Bayesian Learning (732A91) Lab2

Anubhav Dikshit(anudi287) and Lennart Schilling (lensc874)

07 May, 2019

Contents

Question 1: Linear and polynomial regression

The dataset TempLinkoping.txt contains daily temperatures (in Celcius degrees) at Malmslatt, Linkoping over the course of the year 2016 (366 days since 2016 was a leap year). The response variable is temp and the covariate is

time = the number of days since beginning of year/366

The task is to perform a Bayesian analysis of a quadratic regression

$$temp = \beta_0 + \beta_1 \cdot time + \beta_2 \cdot time^2 + \epsilon, epsilon \sim N(0, \sigma^2)$$

a) Determining the prior distribution of the model parameters

Use the conjugate prior for the linear regression model. Your task is to set the prior hyperparameters μ_0, Ω_0, ν_0 and σ_0^2 to sensible values. Start with $\mu_0 = (-10, 100, -100)^T$, $\Omega_0 = 0.01 \cdot I_3$, $\nu_0 = 4$ and $\sigma_0^2 = 1$. 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. Do the collection of curves look reasonable? If not, change the prior hyperparameters until the collection of prior regression curves do agree 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.]

Joint Prior are given by the following:

$$\beta|\sigma^2 \sim N(\mu_0, \sigma^2 \cdot \Omega_0^{-1})$$
$$\sigma^2 \sim Inv - \chi^2(\nu_o, \sigma_0^2)$$

Posterior

$$\beta|\sigma^2, y \sim N(\mu_n, \sigma^2 \Omega_n^{-1})$$
$$\sigma^2|y \sim Inv - \chi^2(\nu_n, \sigma_n^2)$$

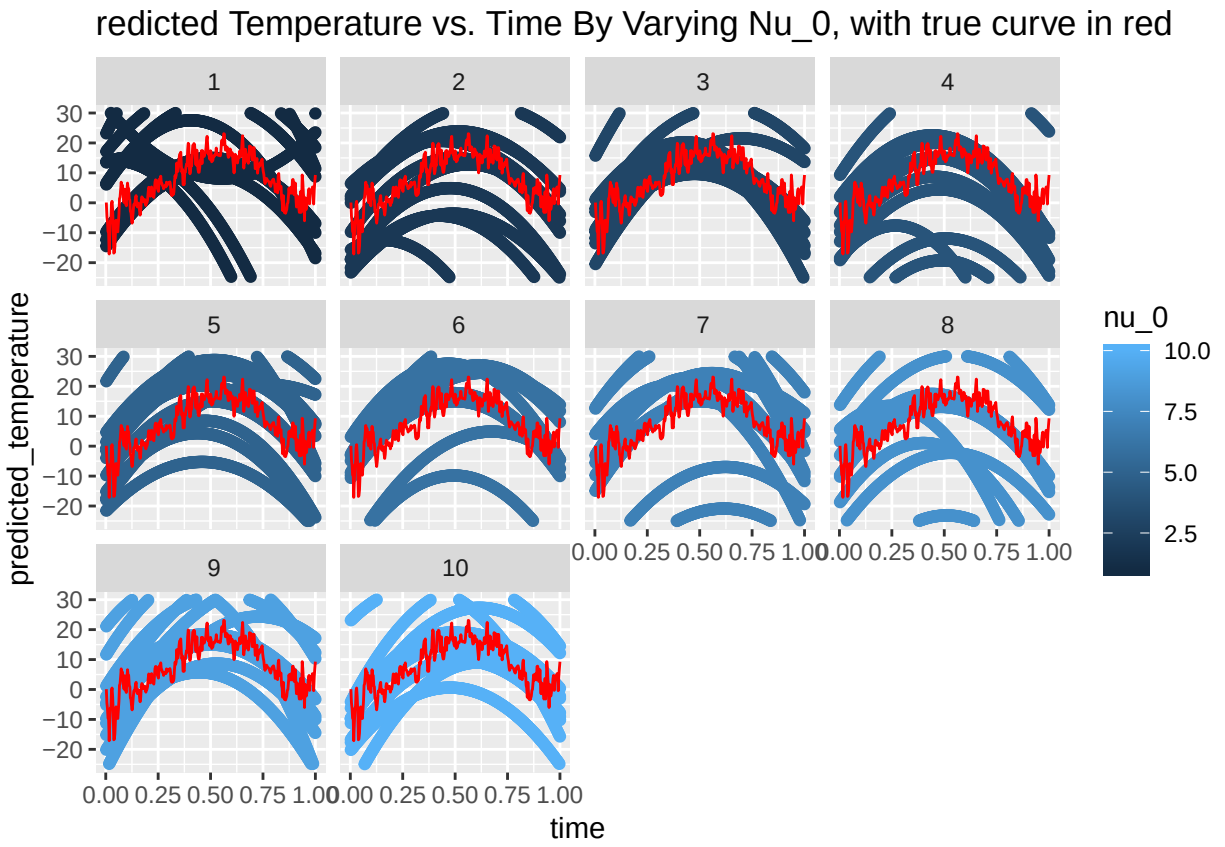
$$\Omega_0 = \lambda I_n$$
$$\mu_n = (X^T X + \Omega_0)^{-1}(X^T X \hat{B} + \Omega_0 \mu_0)$$
$$\Omega_n = X^T X + \Omega_0$$
$$\hat{\beta} = (X^T X)^{-1} X^T Y$$
$$\nu_0 = \nu_0 + n$$
$$\sigma_n^2 = \frac{\nu_0 \sigma_0^2 + (YY^T + \mu_0^T \Omega_0 \mu_0 - \mu_n^T \Omega_n \mu_n)}{\nu_n}$$

Finally

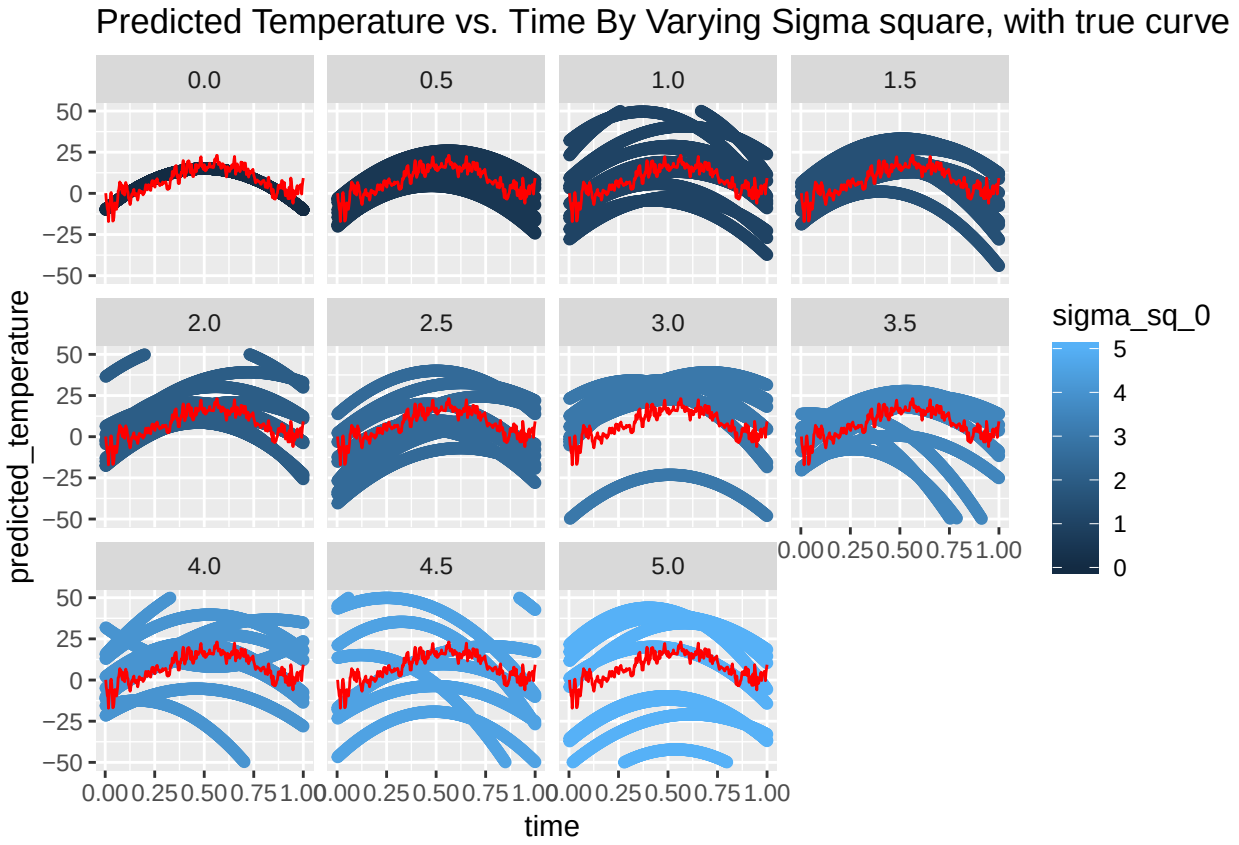
$$Y = X\beta + \epsilon$$
$$\epsilon = N(0, \sigma^2)$$

```
set.seed(12345)

temp_data = read.delim("TempLinkoping2016.txt")
```

```
temp_data$time_square = (temp_data$time)^2
temp_data$intercept = 1

# Initialzing hyper-parameters
mu_0 = t(c(-10,100,-100))
omega_0 = diag(x=0.01,nrow=3,ncol=3)
nu_0 = 4
sigma_sq_0 = 1

# calculating X and other vectors
X = temp_data[,c("intercept", "time", "time_square")] %>% as.matrix()
X_T = t(X)
Y = temp_data[,c("temp")] %>% as.matrix()
Y_T = t(Y)

# defining a function to sample and predict temperature
quad_regression <- function(mu_0, omega_0, nu_0, sigma_sq_0,X,Y){
# sampling sigma squares using inverse chi-square
x = rchisq(n = 1, df = nu_0)
# Calculating sigma_sq.
sigma_sq = (nu_0*sigma_sq_0)/x
beta <- mvtnorm::rmvnorm(n = 1 , mean=mu_0, sigma=sigma_sq * solve(omega_0))
final <-  X %*% t(beta) + rnorm(n=1,mean = 0, sd=sqrt(sigma_sq))
colnames(final) <- c("Predicted_temperature")
return(final)
}

# Define a named list of parameter values
gs = list(nu_0 = seq(1,10,1),
          sigma_sq_0 = seq(0,5,0.5)) %>%
  cross_df() # Convert to data frame grid

final = gs %>% mutate(predicted_temperature = pmap(gs, mu_0 = mu_0,
                                                   omega_0 = omega_0, X=X, Y=Y, quad_regression),
                      actual_temperature = list(temp_data$temp),
                      time = list(temp_data$time)) %>% as_tibble()

final = unnest(final, predicted_temperature, actual_temperature, time)

# Using cut
final$sigma_sq_0_cut <- cut2(final$sigma_sq_0, m=5)
final$nu_0_cut <- cut2(final$nu_0, m=5)

# plotting
final %>% filter(predicted_temperature <30 & predicted_temperature >-25)  %>%
  ggplot(aes(x=time, y = predicted_temperature, color=nu_0)) +
  geom_point() +
  geom_line(aes(y=actual_temperature), color="red") +
  facet_wrap(~nu_0_cut) +
  ggtitle("redicted Temperature vs. Time By Varying Nu_0, with true curve in red")
```




```
final %>% filter(predicted_temperature <50 & predicted_temperature >-50)  %>%
  ggplot(aes(x=time, y = predicted_temperature, color=sigma_sq_0)) +
  geom_point() +
  geom_line(aes(y=actual_temperature), color="red") +
    facet_wrap(~sigma_sq_0_cut) +
  ggtitle("Predicted Temperature vs. Time By Varying Sigma square, with true curve in red")
```

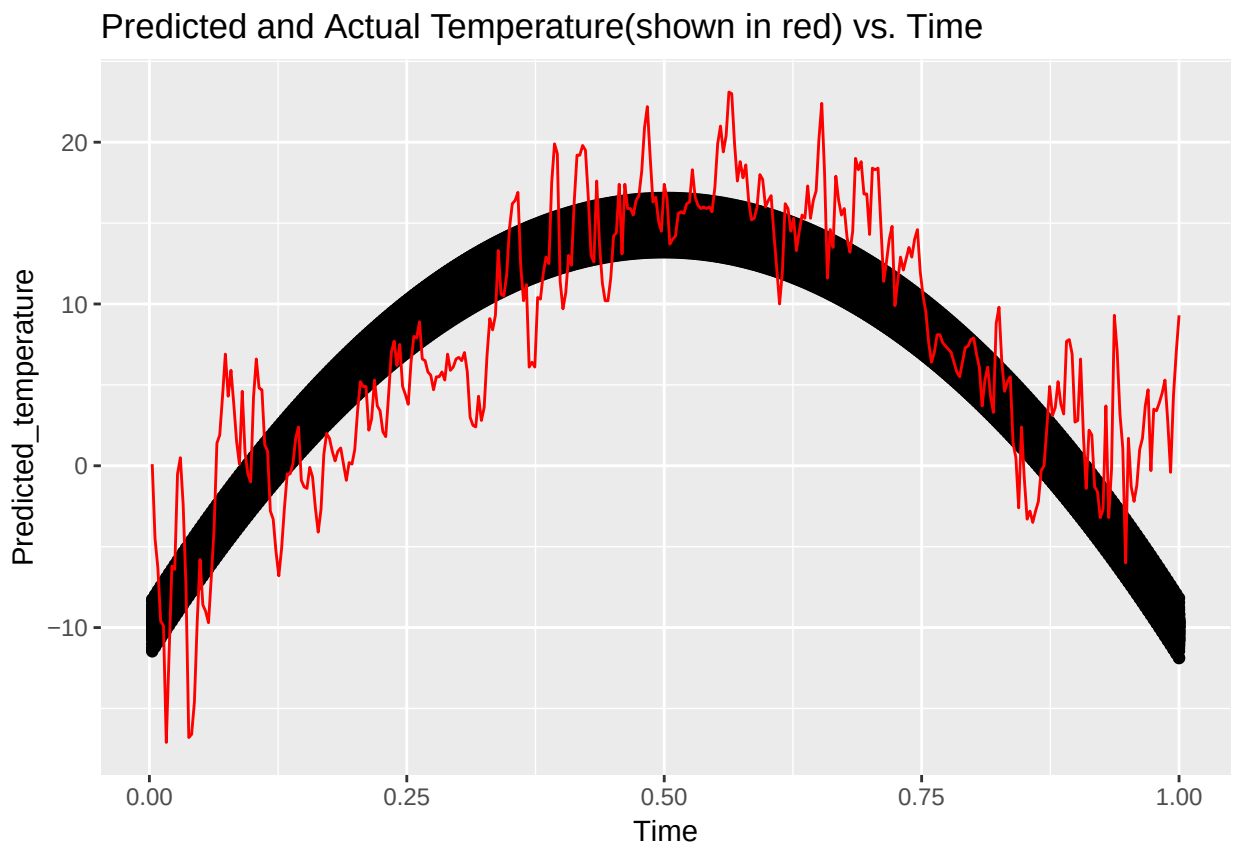


Analysis: Its aparent that low values of sigma work well (0-0.5), thus we will set it to ~0.5 and nu_0 will be ~10, this leads to have the following curve. The impact of omega and mu_0 was not much, so we keep them same.

Best fit

```
set.seed(12345)

# Initialzing final hyper-parameters
mu_0 = t(c(-10,100,-100))
omega_0 = diag(x=50,nrow=3,ncol=3)
nu_0 = 10
sigma_sq_0 = 0.25

best_fit <- NULL
for(i in 1:100){
temp <- quad_regression(mu_0=mu_0, omega_0=omega_0, nu_0=nu_0, sigma_sq_0=sigma_sq_0,X=X,Y=Y)
temp <- temp %>% as.data.frame() %>% mutate(Actual_temperature = temp_data$temp, Time = temp_data$time)
best_fit <- rbind(temp, best_fit)
}

# plotting
best_fit %>%
  ggplot(aes(x=Time, y = Predicted_temperature)) +
  geom_point() +
```

```
  geom_line(aes(y=Actual_temperature), color="red") +
  ggtitle("Predicted and Actual Temperature(shown in red) vs. Time")
```

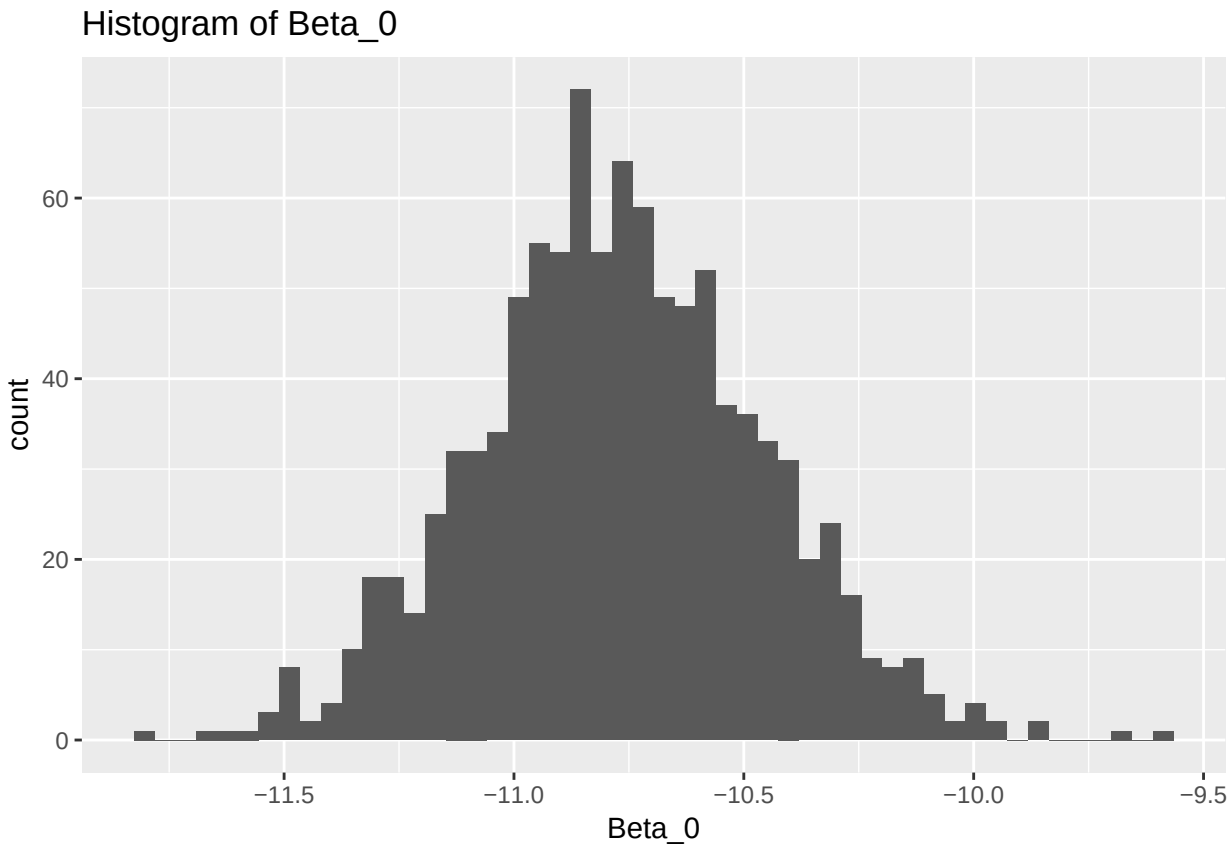


b) Simulating from the joint posterior distribution

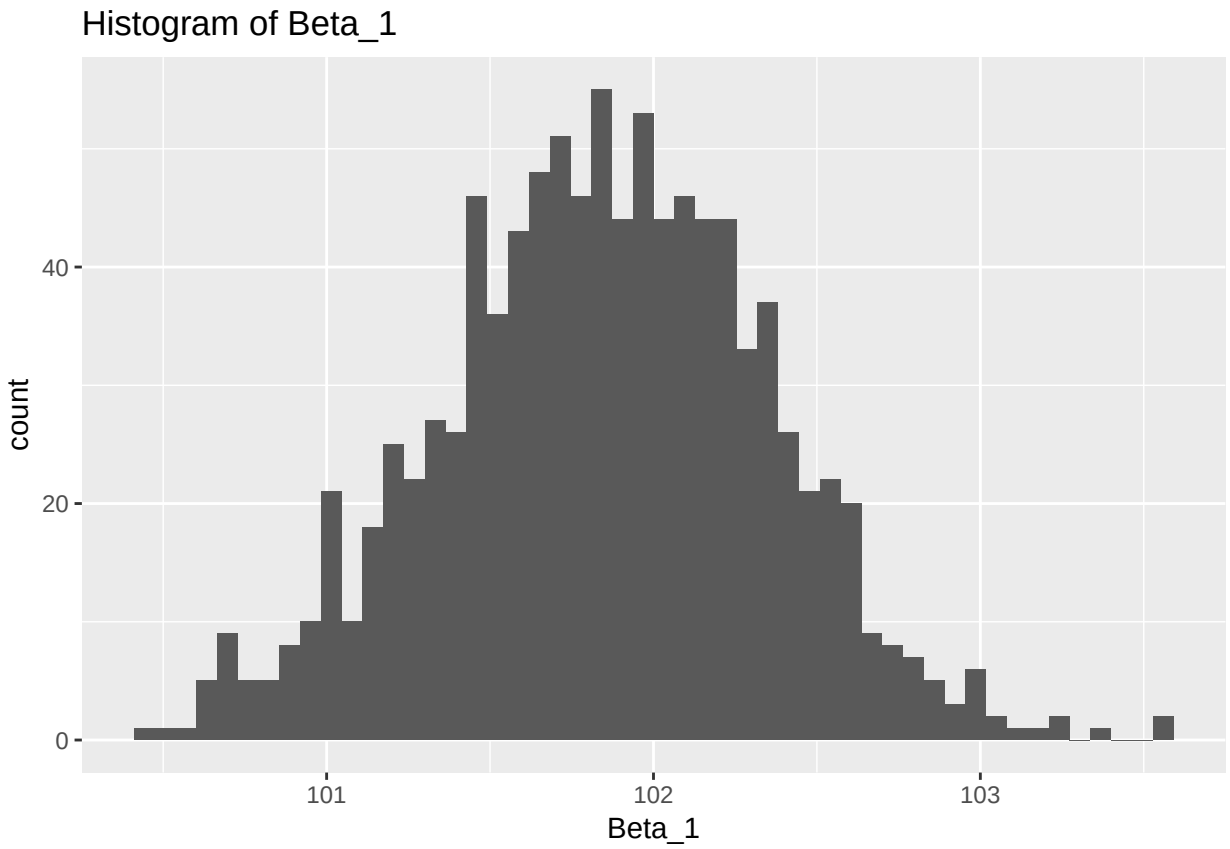
Write a program that simulates from the joint posterior distribution of β_0, β_1, β_2 and σ^2. Plot the marginal posteriors for each parameter as a histogram. Also produce another figure with a scatter plot of the temperature data and overlay a curve for the posterior median of the regression function $f(time) = \beta_0 + \beta_1 \cdot time + \beta_2 \cdot time^2$, computed for every value of time. Also overlay curves for the lower 2.5 percent and upper 97.5 percent posterior credible interval for f(time). That is, compute the 95 percent equal tail posterior probability intervals for every value of time and then connect the lower and upper limits of the interval by curves. Does the interval bands contain most of the data points? Should they?

```
set.seed(12345)
# initialized values from above
mu_0 = c(-10,100,-100)
mu_0_T = t(mu_0)
omega_0 = diag(x=50,nrow=3,ncol=3)
nu_0 = 10
sigma_sq_0 = 0.25

# calculate beta hat given by beta_hat =  (X^{T} X)^{-1} X^T Y
beta_hat = solve(X_T %*% X) %*% X_T %*% Y

# calculate mu_n
```

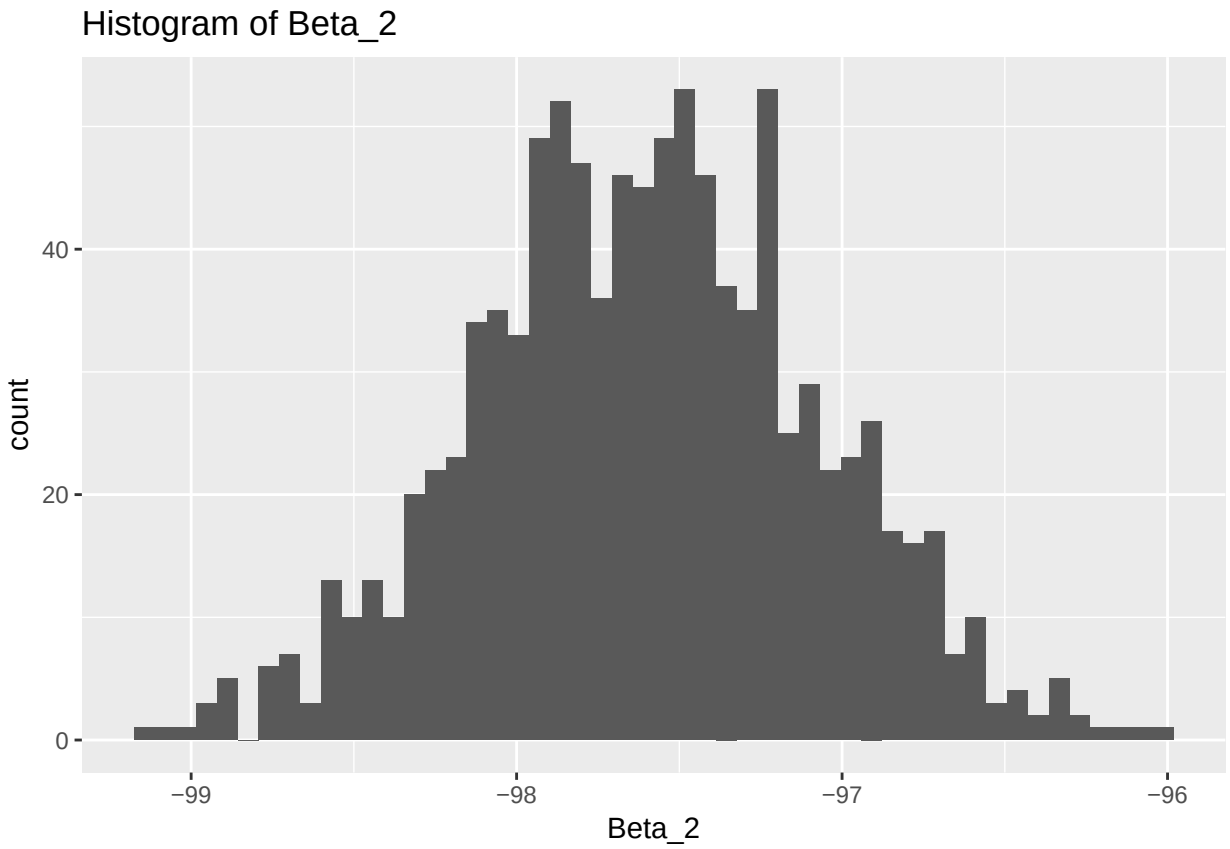
```r
mu_n = solve(X_T %*% X + omega_0) %*% (X_T %*% X %*% beta_hat + omega_0 %*% mu_0)

# calculate omega_n given by Omega_n = X^T X + \Omega_0
omega_n = X_T %*% X + omega_0

# calculate nu_n given by nu_0 + n
nu_n = NROW(X) + nu_0

# calculate sigma_sq_n given by (nu_0 * sigma_sq_0 + (Y %*% Y_T + mu_0^T %*% omega_0 %*% mu_0 - mu_n^T
sigma_sq_n = (nu_0 * sigma_sq_0 + Y_T %*% Y + mu_0_T %*% omega_0 %*% mu_0 - t(mu_n) %*% omega_n %*% mu_

# posterior distribution given by Beta = N(\mu_n, \sigma^2 \Omega_n^{-1})

final <- NULL
temp <- NULL
temp2 <- NULL
predicted_data <- NULL
for(i in 1:1000){
#\sigma^2 =  Inv-\chi^2(nu_n, \sigma^2_n)
x = rchisq(n = 1, df = nu_n)
# Calculating sigma_sq.
sigma_sq = as.vector((nu_n*sigma_sq_n)/x)
betas <- mvtnorm::rmvnorm(n = 1 , mean=mu_n, sigma=sigma_sq * solve(omega_n))
temp2 <-  X %*% t(betas) + rnorm(n=1,mean = 0, sd=sqrt(sigma_sq)) # add the error term you get predicti
temp2 <- cbind(temp2, X, Y)
predicted_data <- rbind(temp2, predicted_data)
temp <- cbind(betas,sigma_sq)
final <- rbind(temp,final)
}

colnames(final) <- c("Beta_0", "Beta_1", "Beta_2", "Sigma_squared")
colnames(predicted_data) <- c("Predicted_Temperature", "Intercept", "Time", "Time_square", "Actual_Temp
final = final %>% as.data.frame()

# calculation of the 95% confidence interval
predicted_data = predicted_data %>% as.data.frame() %>%
group_by(Actual_Temperature, Time) %>%
summarise(temp_hat_median = median(Predicted_Temperature),
temp_hat_l_limit = quantile(x = Predicted_Temperature, probs = 0.025),
temp_hat_u_limit = quantile(x = Predicted_Temperature, probs = 0.975))

ggplot(data=final,aes(Beta_0)) +
  geom_histogram(bins=50) +
  ggtitle("Histogram of Beta_0")
```



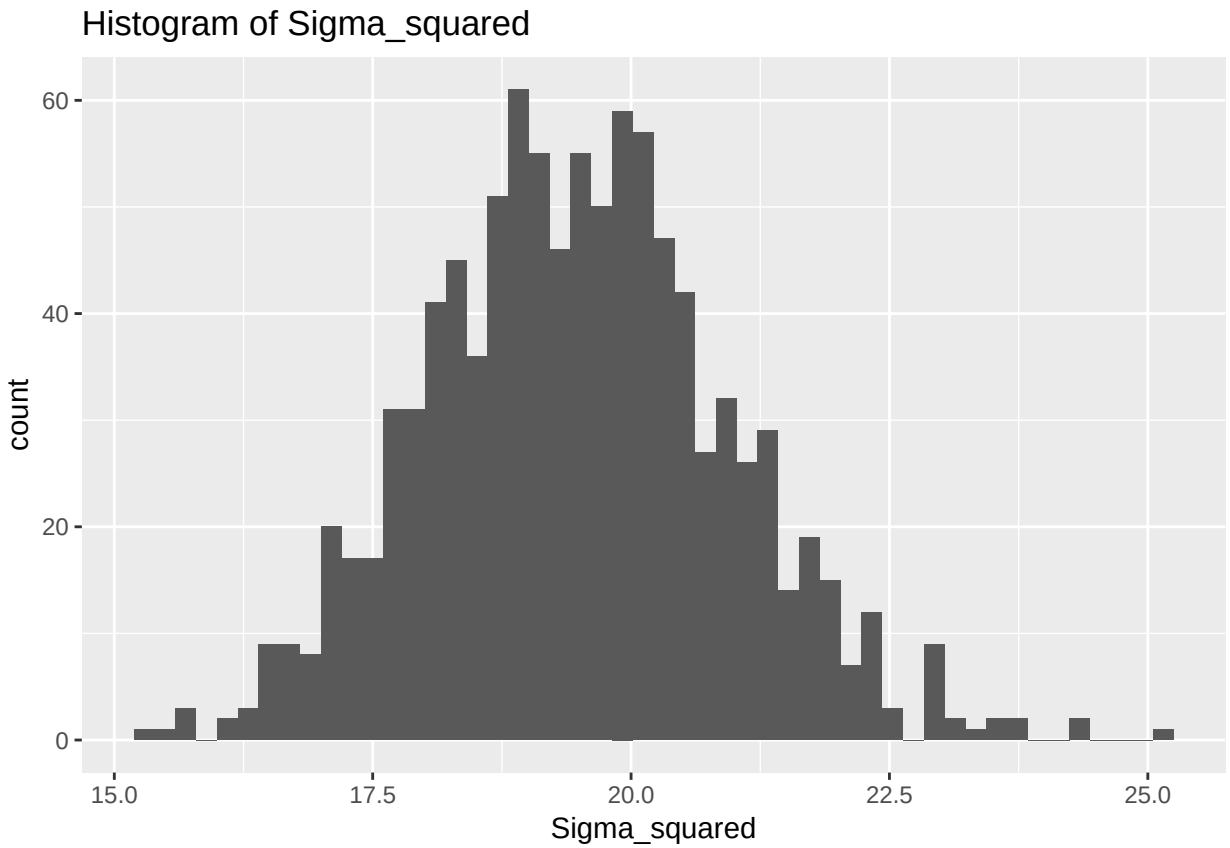

```
ggplot(data=final,aes(Beta_1)) +
  geom_histogram(bins=50) +
  ggtitle("Histogram of Beta_1")
```



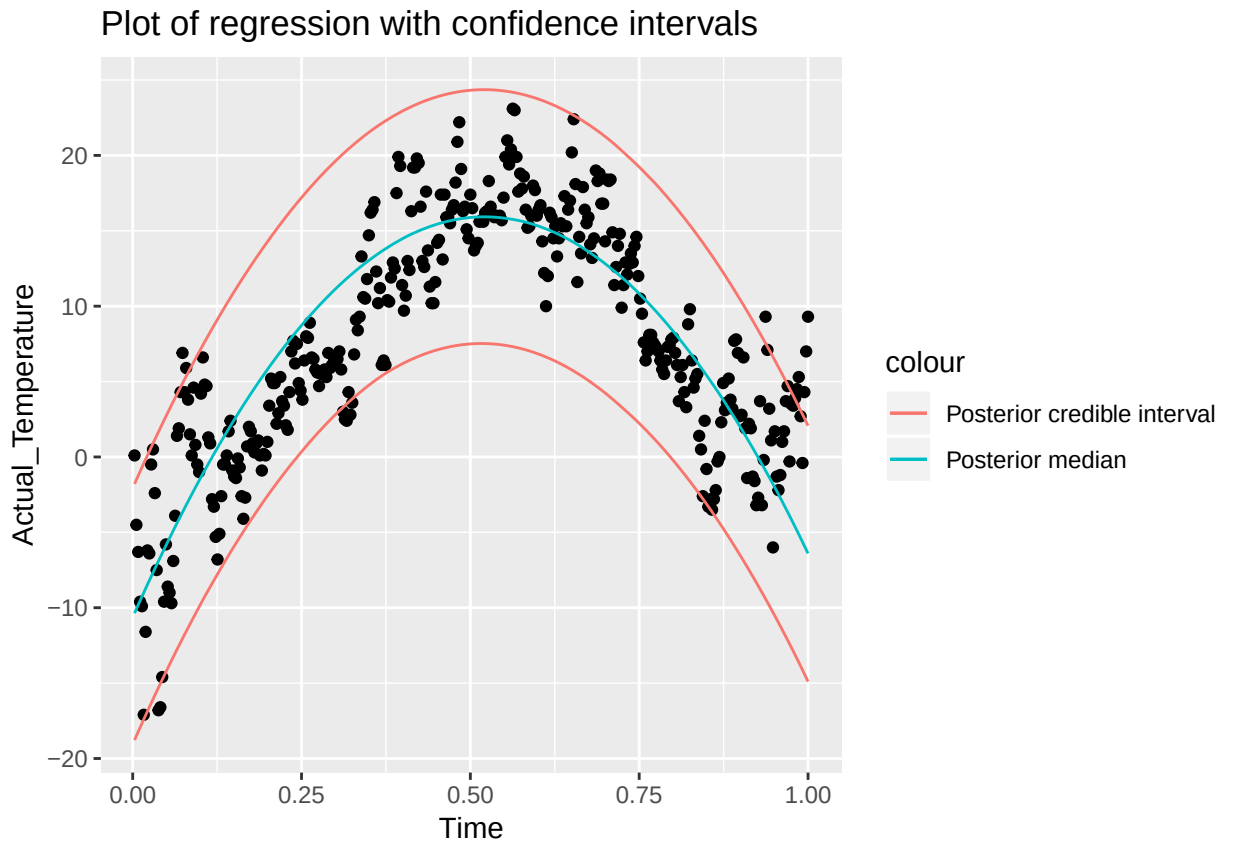

```
ggplot(data=final,aes(Beta_2)) +
  geom_histogram(bins=50) +
  ggtitle("Histogram of Beta_2")
```




```
ggplot(data=final,aes(Sigma_squared)) +
  geom_histogram(bins=50) +
  ggtitle("Histogram of Sigma_squared")
```




```
ggplot(data= predicted_data, aes(x = Time)) + geom_point(aes(y = Actual_Temperature)) +
geom_line(aes(y = temp_hat_median, color = "Posterior median")) +
geom_line(aes(y = temp_hat_l_limit, color = "Posterior credible interval")) +
geom_line(aes(y = temp_hat_u_limit, color = "Posterior credible interval")) +
ggtitle("Plot of regression with confidence intervals")
```



Analysis: No we dont expect 95% of the data inside the confidence band because this is not the prediction band(does not contain the error term)

##c) Simulating from posterior distribution of $\widetilde{x}$

It is of interest to locate the time with the highest expected temperature (that is, the *time* where $f(time)$ is maximal). Let's call this value $\widetilde{x}$. Use the simulations in b) to simulate from the posterior distribution of $\widetilde{x}$. [Hint: the regression curve is a quadratic. You can find a simple formula for $\widetilde{x}$ given β_0, β_1 and β_2.]

Since the given expression is quadratic the first derivate will be zero that is:

$$\begin{aligned} y &= \beta_0 + \beta_1 time + \beta_2 time^2 \\ 0 &= \beta_1 + 2\beta_2 time \\ time &= -0.5 \cdot \frac{\beta_1}{\beta_2} \end{aligned}$$

As evident from the graph this will be close to ~0.50, calculating using the median values we get.

```
cat("The highest expected temperature is:", - 0.5 * mean(final$Beta_1/final$Beta_2))
```

```
## The highest expected temperature is: 0.521753
```

##d) Suggesting prior to estimate a high-order polynomial model

Say now that you want to estimate a polynomial model of order 7, but you suspect that higher order terms may not be needed, and you worry about overfitting. 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 Ω_0 in a smart way.]

This is equivalent of using regularisation, the posterior where Beta are given by

Lasso:

$$\beta_i|\sigma^2 \sim Laplace(0, \frac{\sigma^2}{\lambda})$$

or simple ridge given by:

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

Question 2: Posterior approximation for classification with logistic regression

##a) Implementing logistic regression model

Consider the logistic regression

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

where y is the binary variable with y = 1 if the woman works and y = 0 if she does not. x is a 8-dimensional vector containing the eight features (including a one for the constant term that models the intercept). Fit the logistic regression using maximum likelihood estimation by the command: *glmModel <- glm(Work 0 + ., data = WomenWork, family = binomial).* Note how I added a zero in the model formula so that R doesn't add an extra intercept (we already have an intercept term from the Constant feature). Note also that a dot (.) in the model formula means to add all other variables in the dataset as features. family = binomial tells R that we want to fit a logistic regression.

```
set.seed(12345)
data <- read.csv("WomenWork.csv")
model <- glm(Work~0 +., data = data, family = binomial)
summary(model)
```

```
##
## Call:
## glm(formula = Work ~ 0 + ., family = binomial, data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1666  -0.9298   0.4391   0.9491   2.0579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## Constant     0.64433    1.52306   0.423 0.672258
## HusbandInc  -0.01982    0.01591  -1.246 0.212743
## EducYears    0.18000    0.07914   2.274 0.022938 *
## ExpYears     0.16737    0.06593   2.539 0.011130 *
## ExpYears2   -0.14380    0.23551  -0.611 0.541474
## Age         -0.08234    0.02699  -3.050 0.002286 **
## NSmallChild -1.36248    0.38996  -3.494 0.000476 ***
## NBigChild   -0.02541    0.14171  -0.179 0.857707
## ---
```

```
## 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
```

b) Approximating the posterior distribution of β with a multivariate normal distribution

Now the fun begins. Our goal is to approximate the posterior distribution of the 8-dim parameter vector β with a multivariate normal distribution

$$\beta|y, X \sim N\left(\widetilde{\beta}, J_y^{-1}(\widetilde{\beta})\right)$$

where $\widetilde{\beta}$ is the posterior mode and $J(\widetilde{\beta}) = -\frac{\partial^2 lnp(\beta|y)}{\partial\beta\partial\beta^T}|_{\beta=\widetilde{\beta}}$ is the observed Hessian evaluated at the posterior mode. Note that $\frac{\partial^2 lnp(\beta|y)}{\partial\beta\partial\beta^T}$ is an 8x8 matrix with second derivatives on the diagonal and cross-derivatives $\frac{\partial^2 lnp(\beta|y)}{\partial\beta_i\partial\beta_j}$ on the offdiagonal. It is actually not hard to compute this derivative by hand, but don't worry, we will let the computer do it numerically for you. Now, both $\widetilde{\beta}$ and $J(\widetilde{\beta})$ are computed by the optim function in R. See my code https://github.com/mattiasvillani/BayesLearnCourse/raw/master/Code/MainOptimizeSpam.zip where I have coded everything up for the spam prediction example (it also does probit regression, but that is not needed here). I want you to implement you own version of this. You can use my code as a template, but I want you to write your own file so that you understand every line of your code. Don't just copy my code. Use the prior $\beta \sim N(0, \tau^2 I)$, with $\tau = 10$. Your report should include your code as well as numerical values for $\widetilde{\beta}$ and $J_y^{-1}(\widetilde{\beta})$ for the WomenWork data. Compute an approximate 95 percent credible interval for the variable NSmallChild. Would you say that this feature is an important determinant of the probability that a women works?

Lets derive the formula for the log-likelihood

In our data 'Work' is a binary function thus

$$P(Work = 1|x_i) = \frac{exp(X^T \cdot \beta)}{1 + exp(X^T \cdot \beta)}$$
$$P(Work = 0|x_i) = \frac{1}{1 + exp(X^T \cdot \beta)}$$

$$Since\ P(Work = 1|x_i) + P(Work = 0|x_i) = 1$$

Loss function is cross entropy loss function

$$Loss = y_i \cdot \log P(Work = 1|x_i) + (1 - y_i) \cdot \log P(Work = 0|x_i)$$

Likelihood:

$$p(Y|x,\beta) = \sum_{i=1}^{n=N} y_i \cdot \log(P(Work = 1|x_i)) + (1-y_i) \cdot \log(P(Work = 0|x_i))$$

$$p(Y|x,\beta) = \sum_{i=1}^{n=N} y_i \cdot \log(\frac{exp(X^T \cdot \beta)}{1+exp(X^T \cdot \beta)}) + (1-y_i) \cdot \log(\frac{1}{1+exp(X^T \cdot \beta)})$$

$$p(Y|x,\beta) = \sum_{i=1}^{n=N} y_i \cdot \log(\frac{1}{1+exp(-X^T \cdot \beta)}) + (1-y_i) \cdot \log(\frac{1}{1+exp(X^T \cdot \beta)})$$

$$p(Y|x,\beta) = \sum_{i=1}^{n=N} y_i \cdot [\log(\frac{1}{1+exp(-X^T \cdot \beta)}) - \log(\frac{1}{1+exp(X^T \cdot \beta)})] + \log(\frac{1}{1+exp(X^T \cdot \beta)})$$

$$p(Y|x,\beta) = \sum_{i=1}^{n=N} y_i \cdot [\log(\frac{\frac{1}{1+exp(-X^T \cdot \beta)})}{(\frac{1}{1+exp(X^T \cdot \beta)})}] + \log(\frac{1}{1+exp(X^T \cdot \beta)})$$

$$p(Y|x,\beta) = \sum_{i=1}^{n=N} y_i \cdot log(exp(-X^T \cdot \beta)) + \log(1) - \log(1+exp(X^T \cdot \beta))$$

$$p(Y|x,\beta) = \sum_{i=1}^{n=N} y_i \cdot X^T \cdot \beta - \log(1+exp(X^T \cdot \beta))$$

Further more

In the next step, the goal is to approximate the posterior distribution of β with a multivariate normal distribution

$$\beta|y,X \sim N\left(\widetilde{\beta}, J_y^{-1}(\widetilde{\beta})\right)$$

where $\widetilde{\beta}$ is the posterior mode and $J(\widetilde{\beta}) = -\frac{\partial^2 lnp(\beta|y)}{\partial\beta\partial\beta^T}|_{\beta=\widetilde{\beta}}$ is the observed Hessian evaluated at the posterior mode. Both $\widetilde{\beta}$ and $J(\widetilde{\beta})$ are computed by the `optim`-function in R. We use the prior $\beta \sim N(0, \tau^2 I)$ with $\tau = 10$.

```
set.seed(12345)
# creating matrix for calculation
X <- data[,c("Constant", "HusbandInc", "EducYears", "ExpYears",
             "ExpYears2", "Age", "NSmallChild", "NBigChild" )] %>% as.matrix()
Y <- data[,c("Work")] %>% as.matrix()

# Prior.
tau = 10
mu_0 = rep(0, ncol(X))
sigma_0 = tau^2*diag(ncol(X))
beta_init = rep(0, ncol(X))

log_posterior <- function(betaVect,Y,X,mu,sigma){

  nPara <- length(betaVect);
  linPred <- X %*% betaVect;

  # evaluating the log-likelihood
  logLik <- sum( linPred*Y -log(1 + exp(linPred)));
  if (abs(logLik) == Inf) logLik = -20000; # Likelihood is not finite, stear the optimizer away from he
```

```
  # evaluating the prior
  logPrior <- dmvnorm(betaVect, matrix(0,nPara,1), sigma, log=TRUE);

  # add the log prior and log-likelihood together to get log posterior
  return(logLik + logPrior)
}

results_optim = optim(par = beta_init,
                      fn = log_posterior,
                      Y = Y,
                      X = X,
                      mu = mu_0,
                      sigma = sigma_0,
                      method=c("BFGS"),
                      # Multiplying objective function by -1 to find maximum instead of minimum.
                      control=list(fnscale=-1),
                      hessian=TRUE)
```

The optim-function returns the mode for every β, $\widetilde{\beta}$. Since the function returns $-J_y(\widetilde{\beta})$, we still need to transform this matrix to get the desired $J_y^{-1}(\widetilde{\beta})$ before printing.

```
# Printing results.
  # Beta_mode.
  knitr::kable(
    data.frame(
      beta_ = seq(from = 0,to = length(results_optim$par)-1),
      posterior_mode = results_optim$par),
    caption = "Numerical values for beta_mode")
```

Table 1: Numerical values for beta_mode

beta_	posterior_mode
0	0.6267583
1	-0.0198388
2	0.1803346
3	0.1674284
4	-0.1440383
5	-0.0820616
6	-1.3591145
7	-0.0246623

```
# Adjusted hessian. # Posterior covariance (J^-1(beta hat))
print("Adjusted hessian:")
```

```
## [1] "Adjusted hessian:"
```

```
hessian =  -solve(results_optim$hessian)
hessian
```

```
##              [,1]           [,2]            [,3]           [,4]
## [1,]  2.265999213  0.00333682417 -0.065449232806 -0.01177765557
## [2,]  0.003336824  0.00025290866 -0.000561152701 -0.00003146443
## [3,] -0.065449233 -0.00056115270  0.006218404726 -0.00035435284
## [4,] -0.011777656 -0.00003146443 -0.000354352838  0.00434298028
```

```
## [5,]  0.045717513  0.00014228726  0.001889764740 -0.01421143177
## [6,] -0.030294134 -0.00003586266 -0.000003319259 -0.00013398170
## [7,] -0.188769159  0.00050695295 -0.006135548472 -0.00146499283
## [8,] -0.098022329 -0.00014422783  0.001752241397  0.00054447141
##               [,5]            [,6]         [,7]          [,8]
## [1,]  0.0457175134 -0.030294134435 -0.188769159 -0.0980223288
## [2,]  0.0001422873 -0.000035862665  0.000506953 -0.0001442278
## [3,]  0.0018897647 -0.000003319259 -0.006135548  0.0017522414
## [4,] -0.0142114318 -0.000133981701 -0.001464993  0.0005444714
## [5,]  0.0554187693 -0.000330219635  0.003192093  0.0005086101
## [6,] -0.0003302196  0.000718477547  0.005184373  0.0010952774
## [7,]  0.0031920930  0.005184372595  0.151262310  0.0067689298
## [8,]  0.0005086101  0.001095277394  0.006768930  0.0199714842
```

```
approx_post_std_dev = sqrt(diag(hessian))

  knitr::kable(approx_post_std_dev, caption="Observed hessian evaluated at the posterior mode")
```

Table 2: Observed hessian evaluated at the posterior mode

x
1.5053236
0.0159031
0.0788569
0.0659013
0.2354119
0.0268044
0.3889246
0.1413205

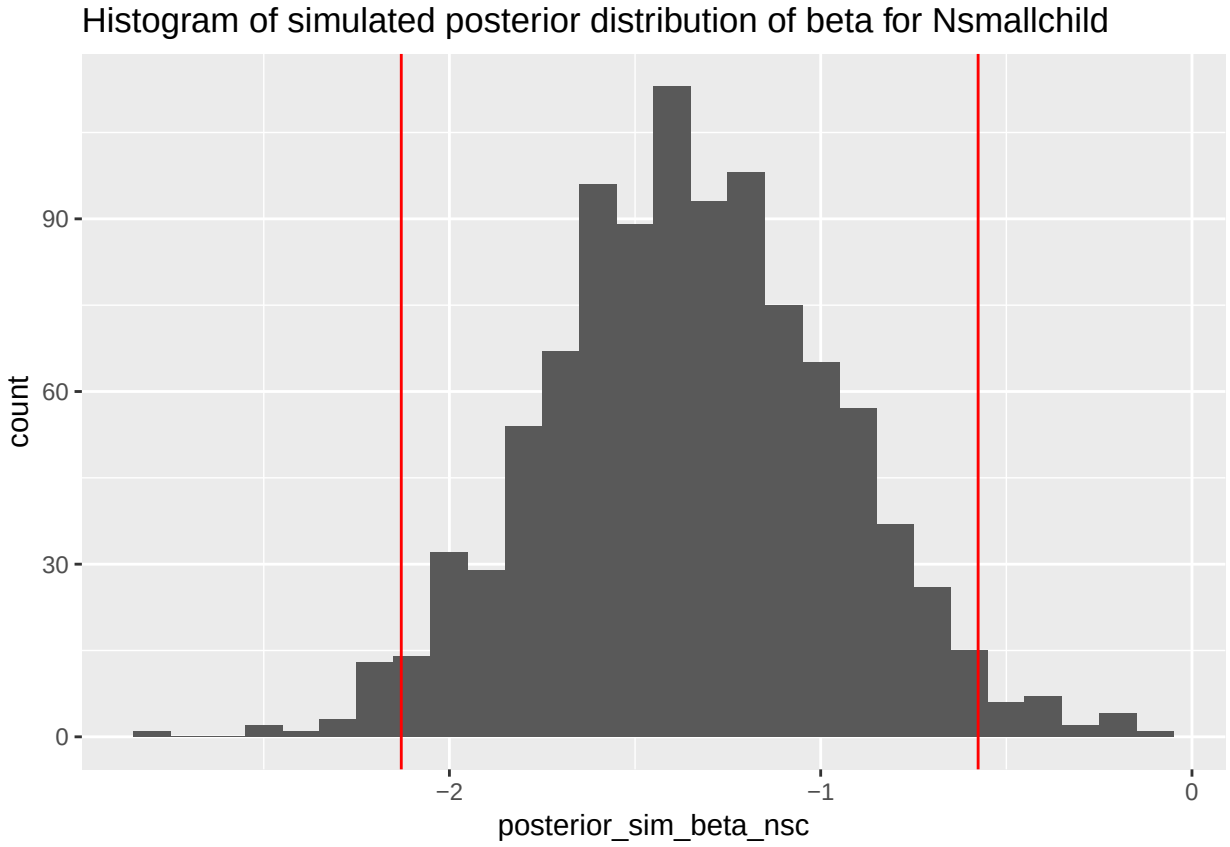
To compute an approximate 95% credible interval for the coefficients of the variable *NSmallChild*, we first need to simulate from the posterior. We do this by simulating from the posterior

$$\beta|y, X \sim N\left(\widetilde{\beta}, J_y^{-1}(\widetilde{\beta})\right)$$

by usage of our obtained parameters $\widetilde{\beta}$ and $J_y^{-1}(\widetilde{\beta})$.

```
set.seed(12345)
# Simulating 1000 times from approximated posterior.
posterior_sim_beta_all = rmvnorm(n = 1000,
                                 mean = results_optim$par,
                                 sigma = -solve(results_optim$hessian))

# Extracting simulated posterior beta values for variable NSmallChild.
posterior_sim_beta_nsc = posterior_sim_beta_all[, 7]

# Computing 95% credible interval bounds.
interval_bounds = quantile(x = posterior_sim_beta_nsc,
                           probs = c(0.025, 0.975))

# Plotting simulated beta values with 95% credible interval for variable NSmallChild.
ggplot() +
```

```
  geom_histogram(aes(x = posterior_sim_beta_nsc),
                 binwidth = 0.1) +
  geom_vline(xintercept = interval_bounds[1],
             color = "red") +
  geom_vline(xintercept = interval_bounds[2],
             color = "red") +
  ggtitle("Histogram of simulated posterior distribution of beta for Nsmallchild")
```




```
# Printing interval bounds.
knitr::kable(as.data.frame(interval_bounds), caption = "Computed credible interval bounds")
```

Table 3: Computed credible interval bounds

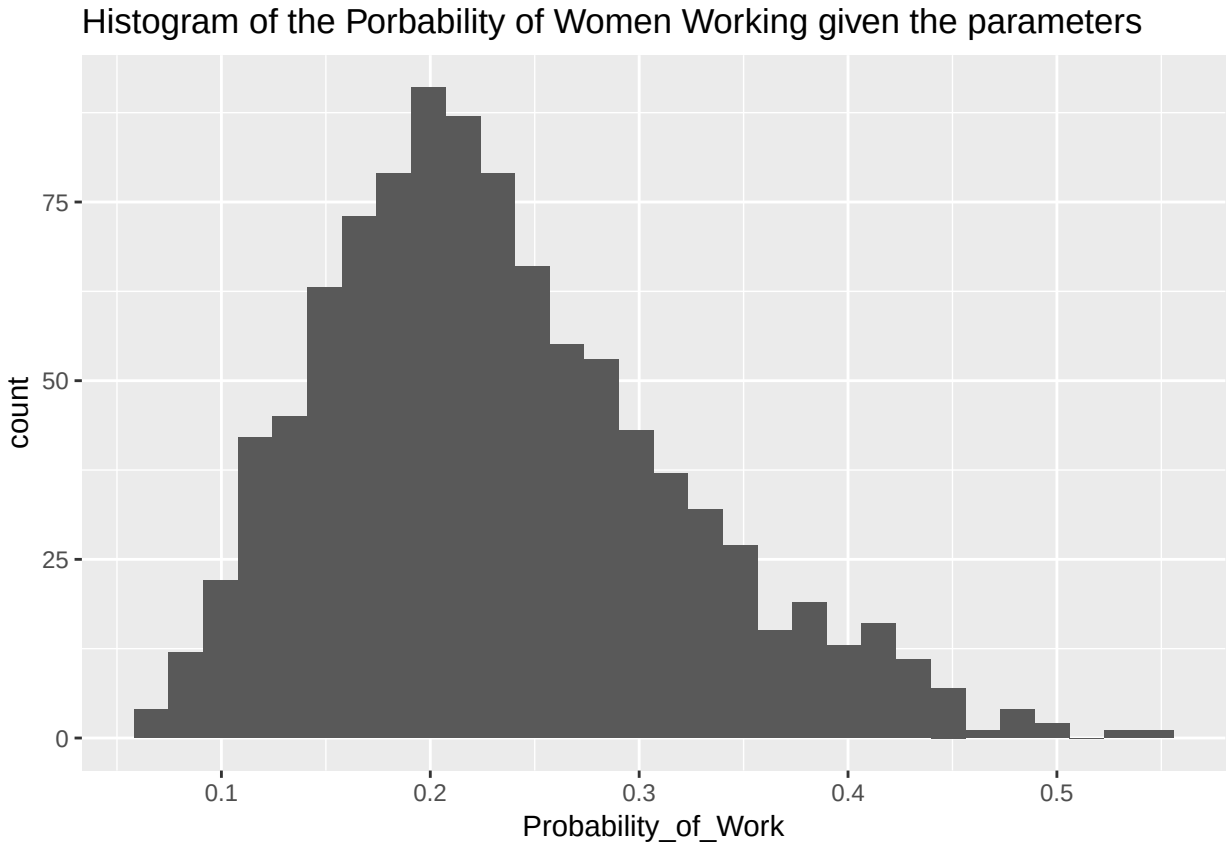
	interval_bounds
2.5%	-2.1296565
97.5%	-0.5759335

##c) Simulating from the predictive distribution of the response variable in a logistic regression

Write a function that simulates from the predictive distribution of the response variable in a logistic regression. Use your normal approximation from 2(b). Use that function to simulate and plot the predictive distribution for the Work variable for a 40 year old woman, with two children (3 and 9 years old), 8 years of education, 10 years of experience. and a husband with an income of 10. [Hint: the R package mvtnorm will again be handy. And remember my discussion on how Bayesian prediction can be done by simulation.]

To perform a simulation from the predictive distribution of the response variable in a logistic regression, we use the drawn posterior β-coefficiencts (`posterior_sim_beta_all`) from *2b)* to predict the *Work*-value. For every drawn β-vector, we calculate $Pr(y = 1|x) = \frac{exp(x^T\beta)}{1+exp(x^T\beta)}$. Since the goal is to predict the *Work*-value for 40-year-old woman with two children (3 and 9 years old), 8 years of education, 10 years of experience and a husband with an income of 10, we use this data for every different drawn β-vector.

```
set.seed(12345)
# Implementing specified data.
X = matrix(c(Constant = 1,
             HusbandInc = 10,
             EducYears = 8,
             ExpYears = 10,
             ExpYears2 = 1.00,
             Age = 40,
             NSmallChild = 1,
             NBigChild = 1),
           nrow = 1)

X_T <- t(X)

colnames(posterior_sim_beta_all) <- c("Constant", "HusbandInc", "EducYears",
                                      "ExpYears", "ExpYears2", "Age", "NSmallChild", "NBigChild")

# Using all drawn posterior beta coeffiecients and specified data vector
# to compute distribution for Pr(y=1|x).
predictive_sim_work = c()
for(sim in 1:nrow(posterior_sim_beta_all)) {
    num <- exp(posterior_sim_beta_all[sim,] %*% X_T)
  predictive_sim_work[sim] <- num/(1+num)
}

predictive_sim_work = predictive_sim_work %>% as.data.frame()
colnames(predictive_sim_work) <- c("Probability_of_Work")

# Plotting histogram of predictive distribution.
ggplot(data=predictive_sim_work, aes(Probability_of_Work)) +
geom_histogram(bins = 30) +
ggtitle("Histogram of the Porbability of Women Working given the parameters")
```



#Appendix

```
knitr::opts_chunk$set(echo = TRUE)
options(scipen=999)
set.seed(12345)

library("ggplot2")
library("tidyverse")
library("LaplacesDemon")
library("mvtnorm")
library("Hmisc")

# The palette with black:
cbbPalette <- c("#000000", "#E69F00", "#56B4E9", "#009E73",
                "#F0E442", "#0072B2", "#D55E00", "#CC79A7")
set.seed(12345)

temp_data = read.delim("TempLinkoping2016.txt")
temp_data$time_square = (temp_data$time)^2
temp_data$intercept = 1

# Initialzing hyper-parameters
mu_0 = t(c(-10,100,-100))
omega_0 = diag(x=0.01,nrow=3,ncol=3)
nu_0 = 4
sigma_sq_0 = 1
```

```r
# calculating X and other vectors
X = temp_data[,c("intercept", "time", "time_square")] %>% as.matrix()
X_T = t(X)
Y = temp_data[,c("temp")] %>% as.matrix()
Y_T = t(Y)

# defining a function to sample and predict temperature
quad_regression <- function(mu_0, omega_0, nu_0, sigma_sq_0,X,Y){
# sampling sigma squares using inverse chi-square
x = rchisq(n = 1, df = nu_0)
# Calculating sigma_sq.
sigma_sq = (nu_0*sigma_sq_0)/x
beta <- mvtnorm::rmvnorm(n = 1 , mean=mu_0, sigma=sigma_sq * solve(omega_0))
final <-  X %*% t(beta) + rnorm(n=1,mean = 0, sd=sqrt(sigma_sq))
colnames(final) <- c("Predicted_temperature")
return(final)
}

# Define a named list of parameter values
gs = list(nu_0 = seq(1,10,1),
          sigma_sq_0 = seq(0,5,0.5)) %>%
  cross_df() # Convert to data frame grid

final = gs %>% mutate(predicted_temperature = pmap(gs, mu_0 = mu_0,
                                                   omega_0 = omega_0, X=X, Y=Y, quad_regression),
                      actual_temperature = list(temp_data$temp),
                      time = list(temp_data$time)) %>% as_tibble()

final = unnest(final, predicted_temperature, actual_temperature, time)

# Using cut
final$sigma_sq_0_cut <- cut2(final$sigma_sq_0, m=5)
final$nu_0_cut <- cut2(final$nu_0, m=5)

# plotting
final %>% filter(predicted_temperature <30 & predicted_temperature >-25)  %>%
  ggplot(aes(x=time, y = predicted_temperature, color=nu_0)) +
  geom_point() +
  geom_line(aes(y=actual_temperature), color="red") +
  facet_wrap(~nu_0_cut) +
  ggtitle("redicted Temperature vs. Time By Varying Nu_0, with true curve in red")


final %>% filter(predicted_temperature <50 & predicted_temperature >-50)  %>%
  ggplot(aes(x=time, y = predicted_temperature, color=sigma_sq_0)) +
  geom_point() +
  geom_line(aes(y=actual_temperature), color="red") +
    facet_wrap(~sigma_sq_0_cut) +
  ggtitle("Predicted Temperature vs. Time By Varying Sigma square, with true curve in red")

set.seed(12345)

# Initialzing final hyper-parameters
```

```
mu_0 = t(c(-10,100,-100))
omega_0 = diag(x=50,nrow=3,ncol=3)
nu_0 = 10
sigma_sq_0 = 0.25

best_fit <- NULL
for(i in 1:100){
temp <- quad_regression(mu_0=mu_0, omega_0=omega_0, nu_0=nu_0, sigma_sq_0=sigma_sq_0,X=X,Y=Y)
temp <- temp %>% as.data.frame() %>% mutate(Actual_temperature = temp_data$temp, Time = temp_data$time)
best_fit <- rbind(temp, best_fit)
}

# plotting
best_fit %>%
  ggplot(aes(x=Time, y = Predicted_temperature)) +
  geom_point() +
  geom_line(aes(y=Actual_temperature), color="red") +
  ggtitle("Predicted and Actual Temperature(shown in red) vs. Time")
set.seed(12345)
# initialized values from above
mu_0 = c(-10,100,-100)
mu_0_T = t(mu_0)
omega_0 = diag(x=50,nrow=3,ncol=3)
nu_0 = 10
sigma_sq_0 = 0.25

# calculate beta hat given by beta_hat =  (X^{T} X)^{-1} X^T Y
beta_hat = solve(X_T %*% X) %*% X_T %*% Y

# calculate mu_n
mu_n = solve(X_T %*% X + omega_0) %*% (X_T %*% X %*% beta_hat + omega_0 %*% mu_0)

# calculate omega_n given by Omega_n = X^T X + \Omega_0
omega_n = X_T %*% X + omega_0

# calculate nu_n given by nu_0 + n
nu_n = NROW(X) + nu_0

# calculate sigma_sq_n given by (nu_0 * sigma_sq_0 + (Y %*% Y_T + mu_0^T %*% omega_0 %*% mu_0 - mu_n^T
sigma_sq_n = (nu_0 * sigma_sq_0 + Y_T %*% Y + mu_0_T %*% omega_0 %*% mu_0 - t(mu_n) %*% omega_n %*% mu_

# posterior distribution given by Beta = N(\mu_n, \sigma^2 \Omega_n^{-1})

final <- NULL
temp <- NULL
temp2 <- NULL
predicted_data <- NULL
for(i in 1:1000){
#\sigma^2 =  Inv-\chi^2(nu_n, \sigma^2_n)
x = rchisq(n = 1, df = nu_n)
# Calculating sigma_sq.
sigma_sq = as.vector((nu_n*sigma_sq_n)/x)
betas <- mvtnorm::rmvnorm(n = 1 , mean=mu_n, sigma=sigma_sq * solve(omega_n))
```

```
temp2 <-  X %*% t(betas) + rnorm(n=1,mean = 0, sd=sqrt(sigma_sq)) # add the error term you get predicti
temp2 <- cbind(temp2, X, Y)
predicted_data <- rbind(temp2, predicted_data)
temp <- cbind(betas,sigma_sq)
final <- rbind(temp,final)
}

colnames(final) <- c("Beta_0", "Beta_1", "Beta_2", "Sigma_squared")
colnames(predicted_data) <- c("Predicted_Temperature", "Intercept", "Time", "Time_square", "Actual_Temp
final = final %>% as.data.frame()

# calculation of the 95% confidence interval
predicted_data = predicted_data %>% as.data.frame() %>%
group_by(Actual_Temperature, Time) %>%
summarise(temp_hat_median = median(Predicted_Temperature),
temp_hat_l_limit = quantile(x = Predicted_Temperature, probs = 0.025),
temp_hat_u_limit = quantile(x = Predicted_Temperature, probs = 0.975))

ggplot(data=final,aes(Beta_0)) +
  geom_histogram(bins=50) +
  ggtitle("Histogram of Beta_0")

ggplot(data=final,aes(Beta_1)) +
  geom_histogram(bins=50) +
  ggtitle("Histogram of Beta_1")

ggplot(data=final,aes(Beta_2)) +
  geom_histogram(bins=50) +
  ggtitle("Histogram of Beta_2")

ggplot(data=final,aes(Sigma_squared)) +
  geom_histogram(bins=50) +
  ggtitle("Histogram of Sigma_squared")

ggplot(data= predicted_data, aes(x = Time)) + geom_point(aes(y = Actual_Temperature)) +
geom_line(aes(y = temp_hat_median, color = "Posterior median")) +
geom_line(aes(y = temp_hat_l_limit, color = "Posterior credible interval")) +
geom_line(aes(y = temp_hat_u_limit, color = "Posterior credible interval")) +
ggtitle("Plot of regression with confidence intervals")

cat("The highest expected temperature is:", - 0.5 * mean(final$Beta_1/final$Beta_2))
set.seed(12345)
data <- read.csv("WomenWork.csv")
model <- glm(Work~0 +., data = data, family = binomial)
summary(model)

set.seed(12345)
# creating matrix for calculation
X <- data[,c("Constant", "HusbandInc", "EducYears", "ExpYears",
             "ExpYears2", "Age", "NSmallChild", "NBigChild" )] %>% as.matrix()
Y <- data[,c("Work")] %>% as.matrix()

# Prior.
```

```r
tau = 10
mu_0 = rep(0, ncol(X))
sigma_0 = tau^2*diag(ncol(X))
beta_init = rep(0, ncol(X))

log_posterior <- function(betaVect,Y,X,mu,sigma){

  nPara <- length(betaVect);
  linPred <- X %*% betaVect;

  # evaluating the log-likelihood
  logLik <- sum( linPred*Y -log(1 + exp(linPred)));
  if (abs(logLik) == Inf) logLik = -20000; # Likelihood is not finite, stear the optimizer away from he

  # evaluating the prior
  logPrior <- dmvnorm(betaVect, matrix(0,nPara,1), sigma, log=TRUE);

  # add the log prior and log-likelihood together to get log posterior
  return(logLik + logPrior)
}

results_optim = optim(par = beta_init,
                      fn = log_posterior,
                      Y = Y,
                      X = X,
                      mu = mu_0,
                      sigma = sigma_0,
                      method=c("BFGS"),
                      # Multiplying objective function by -1 to find maximum instead of minimum.
                      control=list(fnscale=-1),
                      hessian=TRUE)


# Printing results.
  # Beta_mode.
  knitr::kable(
    data.frame(
      beta_ = seq(from = 0,to = length(results_optim$par)-1),
      posterior_mode = results_optim$par),
    caption = "Numerical values for beta_mode")


# Adjusted hessian. # Posterior covariance (J^-1(beta hat))
print("Adjusted hessian:")
hessian =  -solve(results_optim$hessian)
hessian
approx_post_std_dev = sqrt(diag(hessian))

  knitr::kable(approx_post_std_dev, caption="Observed hessian evaluated at the posterior mode")

set.seed(12345)
# Simulating 1000 times from approximated posterior.
posterior_sim_beta_all = rmvnorm(n = 1000,
```

```
                                  mean = results_optim$par,
                                  sigma = -solve(results_optim$hessian))

# Extracting simulated posterior beta values for variable NSmallChild.
posterior_sim_beta_nsc = posterior_sim_beta_all[, 7]

# Computing 95% credible interval bounds.
interval_bounds = quantile(x = posterior_sim_beta_nsc,
                           probs = c(0.025, 0.975))

# Plotting simulated beta values with 95% credible interval for variable NSmallChild.
ggplot() +
  geom_histogram(aes(x = posterior_sim_beta_nsc),
                 binwidth = 0.1) +
  geom_vline(xintercept = interval_bounds[1],
             color = "red") +
  geom_vline(xintercept = interval_bounds[2],
             color = "red") +
  ggtitle("Histogram of simulated posterior distribution of beta for Nsmallchild")


# Printing interval bounds.
knitr::kable(as.data.frame(interval_bounds), caption = "Computed credible interval bounds")
set.seed(12345)
# Implementing specified data.
X = matrix(c(Constant = 1,
             HusbandInc = 10,
             EducYears = 8,
             ExpYears = 10,
             ExpYears2 = 1.00,
             Age = 40,
             NSmallChild = 1,
             NBigChild = 1),
           nrow = 1)

X_T <- t(X)

colnames(posterior_sim_beta_all) <- c("Constant", "HusbandInc", "EducYears",
                                      "ExpYears", "ExpYears2", "Age", "NSmallChild", "NBigChild")

# Using all drawn posterior beta coeffiecients and specified data vector
# to compute distribution for Pr(y=1|x).
predictive_sim_work = c()
for(sim in 1:nrow(posterior_sim_beta_all)) {
    num <- exp(posterior_sim_beta_all[sim,] %*% X_T)
  predictive_sim_work[sim] <- num/(1+num)
}

predictive_sim_work = predictive_sim_work %>% as.data.frame()
colnames(predictive_sim_work) <- c("Probability_of_Work")

# Plotting histogram of predictive distribution.
ggplot(data=predictive_sim_work, aes(Probability_of_Work)) +
```

```
geom_histogram(bins = 30) +
ggtitle("Histogram of the Porbability of Women Working given the parameters")
```