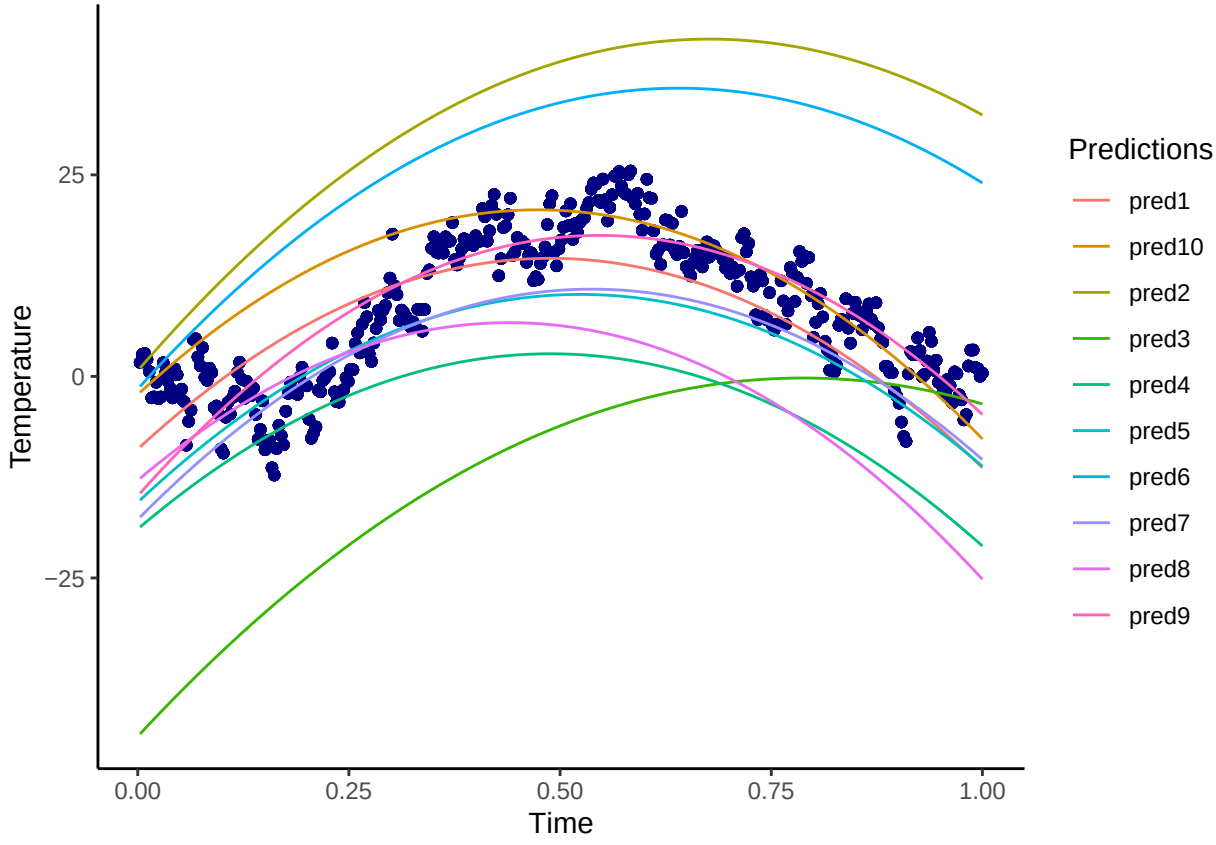
```
# given parameters
m_0 <- c(-10, 100, -100)
omega_0 <- 0.02 * diag(3)
n_0 <- 3
sigma2_0 <- 2
```

```
set.seed(123456)

library(mvtnorm)

# number of draws
N <- 10

# matrix to save b0, b1, b2
b <- matrix(NA, nrow = N, ncol = length(m_0))

# matrix to save the predicted temperatures
pred_temp <- matrix(NA, nrow = N, ncol = nrow(temperatures))

for (i in 1:N) {

    # Inv-x simulator from lab 1
    sigma2 <- n_0 * sigma2_0/rchisq(1, n_0)

    # generate b0,b1,b2
    b[i, ] <- rmvnorm(1, mean = m_0, sigma = sigma2 * solve(omega_0))

    # quadratic regression model
```

```
    pred_temp[i, ] <- b[i, 1] + b[i, 2] * temperatures$time +
        b[i, 3] * (temperatures$time^2) + rnorm(1, 0, sigma2)
}
```

```
# preparing data for plot
rownames(pred_temp) <- c("pred1", "pred2", "pred3", "pred4",
    "pred5", "pred6", "pred7", "pred8", "pred9", "pred10")
pred_temp <- t(pred_temp)

plot_data <- cbind(temperatures, pred_temp)

library(tidyr)

plot_data <- pivot_longer(plot_data, c(3:12))
```

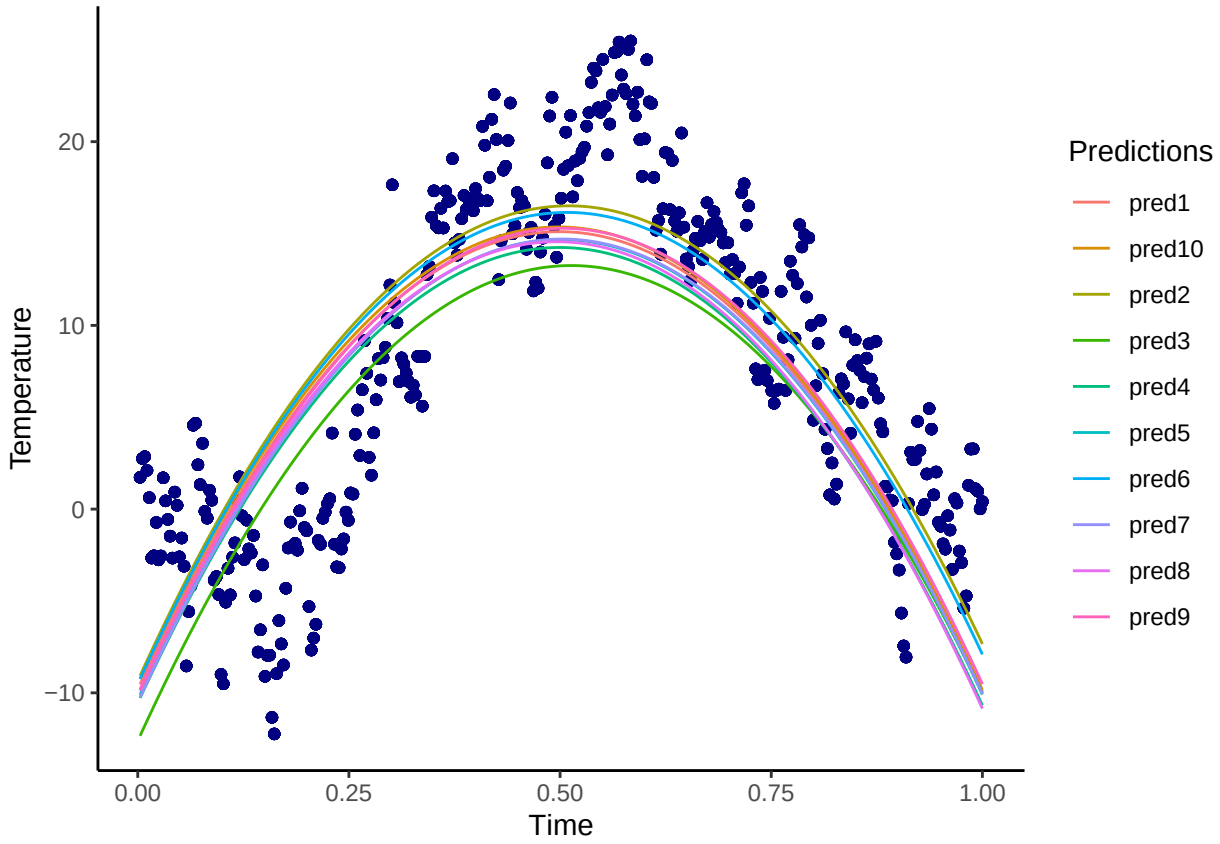
```
ggplot(plot_data) + geom_point(aes(x = time, y = temp), color = "navy") +
    geom_line(aes(x = time, y = value, color = name)) + theme(legend.position = "right") +
    guides(color = guide_legend("Predictions")) + xlab("Time") +
    ylab("Temperature") + theme_classic()
```



The above graph illustrates the actual temperatures (blue points) and a collection of regression curves, one for each draw from the prior. The graph provides mixed results almost half of the curves are not placed in the region of the actual temperatures, but it is not that bad. Two represent a little higher temperatures but follow the same pattern as the actual ones. Additionally, two curves represent the actual temperatures initially, but as time passes, they illustrate lower temperatures than the actual ones. Finally, only one curve shows poor results; in the very beginning has low temperatures, but in the end, the curve is in the region of the actual temperatures. Thus, the prior hyper-parameters need small configurations.

```r
# modefied parameters
m_0_mod <- c(-10, 100, -100)
omega_0_mod <- 0.5 * diag(3)  #before 0.2
n_0_mod <- 3
sigma2_0_mod <- 0.2  #before 2

set.seed(123456)

# number of draws
N <- 10

# matrix to save b0, b1, b2
b_mod <- matrix(NA, nrow = N, ncol = length(m_0_mod))

# matrix to save the predicted temperatures
pred_temp_mod <- matrix(NA, nrow = N, ncol = nrow(temperatures))

for (i in 1:N) {

    # Inv-x simulator from lab 1
    sigma2_mod <- n_0_mod * sigma2_0_mod/rchisq(1, n_0)

    # generate b0,b1,b2
    b_mod[i, ] <- rmvnorm(1, mean = m_0_mod, sigma = sigma2_mod *
        solve(omega_0_mod))

    # quadratic regression model
    pred_temp_mod[i, ] <- b_mod[i, 1] + b_mod[i, 2] * temperatures$time +
        b_mod[i, 3] * (temperatures$time^2) + rnorm(1, 0, sigma2_mod)
}
```

```r
# preparing data for plot
rownames(pred_temp_mod) <- c("pred1", "pred2", "pred3", "pred4",
    "pred5", "pred6", "pred7", "pred8", "pred9", "pred10")

pred_temp_mod <- t(pred_temp_mod)

plot_data_mod <- cbind(temperatures, pred_temp_mod)

plot_data_mod <- pivot_longer(plot_data_mod, c(3:12))

ggplot(plot_data_mod) + geom_point(aes(x = time, y = temp), color = "navy") +
    geom_line(aes(x = time, y = value, color = name)) + theme(legend.position = "right") +
    guides(color = guide_legend("Predictions")) + xlab("Time") +
    ylab("Temperature") + theme_classic()
```



The above graph illustrates the actual temperatures (blue points) and a collection of regression curves, one for each draw from the prior, but with the modified hyper-parameters. Only two parameters were changed, $\Omega_0 = 0.05$ and $\sigma^2 = 0.2$. As a result, all the curves are in the actual temperature region.

Task 1b

Question:Write a function that simulates draws from the joint posterior distribution of $\beta_0, \beta_1, \beta_2$ and σ^2.
i)Plot the marginal posteriors for each parameter as a histogram.
ii)Make a scatter plot of the temperature data and overlay a curve for the posterior median of the regression function $f(\ time) = E[temp|time] = \beta_0 + \beta_1 \cdot\ time + \beta_2 \cdot\ time^2$, i.e. the median of $f(time)$ is computed for every value of time. In addition, overlay curves for the 95% equal tail posterior probability intervals of $f(time)$, i.e. the 2.5 and 97.5 posterior percentiles of $f(time)$ is computed for every value of time. Does the posterior probability intervals contain most of the data points? Should they?

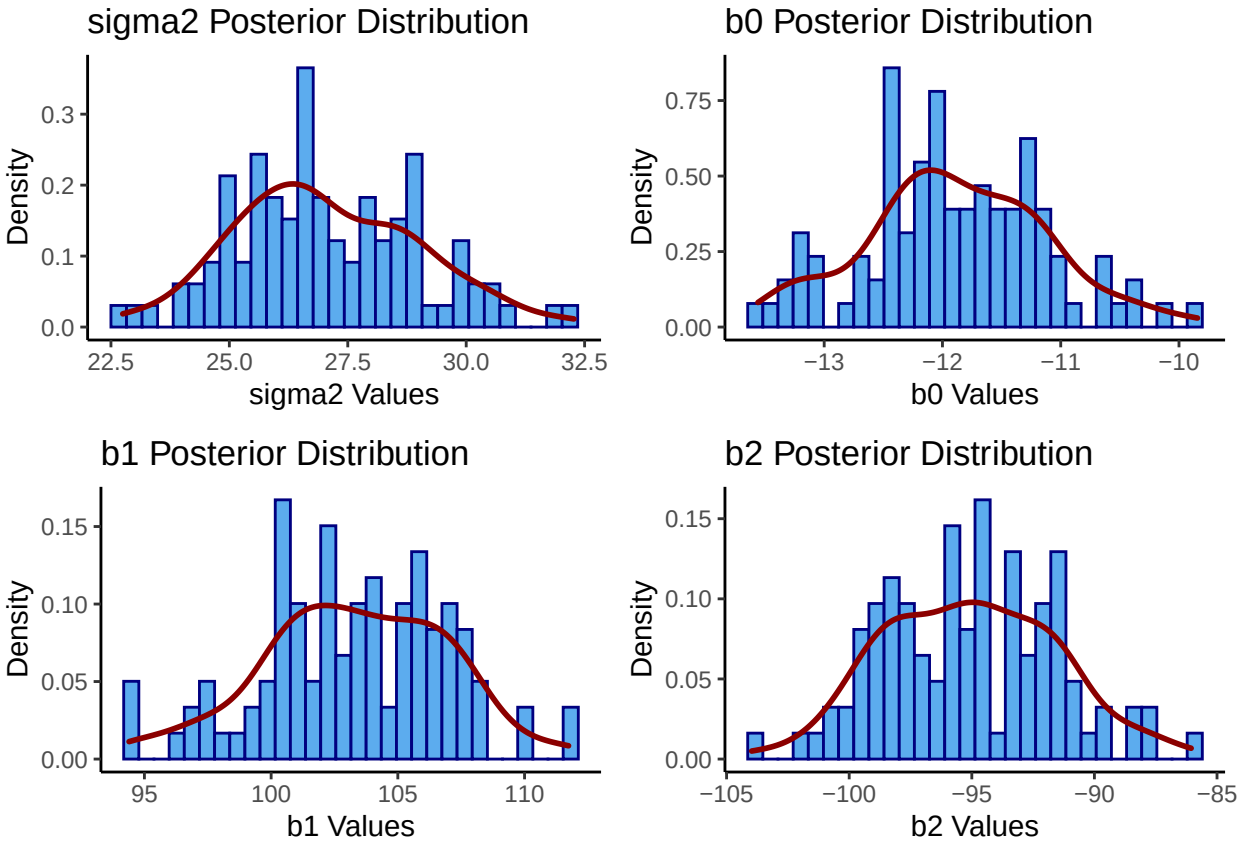
i)

```
set.seed(123456)

# matrix with the times (1st collum with 1 because of the
# intercept)
X <- cbind(rep(1, nrow(temperatures)), temperatures$time, temperatures$time^2)
# vector with target values
y <- temperatures$temp
# calculate b hat
b_hat <- solve(t(X) %*% X) %*% t(X) %*% y

# number of samples
```

```r
N <- 100

# matrix to store the beta values
b <- matrix(0, N, length(m_0))
# vector to store sigma2 values
sigma2 <- c()

for (i in 1:N) {

    # calculate mu_n
    m_n <- solve(t(X) %*% X + omega_0) %*% (t(X) %*% X %*% b_hat +
        omega_0 %*% m_0)

    # calculate omega_n
    omega_n <- t(X) %*% X + omega_0

    # calculate n_n
    n_n <- n_0 + nrow(temperatures)

    # calculate sigma_n
    sigma2_n <- (n_0 * sigma2_0 + (t(y) %*% y + t(m_0) %*% omega_0 %*%
        m_0 - t(m_n) %*% omega_n %*% m_n))/n_n

    # Inv-x simulator from lab 1
    sigma2[i] <- n_n * sigma2_n/rchisq(1, n_n)

    # generate b0,b1,b2
    b[i, ] <- rmvnorm(1, mean = m_n, sigma = sigma2[i] * solve(omega_n))

}
```

```r
library(gridExtra)

# df of sigma2 for plot
plot_df_sigma2 <- data.frame(sigma2 = sigma2)

# histogram of sigma2
sigma2_hist <- ggplot(plot_df_sigma2, aes(x = sigma2)) + geom_histogram(bins = 30,
    color = "navy", fill = "steelblue2", aes(y = ..density..)) +
    geom_density(colour = "red4", size = 1) + ggtitle("sigma2 Posterior Distribution") +
    xlab("sigma2 Values") + ylab("Density") + theme_classic()

# df of betas for plot
plot_df_b <- data.frame(b0 = b[, 1], b1 = b[, 2], b2 = b[, 3])

# histogram of b0
b0_hist <- ggplot(plot_df_b, aes(x = b0)) + geom_histogram(bins = 30,
    color = "navy", fill = "steelblue2", aes(y = ..density..)) +
    geom_density(colour = "red4", size = 1) + ggtitle("b0 Posterior Distribution") +
    xlab("b0 Values") + ylab("Density") + theme_classic()

# histogram of b1
b1_hist <- ggplot(plot_df_b, aes(x = b1)) + geom_histogram(bins = 30,
    color = "navy", fill = "steelblue2", aes(y = ..density..)) +
```

```
    geom_density(colour = "red4", size = 1) + ggtitle("b1 Posterior Distribution") +
    xlab("b1 Values") + ylab("Density") + theme_classic()

# histogram of b2
b2_hist <- ggplot(plot_df_b, aes(x = b2)) + geom_histogram(bins = 30,
    color = "navy", fill = "steelblue2", aes(y = ..density..)) +
    geom_density(colour = "red4", size = 1) + ggtitle("b2 Posterior Distribution") +
    xlab("b2 Values") + ylab("Density") + theme_classic()

grid.arrange(sigma2_hist, b0_hist, b1_hist, b2_hist, nrow = 2)
```

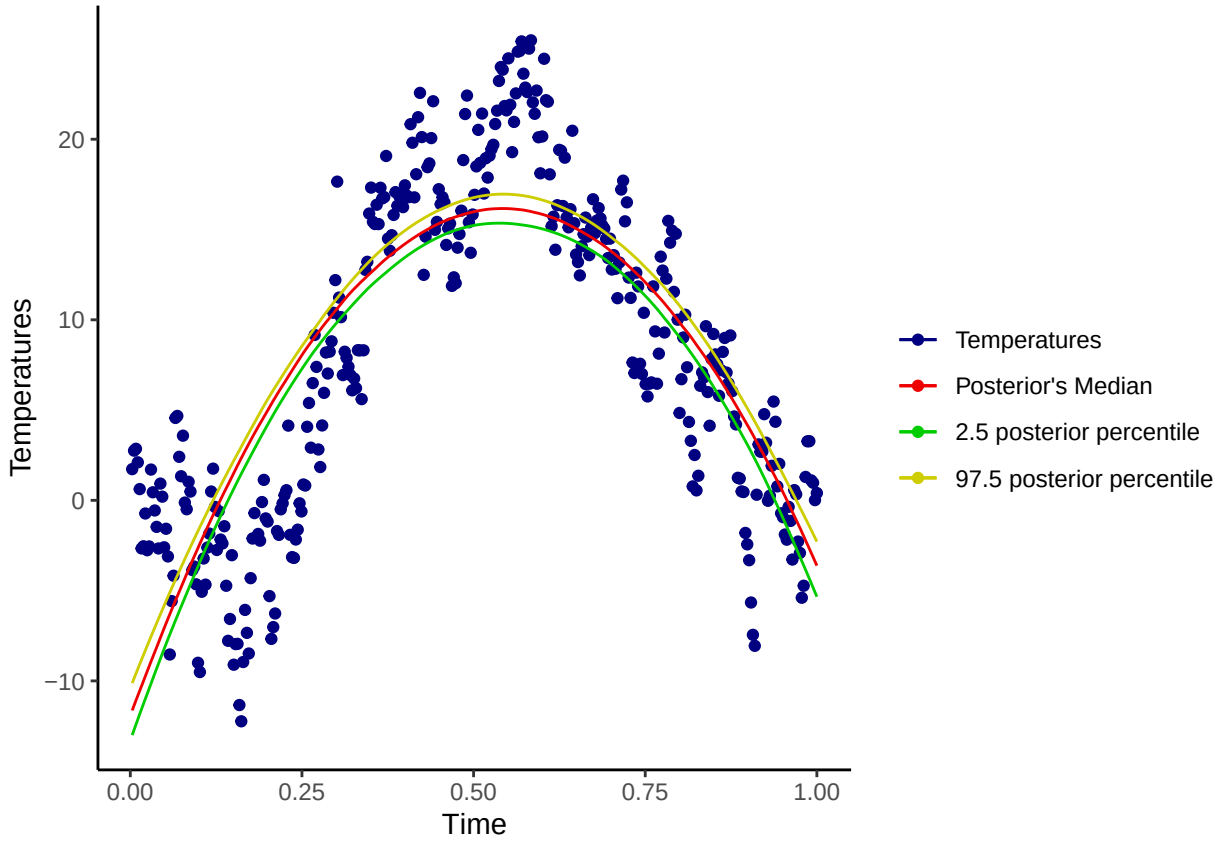


ii)

```
# matrix to store the predicted values
pred_temp2 <- matrix(NA, nrow(b), nrow(temperatures))

# calculate the predicted values
for (i in 1:nrow(b)) {
    pred_temp2[i, ] <- b[i, 1] + b[i, 2] * temperatures$time +
        b[i, 3] * (temperatures$time^2)
}

# calculate posterior's median for every value of time
post_median <- c()
for (i in 1:ncol(pred_temp2)) {
    post_median[i] <- median(pred_temp2[, i])
```

```
}

# calculate the 2.5 and 97.5 posterior percentiles of f
# (time)
intervals <- matrix(NA, nrow = 2, ncol = ncol(pred_temp2))
for (i in 1:ncol(pred_temp2)) {
    intervals[, i] <- quantile(pred_temp2[, i], probs = c(0.025,
        0.975))
}
```

```
# df for plot
plot_df_1b2 <- data.frame(temperatures = temperatures$temp, time = temperatures$time,
    interval1 = intervals[1, ], interval2 = intervals[2, ], medians = post_median)

# scatterplot of the temperatures & curve of the posterior
# median & curves for the 95% equal tail posterior
# probability intervals
ggplot(data = plot_df_1b2) + geom_point(aes(x = time, y = temperatures,
    color = "navy")) + geom_line(aes(x = time, y = medians, color = "red2")) +
    geom_line(aes(x = time, y = interval1, color = "green3")) +
    geom_line(aes(x = time, y = interval2, color = "yellow3")) +
    theme(legend.position = "right") + scale_color_identity(guide = "legend",
    name = "", breaks = c("navy", "red2", "green3", "yellow3"),
    labels = c("Temperatures", "Posterior's Median", "2.5 posterior percentile",
        "97.5 posterior percentile")) + xlab("Time") + ylab("Temperatures") +
    theme_classic()
```



It is evident that 95% equal tail posterior probability intervals do not contain most data points. It should not contain most data points because the interval illustrates the uncertainty around the median value.

Task c

Question:It is of interest to locate the time with the highest expected temperature (i.e. the time where $f(time)$ is maximal). Let's call this value $\tilde{x}$. Use the simulated draws in (b) to simulate from the posterior distribution of $\tilde{x}$. [Hint: the regression curve is a quadratic polynomial. Given each posterior draw of β_0, β_1 and β_2, you can find a simple formula for $\tilde{x}$.]

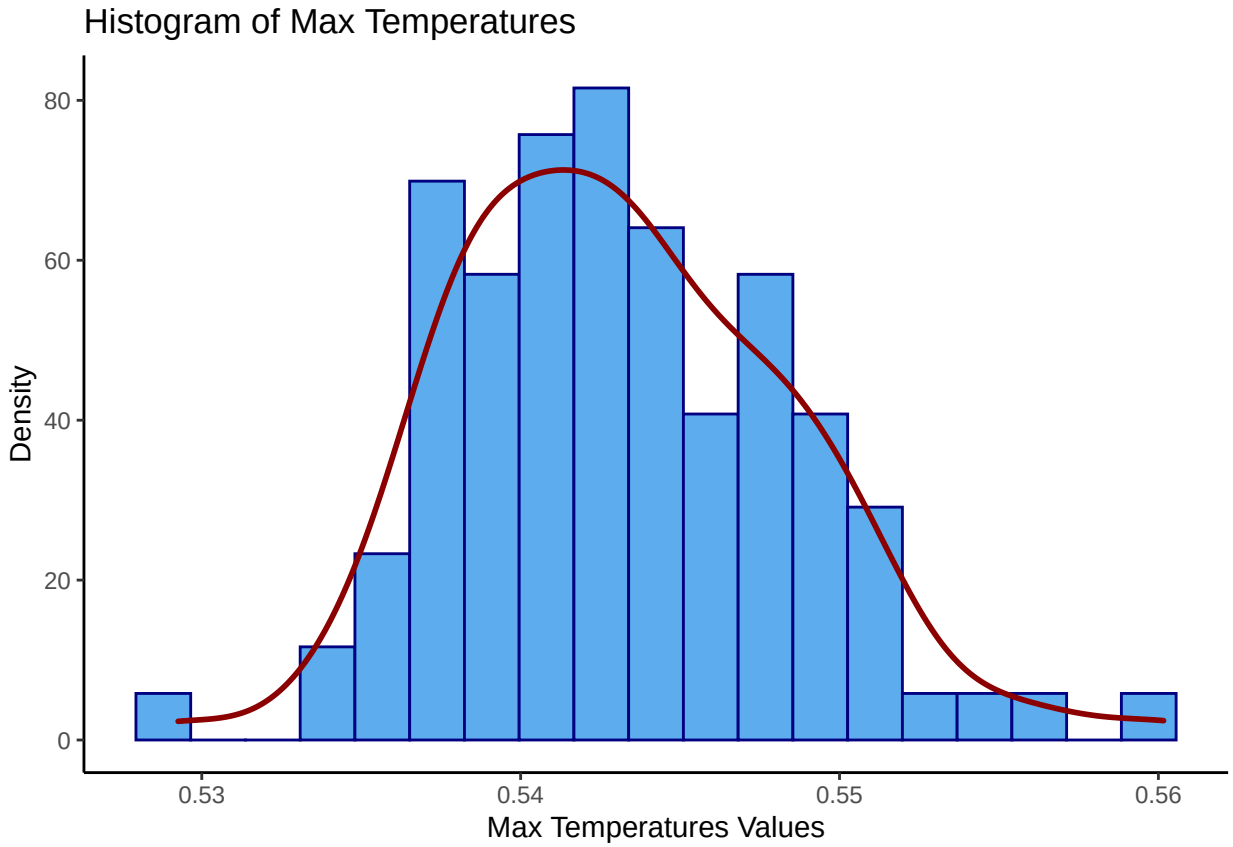
It is given that the regression function is $f(time) = \beta_0 + \beta_1 \cdot time + \beta_2 \cdot time^2$. In order to locate the time with the highest expected temperature; the time, where f(time) is maximal, is needed to be found. Thus, the derivative of $f(time)$ is needed to be found and set it equal to zero to find the maximal.

Set $time = x$; thus, $f(x) = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot x^2$

Calculate the derivative, $f'(x) = \beta_1 + 2 \cdot \beta_2 \cdot x$

Find the maximal: $f'(x) = 0 \Leftrightarrow \beta_1 + 2 \cdot \beta_2 \cdot x = 0 \Leftrightarrow x = \frac{-\beta_1}{2 \cdot \beta_2}$

```
# getting the time with the highest expected temperature by
# using the above expression
max_temp <- c()

for (i in 1:N) {
    max_temp[i] <- max(-b[i, 2]/(2 * b[i, 3]))
}

df_max <- data.frame(max_temp = max_temp)

# histogram of max temps
ggplot(df_max, aes(x = max_temp)) + geom_histogram(bins = 19,
    color = "navy", fill = "steelblue2", aes(y = ..density..)) +
    geom_density(colour = "red4", size = 1) + ggtitle("Histogram of Max Temperatures") +
    xlab("Max Temperatures Values") + ylab("Density") + theme_classic()
```



Task d

Question:Say now that you want to estimate a polynomial regression of order 8, but you suspect that higher order terms may not be needed, and you worry about overfitting the data. Suggest a suitable prior that mitigates this potential problem. You do not need to compute the posterior. Just write down your prior. [Hint: the task is to specify μ_0 and $Omega_0$ in a suitable way.]

Estimating a polynomial regression of order 8 with the suspicion that higher-order terms may not be needed because it might lead to overfitting the data. The main problem that could lead to overfitting is the number of knots. A solution to this problem is to introduce a regularised prior, $\beta_i|\sigma^2 \sim N(\mu_0, \frac{\sigma^2}{\lambda})$, where $Omega_0 = \lambda \cdot I_3$. Where the parameter λ will control the variance of β_i. If λ is larger, β_i would be much closer to μ_0.

Assignment 2 *Posterior approximation for classification with logistic regression*

The dataset WomenAtWork.dat contains n = 168 observations on the following eight variables related to women:

```
# reading data
women <- read.table("WomenAtWork.dat", header = TRUE)

# given parameteres
tau <- 5
```

Task a

Question:Consider the logistic regression model:

$$Pr(y = 1 \mid x) = \frac{\exp(x^T\beta)}{1 + \exp(x^T\beta)}$$

where y equals 1 if the woman works and 0 if she does not. x is a 7-dimensional vector containing the seven features (including a 1 to model the intercept). The goal is to approximate the posterior distribution of the parameter vector β with a multivariate normal distribution $\beta|y, x \sim N\big(\tilde{\beta}, J_y^{-1}(\tilde{\beta})\big)$, y where $\tilde{\beta}$ is the posterior mode and $J(\tilde{\beta}) = -\frac{d^2 lnp(\beta|y)}{d\beta \cdot d\beta^T}$ is the negative of the observed Hessian evaluated at the posterior mode. Note that $\frac{d^2 lnp(\beta|y)}{d\beta \cdot d\beta^T}$ is a 7×7 matrix with second derivatives on the diagonal and cross-derivatives $\frac{d^2 lnp(\beta|y)}{d\beta \cdot d\beta^T}$ on the off-diagonal. You can compute this derivative by hand, but we will let the computer do it numerically for you. Calculate both $\tilde{\beta}$ and $J(\tilde{\beta})$ by using the **optim** function in R. [Hint: You may use code snippets from my demo of logistic regression in Lecture 6.] Use the prior $\beta \sim N(0, \tau^2 I)$, where $tau^2 = 5$. Present the numerical values of $\tilde{\beta}$ and $J^{-1}(\tilde{\beta})$ for the WomenAtWork data. Compute an approximate 95% equal tail posterior probability interval for the regression coefficient to the variable NSmallChild. Would you say that this feature is of importance for the probability that a woman works? [Hint: You can verify that your estimation results are reasonable by comparing the posterior means to the maximum likelihood estimates, given by: glmModel <- glm(Work ~ 0 + ., data = WomenAtWork, family = binomial).]

```
# the below code snippets from a demo of logistic
# regression in Lecture 6

# target value
y <- women$Work

# prior inputs Select which covariates/features to include
X <- as.matrix(women[2:8])
Xnames <- colnames(X)

Npar <- dim(X)[2]

# Setting up the prior
mu <- as.matrix(rep(0, Npar))  # Prior mean vector
Sigma <- tau^2 * diag(Npar)  # Prior covariance matrix

# Functions that returns the log posterior for the logistic
# and probit regression.  First input argument of this
# function must be the parameters we optimize on, i.e. the
# regression coefficients beta.

LogPostLogistic <- function(betas, y, X, mu, Sigma) {
    linPred <- X %*% betas
    logLik <- sum(linPred * y - log(1 + exp(linPred)))
    if (abs(logLik) == Inf)
        logLik = -20000  # Likelihood is not finite, stear the optimizer away from here!
    logPrior <- dmvnorm(betas, mu, Sigma, log = TRUE)

    return(logLik + logPrior)
}

# Select the initial values for beta
initVal <- matrix(0, Npar, 1)

# The argument control is a list of options to the
```

```
# optimizer optim, where fnscale=-1 means that we minimize
# the negative log posterior. Hence, we maximize the log
# posterior.
OptimRes <- optim(initVal, LogPostLogistic, gr = NULL, y, X,
    mu, Sigma, method = c("BFGS"), control = list(fnscale = -1),
    hessian = TRUE)
```

```
names(OptimRes$par) <- Xnames  # Naming the coefficient by covariates
approxPostStd <- sqrt(diag(-solve(OptimRes$hessian)))  # Computing approximate standard deviations.
names(approxPostStd) <- Xnames  # Naming the coefficient by covariates
# print('The posterior mode is:') print(OptimRes$par)
# print('The approximate posterior standard deviation is:')
approxPostStd <- sqrt(diag(-solve(OptimRes$hessian)))
# print(approxPostStd)
```

The below table illustrates the $J_y^{-1}(\widetilde{\beta})$.

```
invHessian <- data.frame(invhes = -solve(OptimRes$hessian))
colnames(invHessian) <- c("Constant", "HusbandInc", "EducYears",
    "ExpYears", "Age", "NSmallChild", "NBigChild")
knitr::kable(invHessian)
```

Constant	HusbandInc	EducYears	ExpYears	Age	NSmallChild	NBigChild
2.5292633	0.0039371	-0.0775073	0.0008893	-0.0341000	-0.2281033	-0.1137008
0.0039371	0.0003915	-0.0008069	-0.0000011	-0.0000516	0.0011500	-0.0000640
-0.0775073	-0.0008069	0.0073415	0.0000610	0.0000496	-0.0080245	0.0019256
0.0008893	-0.0000011	0.0000610	0.0009006	-0.0002492	-0.0009930	0.0006122
-0.0341000	-0.0000516	0.0000496	-0.0002492	0.0008072	0.0060957	0.0012787
-0.2281033	0.0011500	-0.0080245	-0.0009930	0.0060957	0.1918381	0.0094209
-0.1137008	-0.0000640	0.0019256	0.0006122	0.0012787	0.0094209	0.0222163

The below table illustrates the posterior mode values of every feature of the dataset.

```
df_post_mode <- data.frame(post_mode = OptimRes$par)
colnames(df_post_mode) <- c("Value")
rownames(df_post_mode) <- c("Constant", "HusbandInc", "EducYears",
    "ExpYears", "Age", "NSmallChild", "NBigChild")
knitr::kable(df_post_mode)
```

	Value
Constant	0.9929529
HusbandInc	-0.0343502
EducYears	0.1794176
ExpYears	0.1230628
Age	-0.0727923
NSmallChild	-1.6227735
NBigChild	-0.0838883

From the above table, it could be seen that the posterior mode of the feature *NSmallChild* has the lowest value; thus, it could be assumed that this variable plays a significant role if a woman is employed.

The below table illustrates the approximate posterior standard deviation values of every feature of the dataset.

```r
df_approxPostStd <- data.frame(approxPostStd = sqrt(diag(-solve(OptimRes$hessian))))
colnames(df_approxPostStd) <- c("Value")
rownames(df_approxPostStd) <- c("Constant", "HusbandInc", "EducYears",
    "ExpYears", "Age", "NSmallChild", "NBigChild")
knitr::kable(df_approxPostStd)
```

	Value
Constant	1.5903658
HusbandInc	0.0197857
EducYears	0.0856826
ExpYears	0.0300097
Age	0.0284107
NSmallChild	0.4379933
NBigChild	0.1490514

The approximate 95% equal tail posterior probability interval for the regression coefficient to the variable *NSmallChild* has a lower and an upper bound less than 0; as a result, it strengthens the assumption mentioned above that the *NSmallChild* feature plays a significant role if a woman is employed.

```r
# draw b
b_draws <- rmvnorm(n = 1000, mean = OptimRes$par, sigma = -solve(OptimRes$hessian))

# calculate quantiles
interval <- quantile(b_draws[, 6], c(0.025, 0.975))
df_interval <- data.frame(lower_bound = interval[1], upper_bound = interval[2])
colnames(df_interval) <- c("lower bound", "upper bound")
rownames(df_interval) <- c("95% Equal Tail Credible Interval")
knitr::kable(df_interval)
```

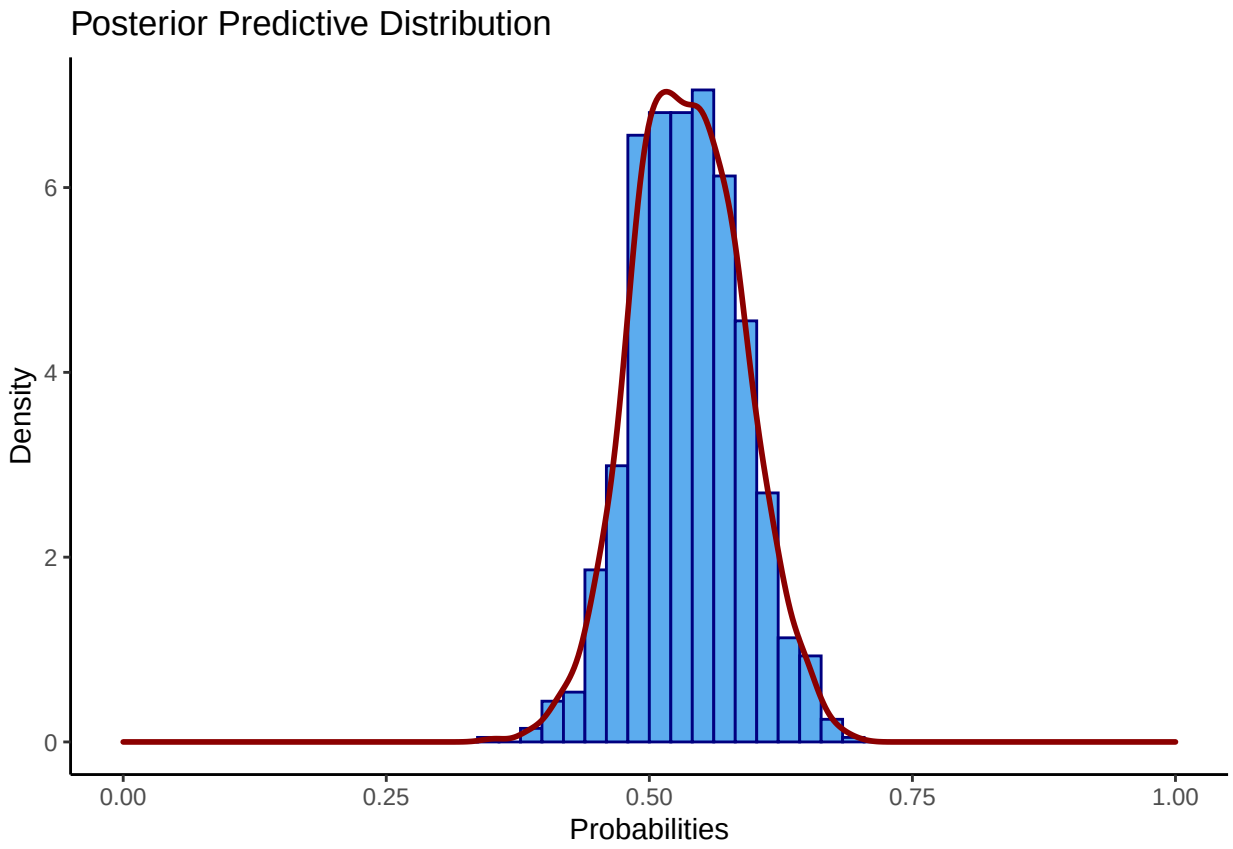
	lower bound	upper bound
95% Equal Tail Credible Interval	-2.445008	-0.8075428

Task b

Question:Use your normal approximation to the posterior from (a). Write a function that simulate draws from the posterior predictive distribution of $Pr(y = 1|x)$, where the values of x corresponds to a 43-year-old woman, with two children (7 and 10 years old), 12 years of education, 8 years of experience, and a husband with an income of 20. Plot the posterior predictive distribution of $Pr(y = 1|x)$ for this woman. [Hints: The R package mvtnorm will be useful. Remember that $Pr(y = 1|x)$ can be calculated for each posterior draw of β.]

```r
set.seed(1234567)
women2 <- c(1, 20, 12, 8, 43, 0, 2)

# function that classifies if the women work or not
prediction <- function(N, data, posterior_mode, posterior_covariates) {

    # generate betas
    b <- rmvnorm(N, mean = posterior_mode, sigma = posterior_covariates)

    # calculating the the logistic regression model
    res <- exp(data %*% t(b))/(1 + exp(data %*% t(b)))
```

```
    return(res)
}

# predictions
pred <- prediction(1000, women2, OptimRes$par, -solve(OptimRes$hessian))

# df for plot
pred_women <- data.frame(pred = t(pred))

ggplot(pred_women, aes(x = pred)) + geom_histogram(bins = 50,
    color = "navy", fill = "steelblue2", aes(y = ..density..)) +
    geom_density(colour = "red4", size = 1) + scale_x_continuous(limits = c(0,
    1)) + ggtitle("Posterior Predictive Distribution") + xlab("Probabilities") +
    ylab("Density") + theme_classic()
```

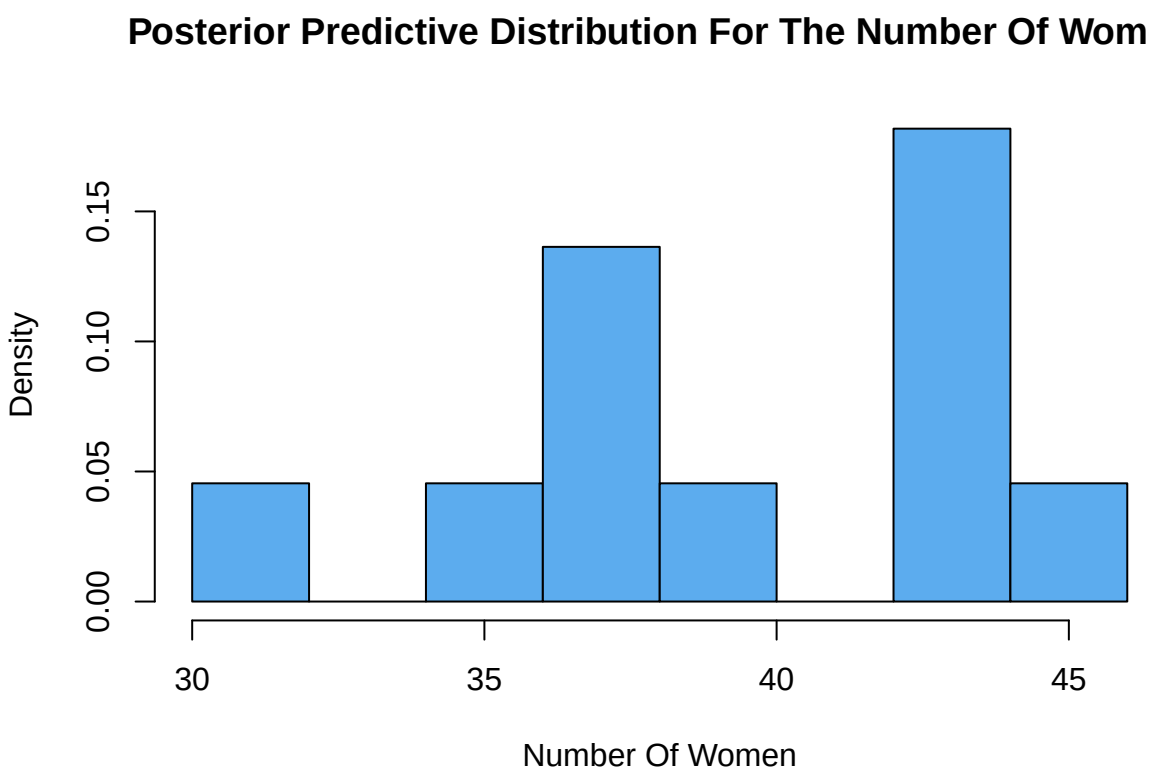


From the above plot, it could be assumed that a 43-year-old woman with two children (7 and 10 years old), 12 years of education, 8 years of experience, and a husband with an income of 20.000 SEK; is more likely to be employed, but there is still a considerable number of occasions that she might without work.

Task 2c

Question:Now, consider 11 women which all have the same features as the woman in (b). Rewrite your function and plot the posterior predictive distribution for the number of women, out of these 11, that are working. [Hint: Simulate from the binomial distribution, which is the distribution for a sum of Bernoulli random variables.]

```r
set.seed(1234567)

prediction2 <- function(N, data, posterior_mode, posterior_covariates,
    nwomen) {

    # generate betas
    b <- rmvnorm(N, mean = posterior_mode, sigma = posterior_covariates)

    # calculating the the logistic regression model
    pred <- exp(data %*% t(b))/(1 + exp(data %*% t(b)))

    # vector to store results
    res <- c()

    for (i in 1:nwomen) {
        # binomial distribution
        res[i] <- sum(rbinom(n = 7, size = 11, prob = pred))
    }

    return(res)
}

# predictions
pred2 <- prediction2(1000, women2, OptimRes$par, -solve(OptimRes$hessian),
    11)

hist(pred2, col = "steelblue2", main = "Posterior Predictive Distribution For The Number Of Women",
    xlab = "Number Of Women", ylab = "Density", probability = TRUE)
```



Lab 3

Assignment 1 *Gibbs sampler for a normal model*

The dataset Precipitation.rds consists of daily records of weather with rain or snow (in units of mm) from the beginning of 1962 to the end of 2008 in a certain area. Assume the natural log of the daily precipitation $y1, ..., yn$ to be independent normally distributed, $lny_n|\mu, \sigma^2 \overset{\text{iid}}{\sim} N(\mu, \sigma^2)$, where both μ and σ^2 are unknown. Let $\mu \sim N(\mu_0, \tau_0)$ independently of $sigma^2 \sim Inv - \chi^2(\nu_0, \sigma_0^2)$.

```
# Reading & preparing data
precipitation <- as.data.frame(readRDS("Precipitation.rds"))
colnames(precipitation) <- "records"
log_records <- log(precipitation$records)
```

Task a

Question: Implement (code!) a Gibbs sampler that simulates from the joint posterior $p(\mu, \sigma^2|lny_1, ..., lny_n)$. The full conditional posteriors are given on the slides from Lecture 7. Evaluate the convergence of the Gibbs sampler by calculating the Inefficiency Factors (IFs) and by plotting the trajectories of the sampled Markov chains.

The full conditional posteriors are:

$$\mu|\sigma^2, x \sim N(\mu_n, \tau_n^2)$$
$$\sigma^2|\mu, x \sim Inv - \chi^2\left(\nu_n, \frac{\nu_0\sigma_0^2 + \sum_{i=1}^{n}(x_i - \mu)^2}{n + \nu_0}\right)$$

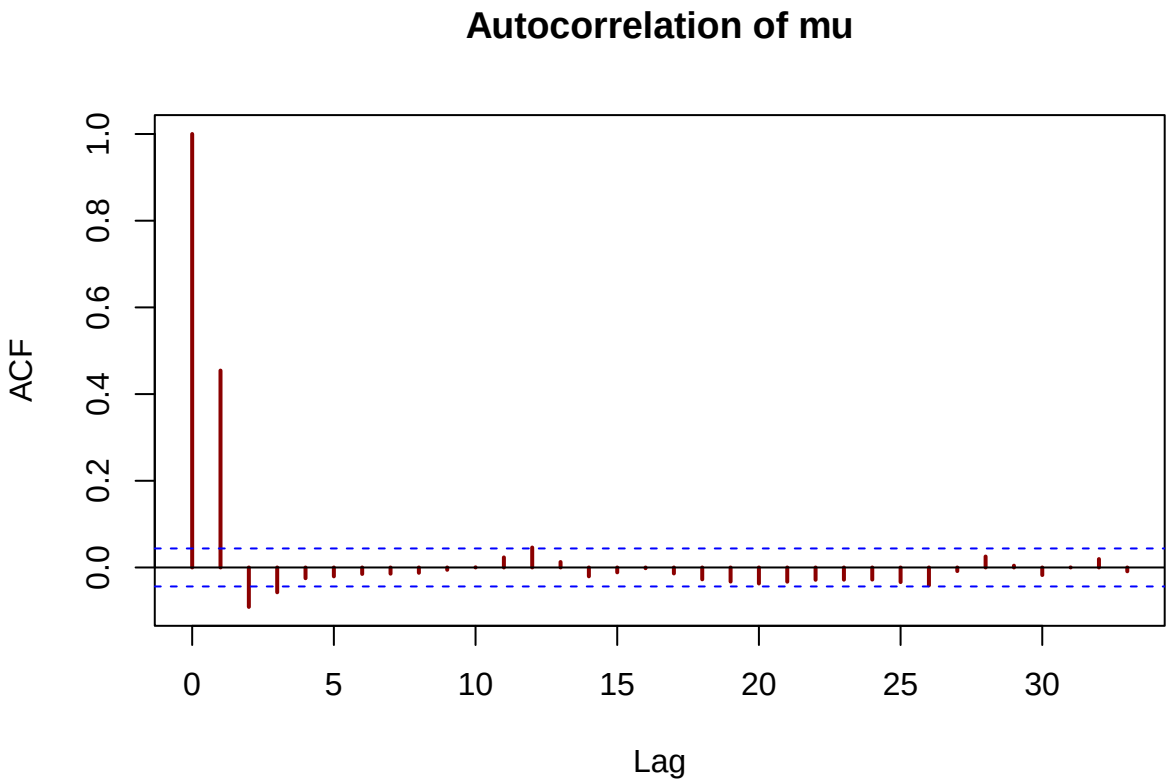
Where:

$$w = \frac{\frac{n}{\sigma^2}}{\frac{n}{\sigma^2} + \frac{1}{\tau_0^2}}$$
$$\mu_n = w\overline{x} + (1 - w)\mu_0$$
$$\tau_n^2 = \frac{1}{\frac{n}{\sigma^2} + \frac{1}{\tau_0^2}}$$
$$\nu_n = \nu_0 + n$$

```
set.seed(1234567890)

# setting up prior parameters
n <- length(log_records)
n_0 <- 1
mu_0 <- mean(log_records)
sigma2_0 <- var(log_records)
tau2_0 <- 1

# normal model with conditionally conjugate prior mu_prop
# <- rnorm(n = 1, mean = mu_0, sqrt(tau2_0)) sigma2_prop <-
# n_0*sigma2_0 / rchisq(1,n_0)

# sigma2 starting value
sigma2 <- sigma2_0

# vectors to store the samples
```

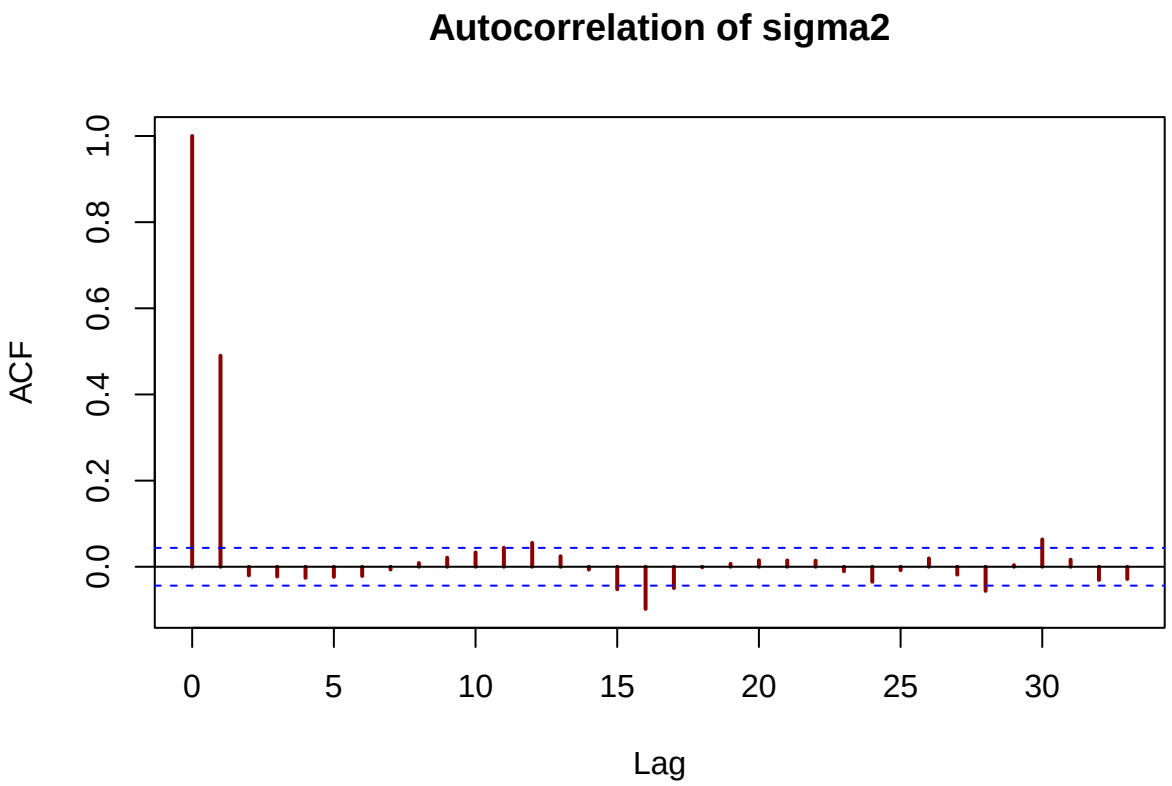
```r
mu_gibbs <- c()
sigma2_gibbs <- c(sigma2)

N <- 1000
for (i in 1:N) {

    # calculating parameters
    w <- (n/sigma2)/((n/sigma2) + (1/tau2_0))
    mu_n <- w * mean(log_records) + (1 - w) * mu_0
    tau2_n <- 1/(n/sigma2 + 1/tau2_0)

    # calculating mu
    mu <- rnorm(n = 1, mean = mu_n, sd = tau2_n)

    # temporaly store the previous sigma2
    previous_sigma2 <- sigma2

    # calculating parameter
    n_n <- n_0 + n

    # calculating sigma2
    sigma2 <- (n_n * ((n_0 * sigma2_0 + sum((log_records - mu)^2))/n_n))/rchisq(1,
        n_n)

    # conditions of how to store the samples
    if (i == 1) {
        mu_gibbs <- c(mu)
        sigma2_gibbs <- c(sigma2)
    } else {
        mu_gibbs <- append(mu_gibbs, c(mu, mu))
        sigma2_gibbs <- append(sigma2_gibbs, c(previous_sigma2,
            sigma2))
    }
}
```

```r
# calculating autocorrelation for mu
mu_r <- acf(mu_gibbs, main = "Autocorrelation of mu", col = "red4",
    lwd = 2)
```




```
# callculating the inefficiency factor of mu
IF_mu <- 1 + 2 * sum(mu_r$acf[-1])
```

```
# calculating autocorrelation for sigma2
sigma2_r <- acf(sigma2_gibbs, main = "Autocorrelation of sigma2",
    col = "red4", lwd = 2)
```




```
# callculating the inefficiency factor of sigma2
IF_sigma2 <- 1 + 2 * sum(sigma2_r$acf[-1])
```

The above graphs illustrate the correlation coefficient of μ and σ^2 against the lag. The red bars illustrate the correlation coefficient of the parameters of each lag, and the blue lines represent the statistically significant boundaries. In both graphs, the ACF equals 1 for a lag 0, which is reasonable because the "first step" is always perfectly correlated with itself (the lag 0 autocorrelation is fixed at one by convention). More specifically, in the first graph, the correlation coefficient of μ is inside the boundaries when lag is greater than 3, which means the values are unrelated between them. Likewise, in the second graph, the correlation coefficient of σ^2 gets closer to 0 after lag is 2, which means the values are unrelated between them. Finally, however, the ACF value crosses the "boundaries" in some cases, which means that the variables are related when lag equals 12, 16 and 30.

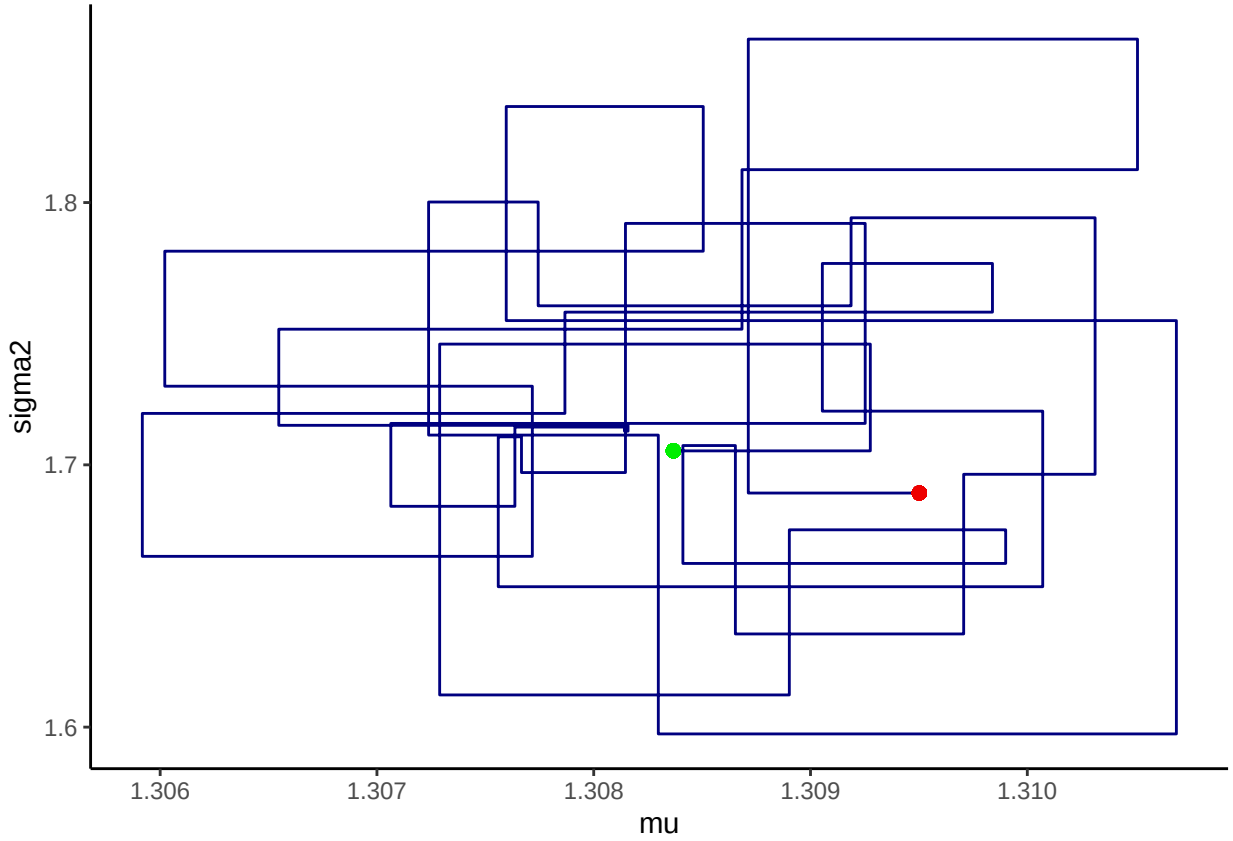
The below table illustrates the inefficiency factors of μ and σ^2.

```
# df with the inefficiency factors of mu & sigma
IFs_df <- data.frame(mu = IF_mu, sigma2 = IF_sigma2)
rownames(IFs_df) <- c("Inefficiency Factors")
knitr::kable(IFs_df)
```

	mu	sigma2
Inefficiency Factors	0.9468888	1.635177

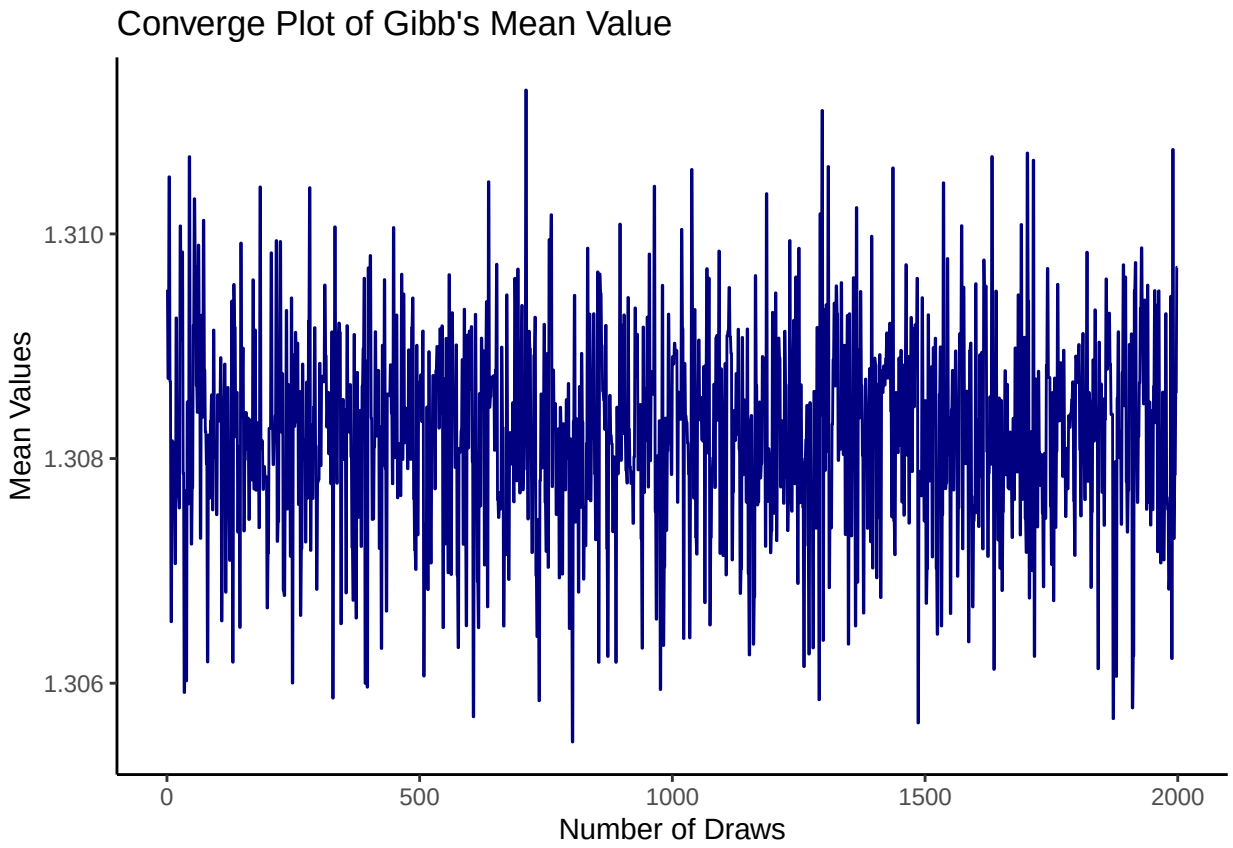
The inefficiency factor equals 1 if there is not any autocorrelation. The inefficiency factor of μ is close to 1, which means this is an efficient sample. However, the inefficiency factor of σ^2 equals approximately 1.63, which means that the sample is not efficient enough; this is because the ACF values of σ^2 cross the statistically significant boundaries when lag equals 12, 16 and 30.

```
# df for plot
plot_df_traj <- data.frame(mu = mu_gibbs, sigma2 = sigma2_gibbs)

# plotting the first 70 samples plotting all the samples
# would be a mess
plot_df_traj <- plot_df_traj[1:70, ]

# trajectories of the sampled Markov chains.
ggplot(plot_df_traj) + geom_path(aes(x = mu, y = sigma2), color = "navy") +
    geom_point(aes(x = mu[1], y = sigma2[1]), color = "red2",
        size = 2) + geom_point(aes(x = mu[70], y = sigma2[70]),
    color = "green2", size = 2) + ylab("sigma2") + xlab("mu") +
    theme_classic()
```

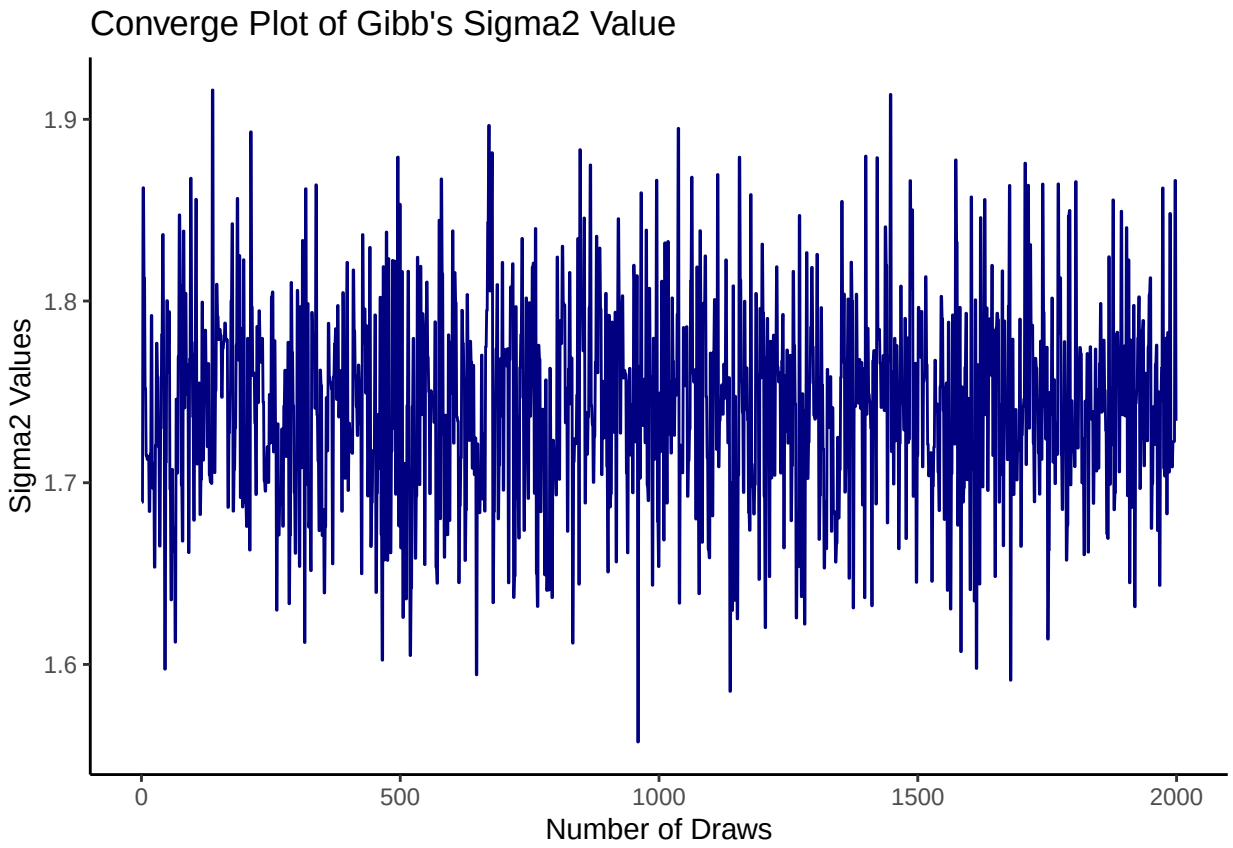


The above plot illustrates the trajectories of the first 70 point of the sample, the red dot is the starting point and the green dot is the ending point.

```
# data for converge plots
df_plot_conv <- data.frame(draws = 1:1999, mean_gibbs = mu_gibbs,
    sigma2_gibbs = sigma2_gibbs)

# converge plot of Gibbs's mu
ggplot(df_plot_conv) + geom_line(aes(x = draws, y = mean_gibbs),
    color = "navy") + theme(legend.position = "right") + ggtitle("Converge Plot of Gibb's Mean Value") +
    xlab("Number of Draws") + ylab("Mean Values") + theme_classic()
```



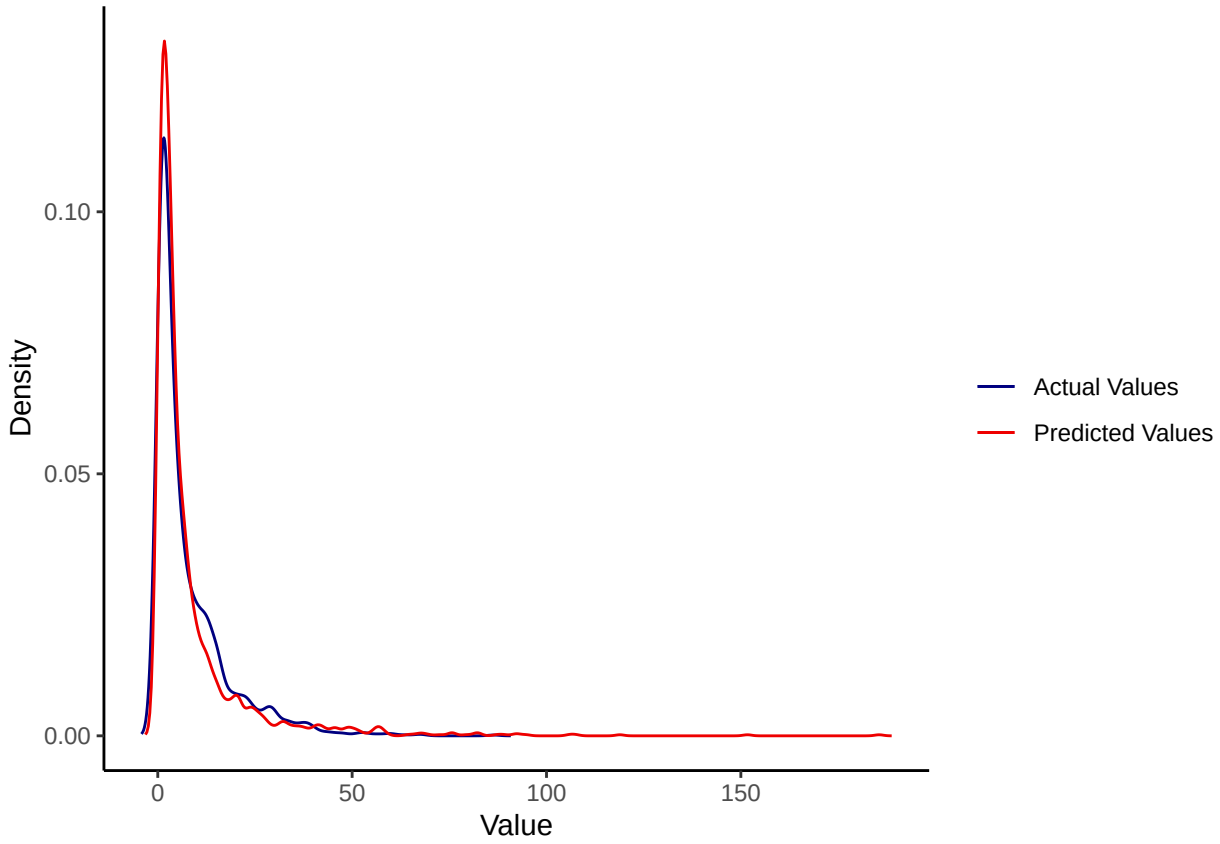

```
# converge plot of Gibbs's sigma2
ggplot(df_plot_conv) + geom_line(aes(x = draws, y = sigma2_gibbs),
    color = "navy") + theme(legend.position = "right") + ggtitle("Converge Plot of Gibb's Sigma2 Value")
    xlab("Number of Draws") + ylab("Sigma2 Values") + theme_classic()
```



Task b

Question:Plot the following in one figure: 1) a histogram or kernel density estimate of the daily precipitation $y1, ..., yn$. 2) The resulting posterior predictive density $p(\tilde{y}|y_1, ..., y_n)$ using the simulated posterior draws from (a). How well does the posterior predictive density agree with this data?

```
set.seed(1234567890)

# predicted draws
predictions <- rnorm(nrow(precipitation), mu_gibbs, sqrt(sigma2_gibbs))

# kernel density estimation
density_actual <- density(precipitation$records)
density_pred <- density(exp(predictions))

# df for plot
plot_df_dens <- data.frame(actual_x = density_actual$x, actual_y = density_actual$y,
    pred_x = density_pred$x, pred_y = density_pred$y)

ggplot(plot_df_dens) + geom_line(aes(x = actual_x, y = actual_y,
    color = "navy")) + geom_line(aes(x = pred_x, y = pred_y,
    color = "red2")) + theme(legend.position = "right") + scale_color_manual(values = c("navy",
    "red2"), name = "", labels = c("Actual Values", "Predicted Values")) +
    xlab("Value") + ylab("Density") + theme_classic()
```



The posterior predictive density has almost the same pattern as the density of the actual values; thus the posterior's predictive values agree very well with the data.

Assignment 2 *Metropolis Random Walk for Poisson regression*

Consider the following Poisson regression model

$$y_i \mid \beta \overset{\text{iid}}{\sim} Poisson\left[\, exp(\mathbf{x}_i^T \beta)\right],\ i = 1, ..., n,$$

where y_i is the count for the ith observation in the sample and xi is the p-dimensional vector with covariate observations for the ith observation. Use the data set eBayNumberOfBidderData.dat. This dataset contains observations from 1000 eBay auctions of coins. The response variable is nBids and records the number of bids in each auction. The remaining variables are features/covariates (x):
Const (for the intercept) PowerSeller (equal to 1 if the seller is selling large volumes on eBay)
VerifyID (equal to 1 if the seller is a verified seller by eBay)
Sealed (equal to 1 if the coin was sold in an unopened envelope)
MinBlem (equal to 1 if the coin has a minor defect)
MajBlem (equal to 1 if the coin has a major defect)
LargNeg (equal to 1 if the seller received a lot of negative feedback from customers)
LogBook (logarithm of the book value of the auctioned coin according to expert sellers. Standardized).
MinBidShare (ratio of the minimum selling price (starting price) to the book value. Standardized).

```
# Reading data
ebay <- read.table("eBayNumberOfBidderData.dat", header = TRUE)
```

Task a

Question:Obtain the maximum likelihood estimator of β in the Poisson regression model for the eBay data [Hint: glm.R, don't forget that glm() adds its own intercept so don't input the covariate Const]. Which covariates are significant?

Below the summary of the glm is represented.

```
# glm function, + 0 in order to do not input the covariate
# Const
model <- glm(formula = nBids ~ . - Const, data = ebay, family = poisson)

# print the summary of the model
summary(model)
```

```
##
## Call:
## glm(formula = nBids ~ . - Const, family = poisson, data = ebay)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5800  -0.7222  -0.0441   0.5269   2.4605
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.07244    0.03077  34.848  < 2e-16 ***
## PowerSeller -0.02054    0.03678  -0.558   0.5765
## VerifyID    -0.39452    0.09243  -4.268 1.97e-05 ***
## Sealed       0.44384    0.05056   8.778  < 2e-16 ***
## Minblem     -0.05220    0.06020  -0.867   0.3859
## MajBlem     -0.22087    0.09144  -2.416   0.0157 *
## LargNeg      0.07067    0.05633   1.255   0.2096
## LogBook     -0.12068    0.02896  -4.166 3.09e-05 ***
## MinBidShare -1.89410    0.07124 -26.588  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 2151.28  on 999  degrees of freedom
## Residual deviance:  867.47  on 991  degrees of freedom
## AIC: 3610.3
##
## Number of Fisher Scoring iterations: 5
```

Task b

Question:Let's do a Bayesian analysis of the Poisson regression. Let the prior be $\beta \sim N[0, 100\mathring{u}(X^T \cdot X)^{-1}]$, where X is the n $\times$ p covariate matrix. This is a commonly used prior, which is called Zellner's g-prior. Assume first that the posterior density is approximately multivariate normal:

$$\beta|y \sim N(\tilde{\beta}, J_y^{-1}(\tilde{\beta})$$

, where $\tilde{\beta}$ is the posterior mode and $J_y(\tilde{\beta})$ is the negative Hessian at the posterior mode. $\tilde{\beta}$ and $J_y(\tilde{\beta})$ can be obtained by numerical optimization (optim.R) exactly like you already did for the logistic regression in Lab 2 (but with the log posterior function replaced by the corresponding one for the Poisson model, which you have to code up.).

The likelihood function of the Poisson regression model is:

$$L(\beta|X,Y) = \prod_{i}^{n} \frac{e^{Y_i \beta^T X_i} e^{-\beta^T X_i}}{Y_i!}$$

The log-likelihood of the Poisson regression model is:

$$l(\beta|X,Y) = log\Big(L(\beta|X,Y)\Big) = \sum_{i}^{n} \Big(Y_i \beta^T X_i - e^{\beta^T Xi} - log(Y_i)\Big)$$

In the above formula, θ appears in the first two terms; therefore, given that we are only interested in the finding of the best value of θ, we drop the $log(Y_i)$.

The final log-likelihood of the Poisson regression model is:

$$l(\beta|X,Y) = \sum_{i}^{n} \Big(Y_i \beta^T X_i - e^{\beta^T Xi}\Big)$$

```
# the below code snippets from a demo of logistic
# regression in Lecture 6

# target value
y <- ebay$nBids

# prior inputs Select which covariates/features to include
X <- as.matrix(ebay[2:10])
Xnames <- colnames(X)

Npar <- dim(X)[2]

# Setting up the prior
mu <- as.matrix(rep(0, Npar))  # Prior mean vector
Sigma <- 100 * solve(t(X) %*% X)  # Prior covariance matrix

# Functions that returns the log posterior for the poisson
# and probit regression.  First input argument of this
# function must be the parameters we optimize on, i.e. the
# regression coefficients beta.

LogPostPoisson <- function(betas, y, X, mu, Sigma) {
    linPred <- X %*% betas
    # loglikelihood of Poisson regression
    logLik <- sum(linPred * y - exp(linPred))
    if (abs(logLik) == Inf)
        logLik = -20000  # Likelihood is not finite, stear the optimizer away from here!
    logPrior <- dmvnorm(betas, mu, Sigma, log = TRUE)

    return(logLik + logPrior)
}

# Select the initial values for beta
initVal <- matrix(0, Npar, 1)
```

```
# The argument control is a list of options to the
# optimizer optim, where fnscale=-1 means that we minimize
# the negative log posterior. Hence, we maximize the log
# posterior.
OptimRes <- optim(initVal, LogPostPoisson, gr = NULL, y, X, mu,
    Sigma, method = c("BFGS"), control = list(fnscale = -1),
    hessian = TRUE)
```

```
names(OptimRes$par) <- Xnames  # Naming the coefficient by covariates
approxPostStd <- sqrt(diag(-solve(OptimRes$hessian)))  # Computing approximate standard deviations.
names(approxPostStd) <- Xnames  # Naming the coefficient by covariates
# print('The posterior mode is:') print(OptimRes$par)
# print('The approximate posterior standard deviation is:')
approxPostStd <- sqrt(diag(-solve(OptimRes$hessian)))
# print(approxPostStd)
```

The below table illustrates the $J_y^{-1}(\widetilde{\beta})$.

```
invHessian <- round(-solve(OptimRes$hessian), 5)
invHessian <- data.frame(invhes = invHessian)
colnames(invHessian) <- c("Constant", "PowerSeller", "VerifyID",
    "Sealed", "MinBlem", "MajBlem", "LargNeg", "LogBook", "MinBidShare")
knitr::kable(invHessian)
```

Constant	PowerSeller	VerifyID	Sealed	MinBlem	MajBlem	LargNeg	LogBook	MinBidShare
0.00095	-0.00071	-0.00027	-0.00027	-0.00045	-0.00028	-0.00051	0.00006	0.00111
-0.00071	0.00135	0.00004	-0.00029	0.00011	-0.00021	0.00028	0.00012	-0.00057
-0.00027	0.00004	0.00852	-0.00078	-0.00010	0.00023	0.00033	-0.00032	-0.00043
-0.00027	-0.00029	-0.00078	0.00256	0.00036	0.00045	0.00034	-0.00013	-0.00006
-0.00045	0.00011	-0.00010	0.00036	0.00362	0.00035	0.00006	0.00006	-0.00006
-0.00028	-0.00021	0.00023	0.00045	0.00035	0.00837	0.00040	-0.00009	0.00026
-0.00051	0.00028	0.00033	0.00034	0.00006	0.00040	0.00318	-0.00025	-0.00011
0.00006	0.00012	-0.00032	-0.00013	0.00006	-0.00009	-0.00025	0.00084	0.00104
0.00111	-0.00057	-0.00043	-0.00006	-0.00006	0.00026	-0.00011	0.00104	0.00505

The below table illustrates the posterior mode values of the features of the dataset, which are almost similar to the actual mode of the true posterior (see the summary of the glm model).

```
df_post_mode <- data.frame(post_mode = OptimRes$par)
colnames(df_post_mode) <- c("Value")
rownames(df_post_mode) <- c("Constant", "PowerSeller", "VerifyID",
    "Sealed", "MinBlem", "MajBlem", "LargNeg", "LogBook", "MinBidShare")
knitr::kable(df_post_mode)
```

	Value
Constant	1.0698412
PowerSeller	-0.0205125
VerifyID	-0.3930060
Sealed	0.4435555
MinBlem	-0.0524663
MajBlem	-0.2212384
LargNeg	0.0706968
LogBook	-0.1202177

	Value
MinBidShare	-1.8919850

The below table illustrates the approximate posterior standard deviation values of every feature of the dataset.

```
df_approxPostStd <- data.frame(approxPostStd = sqrt(diag(-solve(OptimRes$hessian))))
colnames(df_approxPostStd) <- c("Value")
rownames(df_approxPostStd) <- c("Constant", "PowerSeller", "VerifyID",
    "Sealed", "MinBlem", "MajBlem", "LargNeg", "LogBook", "MinBidShare")
knitr::kable(df_approxPostStd)
```

	Value
Constant	0.0307484
PowerSeller	0.0367842
VerifyID	0.0922787
Sealed	0.0505745
MinBlem	0.0602047
MajBlem	0.0914607
LargNeg	0.0563477
LogBook	0.0289564
MinBidShare	0.0710968

Task c

Questio: Let's simulate from the actual posterior of β using the Metropolis algorithm and compare the results with the approximate results in b). Program a general function that uses the Metropolis algorithm to generate random draws from an arbitrary posterior density. In order to show that it is a general function for any model, we denote the vector of model parameters by θ. Let the proposal density be the multivariate normal density mentioned in Lecture 8 (random walk Metropolis):

$$\theta_p|theta_{(i-1)} \sim N(\theta_{(i-1)}, c \cdot \Sigma$$

, where $\Sigma = J_y^{-1}(\tilde{\beta})$ was obtained in b). The value c is a tuning parameter and should be an input to your Metropolis function. The user of your Metropolis function should be able to supply her own posterior density function, not necessarily for the Poisson regression, and still be able to use your Metropolis function. This is not so straightforward, unless you have come across function objects in R. The note HowToCodeRWM.pdf in Lisam describes how you can do this in R. Now, use your new Metropolis function to sample from the posterior of β in the Poisson regression for the eBay dataset. Assess MCMC convergence by graphical methods.

For the Random walk Metropolis algorithm:

1) Initialize β^0 and iterate for i = 1, 2, ...

2) Sample proposal: $\beta_p|\beta^{(i-1)} \sim N(\beta^{(i-1)}, c \cdot \Sigma)$, where $\Sigma = J^{-1}_{\hat{\beta},y}$

3) Compute the acceptance probability: $\alpha = min\left(1, \frac{p(\beta p|y)}{p(\beta^{(i-1)}|y)}\right)$, where:

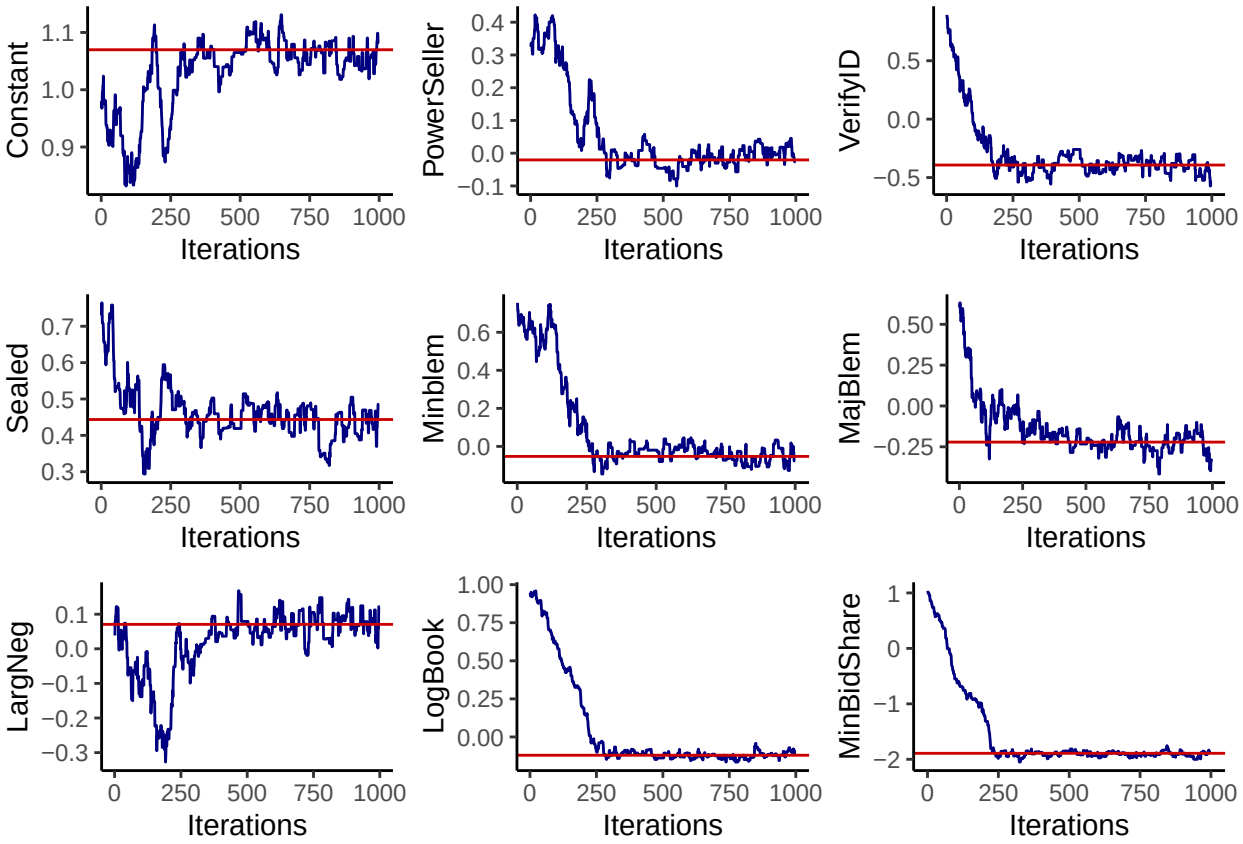
$$\frac{p(\beta_p|y)}{p(\beta^{(i-1)}|y)} = exp\Big[log(p(\beta_p|y)) - log(p(\beta^{(i-1)}|y))$$

4) With probability α set $\beta^{(i)} = \beta_p$ and $\beta^{(i)} = \beta^{(i-1)}$

```r
set.seed(1234567890)

RWMSampler <- function(N, logPostFunc, init_beta, constant, cov_mat,
    target_val, features, mu) {

    init_sample = rmvnorm(1, init_beta, cov_mat)

    # matrix to store the betas
    betas <- matrix(nrow = N, ncol = length(init_sample))

    betas[1, ] <- init_sample

    for (i in 2:N) {

        # sample proposal
        sample_proposal <- as.vector(rmvnorm(1, betas[i - 1,
            ], constant * cov_mat))

        # LogPostPoisson <- function(betas,y,X,mu,Sigma)
        # calculating the new posterior
        post_new <- logPostFunc(sample_proposal, target_val,
            features, mu, cov_mat)

        # calculating the old posterior
        post_old <- logPostFunc(betas[i - 1, ], target_val, features,
            mu, cov_mat)

        # acceptance probability
        a <- min(1, exp(post_new - post_old))

        u <- runif(1, 0, 1)

        if (u < a) {
            betas[i, ] <- sample_proposal
        } else {
            betas[i, ] <- betas[i - 1, ]
        }
    }
    return(betas)
}


RWMbetas <- RWMSampler(1000, LogPostPoisson, runif(9, 0, 1),
    0.35, -solve(OptimRes$hessian), ebay$nBids, as.matrix(ebay[2:10]),
    OptimRes$par)
```

```r
plot_df_RWM <- data.frame(RWMbetas)
colnames(plot_df_RWM) <- c("Constant", "PowerSeller", "VerifyID",
    "Sealed", "Minblem", "MajBlem", "LargNeg", "LogBook", "MinBidShare")

plot_df_RWM$Iterations <- 1:1000

p1 <- ggplot(plot_df_RWM) + geom_line(aes(x = Iterations, y = Constant),
    color = "navy") + geom_hline(yintercept = OptimRes$par[1,
```

```
    ], color = "red3") + theme_classic()

p2 <- ggplot(plot_df_RWM) + geom_line(aes(x = Iterations, y = PowerSeller),
    color = "navy") + geom_hline(yintercept = OptimRes$par[2,
    ], color = "red3") + theme_classic()

p3 <- ggplot(plot_df_RWM) + geom_line(aes(x = Iterations, y = VerifyID),
    color = "navy") + geom_hline(yintercept = OptimRes$par[3,
    ], color = "red3") + theme_classic()

p4 <- ggplot(plot_df_RWM) + geom_line(aes(x = Iterations, y = Sealed),
    color = "navy") + geom_hline(yintercept = OptimRes$par[4,
    ], color = "red3") + theme_classic()

p5 <- ggplot(plot_df_RWM) + geom_line(aes(x = Iterations, y = Minblem),
    color = "navy") + geom_hline(yintercept = OptimRes$par[5,
    ], color = "red3") + theme_classic()

p6 <- ggplot(plot_df_RWM) + geom_line(aes(x = Iterations, y = MajBlem),
    color = "navy") + geom_hline(yintercept = OptimRes$par[6,
    ], color = "red3") + theme_classic()

p7 <- ggplot(plot_df_RWM) + geom_line(aes(x = Iterations, y = LargNeg),
    color = "navy") + geom_hline(yintercept = OptimRes$par[7,
    ], color = "red3") + theme_classic()

p8 <- ggplot(plot_df_RWM) + geom_line(aes(x = Iterations, y = LogBook),
    color = "navy") + geom_hline(yintercept = OptimRes$par[8,
    ], color = "red3") + theme_classic()

p9 <- ggplot(plot_df_RWM) + geom_line(aes(x = Iterations, y = MinBidShare),
    color = "navy") + geom_hline(yintercept = OptimRes$par[9,
    ], color = "red3") + theme_classic()

grid.arrange(p1, p2, p3, p4, p5, p6, p7, p8, p9, nrow = 3)
```



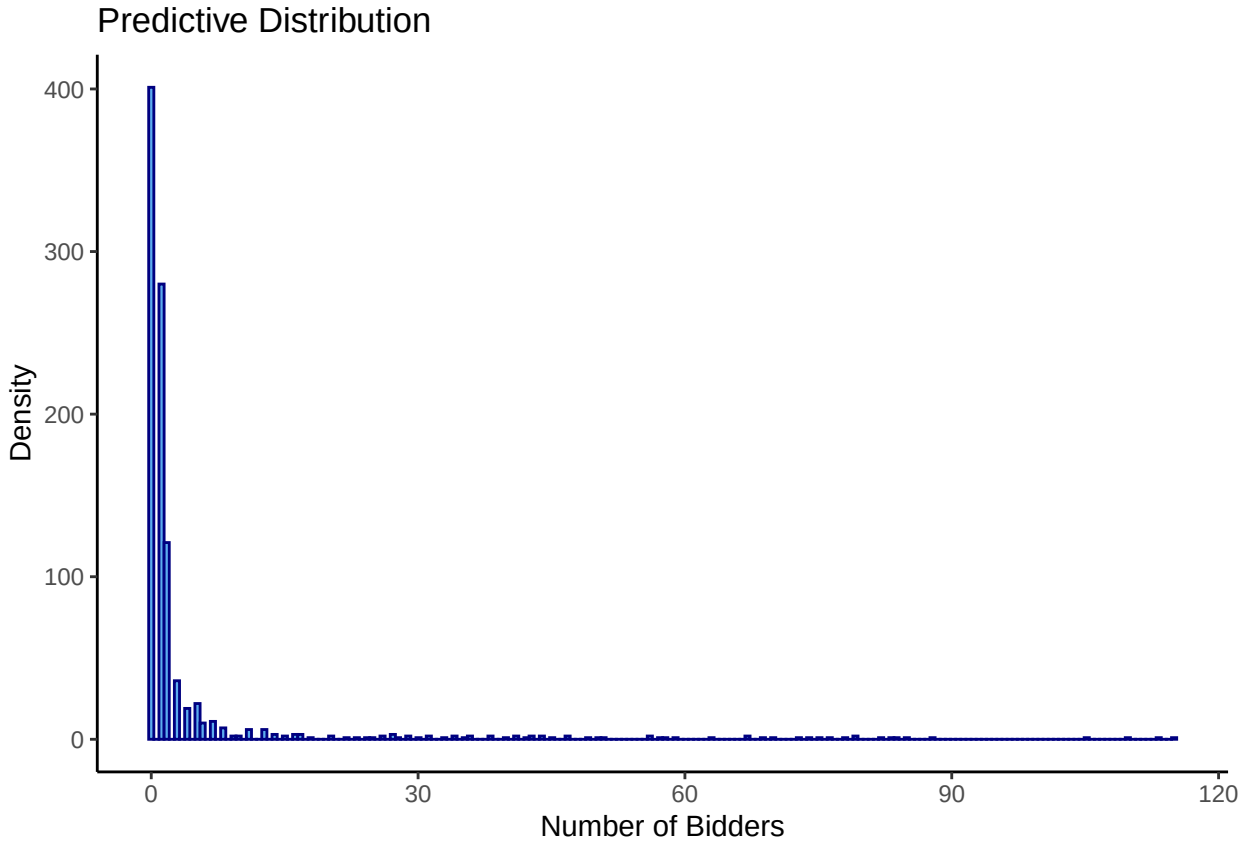
In the above plots, the blue lines illustrate the convergence of each $\tilde{\beta}$ sampled by the Random Walk Metropolis algorithm, and the red lines represent the true posterior mode. More specifically, it could be assumed that the mode of all $\tilde{\beta}$ converges to the true mode, although it is not great in some cases.

Task d

Question: Use the MCMC draws from c) to simulate from the predictive distribution of the number of bidders in a new auction with the characteristics below. Plot the predictive distribution. What is the probability of no bidders in this new auction?
PowerSeller = 1
VerifyID = 0
Sealed = 1
MinBlem = 0
MajBlem = 1
LargNeg = 0
LogBook = 1.2
MinBidShare = 0.8

```
set.seed(1234567890)

# given auction
auction <- c(1, 1, 0, 1, 0, 1, 0, 1.2, 0.8)

nbidders = c()

for (i in 1:nrow(RWMbetas)) {
    nbidders[i] <- rpois(1, exp(RWMbetas[i, ] %*% auction))
```

```
}

nbidders <- data.frame(nbidders)

# calculating the probability of no bidders
zero_bidders <- which(nbidders$nbidders == 0)
prob <- length(zero_bidders)/(nrow(nbidders))

ggplot(nbidders, aes(x = nbidders)) + geom_histogram(bins = 200,
    color = "navy", fill = "steelblue2") + ggtitle("Predictive Distribution") +
    xlab("Number of Bidders") + ylab("Density") + theme_classic()
```



The probability of no bidders in this new auction is approximately 40%.

Assignment 3 *Time series models in Stan*

Task a

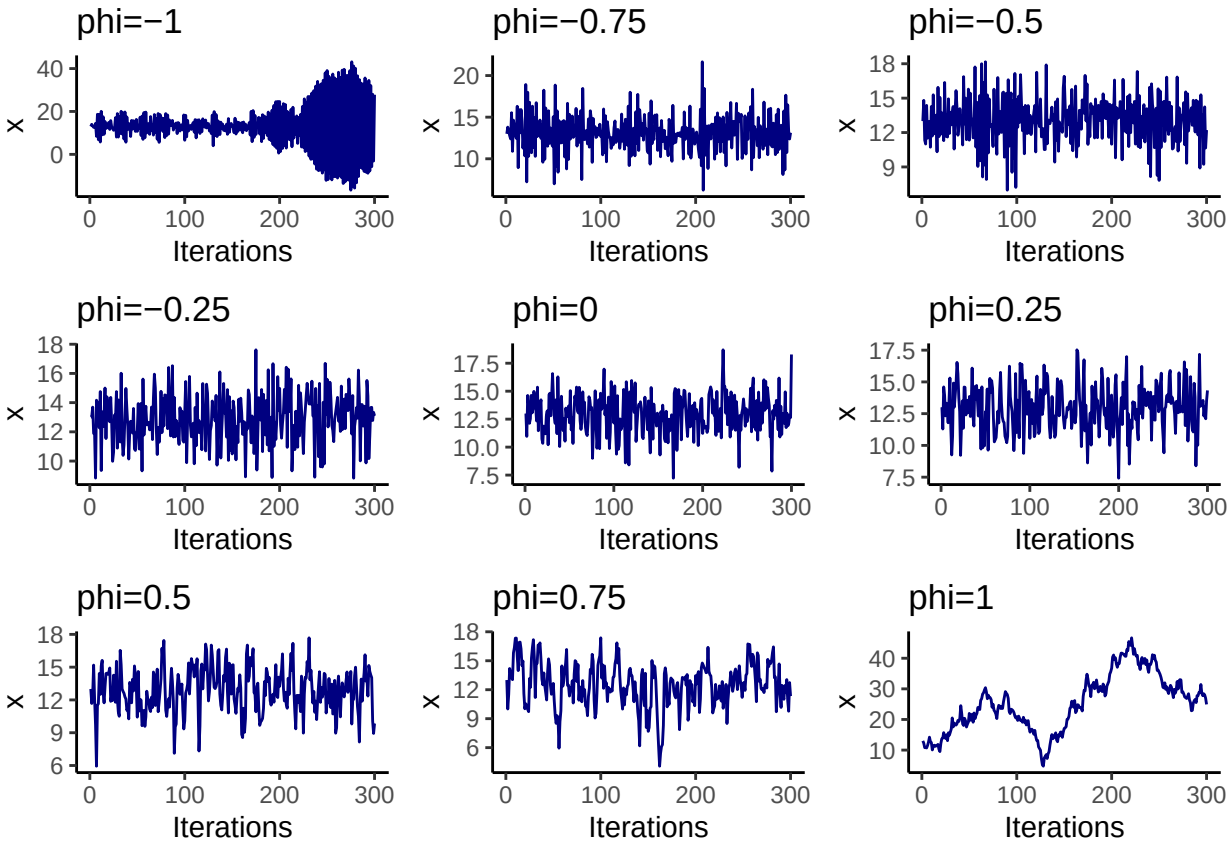
Question:Write a function in R that simulates data from the AR(1)-process:

$$x_t = \mu + \phi(x_{t-1} - \mu) + \epsilon_t, \epsilon_t \overset{\text{iid}}{\sim} N(0, \sigma^2)$$

for given values of μ, ϕ and σ^2. Start the process at $x_1 = \mu$ and then simulate values for x_t for t = 2, 3 . . . , T and return the vector $x_{1:T}$ containing all time points. Use $\mu = 13$, $\sigma^2 = 3$ and T = 300 and look at some different realizations (simulations) of $x_{1:T}$ for values of ϕ between -1 and 1 (this is the interval of ϕ where the AR(1)-process is stationary). Include a plot of at least one realization in the report. What effect does the value of ϕ have on $x_{1:T}$?

```r
set.seed(12345)

# AR(1)-process
ar_process <- function(t, mu, phi, sigma2) {
    x <- c()
    x[1] <- mu
    for (i in 2:t) {
        x[i] <- mu + phi * (x[i - 1] - mu) + rnorm(1, 0, sqrt(sigma2))
    }
    return(x)
}

# given parameters
mu <- 13
sigma2 <- 3
t <- 300
phi <- seq(-1, 1, 0.25)

res <- as.data.frame(sapply(phi, function(phi) ar_process(t,
    mu, phi, sigma2)))

res$iterations <- 1:t
```

```r
# Time series plots of different phis
p1 <- ggplot(res) + geom_line(aes(x = iterations, y = V1), color = "navy") +
    ggtitle("phi=-1") + xlab("Iterations") + ylab("x") + theme_classic()

p2 <- ggplot(res) + geom_line(aes(x = iterations, y = V2), color = "navy") +
    ggtitle("phi=-0.75") + xlab("Iterations") + ylab("x") + theme_classic()

p3 <- ggplot(res) + geom_line(aes(x = iterations, y = V3), color = "navy") +
    ggtitle("phi=-0.5") + xlab("Iterations") + ylab("x") + theme_classic()

p4 <- ggplot(res) + geom_line(aes(x = iterations, y = V4), color = "navy") +
    ggtitle("phi=-0.25") + xlab("Iterations") + ylab("x") + theme_classic()

p5 <- ggplot(res) + geom_line(aes(x = iterations, y = V5), color = "navy") +
    ggtitle("phi=0") + xlab("Iterations") + ylab("x") + theme_classic()

p6 <- ggplot(res) + geom_line(aes(x = iterations, y = V6), color = "navy") +
    ggtitle("phi=0.25") + xlab("Iterations") + ylab("x") + theme_classic()

p7 <- ggplot(res) + geom_line(aes(x = iterations, y = V7), color = "navy") +
    ggtitle("phi=0.5") + xlab("Iterations") + ylab("x") + theme_classic()

p8 <- ggplot(res) + geom_line(aes(x = iterations, y = V8), color = "navy") +
    ggtitle("phi=0.75") + xlab("Iterations") + ylab("x") + theme_classic()

p9 <- ggplot(res) + geom_line(aes(x = iterations, y = V9), color = "navy") +
    ggtitle("phi=1") + xlab("Iterations") + ylab("x") + theme_classic()

grid.arrange(p1, p2, p3, p4, p5, p6, p7, p8, p9, nrow = 3)
```



The autoregressive model specifies that the output variable depends linearly on its own previous values and on a stochastic term. In the above plots, when ϕ has a negative value and is close to 0, then the process still looks like white noise, but as ϕ approaches 1, there seems to be a positive correlated with the previous terms.

Task b

Question: Use your function from a) to simulate two AR(1)-processes, $x_{1:T}$ with $\phi = 0.2$ and $y_{1:T}$ with $\phi = 0.95$. Now, treat your simulated vectors as synthetic data, and treat the values of μ, ϕ and σ^2 as unknown parameters. Implement Stan code that samples from the posterior of the three parameters, using suitable non-informative priors of your choice. [Hint: Look at the time-series models examples in the Stan user's guide/reference manual, and note the different parameterization used here.]
i) Report the posterior mean, 95% credible intervals and the number of effective posterior samples for the three inferred parameters for each of the simulated AR(1)-process. Are you able to estimate the true values?
ii) For each of the two data sets, evaluate the convergence of the samplers and plot the joint posterior of μ and ϕ. Comments?

i)

Table 11: Table for phi=0.2

	mean	2.5%	97.5%	n_eff
mu	13.1299917	12.8534340	13.4180775	3761.651
sigma2	2.9783277	2.5242322	3.5039683	3781.579
phi	0.2846257	0.1753828	0.3937556	3422.264

```
fit2_summary <- summary(fit2)
phi2_stats <- fit2_summary$summary[1:3, c(1, 4, 8, 9)]

knitr::kable(phi2_stats, caption = "Table for phi=0.95")
```

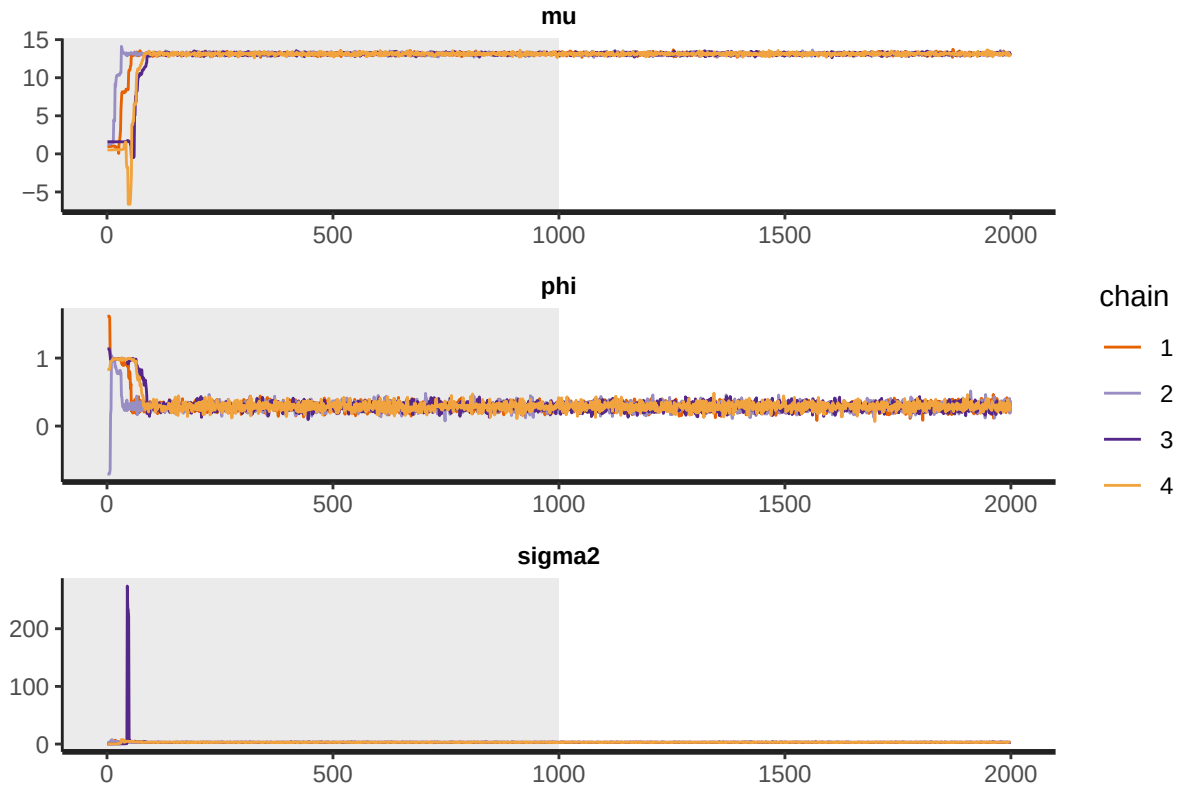
Table 12: Table for phi=0.95

	mean	2.5%	97.5%	n_eff
mu	17.0385695	-79.5703024	145.321794	332.6698
sigma2	3.3502708	2.8547802	3.960345	1429.9033
phi	0.9855417	0.9550611	1.004276	429.0814

From the above tables, the posterior's means are similar to the actual value for each process, except in one case. However, the number of effective posterior samples for the three inferred parameters differ; the number of effective posterior samples is much greater when ϕ equals 0.2 than when ϕ is 0.95. This is because the 95% credible intervals are smaller when ϕ equals 0.2. However, the 95% credible interval of ϕ when ϕ is 0.95 have greater lower and upper bounds, and this might be the reason for a low number of effective posterior samples. Both processes simulated the means of ϕ and σ^2 values pretty well. Moreover, μ is estimated close to its actual value when ϕ is 0.2, compared to μ when ϕ equals 0.95, which differs significantly; it was expected because of its "large" 95% credible interval.

ii)

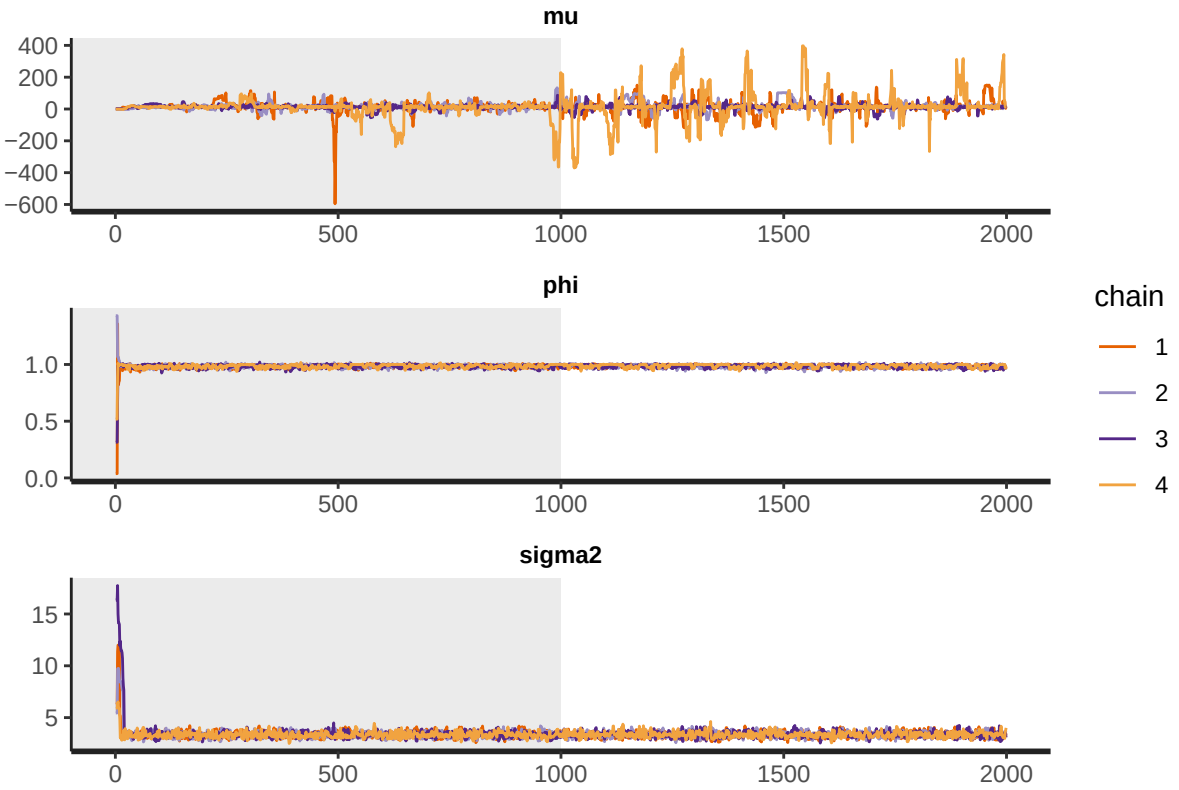
```
traceplot(fit1, pars = c("mu", "phi", "sigma2"), inc_warmup = TRUE,
    nrow = 3)
```



The above traceplots represent the convergence of the samplers when ϕ equals 0.2. After approximately

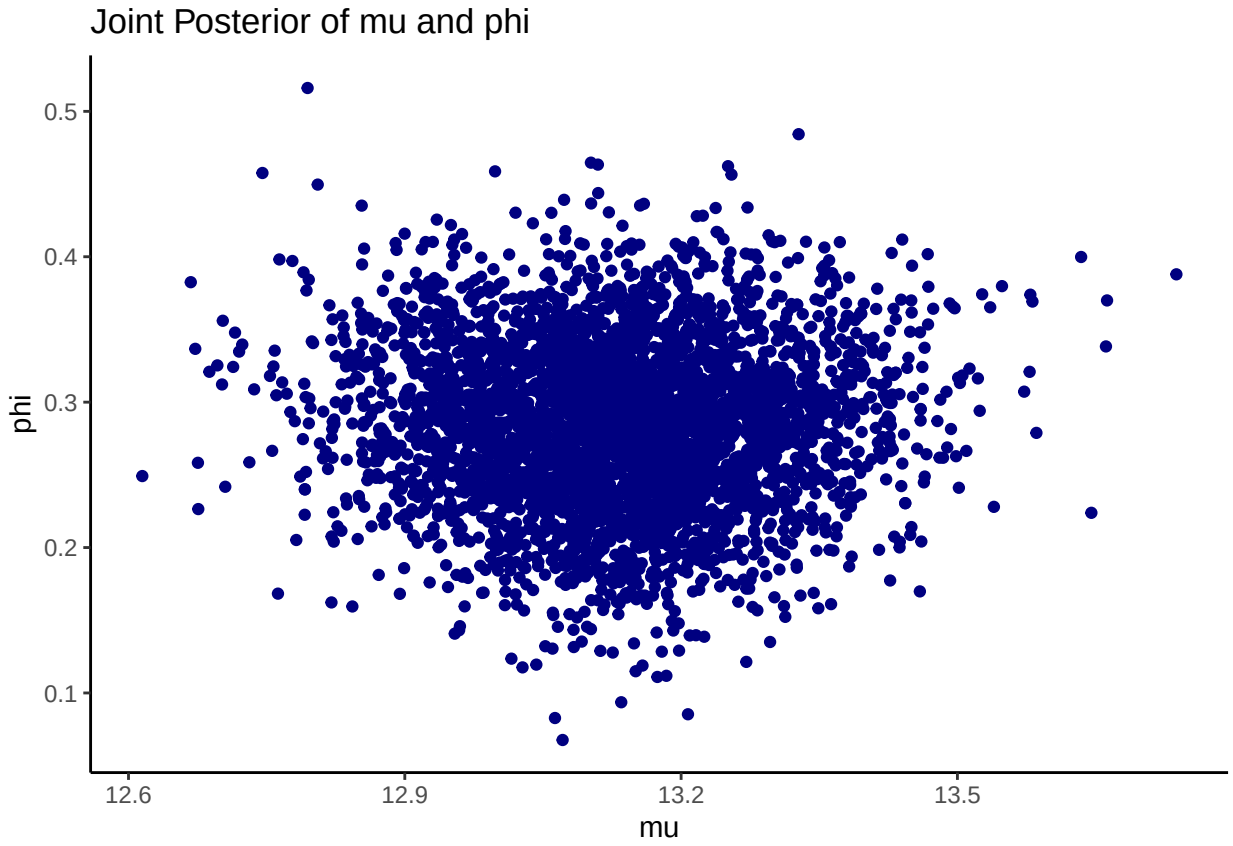
150 iterations all values seems to convergence to the actual ones. (The gray area illustrates the warm-up iterations).

```
traceplot(fit2, pars = c("mu", "phi", "sigma2"), inc_warmup = TRUE,
    nrow = 3)
```



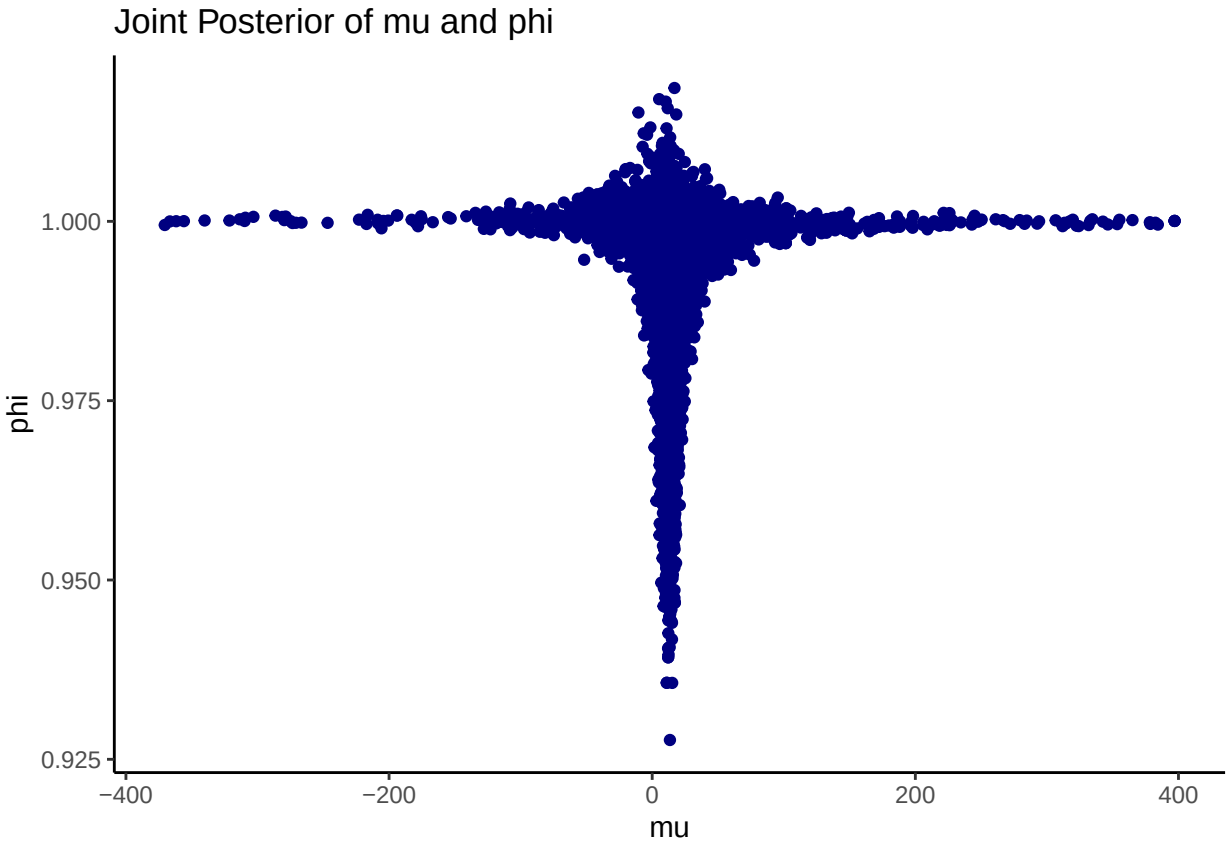
The above traceplots represent the convergence of the samplers when ϕ equals 0.95. After approximately 150 iterations all values seems to convergence to the actual ones. An interesting feature is that μ has extreme values in same cases, which is expected because of its "big" 95% credible interval. (The gray area illustrates the warm-up iterations).

```
draws1 <- extract(fit1)
draws2 <- extract(fit2)

df_draws1 <- data.frame(mu = draws1$mu, phi = draws1$phi)
df_draws2 <- data.frame(mu = draws2$mu, phi = draws2$phi)

ggplot(df_draws1) + geom_point(aes(x = mu, y = phi), color = "navy") +
    ggtitle("Joint Posterior of mu and phi") + theme_classic()
```



In the plot of the joint posterior of μ and ϕ when ϕ is 0.2, it is evident that most of the observations are gathered around in the region of the actual values of μ and ϕ.

```
ggplot(df_draws2) + geom_point(aes(x = mu, y = phi), color = "navy") +
    ggtitle("Joint Posterior of mu and phi") + theme_classic()
```



In the plot of the joint posterior of μ and ϕ when ϕ is 0.95, the observations are plotted close to the actual value of ϕ. However, the observations are spread because of the extremely low and upper bound of the 95% credible interval of μ.