Bayesian learning Lab 2

Mohamed Ali - Mohal954

2023-05-10

Linear and polynomial regression

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

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

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

$$temp = \beta_0 + \beta_1.time\ + e,\ e \sim N(0, \sigma^2)$$

To answer this question a conjugate prior for the linear regression model will be used in which: The joint prior for $\beta\ and\ \sigma^2$:

$$\begin{aligned} \beta|\sigma^2 &\sim\ \mathcal{N}(\mu_0, \sigma^2{\Omega_0}^{-1}) \\ \sigma^2 &\sim Inv - \chi^2(v_0, \sigma^2) \end{aligned}$$

While the posterior:

$$\begin{aligned} \beta|\sigma^2, y &\sim\ \mathcal{N}(\mu_n, \sigma^2{\Omega_n}^{-1}) \\ \sigma^2 &\sim Inv - \chi^2(v_n, {\sigma^2}_n) \\ where: & \\ \mu_n &= (X'X\Omega_0)^{-1}\ (X'X\hat{\beta} + \Omega_0\mu_0) \\ \Omega_n &= X'X + \Omega_0 \\ v_n &= v_0 + n \\ v_n\sigma^2 &= v_n\sigma^2 + (y'y + \mu_0'\Omega_0\mu_0 - \mu_n'\Omega_n\mu_n) \end{aligned}$$

First we start by reading the files and then we Create the covariate_time vaiable as (the number of days sinvce the beginning of the year 365).

```
#reading the files
df<-read_xlsx("Linkoping2022.xlsx")
#Creating the covariate_time vaiable as
#(the number of days sinvce the beginning of the year / 365)
a<- df$datetime
#begunning of the year
b<- '2022-01-01'
a<-format.POSIXlt(strptime(a,'%Y-%m-%d'))
```

```
b<-format.POSIXlt(strptime(b,'%Y-%m-%d'))
#time diff from the
x<-as.vector(difftime(a,b,units='days'))
df$cov_tm<-x/365
```

A

Use the conjugate prior for the linear regression model. The prior hyperparameters $\mu_0, \Omega_0, v_0, \sigma_0^2$ shall be set to sensible values. Start with $\mu_0 = (-10, 100, -100)^T, \Omega_0 = 0.02.I_3, v_0 = 3, \sigma_0^2 = 2$.
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: R package mvtnorm can be used and your $inv - \chi^2$ simulator of random draws from Lab 1.].

We assume to use a conjugate prior from the linear regression in Lec 5 , we have been given the prior hyperparamteres as follow:

```
#We assume to use a conjugate prior from
#the linear regression in Lec 5 ,
#we have been given the prior hyperparamteres as follow:
mu_0= as.matrix(c(0,100,-100),ncol=3)
omega_0=0.01*diag(3)
v_0=1
segma2_0=1
n= length(df$temp)
ndraws=10
```

We have the joint prior for beta and segma2 defined as $\beta|\sigma^2 \sim \mathcal{N}(\mu_0, \sigma^2\Omega_0^{-1})$ and $\sigma^2 \sim Inv - \chi^2(v_o, \sigma_0^2)$ follows Inv-Chi(v0,sigma2_0).

First we draw our random samaple from inv-chi2 using the below defined function from Lec 3 slide 5

```
# Step 1: Draw X folow chi2(n - 1)
draw_chi_sq <- function(n) {
  return(rchisq(1, df = n - 1))
}
# Step 2: Compute sigma2 = (n - 1) * s² / X
compute_sigma_sq <- function(n, segma2_0, X) {
  return((n - 1) * segma2_0 / X)
}

# simulation
segma_estimation <- function(n, mu_0, segma2_0, ndraws) {
  results <- c()

  for (i in 1:ndraws) {
    X <- draw_chi_sq(n)
    sigma_sq <- compute_sigma_sq(n, segma2_0, X)
```

```
    results[i] <- sigma_sq
  }
  return(results)
}

sigma2<-segma_estimation(n, mu_0, segma2_0, ndraws)
```

Now we estimate the betas values using the formula $\beta|\sigma^2 \sim \mathcal{N}(\mu_0, \sigma^2\Omega_0^{-1})$ and we fit the regression based ontemp = beta0 + beta1 time + beta 2 time^2 + erorr.

*Note we have our error follows the normal distrbution by 0 and σ^2.

```
for (i in 1:length(sigma2)) {
  e<- rnorm(1,0,sigma2[i])
  res<-rmvnorm(1,mu_0,sigma2[i]*omega_0)
  temp= x=res[1,1]+res[1,2]*df$cov_tm+res[1,3]*df$cov_tm^2+e
  df[[paste0("temp_fit_",sigma2[i])]]<-temp
}
```

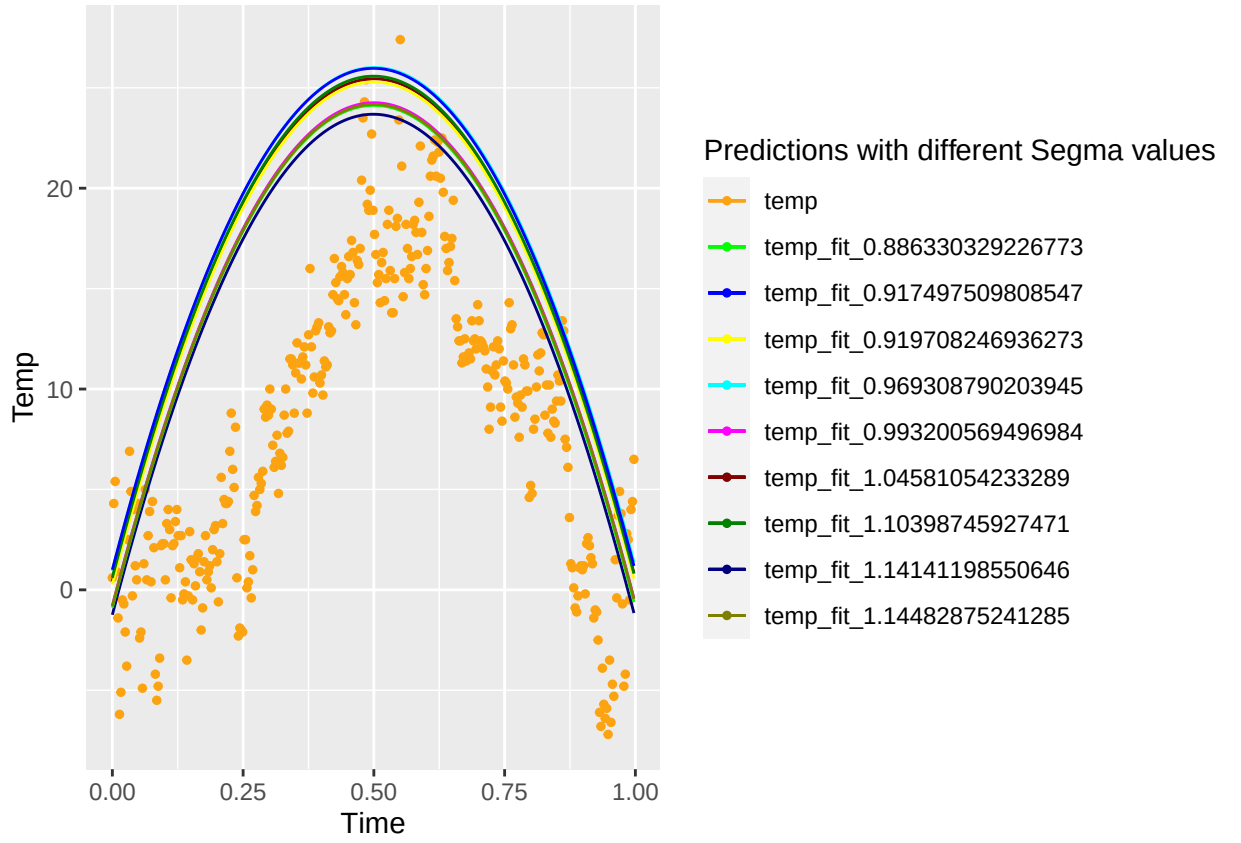
After we done with the draws now for every draw compute the regression curve.

```
## Example function
# x1_grid <- seq(min(X[,2]),max(X[,2]),0.1)
# Mu_draws <- matrix(0,length(x1_grid),2)
# for (ii in 1:length(x1_grid)){
# Curr_x <- c(1,x1_grid[ii],x1_grid[ii]**2,27,27**2,x1_grid[ii]*27)
# CurrMu <- Betas %*% Curr_x
# Mu_draws[ii,] <- quantile(CurrMu,probs=c(0.025,0.975))
# }
# plot(x1_grid,Mu_draws[,1],"n",main="95 % posterior
# probability intervals as a function of x1",
# xlab="x1", ylab="",ylim=c(0,500))
# lines(x1_grid,Mu_draws[,1],col="blue")
# lines(x1_grid,Mu_draws[,2],col="blue")


# Define a vector of colors
colors <-c("#FCA311", "#00FF00", "#0000FF", "#FFFF00", "#00FFFF",
           "#FF00FF", "#800000", "#008000", "#000080", "#808000",
           "#800080", "#008080", "#808080", "#FFC0CB", "#FFA500",
           "#FFD700", "#A52A2A", "#7FFF00", "#FF1493", "#00BFFF")

# Plot with different colored lines
plt <- ggplot(df, aes(x = cov_tm, y = temp)) +
  geom_point(aes(color = factor('temp')), size = 1)

for (i in names(df)[-c(1:4, ncol(df))]) {
  plt <- plt + geom_line(aes_string(y = i, color = factor(i)), linetype = 1)
}
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
```

```
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
# Map colors to the lines
plt <- plt +
  scale_color_manual(values = colors) +
  labs(x = 'Time', y = 'Temp',color='Predictions with different Segma values')
plt
```



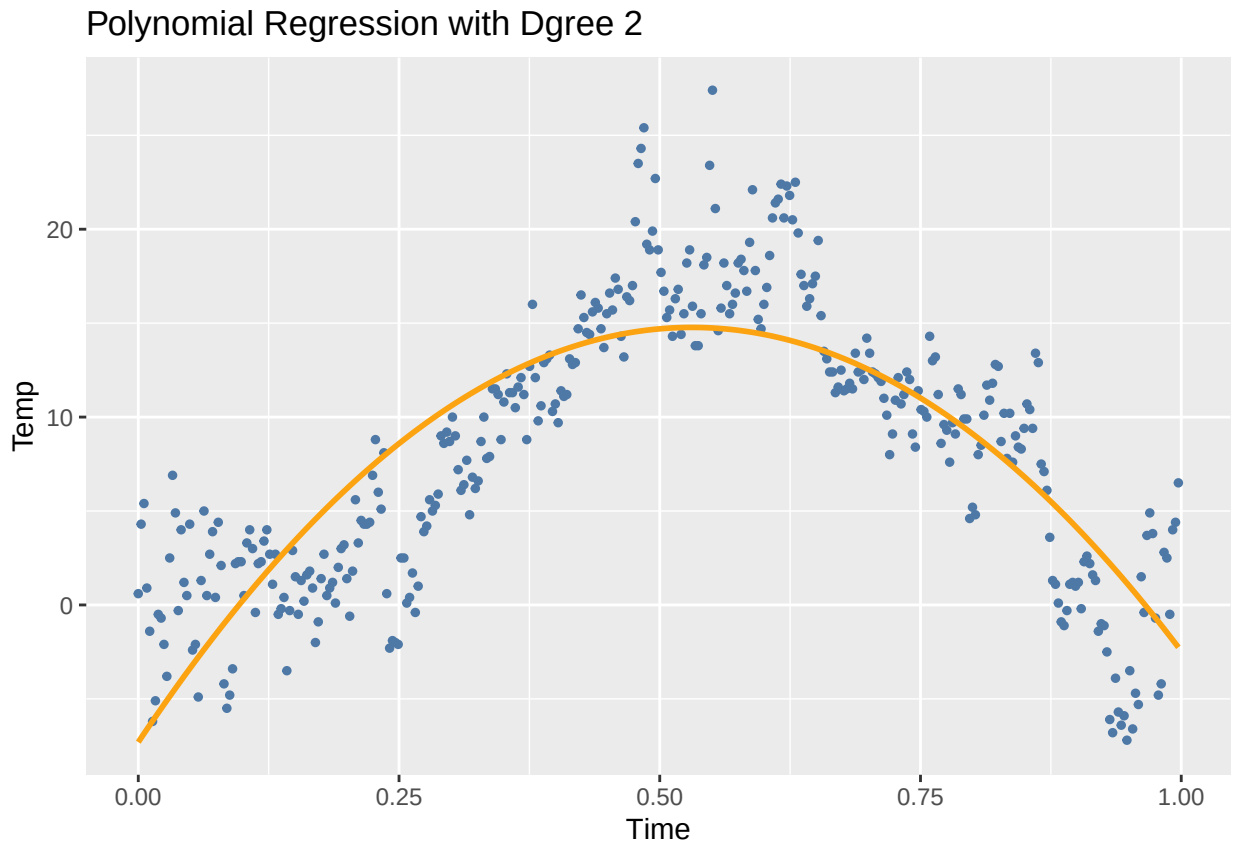
Comparing the above chart with our prior beliefs which we can see by running the polynomial model using the *lm* function in R with df = 2 we can see the fitted regression line with to some extend follow our regression line when σ^2 is equal to *1.037*.

```
degree <- 2  # Set the degree of the polynomial
x= df$cov_tm
y= df$temp
model <- lm(y ~ poly(x, degree, raw = TRUE))
df_plt<- data.frame(x=x,y=y)
z=predict(model)
# Print the model summary
```

```
# Plot the data and regression line
plt <- ggplot(df_plt, aes(x = x, y = y)) +
  geom_point(color = "#4E79A7", size = 1)+
  geom_line(aes(y = z), color = "#FCA311",size=1 ,linetype = 1)+
  labs(x = 'Time', y = 'Temp'
       ,title ='Polynomial Regression with Dgree 2')
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
plt
```



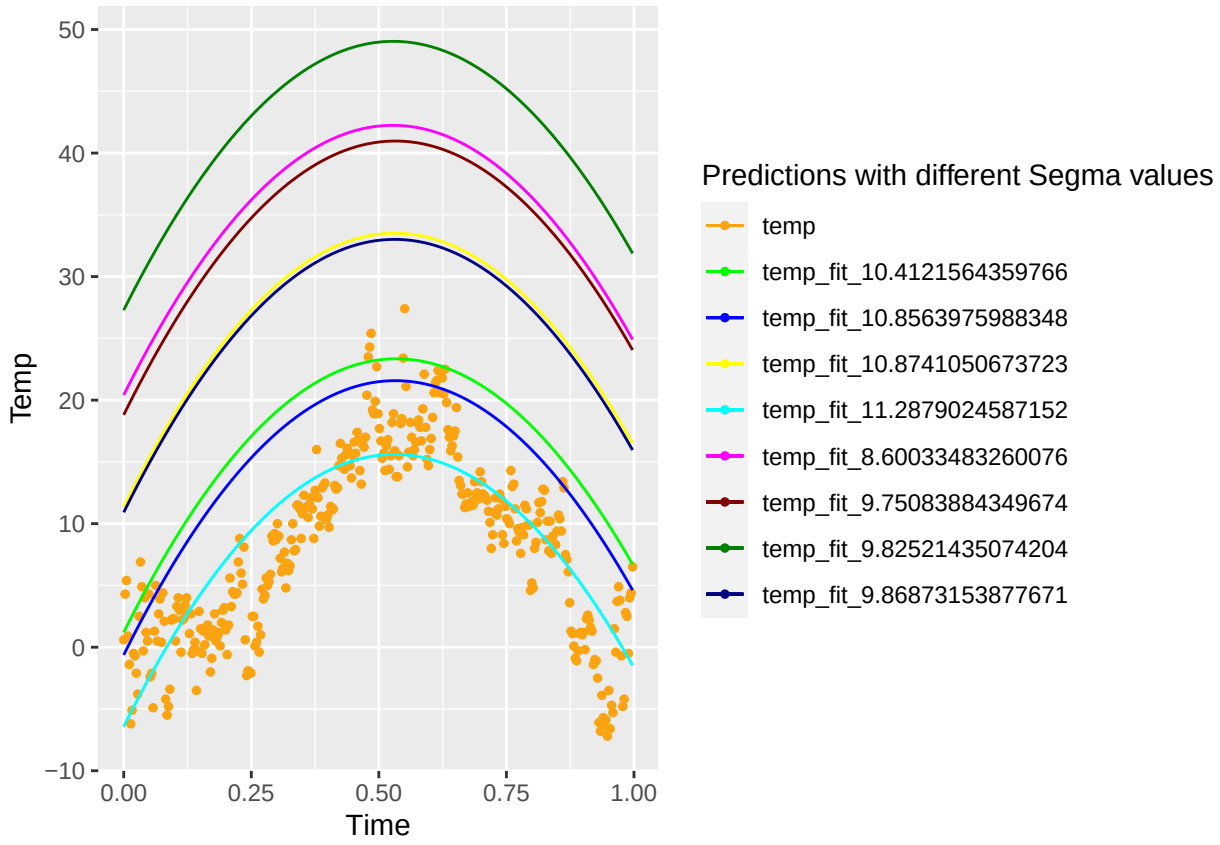

```
summary(model)
```

```
##
## Call:
## lm(formula = y ~ poly(x, degree, raw = TRUE))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -10.6557  -2.8525  -0.1874   2.5052  12.6580
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   -7.3039     0.6582  -11.10   <2e-16 ***
## poly(x, degree, raw = TRUE)1  83.1568     3.0492   27.27   <2e-16 ***
## poly(x, degree, raw = TRUE)2 -78.3093     2.9600  -26.46   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.215 on 362 degrees of freedom
## Multiple R-squared:  0.6726, Adjusted R-squared:  0.6708
## F-statistic: 371.9 on 2 and 362 DF,  p-value: < 2.2e-16
```

Now we draw simulations using the modified μ_0 from the model results we have.

```
mu_news= as.matrix(c(7.3,83.1,-78.3),ncol=3)
segma2_new=10
sigma2<-segma_estimation(n,mu_news, segma2_new, ndraws)

for (i in 1:length(sigma2)) {
  e<- rnorm(1,0,sigma2[i])
  res<-rmvnorm(1,mu_news,sigma2[i]*omega_0)
  temp= x=res[1,1]+res[1,2]*df$cov_tm+res[1,3]*df$cov_tm^2+e
  df[[paste0("temp_fit_",sigma2[i])]]<-temp
}

# Define a vector of colors
colors <-c("#FCA311", "#00FF00", "#0000FF", "#FFFF00", "#00FFFF",
          "#FF00FF", "#800000", "#008000", "#000080", "#808000",
         "#800080", "#008080", "#808080", "#FFC0CB", "#FFA500",
        "#FFD700", "#A52A2A", "#7FFF00", "#FF1493", "#00BFFF")

# Plot with different colored lines
plt <- ggplot(df, aes(x = cov_tm, y = temp)) +
  geom_point(aes(color = factor('temp')), size = 1)

for (i in names(df)[-c(1:15, ncol(df))]) {
  plt <- plt + geom_line(aes_string(y = i, color = factor(i)), linetype = 1)
}

# Map colors to the lines
plt <- plt +
  scale_color_manual(values = colors) +
  labs(x = 'Time', y = 'Temp',color='Predictions with different Segma values')
plt
```



B

Write a function that simulate draws from the joint posterior distribution of $\beta_0, \beta_1, \beta_2, \sigma^2$. i- Plot a histogram for each marginal posterior of the parameters.
ii- Make a scatter plot of the temperature data and overlay a curve for the posterior median of the regression function $f(time) = E[temp|time] = \beta_0 + \beta_1.time + \beta_2.time^2$, 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?

As we want to estimate the uncertainty in the model parameters. Simulating from the joint posterior allows us to obtain a set of plausible values for all the parameters in the model, taking into account the observed data and prior information. to do so we have our non-informative prior:

$$p(\beta, \sigma^2) \propto \sigma^-1$$

Our joint posterior of β and σ^2:

$$\begin{aligned}
\beta|\sigma^2, y &\sim \mathcal{N}(\hat{\beta}, \sigma^2(X\text{t}X)^{-1}) \\
\sigma^2|y &\sim Inv - \chi^{\in}(n-k, s^2) \\
where &: \\
k &= number\ of\ \beta s\ in\ our\ case\ 3 \\
\hat{\beta} &= (X\text{t}X)^{-1}X\text{t}y \\
s^2 &= \frac{1}{n-k}(y - X\hat{\beta})\text{t}(y - X\hat{\beta})
\end{aligned}$$

Thus to simulate from the joint posterior we need to simulate from:
1- $p(\sigma|y$.
2- $p(\beta|\sigma^2, y)$.

And then we find the marginal posterior of β:

$$\beta|y \sim t_n - k(\hat{\beta}, s^2(X\text{t}X)^{-}1)$$

There for to draw a sample from the joint posterior distribution of $\beta_0, \beta_2, \beta_3$ *and* σ^2 we need first to calculate the value of $\hat{\beta} = (X\text{t}X)^{-}1X\text{t}y$:

```
y<-as.matrix(df$temp)
x<-as.matrix(cbind(1,df$cov_tm,df$cov_tm^2))
n<- length(y)
k=3
beta_ht<-(solve(t(x)%*%x))%*%(t(x)%*%y)
s2<-(1/(n-k))*t((y-(x%*%beta_ht)))%*%(y-(x%*%beta_ht))
```

Then we calculate the values of Ω_n , v_n , μ_n *and* ${\sigma^2}_n$ we use the same format as in eq 1

```
omega_n <- t(x) %*% x+omega_0
df_<-(n-k)
lmbda_<-s2
v_n <- v_0 + n
mu_n <- solve(t(x) %*% x + omega_0) %*% (t(x) %*% x %*% beta_ht + omega_0 %*% mu_0)
sigma2_n <- (v_0 * segma2_0 + (t(y) %*% y + t(mu_0) %*% omega_0
                               %*% mu_0 - t(mu_n) %*% omega_n %*% mu_n)) / v_n
```

Now we generate the valuse of β from $t_{n-k}(\hat{\beta},\ s^2(X\text{t}X)^{-1})$ using μ_n as our delta and σ_n^2 as sigma. in R we use function. *mvtnorm::rmvt*:

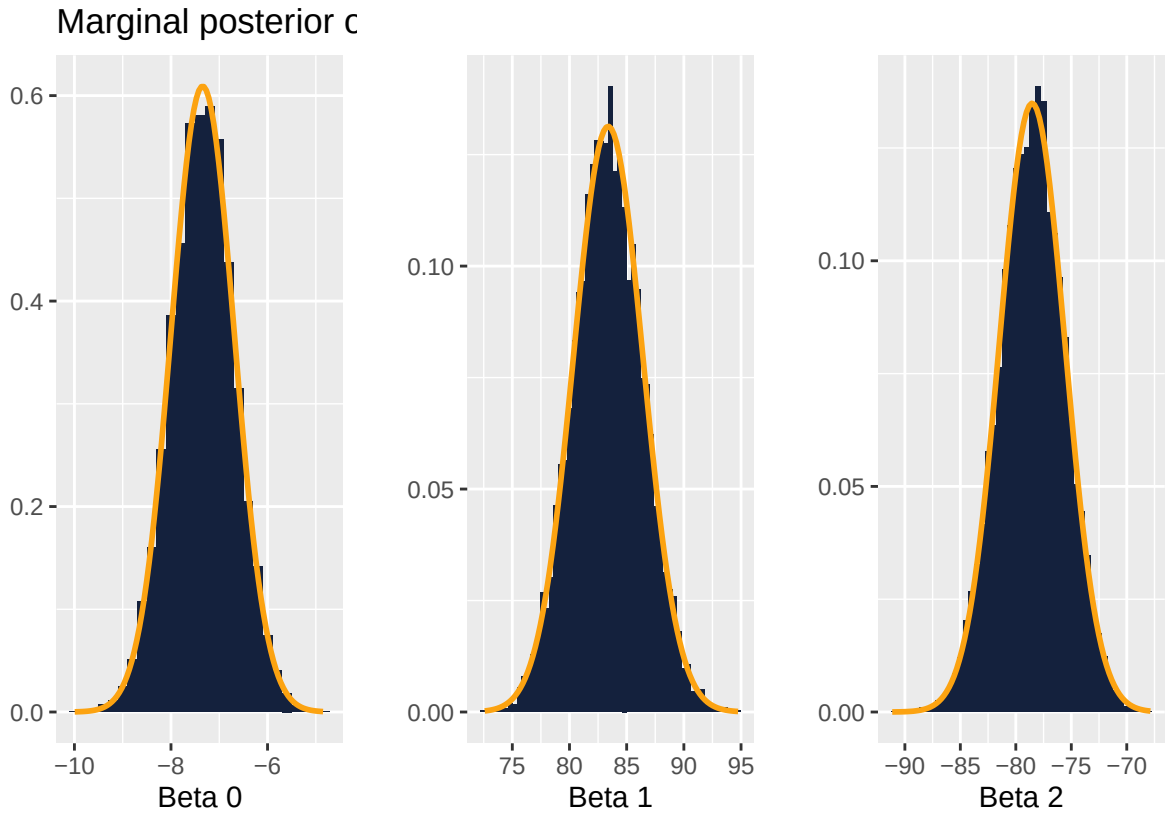
```
res<-data.frame(mvtnorm::rmvt(10000,delta=mu_n,df=df_,sigma=sigma2_n[1,1]*solve(t(x)%*%x)))
```

i/ Now we Plot a histogram for each marginal posterior of the parameters $\beta's$.

```
#We store the value of betas in a df and
#then we use this data to plot the histogarm of the betas
res<-data.frame(mvtnorm::rmvt(10000,delta=mu_n,
                              df=df_,sigma=sigma2_n[1,1]*solve(t(x)%*%x)))
```

```r
names(res)<-c('b_0','b_1','b_2')
plt1 <- ggplot(res,aes(x = b_0)) +geom_histogram(aes(y=..density..),
                  linetype=1,
                  fill='#14213D',binwidth = 0.2)+
  labs(x='Beta 0',y=' ',title ='Marginal posterior of the parameters')+
                stat_function(fun = dnorm, args = list(mean = mean(res$b_0),
                              sd = sd(res$b_0)),
                color = "#FCA311", size = 1)

plt2 <- ggplot(res,aes(x = b_1)) +geom_histogram(aes(y=..density..),
                                                 linetype=1,
                                                 fill='#14213D',binwidth = 0.4)+
  labs(x='Beta 1',y=' ')+
  stat_function(fun = dnorm, args = list(mean = mean(res$b_1),
                                         sd = sd(res$b_1)),
                color = "#FCA311", size = 1)
plt3 <- ggplot(res,aes(x = b_2)) +geom_histogram(aes(y=..density..),
                                                 linetype=1,
                                                 fill='#14213D',binwidth = 0.5)+
  labs(x='Beta 2',y=' ',)+
  stat_function(fun = dnorm, args = list(mean = mean(res$b_2),
                                         sd = sd(res$b_2)),
                color = "#FCA311", size = 1)

plt1+plt2+plt3
```

```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

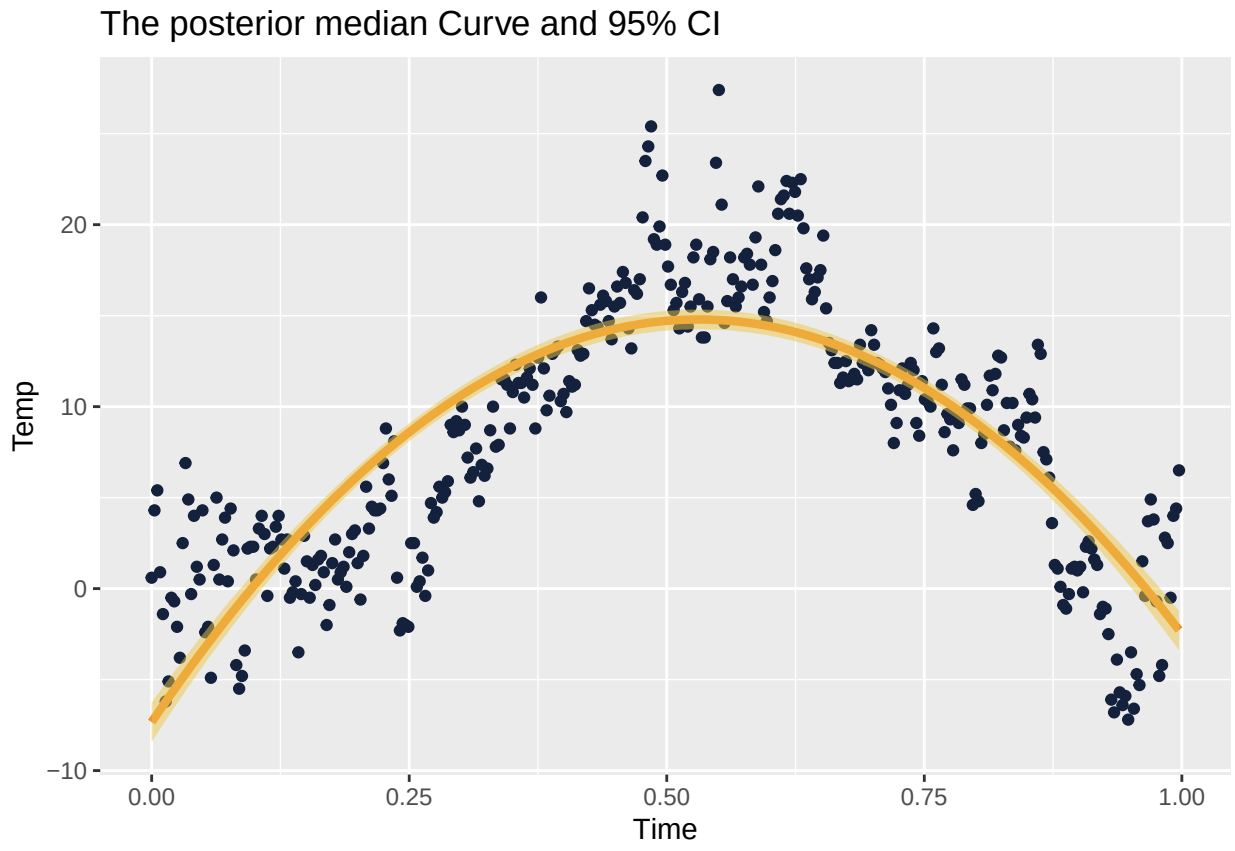


ii/ Make a scatter plot of the temperature data and overlay a curve for the posterior median of the regression function $f(time) = E[temp|time] = \beta_0 + \beta_1.time + \beta_2.time^2$.
First we need to calculate the $f(time)$ by using the results from i we can define:

```
# Calculating the median value point
median=as.matrix(apply(res, 2, median))

# we fint the regression  model P(time)=beta_0+beta_1*time+beta_2*time2
predicted_response <- x%*% median

#storing the median values
posterior_median <- apply(predicted_response, 1, median)

#Finding the predicted interval
prd_int <- data.frame(nrow = n, nrow = 2)
colnames(prd_int) <- c("CI_lower","CI_upper")
preds<- as.matrix(res)%*%t(x)
for(i in 1:nrow(x)){
  data_t <- preds[,i]
  #Here we have 95% CI using the function quantile
  prd_int[i,] <- quantile(data_t, probs = c(0.05,0.95))
}
```

By using the results from above we can fit the curve line for the posterior median and 95% CI on the scatter plot of the temperature data as below:

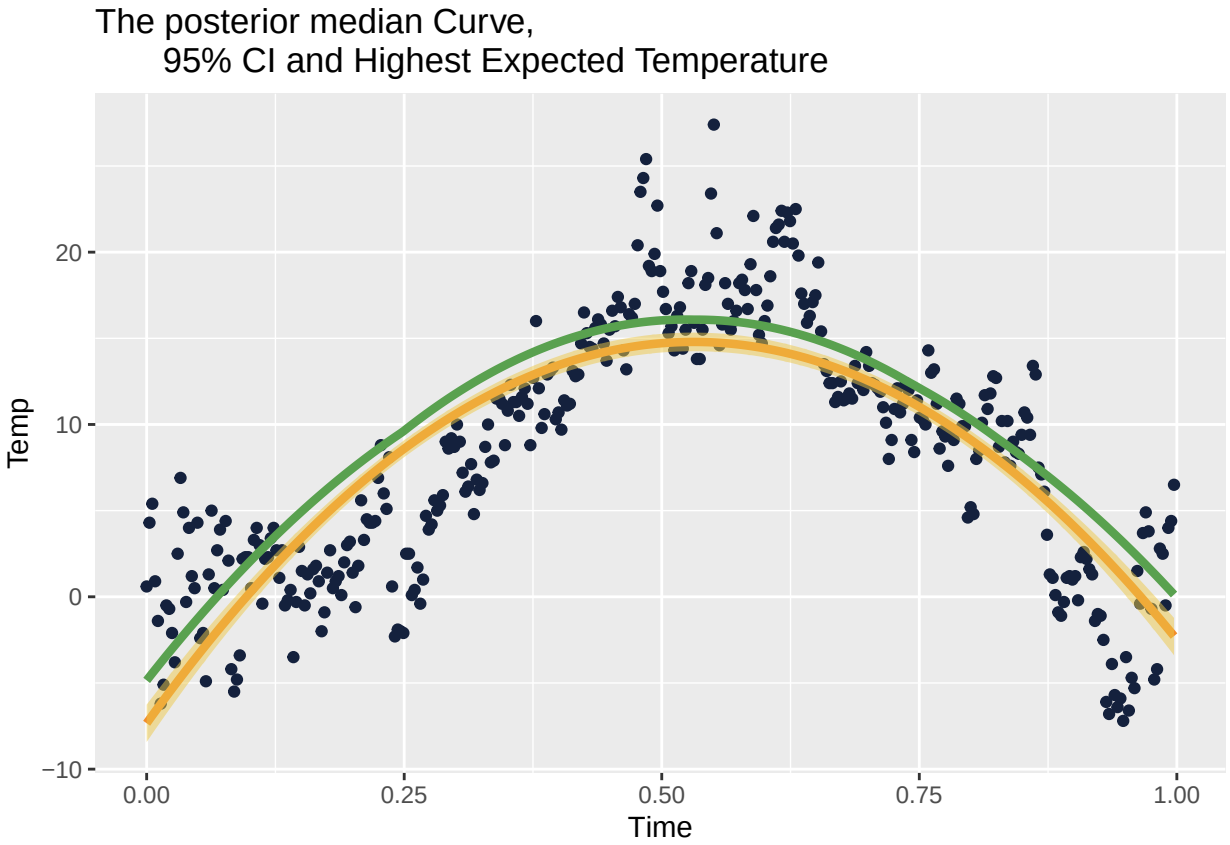
```r
# Storing the data in one data frame

plt_df=data.frame(x=df$cov_tm,y=df$temp,med=posterior_median)
plt_df= cbind(plt_df,prd_int)
# Calculate posterior median of the predicted response


plt <- ggplot(plt_df, aes(x = x, y = y)) +
  geom_point(color = "#14213D", size = 1.5)+
  geom_line(aes(y = med), color = "#F28E2B", linetype = 1,size=1.5)+
  geom_ribbon(aes(ymin = CI_lower, ymax = CI_upper)
              , alpha = 0.5,fill = "#EDC948")+
  labs(x = 'Time', y = 'Temp'
       ,title ='The posterior median Curve and 95% CI')
plt
```



C

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 $\bar{x}$. Use the simulated draws in (b) to simulate from the posterior distribution of $\bar{x}$.
[Hint: the regression curve is a quadratic polynomial. Given each posterior draw of $\beta_0, \beta_1, \beta_2$, you can find a simple formula for $\bar{x}$.]. To simulate from the posterior distribution of the highest expected temperature $\bar{X}$ we can use the all possible prediction values that we generated in the task before, then by taking the max value we get the point point wise highest expected temperature over time:

```r
# Storing the data in one data frame
#Initite the storing vector
het<-c()

#Startingh the for loop
for (i in 1:nrow(x)) {
  het[i]<-max(preds[,i])
}

# binding the data into te ploting data frame

plt_df= cbind(plt_df,het)

#Ploting the data

plt <- ggplot(plt_df, aes(x = x, y = y)) +
  geom_point(color = "#14213D", size = 1.5)+
  geom_line(aes(y = het), color = "#59A14F", linetype = 1,size=1.5)+
  geom_line(aes(y = med), color = "#F28E2B", linetype = 1,size=1.5)+
  geom_ribbon(aes(ymin = CI_lower, ymax = CI_upper)
              , alpha = 0.5,fill = "#EDC948")+
  labs(x = 'Time', y = 'Temp'
       ,title ='The posterior median Curve,
       95% CI and Highest Expected Temperature'
       ,color = "Line Legend") +
  scale_color_manual(values = c("#14213D","#59A14F","#F28E2B","#EDC948")
                     , labels = c("1","2","3","4"))+
  theme(legend.position="bottom")
plt
```



D

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 Ω_0 in a suitable way.].
To mitigate the problem of over fitting when estimating a polynomial resgression of order 10 without being worry of the over fitting, one can use Smoothness/Shrinkage/Regularization of the prior by introducing λ a penalization parameter (See lec 5 Slide 9 Lasso) in this method we have:

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

Here we have a large values of λ gives smoother fit. More shrinkage. where:

$$\mu_0 = 0$$
$$\Omega_0 = \lambda I$$

Which equivalent to *Penalized Likelihood*:

$$-2log\ p(\beta|\sigma^2, y, x) \propto (y - X\beta)'(y - X\beta) + \lambda\beta'\beta$$

Thus, the Posterior mean/mode gives ridge regressoin estimator:

$$\tilde{\beta} = (X\text{t}X + \lambda I)^{-1} X\text{t}y$$
$$if\ X\text{t}X = I$$
$$Then$$
$$\tilde{\beta} = \frac{1}{(1+\lambda)}\hat{\beta}$$

We might also be interested to determine the lambda, which could be by performing a cross validation on the test data pf using the Bayesian inference, where to us a prior for λ. we have this hierarchical setup:

$$\begin{aligned} y|\beta, \sigma^2, x &\sim N(X\beta, \sigma^2 I_n) \\ \beta|\sigma^2, \lambda &\sim N(0, \sigma^2 \lambda^{-1} I_m) \\ \sigma^2 &\sim Inv - \chi^2(v_o, \sigma_o^2) \\ \lambda &\sim Inv - \chi^2(\eta_0, \lambda_0) \\ so,\ \mu_o = 0,\ \Omega &= \lambda I_m \end{aligned}$$

and we have the joint posterior of β, σ^2, and λ is:

$$\begin{aligned} \beta|\sigma^2, \lambda, y &\sim N(\mu_n, \sigma^2 \Omega_n^{-1}) \\ \sigma^2|\lambda, y &\sim Inv - \chi^2(v_n, \sigma_n^2) \\ p(\lambda|y) &\propto \sqrt{\frac{|\Omega_0|}{|X^T X + \Omega_0|}} (\frac{v_n \sigma_n^2}{2})^{\frac{-v_n}{2}} . p(\lambda) \\ where,\ \Omega_0 &= \lambda I_m\ and\ p(\lambda)\ is\ the\ prior\ for\ \lambda\ and: \\ \mu_n &= (X^T X + \Omega_0)^{-1} X^T y \\ \Omega_n &= X^T X + \Omega_0 \\ v_n &= v_0 + n \\ v_n \sigma_n^2 &= v_0 \sigma_0^2 + y^T y - \mu_0^T \Omega_n \mu_n \end{aligned}$$

Posterior approximation for classifcation with logistic regression

A

Firstly we want to calculate the value of $\tilde{\beta}$ the posterior mode and the negative of the observed 7x7 hessian evaluated at the posterior mode $J(\tilde{\beta}) = -\frac{\partial^2\ lnp(\beta|y)}{\partial\beta\partial\beta^T}|_{\beta=\tilde{\beta}}$.
Using code snippets from my demo of logistic regression in Lecture 6. First we want to calculate the vale of $\tilde{\beta}$ and $J(\tilde{\beta}) = -\frac{\partial^2\ lnp(\beta|y)}{\partial\beta\partial\beta^T}|_{\beta=\tilde{\beta}}$ by using the *optim*, Note that we have $\tau = 2$ and $\beta \sim N(0, \frac{1}{\lambda} I)$.

```
# First we want to calculate the vale of beta and the Jacopiang
#inv of beta by using thge optim function and the code from the lec notes
# Note that we have tau = 2 and prior beta follows N(0,tau^2I)

### Select Logistic or Probit regression and install packages ###
Probit <- 0

### Prior and data inputs ###
Covs <- c(2:8) # Select which covariates/features to include
```

2. **Posterior approximation for classification with logistic regression**
The dataset `WomenAtWork.dat` contains $n = 168$ observations on the following eight variables related to women:

Variable	Data type	Meaning	Role
Work	Binary	Whether or not the woman works	Response y
Constant	1	Constant to the intercept	Feature
HusbandInc	Numeric	Husband's income	Feature
EducYears	Counts	Years of education	Feature
ExpYears	Counts	Years of experience	Feature
Age	Counts	Age	Feature
NSmallChild	Counts	Number of child ≤ 6 years in household	Feature
NBigChild	Counts	Number of child > 6 years in household	Feature

(a) Consider the logistic regression model:

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

where y equals 1 if the woman works and 0 if she does not. $\mathbf{x}$ 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

$$\beta|\mathbf{y}, \mathbf{x} \sim N\left(\tilde{\beta}, J_{\mathbf{y}}^{-1}(\tilde{\beta})\right),$$

where $\tilde{\beta}$ is the posterior mode and $J(\tilde{\beta}) = -\frac{\partial^2 \ln p(\beta|\mathbf{y})}{\partial\beta\partial\beta^T}|_{\beta=\tilde{\beta}}$ is the negative of the observed Hessian evaluated at the posterior mode. Note that $\frac{\partial^2 \ln p(\beta|\mathbf{y})}{\partial\beta\partial\beta^T}$ is a 7×7 matrix with second derivatives on the diagonal and cross-derivatives

Figure 1: Alt Text

$\frac{\partial^2 \ln p(\beta|\mathbf{y})}{\partial \beta_i \partial \beta_j}$ 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 $\beta \sim \mathcal{N}(0, \tau^2 I)$, where $\tau = 5$.

Present the numerical values of $\tilde{\beta}$ and $J_{\mathbf{y}}^{-1}(\tilde{\beta})$ 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)`.]

(b) Use your normal approximation to the posterior from (a). Write a function that simulate draws from the posterior predictive distribution of $\Pr(y = 1|\mathbf{x})$, where the values of $\mathbf{x}$ 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 $\Pr(y = 1|\mathbf{x})$ for this woman.

[Hints: The R package `mvtnorm` will be useful. Remember that $\Pr(y = 1|\mathbf{x})$ can be calculated for each posterior draw of β.]

(c) 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.]

Figure 2: Alt Text

```
standardize <- F # If TRUE,covariates/features are standardized
                 #to mean 0 and variance 1
lambda <- 4 # scaling factor for the prior of beta in our case tau = 2

# Loading out data set
wat<-read.table("WomenAtWork.dat",header = T) # read data from file
Nobs <- dim(wat)[1] # number of observations
y <- wat[1] # y=1 if the women is working, otherwise y=0.
X <- as.matrix(wat[,Covs]) # Covs matrix 7*7
Xnames <- colnames(X)
# Standraizing the covs matrix
if (standardize){
  Index <- 2:(length(Covs)-1)
  X[,Index] <- scale(X[,Index])
}
Npar <- dim(X)[2]

#############################################################################
# This is to add y variable as binary response and adding
#intercept, for now it's not needed
# for (ii in 1:Nobs){
#   if (wat$quality[ii] > 5){
#     y[ii] <- 1
#   }
# }
# wat <- data.frame(intercept=rep(1,Nobs),wat) # add intercept
#############################################################################

# Setting up the prior
mu <- as.matrix(rep(0,Npar)) # Prior mean vector
Sigma <- (1/lambda)*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)
}

LogPostProbit <- function(betas,y,X,mu,Sigma){
  linPred <- X%*%betas;
  SmallVal <- .Machine$double.xmin
  logLik <- sum(y*log(pnorm(linPred)+SmallVal) +
                  (1-y)*log(1-pnorm(linPred)+SmallVal) )
  logPrior <- dmvnorm(betas, mu, Sigma, log=TRUE);
  return(logLik + logPrior)
```

```
}

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

if (Probit==1){
  logPost = LogPostProbit;
} else{
  logPost = LogPostLogistic;
}

# 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,logPost,gr=NULL,y,X,mu,
                  Sigma,method=c("BFGS"),control=list(fnscale=-1),hessian=TRUE)

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

```
## [1] "The posterior mode is:"
```

```
print(OptimRes$par)
```

```
##              [,1]
## [1,] -0.056386120
## [2,] -0.031827146
## [3,]  0.122159465
## [4,]  0.117428313
## [5,] -0.034825257
## [6,] -0.843613303
## [7,] -0.004864958
## attr(,"names")
## [1] "Constant"    "HusbandInc"  "EducYears"   "ExpYears"    "Age"
## [6] "NSmallChild" "NBigChild"
```

```
print('The Hessian Matrix:')
```

```
## [1] "The Hessian Matrix:"
```

```
print(OptimRes$hessian)
```

```
##              [,1]        [,2]         [,3]         [,4]        [,5]        [,6]
## [1,]   -30.680116   -529.3142   -327.50822   -236.69284  -1090.0231   -6.677466
## [2,]  -529.314175 -13341.9063  -6766.54274  -4843.93682 -21997.1466 -120.512938
## [3,]  -327.508217  -6766.5427  -4173.35502  -2920.20417 -13322.2994  -89.131991
## [4,]  -236.692843  -4843.9368  -2920.20417  -3210.07312 -10128.0426  -52.468406
```

```
## [5,] -1090.023088 -21997.1466 -13322.29943 -10128.04260 -46241.4185 -222.328139
## [6,]    -6.677466   -120.5129    -89.13199    -52.46841   -222.3281  -12.483127
## [7,]   -35.842534   -680.3382   -427.53675   -263.56474  -1376.1948   -7.970026
##              [,7]
## [1,]   -35.842534
## [2,]  -680.338176
## [3,]  -427.536753
## [4,]  -263.564737
## [5,] -1376.194769
## [6,]    -7.970026
## [7,]   -95.588839
```

```
print('The approximate posterior standard deviation is:')
```

```
## [1] "The approximate posterior standard deviation is:"
```

```
print(approxPostStd)
```

```
##    Constant  HusbandInc   EducYears    ExpYears         Age NSmallChild
##  0.48111702  0.02091553  0.07054393  0.03223060  0.01981162  0.32730151
##   NBigChild
##  0.14263857
```

Now we compare with the results from the regression model: *glmModel<- glm(Work ~ 0 + ., data = WomenAtWork, family = binomial).*

```
glmModel<- glm(Work ~ 0 + ., data = wat, family = binomial)
summary(glmModel)
```

```
##
## Call:
## glm(formula = Work ~ 0 + ., family = binomial, data = wat)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3210  -0.9799   0.4423   0.9707   1.9131
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## Constant     0.02263    1.93083   0.012 0.990649
## HusbandInc  -0.03796    0.02229  -1.703 0.088573 .
## EducYears    0.18447    0.10007   1.844 0.065253 .
## ExpYears     0.12132    0.03353   3.618 0.000297 ***
## Age         -0.04858    0.03323  -1.462 0.143686
## NSmallChild -1.56485    0.51078  -3.064 0.002187 **
## NBigChild   -0.02526    0.17716  -0.143 0.886618
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 182.99  on 132  degrees of freedom
## Residual deviance: 146.73  on 125  degrees of freedom
## AIC: 160.73
##
## Number of Fisher Scoring iterations: 4
```

The Coefficients of model results about shows relationship between each predictor variable and the log-odds of the outcome variable. variables like *HusbandInc,Age,NSmallChild and NBigChild* have a negative impact on the y variable in which the direction of the relationship, while the rest have a positive impact, our results from the teammate tell the same direction of the model estimates (i.e variables with negative impact *HusbandInc,Age,NSmallChild and NBigChild*).
However the magnitude of the coefficient which indicates the strength of the relationship is a bit different.

Now we Compute an approximate 95% equal tail posterior probability interval for the regression coefficient to the variable NSmallChild.

```
# We use the function rmvnorm to generate the
#variates using OptimRes$par as our mean and approxPostStd as sigma
postmode<-as.matrix(OptimRes$par[,1])
poststd<-solve(-OptimRes$hessian)
watvar<-data.frame(rmvnorm(n=10000,mean = postmode, sigma = poststd))

#For a 95% CI, you would typically calculate
#the lower and upper bounds at quantiles 0.025 and 0.975, respectively.
print('An approximate 95% equal tail posterior probability
      interval for the regression coeffcient to the variable NSmallChild is:')
```

```
## [1] "An approximate 95% equal tail posterior probability\n      interval for the regression coeffcie
```

```
print(quantile(watvar[,6],c(0.025,.975)))
```

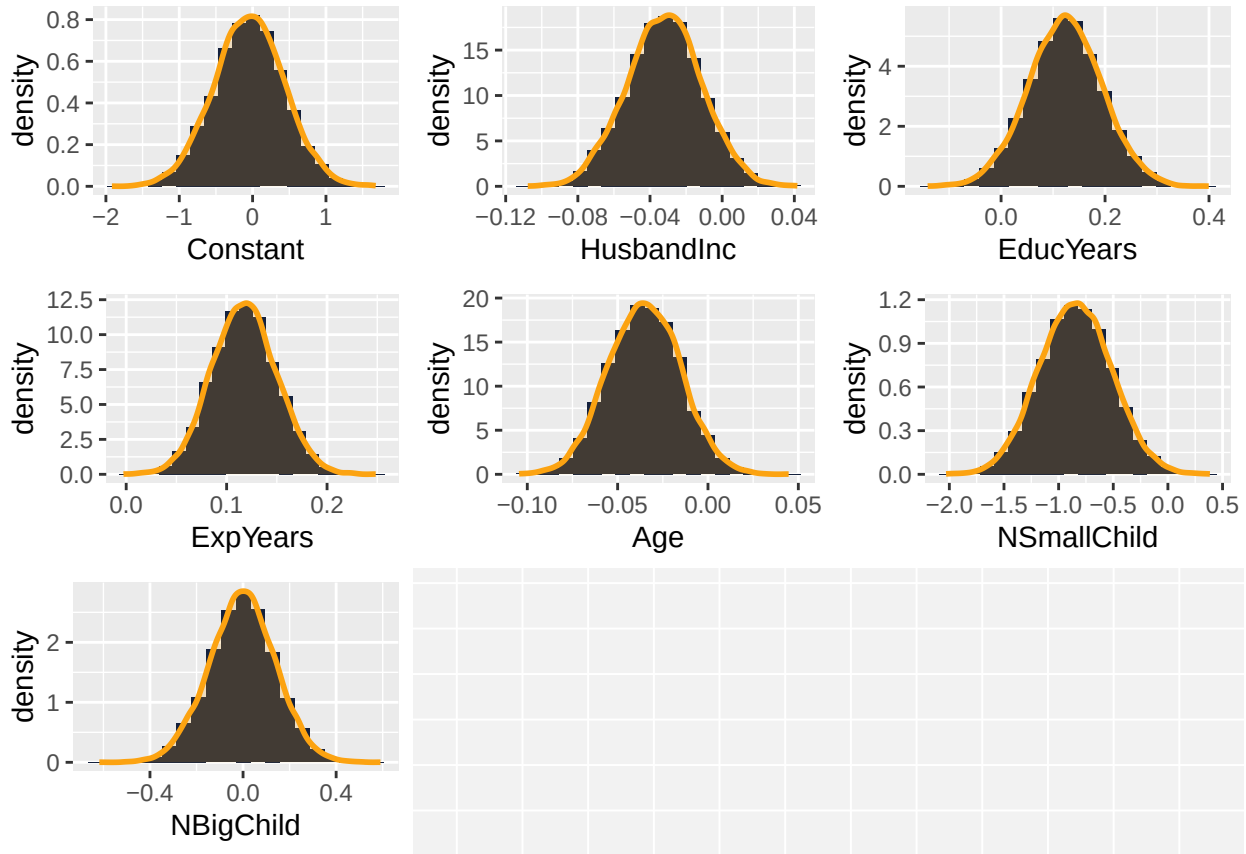
```
##       2.5%      97.5%
## -1.4837914 -0.1907636
```

The results of the CI tell us that the NSmallChild regression coefficient have a significant impact on the log-odds of the outcome variable as we can see the coefficient fall between the lower and the upper bound of the CI, the direction of this impact in negative and have the highest magnitude in the model coefficients.

B

Now we write a function that simulate draws from the posterior predictive distribution of Pr(y = 0|x),where the values of x corresponds to a 40-year-old woman, with two children (4 and 7 years old), 11 years of education, 7 years of experience, and a husband with an income of 18. Plot the posterior predictive distribution of Pr(y = 0|x) for this woman.
First we estimate the values of β using the normal approximation to the posterior from (a) above, the below graph shows the distribution of the estimated βs:

```
betas<-rmvnorm(n=10000,mean = postmode, sigma = poststd)
# ploting the beta distribution

p_data<- as.data.frame(betas)
colnames(p_data)<-colnames(wat[2:8])
names<-colnames(p_data)
p_fun<- function(coln){
          plt <- ggplot(p_data,aes_string(x = coln)) +
              geom_histogram(aes(y=..density..),linetype=1
                             ,fill='#14213D',bins = 20)+
              geom_density(alpha=.2,color="#FCA311",size=1,fill="#FCA311")
plt
}

plot(arrangeGrob(grobs = lapply(names, p_fun)))
```



Now we Write a function that simulate draws from the posterior predictive distribution of Pr(y = 0jx),where the values of x.

```
pred_prob<- function(ndraws,x_new){
        ### Select Logistic or Probit regression and install packages ###
        Probit <- 0
        ### Prior and data inputs ###
        Covs <- c(2:8) # Select which covariates/features to include
        standardize <- F # If TRUE, covariates/features
```

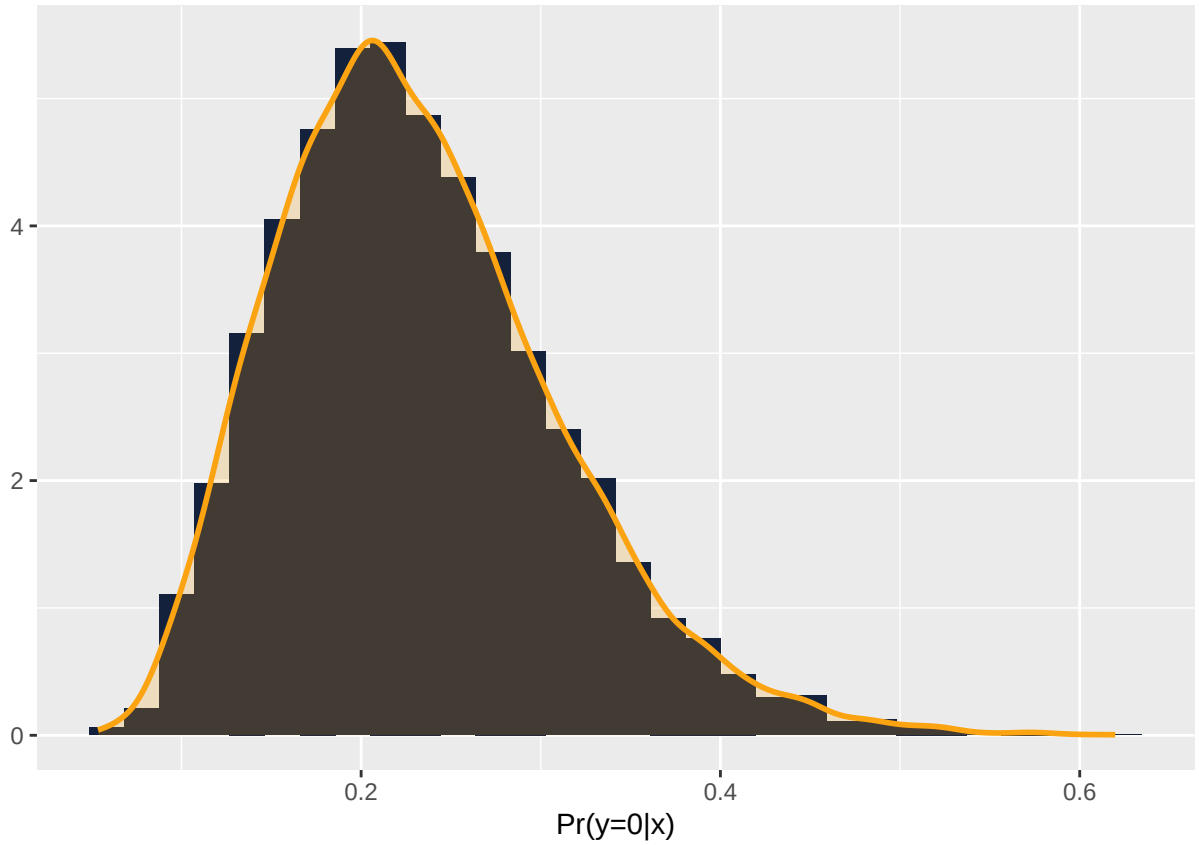
```r
#are standardized to mean 0 and variance 1
lambda <- 2 # scaling factor for the prior of beta in our case tau = 2
# Loading out data set
wat<-read.table("WomenAtWork.dat",header = T) # read data from file
Nobs <- dim(wat)[1] # number of observations
y <- wat[1] # y=1 if the women is working, otherwise y=0.
X <- as.matrix(wat[,Covs]) # Covs matrix 7*7
Xnames <- colnames(X)
# Standraizing the covs matrix
if (standardize){
  Index <- 2:(length(Covs)-1)
  X[,Index] <- scale(X[,Index])
}
Npar <- dim(X)[2]
# Setting up the prior
mu <- as.matrix(rep(0,Npar)) # Prior mean vector
Sigma <- (lambda)^2 *diag(Npar) # Prior covariance matrix
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)
}
# Not in use we change the value to 0 at the beginning of the code
#########################################################################
LogPostProbit <- function(betas,y,X,mu,Sigma){
  linPred <- X%*%betas;
  SmallVal <- .Machine$double.xmin
  logLik <- sum(y*log(pnorm(linPred)+SmallVal) +
                  (1-y)*log(1-pnorm(linPred)+SmallVal))
  logPrior <- dmvnorm(betas, mu, Sigma, log=TRUE);
  return(logLik + logPrior)
}
#########################################################################
# Select the initial values for beta
initVal <- matrix(0,Npar,1)
if (Probit==1){
  logPost = LogPostProbit;
} else{
  logPost = LogPostLogistic;
}
# 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,logPost,gr=NULL,y,X,mu,Sigma,method=c("BFGS")
                  ,control=list(fnscale=-1),hessian=TRUE)

postmode<-as.matrix(OptimRes$par[,1])
poststd<- solve(-OptimRes$hessian)
```

```
        x_new<-as.matrix(x_new,ncol=1)
        betas<-rmvnorm(n=ndraws,mean = postmode, sigma = poststd)
        # Finding the value y givan the new Xs
        pr_y<-data.frame(x=betas%*%x_new)
        # Finding the probabilities using the logistics function
        pr_y$x_logit<-1/(1+exp(-pr_y$x))
        # Ploting the dataset
        plt <- ggplot(pr_y,aes(x = x_logit)) +geom_histogram(aes(y=..density..),
                  linetype=1, fill='#14213D')+
                  geom_density(alpha=.2,color="#FCA311",size=1,fill="#FCA311")+
                  labs(x='Pr(y=0|x)',y=' ',)
        plt
}
```

```
pred_prob(10000,c(1,18,11,7,40,1,1))
```

```
##
```

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

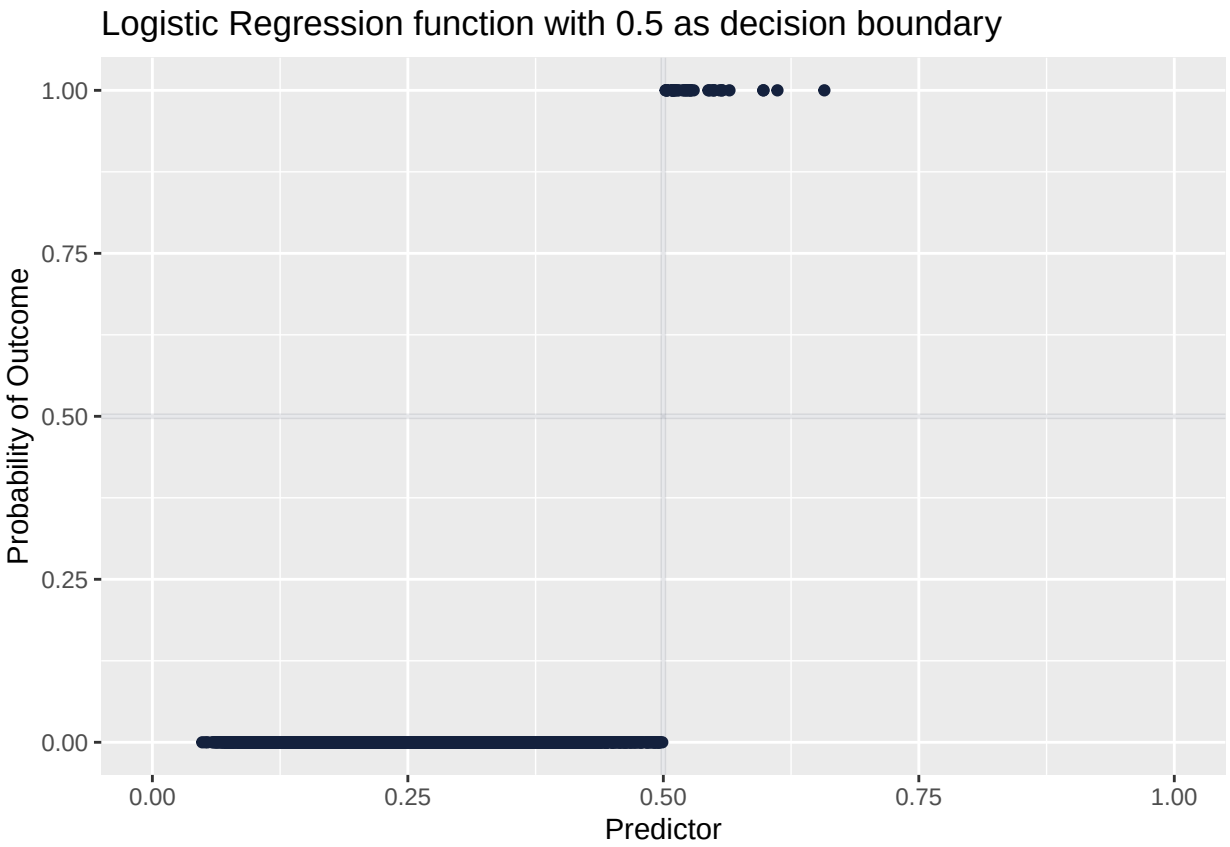


consider 13 women which all have the same features as the woman in (b).Rewrite your function and plot the posterior predictive distribution for the number of women, out of these 13, that are not working.

```
pred_prob2<- function(ndraws,x_new){
      ### Select Logistic or Probit regression and install packages ###
      Probit <- 0
      ### Prior and data inputs ###
      Covs <- c(2:8) # Select which covariates/features to include
      standardize <- F # If TRUE, covariates/features are
                       #standardized to mean 0 and variance 1
      lambda <- 2 # scaling factor for the prior of beta in our case tau = 2
      # Loading out data set
      wat<-read.table("WomenAtWork.dat",header = T) # read data from file
      Nobs <- dim(wat)[1] # number of observations
      y <- wat[1] # y=1 if the women is working, otherwise y=0.
      X <- as.matrix(wat[,Covs]) # Covs matrix 7*7
      Xnames <- colnames(X)
      # Standraizing the covs matrix
      if (standardize){
        Index <- 2:(length(Covs)-1)
        X[,Index] <- scale(X[,Index])
      }
      Npar <- dim(X)[2]
      # Setting up the prior
      mu <- as.matrix(rep(0,Npar)) # Prior mean vector
      Sigma <- (lambda)^2 *diag(Npar) # Prior covariance matrix
      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)
      }
      # Not in use we change the value to 0 at the beginning of the code
      ################################################################################
      LogPostProbit <- function(betas,y,X,mu,Sigma){
        linPred <- X%*%betas;
        SmallVal <- .Machine$double.xmin
        logLik <- sum(y*log(pnorm(linPred)+SmallVal) +
                        (1-y)*log(1-pnorm(linPred)+SmallVal))
        logPrior <- dmvnorm(betas, mu, Sigma, log=TRUE);
        return(logLik + logPrior)
      }
      ################################################################################
      # Select the initial values for beta
      initVal <- matrix(0,Npar,1)
      if (Probit==1){
        logPost = LogPostProbit;
      } else{
        logPost = LogPostLogistic;
      }
      # 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,logPost,gr=NULL,y,X,mu,Sigma,method=c("BFGS")
                          ,control=list(fnscale=-1),hessian=TRUE)

        postmode<-as.matrix(OptimRes$par[,1])
        poststd<- solve(-OptimRes$hessian)

        x_new<-as.matrix(x_new,ncol=1)
        betas<-rmvnorm(n=ndraws,mean = postmode, sigma = poststd)
        # Finding the value y givan the new Xs
        pr_y<-data.frame(x=betas%*%x_new)
        # Finding the probabilities using the logistics function
        pr_y$x_logit<-1/(1+exp(-pr_y$x))
        #Adding the clasifier
        pr_y$job_flag <- ifelse(pr_y$x_logit <= 0.5, 0, 1)
        plt <- ggplot(pr_y, aes(x = x_logit, y = job_flag)) +
          geom_point(colour="#14213D") +
          # stat_smooth(method="glm",colour="#FCA311",
          #             alpha = 0.5, se=FALSE, fullrange=TRUE,
          #             method.args = list(family=binomial)) +
          xlab("Predictor") + xlim(c(0,1))+
          ylab("Probability of Outcome") +
          ggtitle("Logistic Regression function with 0.5 as decision boundary")+
    geom_vline(aes(xintercept = 0.5), color = "#14213D",size=1, alpha = 0.1) +
     geom_hline(aes(yintercept = 0.5), color = "#14213D",size=1, alpha = 0.1)

        plt
  }
```

```
pred_prob2(10000,c(1,18,11,7,40,1,1))
```



References:

1- Bertil Wegmann (2023). Bayesian Learning [Lecture notes]. 732A73, Department of Computer and Information Science,LiU University.

Code Appendix

```
knitr::opts_chunk$set(echo = TRUE)
set.seed(123)
library(ggplot2)
library(mvtnorm)
library(readxl)
library(LaplacesDemon)
library(patchwork)
library(gridExtra)
#reading the files
df<-read_xlsx("Linkoping2022.xlsx")
#Creating the covariate_time vaiable as
#(the number of days sinvce the beginning of the year / 365)
a<- df$datetime
#begunning of the year
```

```r
b<- '2022-01-01'
a<-format.POSIXlt(strptime(a,'%Y-%m-%d'))
b<-format.POSIXlt(strptime(b,'%Y-%m-%d'))
#time diff from the
x<-as.vector(difftime(a,b,units='days'))
df$cov_tm<-x/365
#We assume to use a conjugate prior from
#the linear regression in Lec 5 ,
#we have been given the prior hyperparamteres as follow:
mu_0= as.matrix(c(0,100,-100),ncol=3)
omega_0=0.01*diag(3)
v_0=1
segma2_0=1
n= length(df$temp)
ndraws=10

# Step 1: Draw X folow chi2(n - 1)
draw_chi_sq <- function(n) {
  return(rchisq(1, df = n - 1))
}
# Step 2: Compute sigma2 = (n - 1) * s² / X
compute_sigma_sq <- function(n, segma2_0, X) {
  return((n - 1) * segma2_0 / X)
}

# simulation
segma_estimation <- function(n, mu_0, segma2_0, ndraws) {
  results <- c()

  for (i in 1:ndraws) {
    X <- draw_chi_sq(n)
    sigma_sq <- compute_sigma_sq(n, segma2_0, X)
    results[i] <- sigma_sq
  }
  return(results)
}

sigma2<-segma_estimation(n, mu_0, segma2_0, ndraws)
for (i in 1:length(sigma2)) {
  e<- rnorm(1,0,sigma2[i])
  res<-rmvnorm(1,mu_0,sigma2[i]*omega_0)
  temp= x=res[1,1]+res[1,2]*df$cov_tm+res[1,3]*df$cov_tm^2+e
  df[[paste0("temp_fit_",sigma2[i])]]<-temp
}
## Example function
# x1_grid <- seq(min(X[,2]),max(X[,2]),0.1)
# Mu_draws <- matrix(0,length(x1_grid),2)
# for (ii in 1:length(x1_grid)){
# Curr_x <- c(1,x1_grid[ii],x1_grid[ii]**2,27,27**2,x1_grid[ii]*27)
# CurrMu <- Betas %*% Curr_x
# Mu_draws[ii,] <- quantile(CurrMu,probs=c(0.025,0.975))
# }
# plot(x1_grid,Mu_draws[,1],"n",main="95 % posterior
```

```
# probability intervals as a function of x1",
# xlab="x1", ylab="",ylim=c(0,500))
# lines(x1_grid,Mu_draws[,1],col="blue")
# lines(x1_grid,Mu_draws[,2],col="blue")


# Define a vector of colors
colors <-c("#FCA311", "#00FF00", "#0000FF", "#FFFF00", "#00FFFF",
           "#FF00FF", "#800000", "#008000", "#000080", "#808000",
           "#800080", "#008080", "#808080", "#FFC0CB", "#FFA500",
           "#FFD700", "#A52A2A", "#7FFF00", "#FF1493", "#00BFFF")

# Plot with different colored lines
plt <- ggplot(df, aes(x = cov_tm, y = temp)) +
  geom_point(aes(color = factor('temp')), size = 1)

for (i in names(df)[-c(1:4, ncol(df))]) {
  plt <- plt + geom_line(aes_string(y = i, color = factor(i)), linetype = 1)
}

# Map colors to the lines
plt <- plt +
  scale_color_manual(values = colors) +
  labs(x = 'Time', y = 'Temp',color='Predictions with different Segma values')
plt
degree <- 2  # Set the degree of the polynomial
x= df$cov_tm
y= df$temp
model <- lm(y ~ poly(x, degree, raw = TRUE))
df_plt<- data.frame(x=x,y=y)
z=predict(model)
# Print the model summary


# Plot the data and regression line
plt <- ggplot(df_plt, aes(x = x, y = y)) +
  geom_point(color = "#4E79A7", size = 1)+
  geom_line(aes(y = z), color = "#FCA311",size=1 ,linetype = 1)+
  labs(x = 'Time', y = 'Temp'
       ,title ='Polynomial Regression with Dgree 2')
plt
summary(model)

mu_news= as.matrix(c(7.3,83.1,-78.3),ncol=3)
segma2_new=10
sigma2<-segma_estimation(n,mu_news, segma2_new, ndraws)

for (i in 1:length(sigma2)) {
  e<- rnorm(1,0,sigma2[i])
  res<-rmvnorm(1,mu_news,sigma2[i]*omega_0)
  temp= x=res[1,1]+res[1,2]*df$cov_tm+res[1,3]*df$cov_tm^2+e
  df[[paste0("temp_fit_",sigma2[i])]]<-temp
}
```

```
# Define a vector of colors
colors <-c("#FCA311", "#00FF00", "#0000FF", "#FFFF00", "#00FFFF",
          "#FF00FF", "#800000", "#008000", "#000080", "#808000",
         "#800080", "#008080", "#808080", "#FFC0CB", "#FFA500",
        "#FFD700", "#A52A2A", "#7FFF00", "#FF1493", "#00BFFF")

# Plot with different colored lines
plt <- ggplot(df, aes(x = cov_tm, y = temp)) +
  geom_point(aes(color = factor('temp')), size = 1)

for (i in names(df)[-c(1:15, ncol(df))]) {
  plt <- plt + geom_line(aes_string(y = i, color = factor(i)), linetype = 1)
}

# Map colors to the lines
plt <- plt +
  scale_color_manual(values = colors) +
  labs(x = 'Time', y = 'Temp',color='Predictions with different Segma values')
plt
y<-as.matrix(df$temp)
x<-as.matrix(cbind(1,df$cov_tm,df$cov_tm^2))
n<- length(y)
k=3
beta_ht<-(solve(t(x)%*%x))%*%(t(x)%*%y)
s2<-(1/(n-k))*t((y-(x%*%beta_ht)))%*%(y-(x%*%beta_ht))
omega_n <- t(x) %*% x+omega_0
df_<-(n-k)
lmbda_<-s2
v_n <- v_0 + n
mu_n <- solve(t(x) %*% x + omega_0) %*% (t(x) %*% x %*% beta_ht + omega_0 %*% mu_0)
sigma2_n <- (v_0 * segma2_0 + (t(y) %*% y + t(mu_0) %*% omega_0
                               %*% mu_0 - t(mu_n) %*% omega_n %*% mu_n)) / v_n
res<-data.frame(mvtnorm::rmvt(10000,delta=mu_n,df=df_,sigma=sigma2_n[1,1]*solve(t(x)%*%x)))
#We store the value of betas in a df and
#then we use this data to plot the histogarm of the betas
res<-data.frame(mvtnorm::rmvt(10000,delta=mu_n,
                              df=df_,sigma=sigma2_n[1,1]*solve(t(x)%*%x)))


names(res)<-c('b_0','b_1','b_2')
plt1 <- ggplot(res,aes(x = b_0)) +geom_histogram(aes(y=..density..),
                  linetype=1,
                  fill='#14213D',binwidth = 0.2)+
  labs(x='Beta 0',y=' ',title ='Marginal posterior of the parameters')+
                stat_function(fun = dnorm, args = list(mean = mean(res$b_0),
                              sd = sd(res$b_0)),
                color = "#FCA311", size = 1)

plt2 <- ggplot(res,aes(x = b_1)) +geom_histogram(aes(y=..density..),
                                                 linetype=1,
                                                 fill='#14213D',binwidth = 0.4)+
  labs(x='Beta 1',y=' ')+
  stat_function(fun = dnorm, args = list(mean = mean(res$b_1),
```

```
                                        sd = sd(res$b_1)),
                color = "#FCA311", size = 1)
plt3 <- ggplot(res,aes(x = b_2)) +geom_histogram(aes(y=..density..),
                                                 linetype=1,
                                                 fill='#14213D',binwidth = 0.5)+
  labs(x='Beta 2',y=' ',)+
  stat_function(fun = dnorm, args = list(mean = mean(res$b_2),
                                        sd = sd(res$b_2)),
                color = "#FCA311", size = 1)

plt1+plt2+plt3

# Calculating the median value point
median=as.matrix(apply(res, 2, median))

# we fint the regression  model P(time)=beta_0+beta_1*time+beta_2*time2
predicted_response <- x%*% median

#storing the median values
posterior_median <- apply(predicted_response, 1, median)

#Finding the predicted interval
prd_int <- data.frame(nrow = n, nrow = 2)
colnames(prd_int) <- c("CI_lower","CI_upper")
preds<- as.matrix(res)%*%t(x)
for(i in 1:nrow(x)){
  data_t <- preds[,i]
  #Here we have 95% CI using the function quantile
  prd_int[i,] <- quantile(data_t, probs = c(0.05,0.95))
}
# Storing the data in one data frame

plt_df=data.frame(x=df$cov_tm,y=df$temp,med=posterior_median)
plt_df= cbind(plt_df,prd_int)
# Calculate posterior median of the predicted response


plt <- ggplot(plt_df, aes(x = x, y = y)) +
  geom_point(color = "#14213D", size = 1.5)+
  geom_line(aes(y = med), color = "#F28E2B", linetype = 1,size=1.5)+
  geom_ribbon(aes(ymin = CI_lower, ymax = CI_upper)
              , alpha = 0.5,fill = "#EDC948")+
  labs(x = 'Time', y = 'Temp'
       ,title ='The posterior median Curve and 95% CI')
plt
# Storing the data in one data frame
#Initite the storing vector
het<-c()

#Startingh the for loop
for (i in 1:nrow(x)) {
  het[i]<-max(preds[,i])
}
```

```r
# binding the data into te ploting data frame

plt_df= cbind(plt_df,het)

#Ploting the data

plt <- ggplot(plt_df, aes(x = x, y = y)) +
  geom_point(color = "#14213D", size = 1.5)+
  geom_line(aes(y = het), color = "#59A14F", linetype = 1,size=1.5)+
  geom_line(aes(y = med), color = "#F28E2B", linetype = 1,size=1.5)+
  geom_ribbon(aes(ymin = CI_lower, ymax = CI_upper)
              , alpha = 0.5,fill = "#EDC948")+
  labs(x = 'Time', y = 'Temp'
       ,title ='The posterior median Curve,
       95% CI and Highest Expected Temperature'
       ,color = "Line Legend") +
  scale_color_manual(values = c("#14213D","#59A14F","#F28E2B","#EDC948")
                     , labels = c("1","2","3","4"))+
  theme(legend.position="bottom")
plt
# First we want to calculate the vale of beta and the Jacopiang
#inv of beta by using thge optim function and the code from the lec notes
# Note that we have tau = 2 and prior beta follows N(0,tau^2I)

### Select Logistic or Probit regression and install packages ###
Probit <- 0

### Prior and data inputs ###
Covs <- c(2:8) # Select which covariates/features to include
standardize <- F # If TRUE,covariates/features are standardized
                 #to mean 0 and variance 1
lambda <- 4 # scaling factor for the prior of beta in our case tau = 2

# Loading out data set
wat<-read.table("WomenAtWork.dat",header = T) # read data from file
Nobs <- dim(wat)[1] # number of observations
y <- wat[1] # y=1 if the women is working, otherwise y=0.
X <- as.matrix(wat[,Covs]) # Covs matrix 7*7
Xnames <- colnames(X)
# Standraizing the covs matrix
if (standardize){
  Index <- 2:(length(Covs)-1)
  X[,Index] <- scale(X[,Index])
}
Npar <- dim(X)[2]

#############################################################################
# This is to add y variable as binary response and adding
#intercept, for now it's not needed
# for (ii in 1:Nobs){
#   if (wat$quality[ii] > 5){
#     y[ii] <- 1
#   }
```

```
# }
# wat <- data.frame(intercept=rep(1,Nobs),wat) # add intercept
################################################################################

# Setting up the prior
mu <- as.matrix(rep(0,Npar)) # Prior mean vector
Sigma <- (1/lambda)*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)
}

LogPostProbit <- function(betas,y,X,mu,Sigma){
  linPred <- X%*%betas;
  SmallVal <- .Machine$double.xmin
  logLik <- sum(y*log(pnorm(linPred)+SmallVal) +
                  (1-y)*log(1-pnorm(linPred)+SmallVal) )
  logPrior <- dmvnorm(betas, mu, Sigma, log=TRUE);
  return(logLik + logPrior)
}

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

if (Probit==1){
  logPost = LogPostProbit;
} else{
  logPost = LogPostLogistic;
}

# 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,logPost,gr=NULL,y,X,mu,
                  Sigma,method=c("BFGS"),control=list(fnscale=-1),hessian=TRUE)

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

```
print('The Hessian Matrix:')
print(OptimRes$hessian)
print('The approximate posterior standard deviation is:')
print(approxPostStd)
glmModel<- glm(Work ~ 0 + ., data = wat, family = binomial)
summary(glmModel)
# We use the function rmvnorm to generate the
#variates using OptimRes$par as our mean and approxPostStd as sigma
postmode<-as.matrix(OptimRes$par[,1])
poststd<-solve(-OptimRes$hessian)
watvar<-data.frame(rmvnorm(n=10000,mean = postmode, sigma = poststd))

#For a 95% CI, you would typically calculate
#the lower and upper bounds at quantiles 0.025 and 0.975, respectively.
print('An approximate 95% equal tail posterior probability
      interval for the regression coeffcient to the variable NSmallChild is:')
print(quantile(watvar[,6],c(0.025,.975)))
betas<-rmvnorm(n=10000,mean = postmode, sigma = poststd)
# ploting the beta distribution

p_data<- as.data.frame(betas)
colnames(p_data)<-colnames(wat[2:8])
names<-colnames(p_data)
p_fun<- function(coln){
          plt <- ggplot(p_data,aes_string(x = coln)) +
              geom_histogram(aes(y=..density..),linetype=1
                             ,fill='#14213D',bins = 20)+
              geom_density(alpha=.2,color="#FCA311",size=1,fill="#FCA311")
plt
}

plot(arrangeGrob(grobs = lapply(names, p_fun)))
pred_prob<- function(ndraws,x_new){
        ### Select Logistic or Probit regression and install packages ###
        Probit <- 0
        ### Prior and data inputs ###
        Covs <- c(2:8) # Select which covariates/features to include
        standardize <- F # If TRUE, covariates/features
        #are standardized to mean 0 and variance 1
        lambda <- 2 # scaling factor for the prior of beta in our case tau = 2
        # Loading out data set
        wat<-read.table("WomenAtWork.dat",header = T) # read data from file
        Nobs <- dim(wat)[1] # number of observations
        y <- wat[1] # y=1 if the women is working, otherwise y=0.
        X <- as.matrix(wat[,Covs]) # Covs matrix 7*7
        Xnames <- colnames(X)
        # Standraizing the covs matrix
        if (standardize){
          Index <- 2:(length(Covs)-1)
          X[,Index] <- scale(X[,Index])
        }
        Npar <- dim(X)[2]
        # Setting up the prior
```

```
        mu <- as.matrix(rep(0,Npar)) # Prior mean vector
        Sigma <- (lambda)^2 *diag(Npar) # Prior covariance matrix
        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)
        }
        # Not in use we change the value to 0 at the beginning of the code
        ##########################################################################
        LogPostProbit <- function(betas,y,X,mu,Sigma){
          linPred <- X%*%betas;
          SmallVal <- .Machine$double.xmin
          logLik <- sum(y*log(pnorm(linPred)+SmallVal) +
                          (1-y)*log(1-pnorm(linPred)+SmallVal))
          logPrior <- dmvnorm(betas, mu, Sigma, log=TRUE);
          return(logLik + logPrior)
        }
        ##########################################################################
        # Select the initial values for beta
        initVal <- matrix(0,Npar,1)
        if (Probit==1){
          logPost = LogPostProbit;
        } else{
          logPost = LogPostLogistic;
        }
        # 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,logPost,gr=NULL,y,X,mu,Sigma,method=c("BFGS")
                          ,control=list(fnscale=-1),hessian=TRUE)

        postmode<-as.matrix(OptimRes$par[,1])
        poststd<- solve(-OptimRes$hessian)

        x_new<-as.matrix(x_new,ncol=1)
        betas<-rmvnorm(n=ndraws,mean = postmode, sigma = poststd)
        # Finding the value y givan the new Xs
        pr_y<-data.frame(x=betas%*%x_new)
        # Finding the probabilities using the logistics function
        pr_y$x_logit<-1/(1+exp(-pr_y$x))
        # Ploting the dataset
        plt <- ggplot(pr_y,aes(x = x_logit)) +geom_histogram(aes(y=..density..),
                    linetype=1, fill='#14213D')+
                    geom_density(alpha=.2,color="#FCA311",size=1,fill="#FCA311")+
                    labs(x='Pr(y=0|x)',y=' ',)
        plt
  }
pred_prob(10000,c(1,18,11,7,40,1,1))
  pred_prob2<- function(ndraws,x_new){
```

```
### Select Logistic or Probit regression and install packages ###
Probit <- 0
### Prior and data inputs ###
Covs <- c(2:8) # Select which covariates/features to include
standardize <- F # If TRUE, covariates/features are
                 #standardized to mean 0 and variance 1
lambda <- 2 # scaling factor for the prior of beta in our case tau = 2
# Loading out data set
wat<-read.table("WomenAtWork.dat",header = T) # read data from file
Nobs <- dim(wat)[1] # number of observations
y <- wat[1] # y=1 if the women is working, otherwise y=0.
X <- as.matrix(wat[,Covs]) # Covs matrix 7*7
Xnames <- colnames(X)
# Standraizing the covs matrix
if (standardize){
  Index <- 2:(length(Covs)-1)
  X[,Index] <- scale(X[,Index])
}
Npar <- dim(X)[2]
# Setting up the prior
mu <- as.matrix(rep(0,Npar)) # Prior mean vector
Sigma <- (lambda)^2 *diag(Npar) # Prior covariance matrix
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)
}
# Not in use we change the value to 0 at the beginning of the code
#################################################################################
LogPostProbit <- function(betas,y,X,mu,Sigma){
  linPred <- X%*%betas;
  SmallVal <- .Machine$double.xmin
  logLik <- sum(y*log(pnorm(linPred)+SmallVal) +
                  (1-y)*log(1-pnorm(linPred)+SmallVal))
  logPrior <- dmvnorm(betas, mu, Sigma, log=TRUE);
  return(logLik + logPrior)
}
#################################################################################
# Select the initial values for beta
initVal <- matrix(0,Npar,1)
if (Probit==1){
  logPost = LogPostProbit;
} else{
  logPost = LogPostLogistic;
}
# 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,logPost,gr=NULL,y,X,mu,Sigma,method=c("BFGS")
                          ,control=list(fnscale=-1),hessian=TRUE)

        postmode<-as.matrix(OptimRes$par[,1])
        poststd<- solve(-OptimRes$hessian)

        x_new<-as.matrix(x_new,ncol=1)
        betas<-rmvnorm(n=ndraws,mean = postmode, sigma = poststd)
        # Finding the value y givan the new Xs
        pr_y<-data.frame(x=betas%*%x_new)
        # Finding the probabilities using the logistics function
        pr_y$x_logit<-1/(1+exp(-pr_y$x))
        #Adding the clasifier
        pr_y$job_flag <- ifelse(pr_y$x_logit <= 0.5, 0, 1)
        plt <- ggplot(pr_y, aes(x = x_logit, y = job_flag)) +
          geom_point(colour="#14213D") +
          # stat_smooth(method="glm",colour="#FCA311",
          #             alpha = 0.5, se=FALSE, fullrange=TRUE,
          #             method.args = list(family=binomial)) +
          xlab("Predictor") + xlim(c(0,1))+
          ylab("Probability of Outcome") +
          ggtitle("Logistic Regression function with 0.5 as decision boundary")+
    geom_vline(aes(xintercept = 0.5), color = "#14213D",size=1, alpha = 0.1) +
     geom_hline(aes(yintercept = 0.5), color = "#14213D",size=1, alpha = 0.1)

        plt
  }
################################################################################
        # Example function
        # LogPost <- function(theta,n,Sumx3){
        #
        #     logLik <- n*log(theta) -  Sumx3*theta;
        #     logPrior <- 2*log(theta) - 4*theta;
        #
        #     return(logLik + logPrior)
        #   }
        #   theta_grid <- seq(0.01,2.5,0.01)
        #   PostDens_propto <- exp(LogPost(theta_grid,5,4.084))
        #   PostDens <- PostDens_propto/(0.01*sum(PostDens_propto))
        #   plot(theta_grid,PostDens,main="Posterior distribution"
        #        ,xlab="theta", ylab="")
        #
        # n <- 5
        # Sumx3 <- 4.084
        # OptRes <- optim(0.5,LogPost,gr=NULL,n,Sumx3,method=c("L-BFGS-B")
        #                 ,lower=0.1,control=list(fnscale=-1),hessian=TRUE)
        #
        # plot(theta_grid,PostDens,col="blue",main="Posterior distribution"
        #      ,xlab="theta", ylab="")
        # lines(theta_grid,dnorm(theta_grid
        #         ,mean = OptRes$par,sd = sqrt(-1/OptRes$hessian)),col="red")
        # legend("topleft", legend=c("Approximation", "Exact"
        #                            ,col=c("red", "blue"), lty=1:2, cex=0.8))
```

```
        ##########################################################################
pred_prob2(10000,c(1,18,11,7,40,1,1))
```