Bayesian learning lab 2 report

Nicolas Taba (nicta839), Yuki Washio (timwa902)

19/04/2021

1.

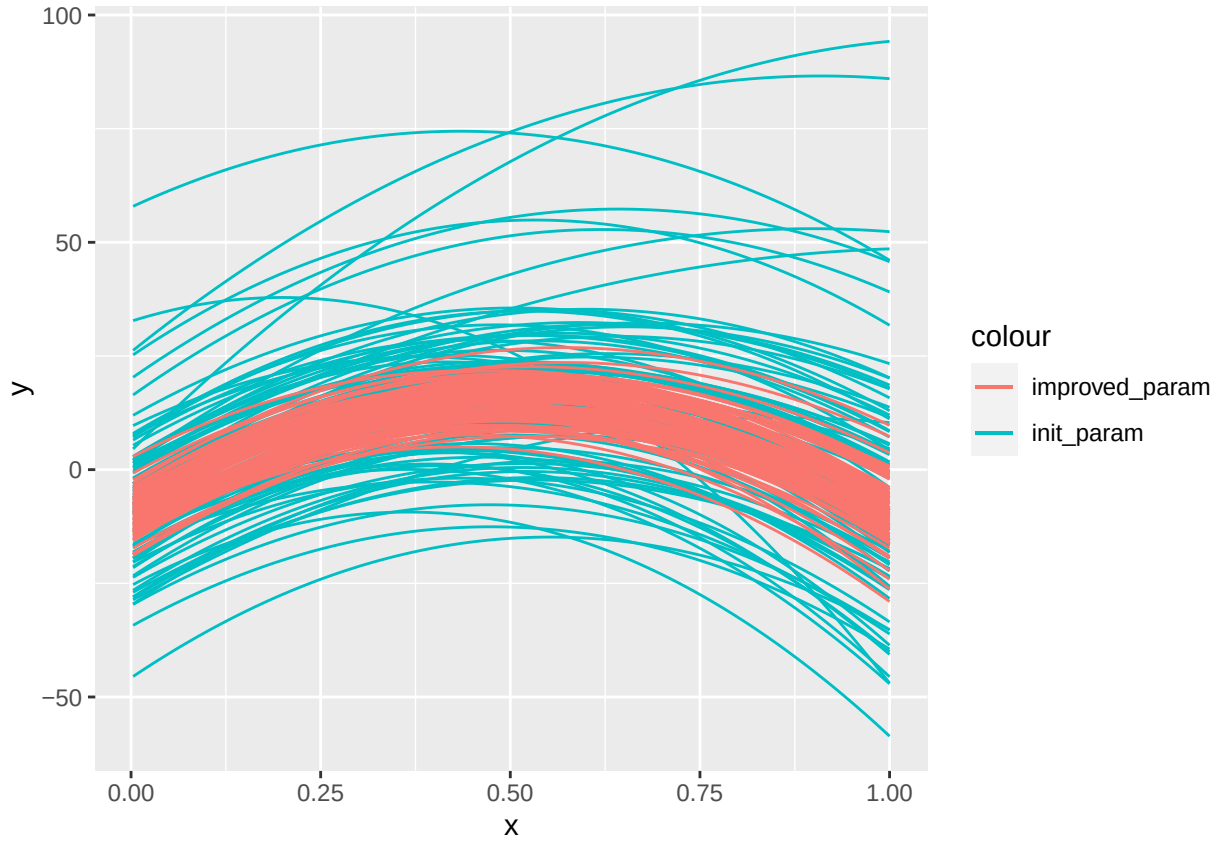
1.a)

```
# rm(list = ls())

library(ggplot2)
library(mvtnorm)
```

```
## Warning: package 'mvtnorm' was built under R version 4.0.3
```

```
# working directory change as needed
setwd("C:/Users/nicol/Documents/MSc/MSc_Statistics_and_Machine_Learning/Bayesian Learning 732A73/lab2")

# Get data
temp_data <- read.table(file = "TempLinkoping.txt", header = TRUE)
time_matrix <- as.matrix(cbind("bias" = 1,"linear" = temp_data$time,"quad" = (temp_data$time)^2))
temp_matrix <- as.matrix(temp_data$temp)

# Lecture 5 slides
#  Inverse chisquare
rInvChisq <- function(draws, n, tau_sq){
  #  n are the degrees of freedom
  X <- rchisq(draws, n)
  sample <- (tau_sq * n)/X
  return(sample)
}
# Variance prior
var_prior <- function(nu, var){
  return(rInvChisq(draws = 1, nu, var))
}
# Beta parameters prior
beta_prior <- function(mu, var, omega){
  return(rmvnorm(1, mu, sigma = var * solve(omega)))
}


# Initial parameters
mu_0 <- matrix(c(-10, 100, -100), nrow = 3)
ohm_0 <- 0.01 * diag(3)
nu_0 <- 4
sigma_0 <- 1
```

```r
set.seed(12345)

# data frames for plotting
df_init <- data.frame(x = temp_data$time, y = 0)
df_imrpove <- data.frame(x = temp_data$time, y = 0)


# generate the variance
sigma_2 <- var_prior(nu = nu_0, var = sigma_0)
# generate the betas using the previous variance
beta_vector <- beta_prior(mu = mu_0, var = sigma_2, omega = ohm_0)
df_init$y <- time_matrix%*%t(beta_vector)
# initialize plot
plot_init <- ggplot(data = df_init, aes(x = x, y = y, col="init_param"))+
  geom_line()

# simulate many draws and plot

# Simulation with initial parameters
nDraws <- 100
for (i in 1:nDraws){
  # generate the variance
  sigma_2 <- var_prior(nu = nu_0, var = sigma_0)
  # generate the betas using the previous variance
  beta_vector <- beta_prior(mu = mu_0, var = sigma_2, omega = ohm_0)
  df_init$y <- time_matrix%*%t(beta_vector)
  plot_init <- plot_init + geom_line(data = df_init, aes(x = x, y = y, col = "init_param"))
}


# Simulation with improved parameters
mu_0_improve <- c(-10, 100, -100)
ohm_0_improve <- 0.1 * diag(3)
nu_0_improve <- 4
sigma_0_improve<- 1

for (i in 1:nDraws){
  # generate the variance
  sigma_2 <- var_prior(nu = nu_0_improve, var = sigma_0_improve)
  # generate the betas using the previous variance
  beta_vector <- beta_prior(mu = mu_0_improve, var = sigma_2, omega = ohm_0_improve)
  df_init$y <- time_matrix%*%t(beta_vector)
  plot_init <- plot_init + geom_line(data = df_init, aes(x = x, y = y, col = "improved_param"))
}

plot_init
```

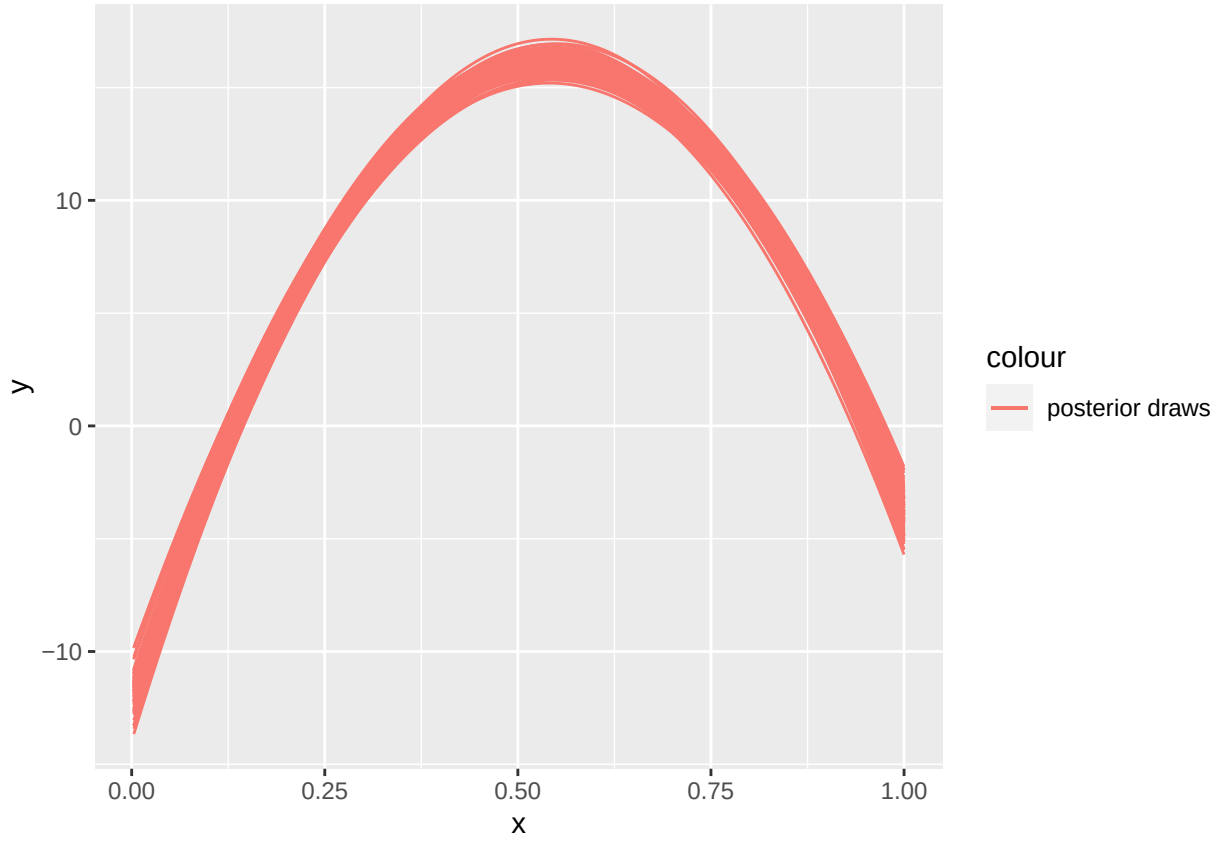


The initial curves present large variations that could influence strongly the posterior distribution such that we would not be able to make accurate or realistic predictions about temperature. We choose to increase the value of Ω_0 by a factor of 10 to get better posterior curves.

1.b)

```
################## 1) B #################
tXX <- t(time_matrix)%*%time_matrix
beta_hat <- solve(tXX)%*%t(time_matrix)%*%temp_matrix
mu_n <- solve(tXX + ohm_0_improve) %*% (tXX%*%beta_hat + ohm_0_improve%*%mu_0)

ohm_n <- tXX + ohm_0_improve

nu_n <- nu_0 + length(temp_matrix)

sigma_n <- (nu_0 * sigma_0 + (t(temp_matrix)%*%temp_matrix + t(mu_0)%*%ohm_0_improve%*%mu_0 - t(mu_n)%*

# posterior
sigma_2_posterior <- var_prior(nu = nu_n, var = sigma_n)
beta_posterior <- rmvnorm(1, mean = mu_n, sigma = sigma_2_posterior[1]*solve(ohm_n))

#posterior data frame and plot initialization
posterior_df <- data.frame(x = temp_data$time, y = time_matrix%*%t(beta_posterior))

# plot posterior
posterior_plot <- ggplot(data = posterior_df, aes(x = x, y = y, col="posterior draws"))+
```

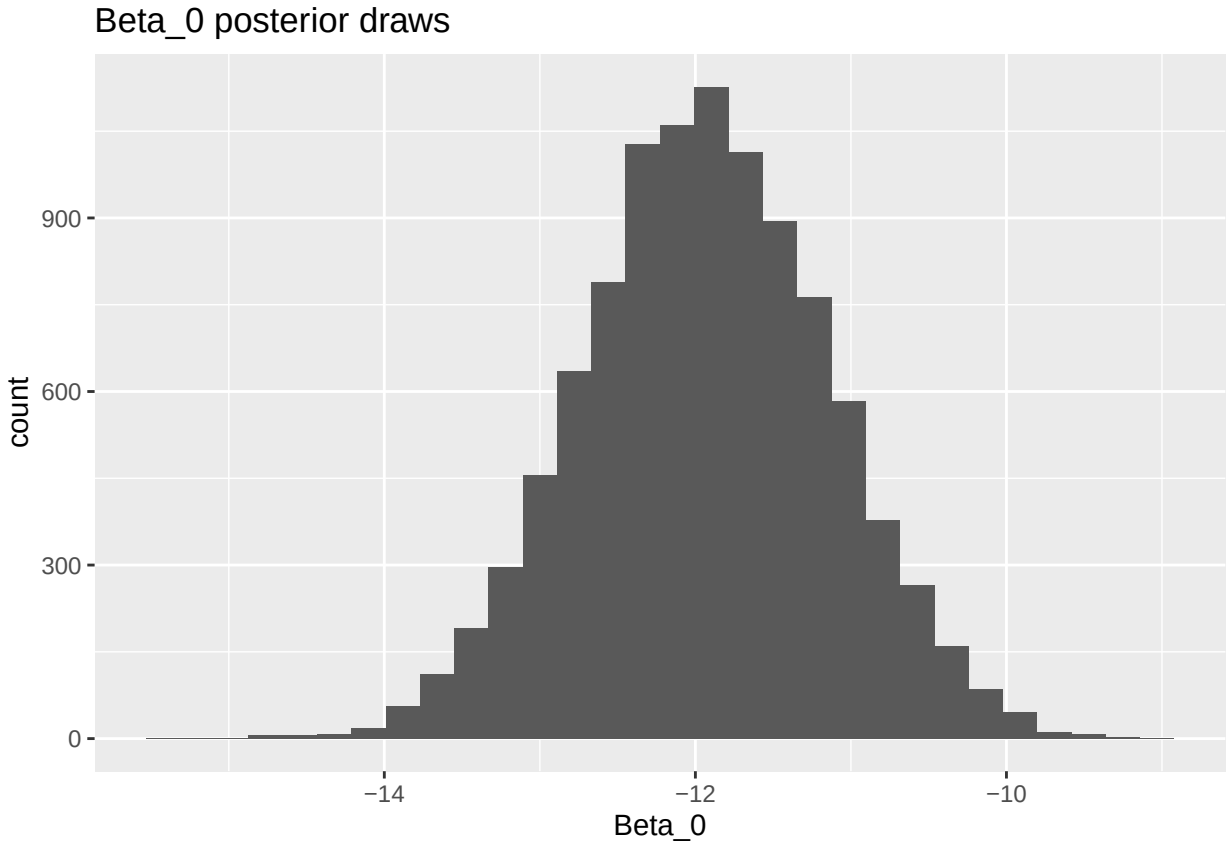
```
  geom_line()

# Multiple draws
nDraws <- 100

for (i in 1:nDraws){
  # generate the variance
  sigma_2_posterior <- var_prior(nu = nu_n, var = sigma_n)
  # generate the betas using the previous variance
  beta_posterior <- rmvnorm(1, mean = mu_n, sigma = sigma_2_posterior[1]*solve(ohm_n))
  posterior_df <- data.frame(x = temp_data$time, y = time_matrix%*%t(beta_posterior))
  posterior_plot <- posterior_plot + geom_line(data = posterior_df, aes(x = x, y = y, col="posterior dra
}

posterior_plot
```




```
### histogram of each posterior parameter
Ndraws <- 10000
df_beta <- data.frame("Beta_0" = 0, "Beta_1" = 0, "Beta_2" = 0, "sigma_2" = 0)

for (i in 1:Ndraws){
  # generate the variance
  sigma_2_posterior <- var_prior(nu = nu_n, var = sigma_n)
  beta_posterior <- rmvnorm(1, mean = mu_n, sigma = sigma_2_posterior[1]*solve(ohm_n))
  df_beta[i, ] <- cbind(beta_posterior, sigma_2_posterior)
}
```

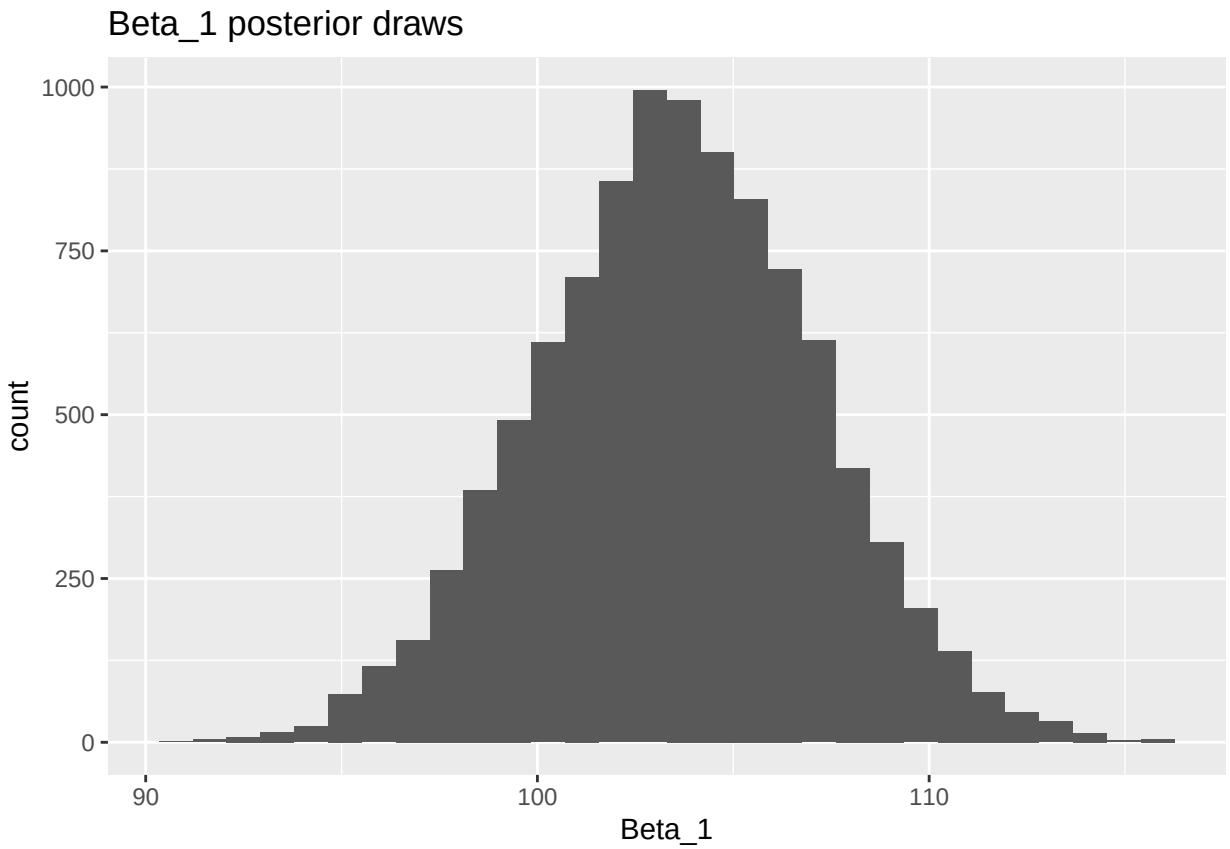
```
histogram_b0 <- ggplot(df_beta, aes(x = Beta_0))+
  geom_histogram()+ labs(title = "Beta_0 posterior draws")

histogram_b0
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



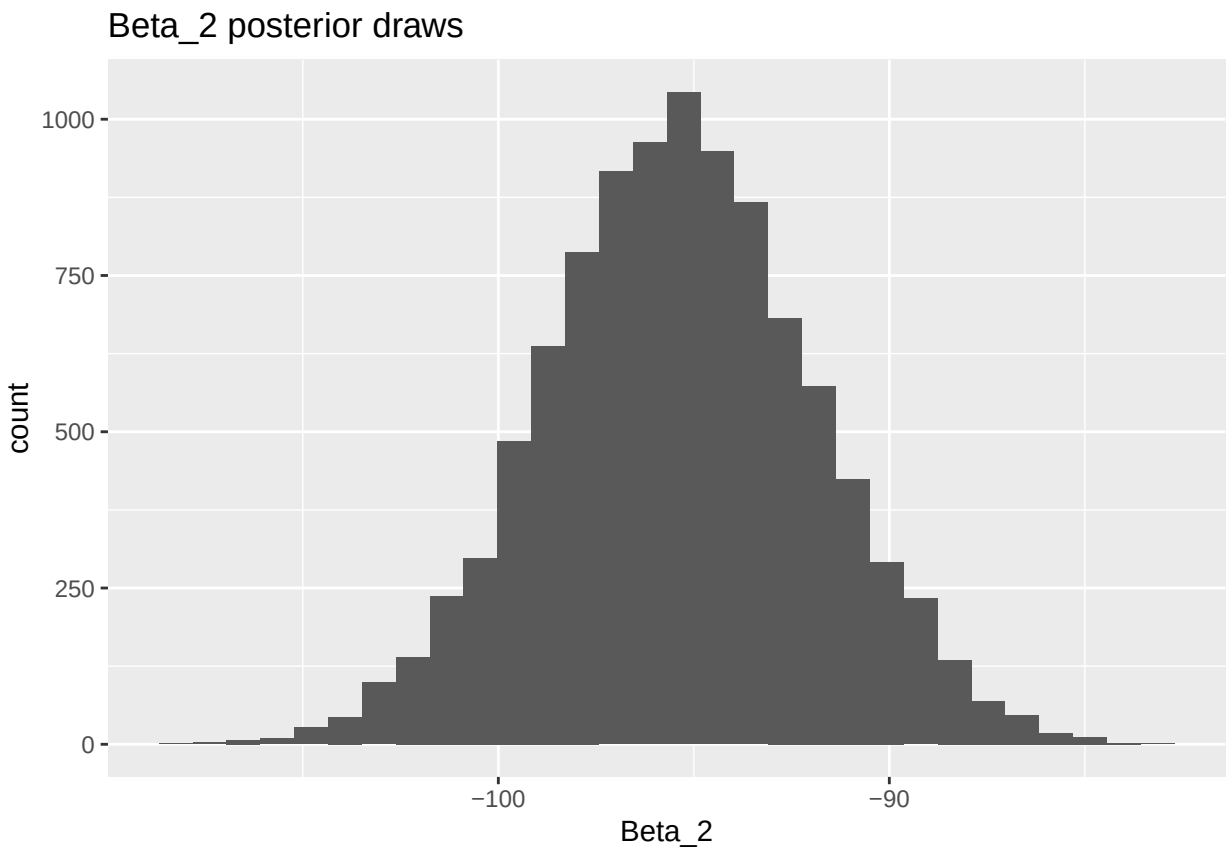

```
histogram_b1 <- ggplot(df_beta, aes(x = Beta_1))+
  geom_histogram()+ labs(title = "Beta_1 posterior draws")

histogram_b1
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



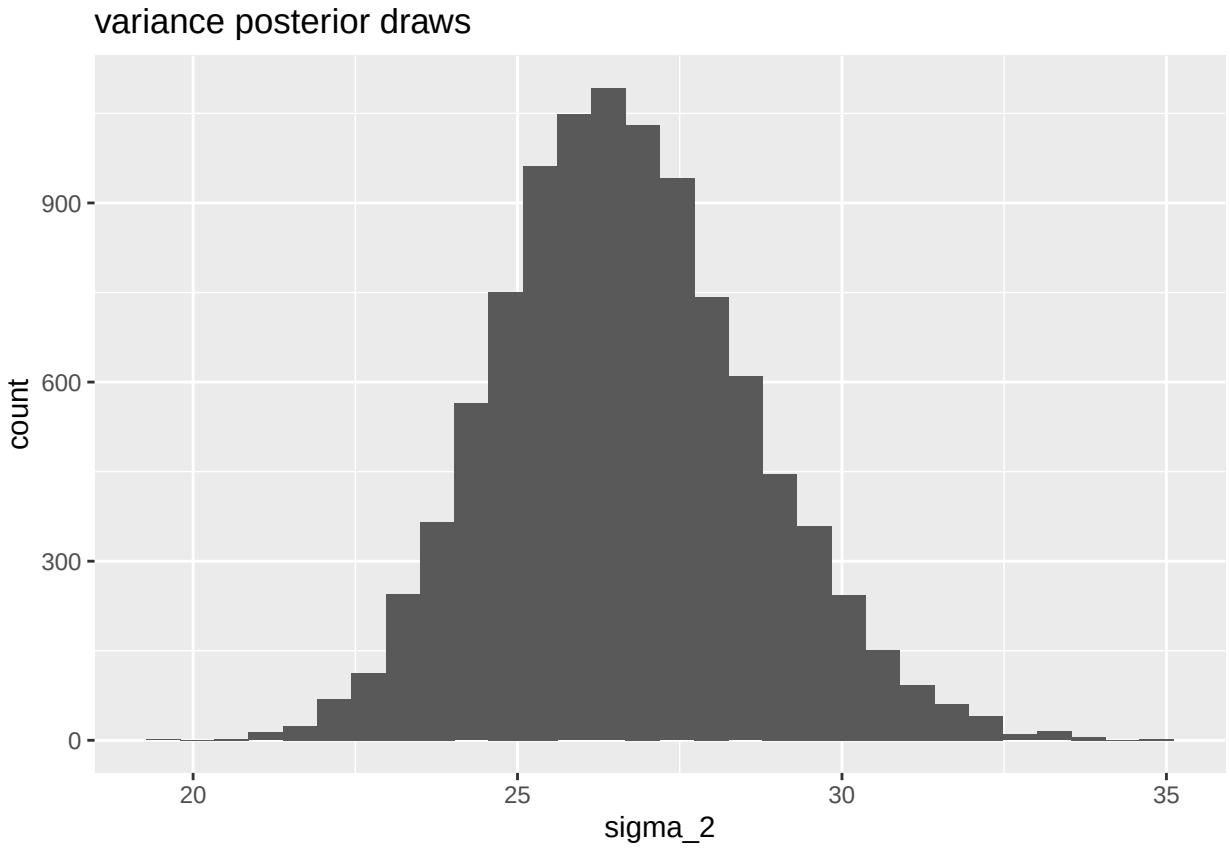

```
histogram_b2 <- ggplot(df_beta, aes(x = Beta_2))+
  geom_histogram()+ labs(title = "Beta_2 posterior draws")

histogram_b2
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

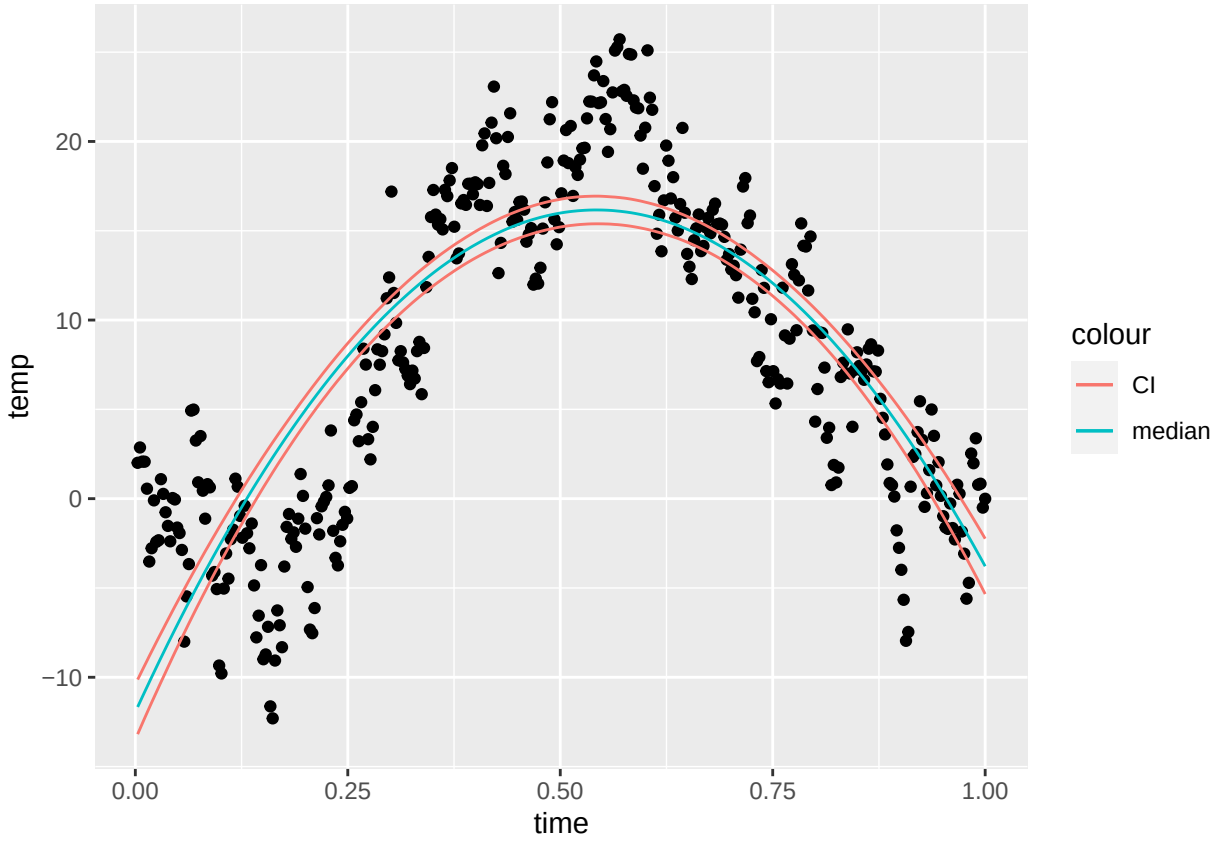



```
histogram_sigma <- ggplot(df_beta, aes(x = sigma_2))+
  geom_histogram()+ labs(title = "variance posterior draws")

histogram_sigma
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
#### scatterplot
temp_posterior <- time_matrix%*%t(as.matrix(df_beta[,1:3]))

df_scatter <- data.frame("median" = rep(0, nrow(temp_posterior)), "upper" = rep(1, nrow(temp_posterior))

for (i in 1:nrow(temp_posterior)){
  df_scatter[i, ] <- cbind(median(temp_posterior[i, ]), quantile(temp_posterior[i, ], probs=0.975), quar
}

df_scatter$temp <- temp_data$temp
df_scatter$time <- temp_data$time

scatter_plot <- ggplot(df_scatter, aes(x=time, y=temp)) +
  geom_point()+
  geom_line(aes(x = time, y = median, col = "median"))+
  geom_line(aes(x = time, y = upper, col = "CI"))+
  geom_line(aes(x = time, y = lower, col ="CI"))



scatter_plot
```

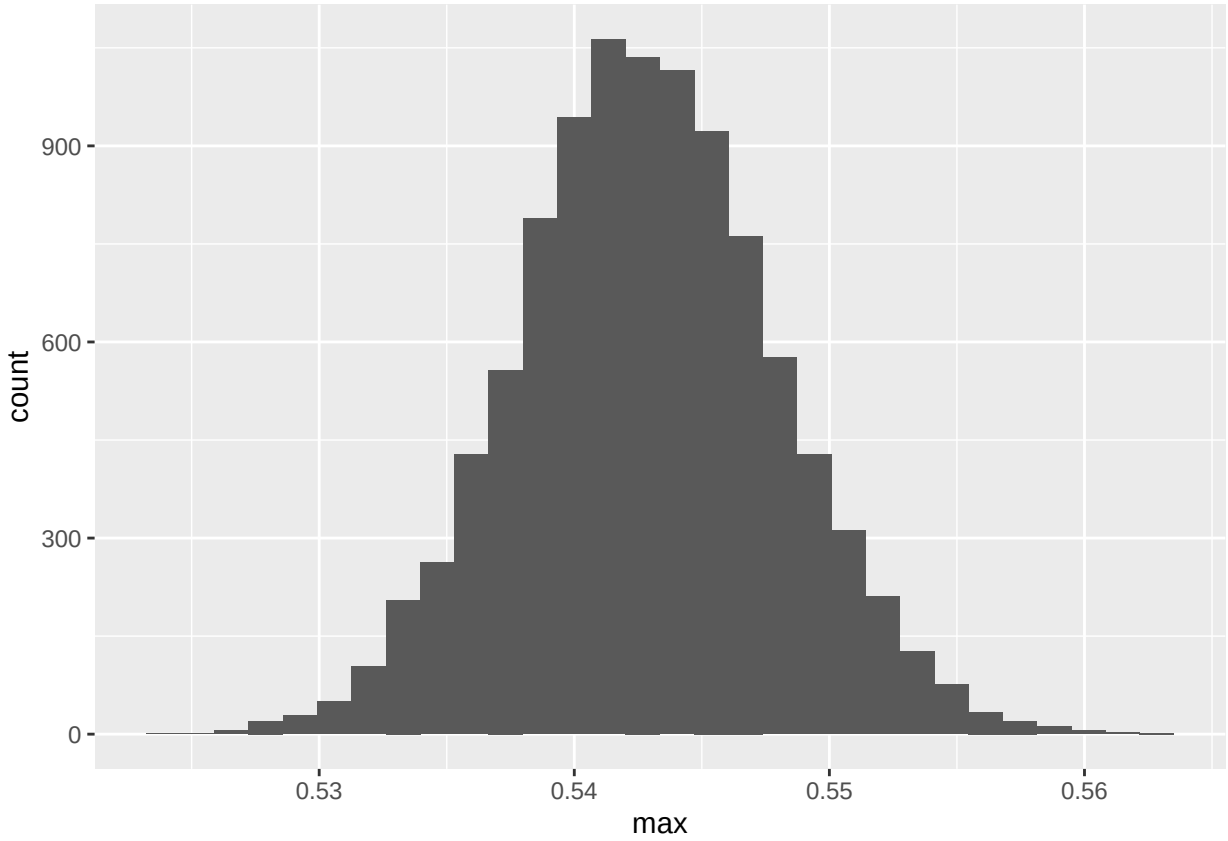


The confidence interval(95% equal tail) curves do not contain most of the data points. This is expected because the credible interval is that of the median of the posterior draws and not the data points. The tighter the bands, the smaller the variance is in our predicted posterior temperature. The regression model asked in this question also does not contain normally distributed noise. Adding this noise would widen the bands and include more data points.

1.c)

```
################ 1C ############
# differentiate the function with respect to x
# The analytical position of the maximum is x = -beta_1/(2*beta_2) (the bias vanishes)

df_beta$max <- -(df_beta$Beta_1)/(2 * df_beta$Beta_2)

plot_max_temp <- ggplot(df_beta, aes(x=max))+
  geom_histogram()

plot_max_temp
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We find that the maximum temperature occurs a little 0.54 year which corresponds to around the second week of June. This is a result that makes sense from how we know the temperature to be around that time of the year in Linköping

1.d)

One way of ensuring that we do not overfit is to use a regularization prior. In our case, we will favor a Normal regularization prior:

$$\beta_j|\sigma^2 \sim N\left(0, \frac{\sigma^2}{\lambda}\right)$$

The posterior mean gives the ridge regression estimator for β. We choose this shrinkage prior because from our earlier analysis, we found tha thte β values were all in the same order of magnitude and that the tails die of rapidly.

2.

2.a)

```
library(ggplot2)
library(mvtnorm)
# working directory change as needed
setwd("C:/Users/nicol/Documents/MSc/MSc_Statistics_and_Machine_Learning/Bayesian Learning 732A73/lab2")
# Read data
data_women <- read.csv("womenWork.dat", sep = "")
```

```r
# glm model
glm_estimate <- glm(formula = Work ~ 0 + ., data = data_women, family = binomial)
summary(glm_estimate)
```

```
##
## Call:
## glm(formula = Work ~ 0 + ., family = binomial, data = data_women)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1662  -0.9299   0.4391   0.9494   2.0582
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## Constant     0.64430    1.52307   0.423 0.672274
## HusbandInc  -0.01977    0.01590  -1.243 0.213752
## EducYears    0.17988    0.07914   2.273 0.023024 *
## ExpYears     0.16751    0.06600   2.538 0.011144 *
## ExpYears2   -0.14436    0.23585  -0.612 0.540489
## Age         -0.08234    0.02699  -3.050 0.002285 **
## NSmallChild -1.36250    0.38996  -3.494 0.000476 ***
## NBigChild   -0.02543    0.14172  -0.179 0.857592
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 277.26  on 200  degrees of freedom
## Residual deviance: 222.73  on 192  degrees of freedom
## AIC: 238.73
##
## Number of Fisher Scoring iterations: 4
```

```r
covariates <- as.matrix(data_women[,2:ncol(data_women)])
target <- data_women[, 1]

Npar <- ncol(covariates)

# Initialize prior
mu <- as.matrix(rep(0, Npar))
tau <- 10
Sigma <- tau^2 * diag(Npar)

# Logposterior for logistic regression (code from Lisam)
LogPostLogistic <- function(betas,y,X,mu,Sigma){
  linPred <- X%*%betas;
  logLik <- sum( linPred*y - log(1 + exp(linPred)) );
  logPrior <- dmvnorm(betas, mu, Sigma, log=TRUE);

  return(logLik + logPrior)
}

# Initialize betas
initVal <- matrix(0, Npar, 1)
```

```r
# Optimizer
OptimRes <- optim(initVal, LogPostLogistic, gr = NULL, y = target, X = covariates, mu = mu, Sigma = Sig

beta_mode <- OptimRes$par
rownames(beta_mode) <- colnames(covariates)
inv_hessian <- solve(-OptimRes$hessian)
rownames(inv_hessian) <- colnames(covariates)
colnames(inv_hessian) <- colnames(covariates)


print(beta_mode)
```

```
##                    [,1]
## Constant     0.62672884
## HusbandInc  -0.01979113
## EducYears    0.18021897
## ExpYears     0.16756670
## ExpYears2   -0.14459669
## Age         -0.08206561
## NSmallChild -1.35913317
## NBigChild   -0.02468351
```

```r
print(inv_hessian)
```

```
##                 Constant    HusbandInc     EducYears      ExpYears
## Constant     2.266022568  3.338861e-03 -6.545121e-02 -1.179140e-02
## HusbandInc   0.003338861  2.528045e-04 -5.610225e-04 -3.125413e-05
## EducYears   -0.065451206 -5.610225e-04  6.218199e-03 -3.558209e-04
## ExpYears    -0.011791404 -3.125413e-05 -3.558209e-04  4.351716e-03
## ExpYears2    0.045780724  1.414915e-04  1.896289e-03 -1.424909e-02
## Age         -0.030293450 -3.588562e-05 -3.240448e-06 -1.340888e-04
## NSmallChild -0.188748354  5.066847e-04 -6.134564e-03 -1.468951e-03
## NBigChild   -0.098023929 -1.444223e-04  1.752732e-03  5.437105e-04
##                 ExpYears2           Age   NSmallChild     NBigChild
## Constant     0.0457807243 -3.029345e-02 -0.1887483542 -0.0980239285
## HusbandInc   0.0001414915 -3.588562e-05  0.0005066847 -0.0001444223
## EducYears    0.0018962893 -3.240448e-06 -0.0061345645  0.0017527317
## ExpYears    -0.0142490853 -1.340888e-04 -0.0014689508  0.0005437105
## ExpYears2    0.0555786706 -3.299398e-04  0.0032082535  0.0005120144
## Age         -0.0003299398  7.184611e-04  0.0051841611  0.0010952903
## NSmallChild  0.0032082535  5.184161e-03  0.1512621814  0.0067688739
## NBigChild    0.0005120144  1.095290e-03  0.0067688739  0.0199722657
```

```r
# 95% interval is 2 std deviation away from mean for normal distribution
upper <- beta_mode[7]+(2*sqrt(inv_hessian[7,7]))
lower <- beta_mode[7]-(2*sqrt(inv_hessian[7,7]))

cat("The 95% posterior probability interval is [", lower, upper,"]")
```

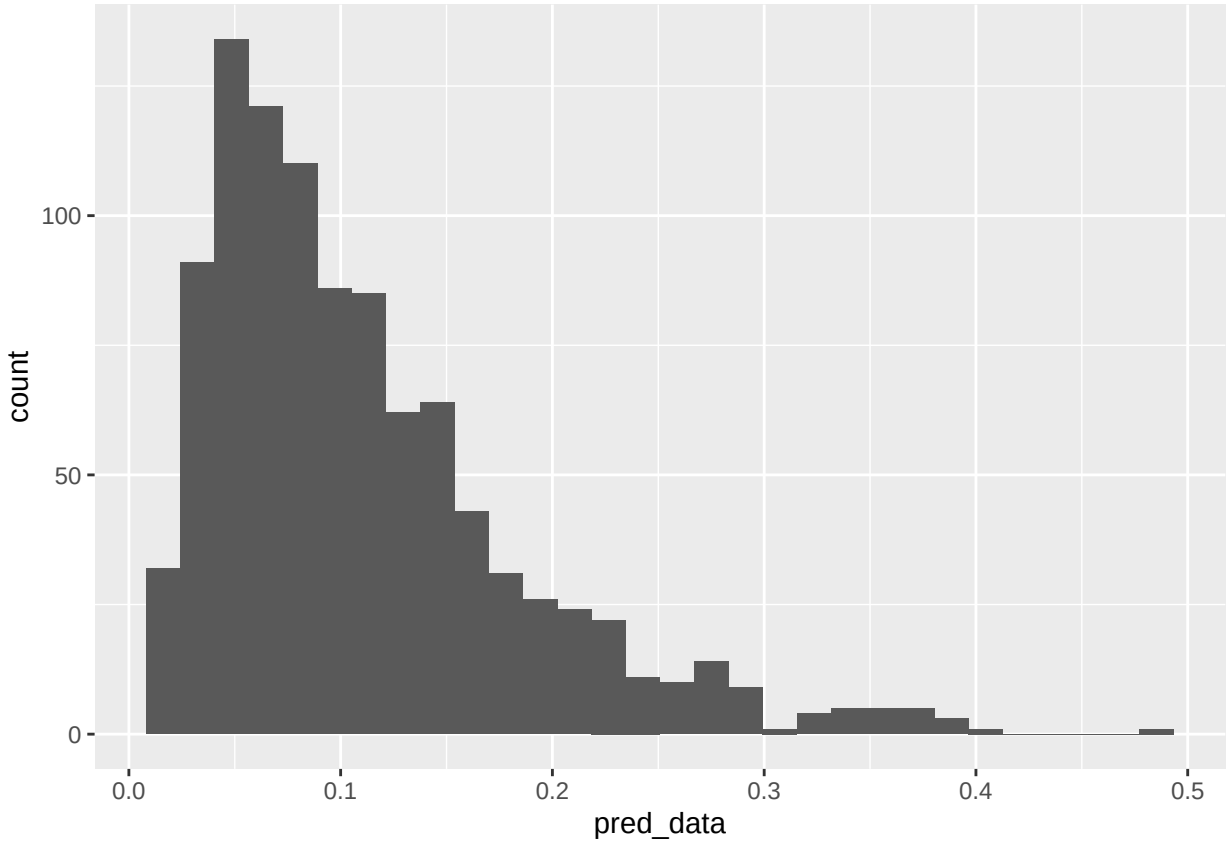
```
## The 95% posterior probability interval is [ -2.136982 -0.5812844 ]
```

The number of small children affects the probability that a woman is predicted to be working given that we have the other data. Furthermore, we can say that having children below the age of 6 negatively impacts the probability that we predict that a woman is working.

2.b)

```
################### 2B

covariates_1 <- c(1, 13, 8, 11, (11/10)^2, 37, 2, 0)

posterior_pred <- function(x, beta){
  return((exp(t(x)%*%beta))/(1 + exp(t(x)%*%beta)))
}


draws <- 1000
pred_data <- rep(0, draws)


for(i in 1:draws){
  draws_data <- rmvnorm(n=1, mean = beta_mode, sigma = inv_hessian)
  pred_data[i] <- posterior_pred(x = covariates_1, beta = t(draws_data))
}


df_pred <- as.data.frame(pred_data)

pred_plot <- ggplot()+
  geom_histogram(data = df_pred, aes(x = pred_data))

pred_plot
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

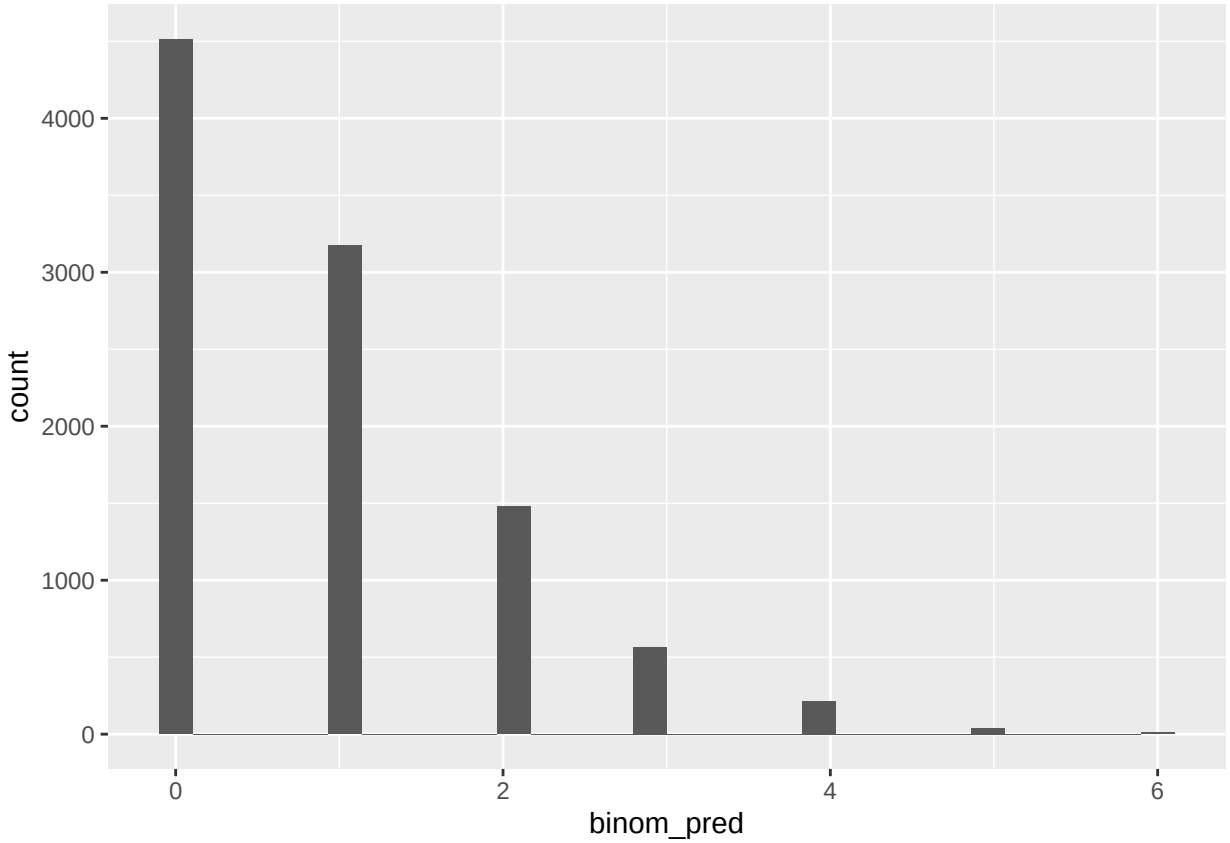


2.c)

We now find the mode of this distribution, which is the value that the parameter takes when the distribution is maximal.

```
n <- 10000
binom_pred <- rep(0, n)
for(i in 1:n){
  binom_pred[i] <- rbinom(1, 8, sample(pred_data, 1))
}

df_binomPred <- as.data.frame(binom_pred)

binom_plot <- ggplot()+
  geom_histogram(data = df_binomPred, aes(x=binom_pred))

binom_plot
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Appendix: All code for this report

```
knitr::opts_chunk$set(echo = TRUE)
# rm(list = ls())

library(ggplot2)
library(mvtnorm)
# working directory change as needed
setwd("C:/Users/nicol/Documents/MSc/MSc_Statistics_and_Machine_Learning/Bayesian Learning 732A73/lab2")

# Get data
temp_data <- read.table(file = "TempLinkoping.txt", header = TRUE)
time_matrix <- as.matrix(cbind("bias" = 1,"linear" = temp_data$time,"quad" = (temp_data$time)^2))
temp_matrix <- as.matrix(temp_data$temp)

# Lecture 5 slides
#  Inverse chisquare
rInvChisq <- function(draws, n, tau_sq){
  #  n are the degrees of freedom
  X <- rchisq(draws, n)
  sample <- (tau_sq * n)/X
  return(sample)
}
# Variance prior
var_prior <- function(nu, var){
```

```
  return(rInvChisq(draws = 1, nu, var))
}
# Beta parameters prior
beta_prior <- function(mu, var, omega){
  return(rmvnorm(1, mu, sigma = var * solve(omega)))
}


# Initial parameters
mu_0 <- matrix(c(-10, 100, -100), nrow = 3)
ohm_0 <- 0.01 * diag(3)
nu_0 <- 4
sigma_0 <- 1

set.seed(12345)

# data frames for plotting
df_init <- data.frame(x = temp_data$time, y = 0)
df_imrpove <- data.frame(x = temp_data$time, y = 0)


# generate the variance
sigma_2 <- var_prior(nu = nu_0, var = sigma_0)
# generate the betas using the previous variance
beta_vector <- beta_prior(mu = mu_0, var = sigma_2, omega = ohm_0)
df_init$y <- time_matrix%*%t(beta_vector)
# initialize plot
plot_init <- ggplot(data = df_init, aes(x = x, y = y, col="init_param"))+
  geom_line()

# simulate many draws and plot

# Simulation with initial parameters
nDraws <- 100
for (i in 1:nDraws){
  # generate the variance
  sigma_2 <- var_prior(nu = nu_0, var = sigma_0)
  # generate the betas using the previous variance
  beta_vector <- beta_prior(mu = mu_0, var = sigma_2, omega = ohm_0)
  df_init$y <- time_matrix%*%t(beta_vector)
  plot_init <- plot_init + geom_line(data = df_init, aes(x = x, y = y, col = "init_param"))
}


# Simulation with improved parameters
mu_0_improve <- c(-10, 100, -100)
ohm_0_improve <- 0.1 * diag(3)
nu_0_improve <- 4
sigma_0_improve<- 1

for (i in 1:nDraws){
  # generate the variance
  sigma_2 <- var_prior(nu = nu_0_improve, var = sigma_0_improve)
```

```
  # generate the betas using the previous variance
  beta_vector <- beta_prior(mu = mu_0_improve, var = sigma_2, omega = ohm_0_improve)
  df_init$y <- time_matrix%*%t(beta_vector)
  plot_init <- plot_init + geom_line(data = df_init, aes(x = x, y = y, col = "improved_param"))
}

plot_init


################## 1) B ################
tXX <- t(time_matrix)%*%time_matrix
beta_hat <- solve(tXX)%*%t(time_matrix)%*%temp_matrix
mu_n <- solve(tXX + ohm_0_improve) %*% (tXX%*%beta_hat + ohm_0_improve%*%mu_0)

ohm_n <- tXX + ohm_0_improve

nu_n <- nu_0 + length(temp_matrix)

sigma_n <- (nu_0 * sigma_0 + (t(temp_matrix)%*%temp_matrix + t(mu_0)%*%ohm_0_improve%*%mu_0 - t(mu_n)%*

# posterior
sigma_2_posterior <- var_prior(nu = nu_n, var = sigma_n)
beta_posterior <- rmvnorm(1, mean = mu_n, sigma = sigma_2_posterior[1]*solve(ohm_n))

#posterior data frame and plot initialization
posterior_df <- data.frame(x = temp_data$time, y = time_matrix%*%t(beta_posterior))

# plot posterior
posterior_plot <- ggplot(data = posterior_df, aes(x = x, y = y, col="posterior draws"))+
  geom_line()

# Multiple draws
nDraws <- 100

for (i in 1:nDraws){
  # generate the variance
  sigma_2_posterior <- var_prior(nu = nu_n, var = sigma_n)
  # generate the betas using the previous variance
  beta_posterior <- rmvnorm(1, mean = mu_n, sigma = sigma_2_posterior[1]*solve(ohm_n))
  posterior_df <- data.frame(x = temp_data$time, y = time_matrix%*%t(beta_posterior))
  posterior_plot <- posterior_plot + geom_line(data = posterior_df, aes(x = x, y = y, col="posterior dra
}

posterior_plot


### histogram of each posterior parameter
Ndraws <- 10000
df_beta <- data.frame("Beta_0" = 0, "Beta_1" = 0, "Beta_2" = 0, "sigma_2" = 0)

for (i in 1:Ndraws){
  # generate the variance
  sigma_2_posterior <- var_prior(nu = nu_n, var = sigma_n)
```

```
  beta_posterior <- rmvnorm(1, mean = mu_n, sigma = sigma_2_posterior[1]*solve(ohm_n))
  df_beta[i, ] <- cbind(beta_posterior, sigma_2_posterior)
}

histogram_b0 <- ggplot(df_beta, aes(x = Beta_0))+
  geom_histogram()+ labs(title = "Beta_0 posterior draws")

histogram_b0

histogram_b1 <- ggplot(df_beta, aes(x = Beta_1))+
  geom_histogram()+ labs(title = "Beta_1 posterior draws")

histogram_b1

histogram_b2 <- ggplot(df_beta, aes(x = Beta_2))+
  geom_histogram()+ labs(title = "Beta_2 posterior draws")

histogram_b2

histogram_sigma <- ggplot(df_beta, aes(x = sigma_2))+
  geom_histogram()+ labs(title = "variance posterior draws")

histogram_sigma

#### scatterplot
temp_posterior <- time_matrix%*%t(as.matrix(df_beta[,1:3]))

df_scatter <- data.frame("median" = rep(0, nrow(temp_posterior)), "upper" = rep(1, nrow(temp_posterior))

for (i in 1:nrow(temp_posterior)){
  df_scatter[i, ] <- cbind(median(temp_posterior[i, ]), quantile(temp_posterior[i, ], probs=0.975), quar
}

df_scatter$temp <- temp_data$temp
df_scatter$time <- temp_data$time

scatter_plot <- ggplot(df_scatter, aes(x=time, y=temp)) +
  geom_point()+
  geom_line(aes(x = time, y = median, col = "median"))+
  geom_line(aes(x = time, y = upper, col = "CI"))+
  geom_line(aes(x = time, y = lower, col ="CI"))



scatter_plot

################ 1C ############
# differentiate the function with respect to x
# The analytical position of the maximum is x = -beta_1/(2*beta_2) (the bias vanishes)

df_beta$max <- -(df_beta$Beta_1)/(2 * df_beta$Beta_2)

plot_max_temp <- ggplot(df_beta, aes(x=max))+
```

```r
  geom_histogram()

plot_max_temp

library(ggplot2)
library(mvtnorm)
# working directory change as needed
setwd("C:/Users/nicol/Documents/MSc/MSc_Statistics_and_Machine_Learning/Bayesian Learning 732A73/lab2")
# Read data
data_women <- read.csv("womenWork.dat", sep = "")

# glm model
glm_estimate <- glm(formula = Work ~ 0 + ., data = data_women, family = binomial)
summary(glm_estimate)

covariates <- as.matrix(data_women[,2:ncol(data_women)])
target <- data_women[, 1]

Npar <- ncol(covariates)

# Initialize prior
mu <- as.matrix(rep(0, Npar))
tau <- 10
Sigma <- tau^2 * diag(Npar)

# Logposterior for logistic regression (code from Lisam)
LogPostLogistic <- function(betas,y,X,mu,Sigma){
  linPred <- X%*%betas;
  logLik <- sum( linPred*y - log(1 + exp(linPred)) );
  logPrior <- dmvnorm(betas, mu, Sigma, log=TRUE);

  return(logLik + logPrior)
}

# Initialize betas
initVal <- matrix(0, Npar, 1)


# Optimizer
OptimRes <- optim(initVal, LogPostLogistic, gr = NULL, y = target, X = covariates, mu = mu, Sigma = Sig

beta_mode <- OptimRes$par
rownames(beta_mode) <- colnames(covariates)
inv_hessian <- solve(-OptimRes$hessian)
rownames(inv_hessian) <- colnames(covariates)
colnames(inv_hessian) <- colnames(covariates)


print(beta_mode)
print(inv_hessian)


# 95% interval is 2 std deviation away from mean for normal distribution
```

```r
upper <- beta_mode[7]+(2*sqrt(inv_hessian[7,7]))
lower <- beta_mode[7]-(2*sqrt(inv_hessian[7,7]))

cat("The 95% posterior probability interval is [", lower, upper,"]")

################### 2B

covariates_1 <- c(1, 13, 8, 11, (11/10)^2, 37, 2, 0)

posterior_pred <- function(x, beta){
  return((exp(t(x)%*%beta))/(1 + exp(t(x)%*%beta)))
}


draws <- 1000
pred_data <- rep(0, draws)


for(i in 1:draws){
  draws_data <- rmvnorm(n=1, mean = beta_mode, sigma = inv_hessian)
  pred_data[i] <- posterior_pred(x = covariates_1, beta = t(draws_data))
}


df_pred <- as.data.frame(pred_data)

pred_plot <- ggplot()+
  geom_histogram(data = df_pred, aes(x = pred_data))

pred_plot
n <- 10000
binom_pred <- rep(0, n)
for(i in 1:n){
  binom_pred[i] <- rbinom(1, 8, sample(pred_data, 1))
}

df_binomPred <- as.data.frame(binom_pred)

binom_plot <- ggplot()+
  geom_histogram(data = df_binomPred, aes(x=binom_pred))

binom_plot
```