Lab 2 Bayesian Learning

Hugo Morvan

2024-04-29

Linear and polynomial regression

The dataset Linkoping2022.xlsx contains daily average temperatures (in degree Celcius) in Linköping over the course of the year 2022. Use the function read_xlsx(), which is included in the R package readxl (install.packages("readxl")), to import the dataset in R. The response variable is temp and the covariate time that you need to create yourself is defined by:

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

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

$$temp = \beta_0 + \beta_1 * time + \beta_2 * time^2 + \epsilon, \epsilon \overset{\text{iid}}{\sim} N(0, \sigma^2).$$

```
library(readxl)
data = as.data.frame(read_xlsx("Linkoping2022.xlsx"))
#365 obs of 3 variables (name, datetime, temp)
time = c(0:364)/365
data$time <- time
```

a)

Use the conjugate prior for the linear regression model. The prior hyperparameters μ_0, Ω_0, ν_0 and σ^2 shall be set to sensible values. Start with $\mu_0 = (0, 100, -100)^T$, $\Omega_0 = 0.01 * I_3$, $\nu_0 = 1$ and $\sigma^2 = 1$. 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. 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.]

Conjugate prior for the linear regression:

- Joint prior for β and σ^2 : $\beta|\sigma^2 \sim N(\mu_0, \sigma^2\Omega_0^{-1})$ where $\sigma^2 \sim Inv - \chi^2(\nu_0, \sigma_0^2)$
- Posterior : $\beta|\sigma^2, y \sim N(\mu_n, \sigma^2\Omega_n^{-1})$ and $\sigma^2|y \sim Inv - \chi^2(\nu_n, \sigma_n^2)$

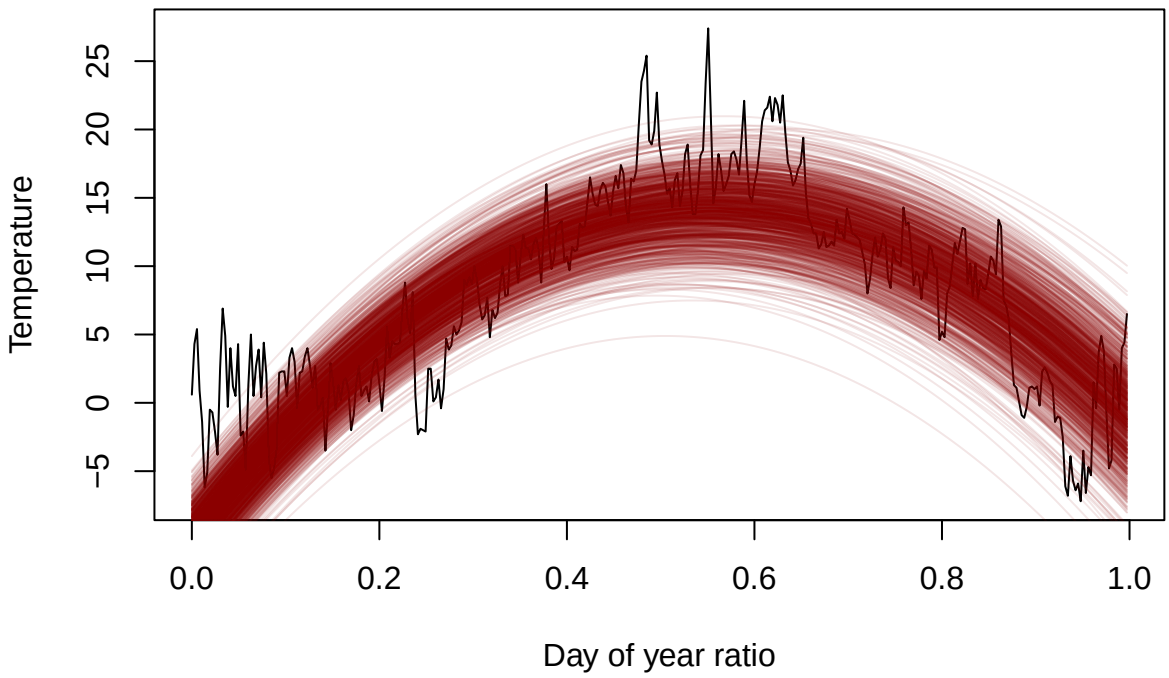
$\mu_n = (X'X + \Omega_0)^{-1}(X'X\hat{\beta} + \Omega_0\mu_0)$

$\Omega_n = X'X + \Omega_0$

$\nu_n = \nu_0 + n$

$\nu_n\sigma_n^2 = \nu_0\sigma_0^2 + (y'y + \mu_0'\Omega_0\mu_0 - \mu_n'\Omega_n\mu_n)$

```r
library(scales)
library(mvtnorm)
# Initializing the parameters
mu_0 = c(-10,87,-78) #(y intercept, next two have to be balanced. High value -> high arch, vice versa)
Omega_0 = as.matrix(0.01 * diag(3)) #thickness (higher = thinner)
nu_0 = 3 #Vertical thickness (higher = thicker)
sigma_sq_0 = 4 #vertical thickness (higher = thicker)
n = 365

X = matrix(c(time*0+1, time, time*time), ncol=3)
y = data$temp

#Prior
sigsq_draws_prior = as.matrix((nu_0*sigma_sq_0) / rchisq(1000, n-1))
beta_draws_prior = matrix(NA,nrow=0, ncol=3)
for(i in (1:length(sigsq_draws_prior))){
  sig = sigsq_draws_prior[i]
  beta_draws_prior = rbind(beta_draws_prior,rmvnorm(n = 1, mean = mu_0, sigma = sig*solve(Omega_0)))
}


#Plot
f = function(x,betas,sigsq){
  betas[1] + betas[2]*x + betas[3]*x^2 + rnorm(1, 0, sigsq)
}

x = time
betas = beta_draws_prior[1,]
plot(x, y, type = 'l', xlab="Day of year ratio", ylab = "Temperature")
for(i in 1:1000){
  x = time
  betas = beta_draws_prior[i,]
  sigsq = sigsq_draws_prior[i,]
  lines(x, f(x, betas, sigsq), type = 'l', col=alpha('darkred', 0.1))
}
```



The collection of curves looks reasonable for most of the data. Only the beginning of the year is not well represented by the prior, but it is hard to find a prior that fits the data perfectly.

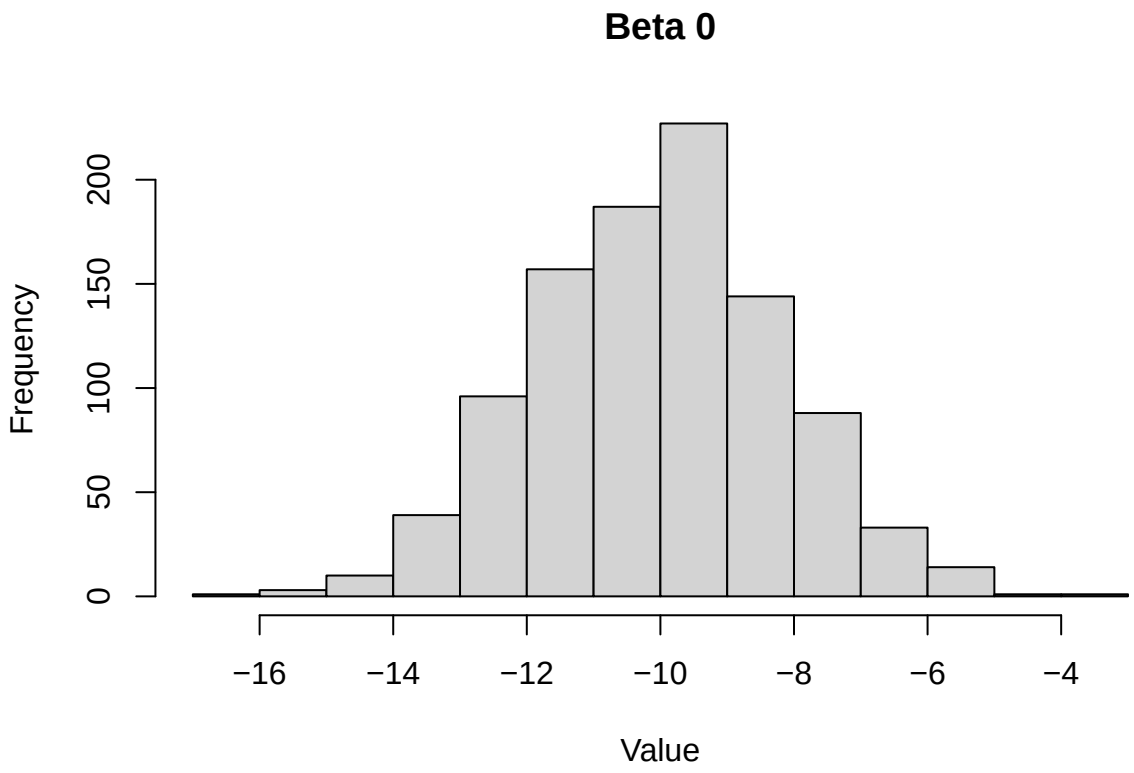
b)

Write a function that simulate draws from the joint posterior distribution of β_0, β_1,β_2 and σ^2.

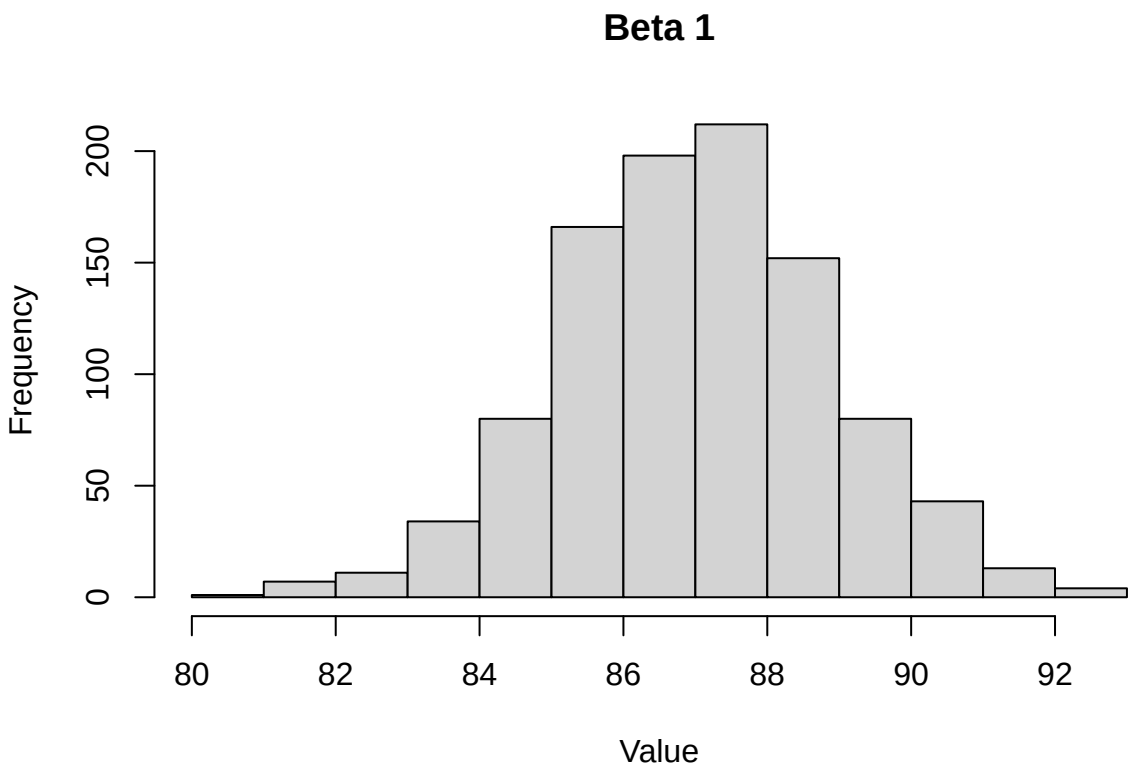
```
#Posterior
beta_draws_post = matrix(0,nrow=0, ncol=3)
sigsq_draw_post = matrix(0,nrow=0, ncol=1)
for(i in 1:1000){
  beta_hat = beta_draws_prior[i,]
  mu_n = solve(t(X) %*% X + Omega_0)%*%(t(X) %*% X %*% beta_hat + Omega_0 %*% mu_0)
  Omega_n = t(X)%*%X + Omega_0
  nu_n = nu_0 + n
  nu_n_sig_n = nu_0%*%sigma_sq_0+(t(y)%*%y + t(mu_0)%*%Omega_0%*%mu_0 - t(mu_n)%*%Omega_n%*%mu_n)

  sig = nu_n_sig_n / rchisq(1, n-1)
  sigsq_draw_post = rbind(sigsq_draw_post, sig)

  beta_draws_post = rbind(beta_draws_prior,
                          rmvnorm(n=1,mean=mu_n, sigma = sigsq_draw_post[1,]*solve(Omega_n)))
}
```

i) Plot a histogram for each marginal posterior of the parameters.

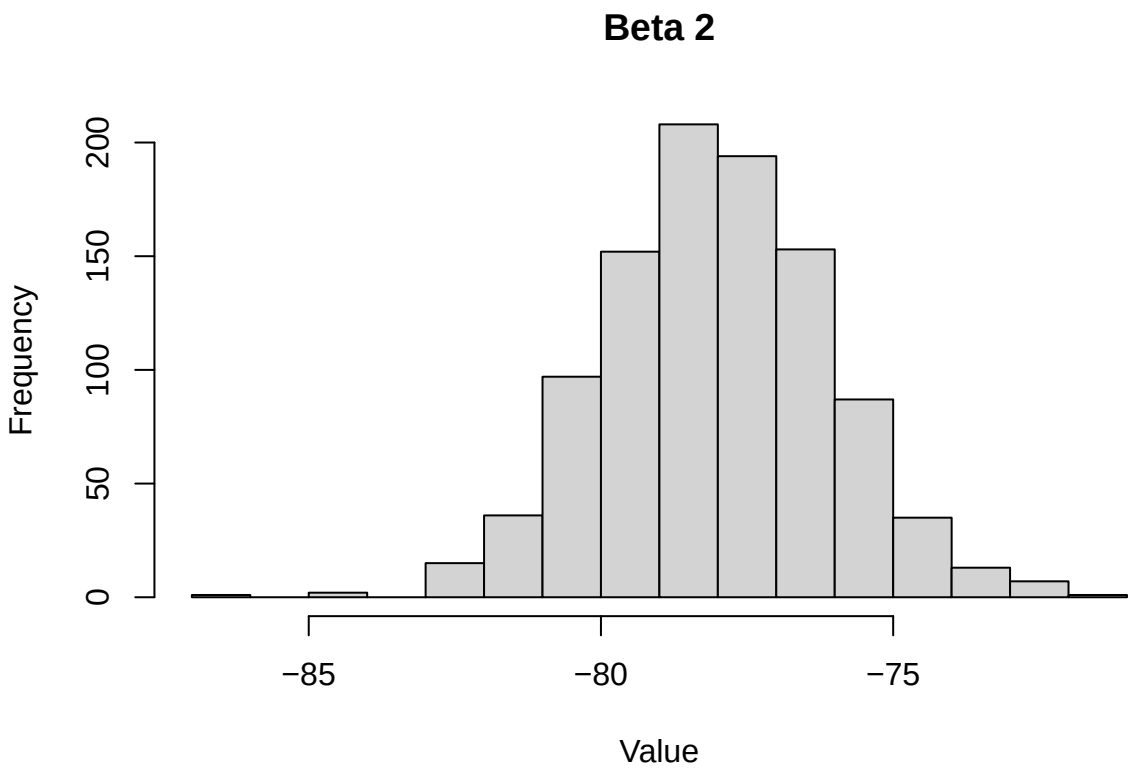
```r
#histogram for each marginal posterior of the parameters
hist(beta_draws_post[,1], main="Beta 0", xlab="Value")
```



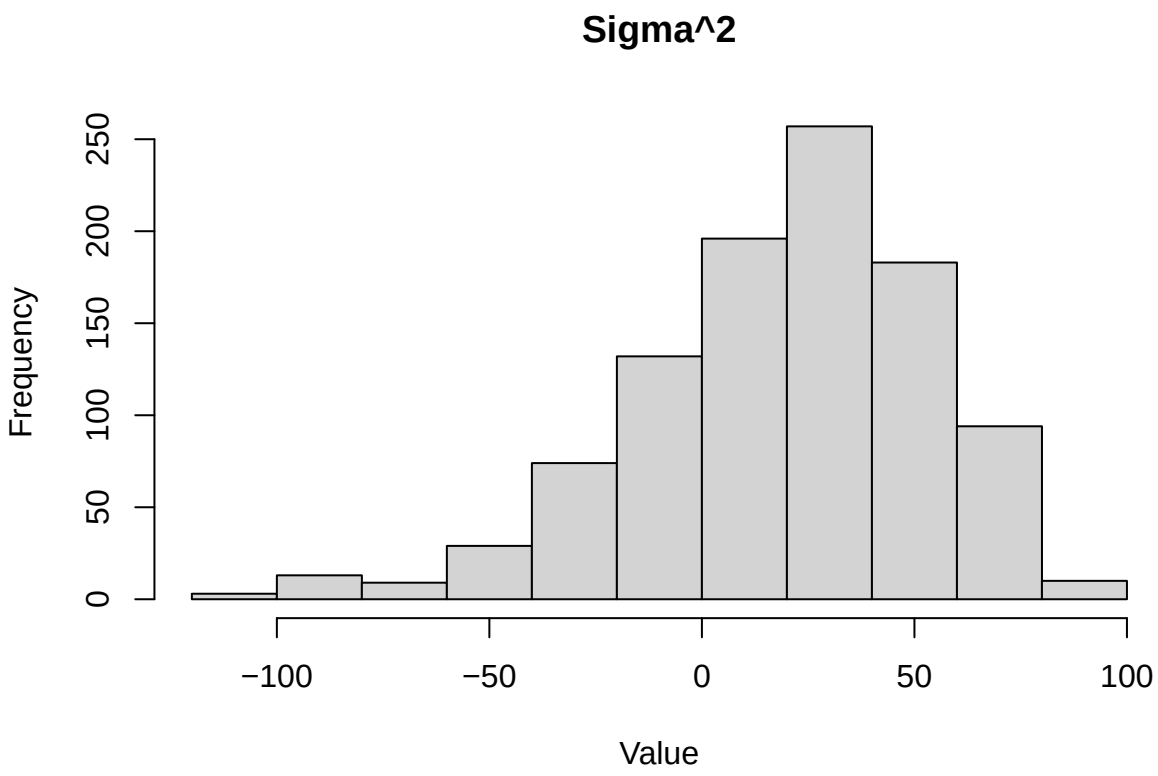

```r
hist(beta_draws_post[,2], main="Beta 1", xlab="Value")
```




```
hist(beta_draws_post[,3], main="Beta 2", xlab="Value")
```

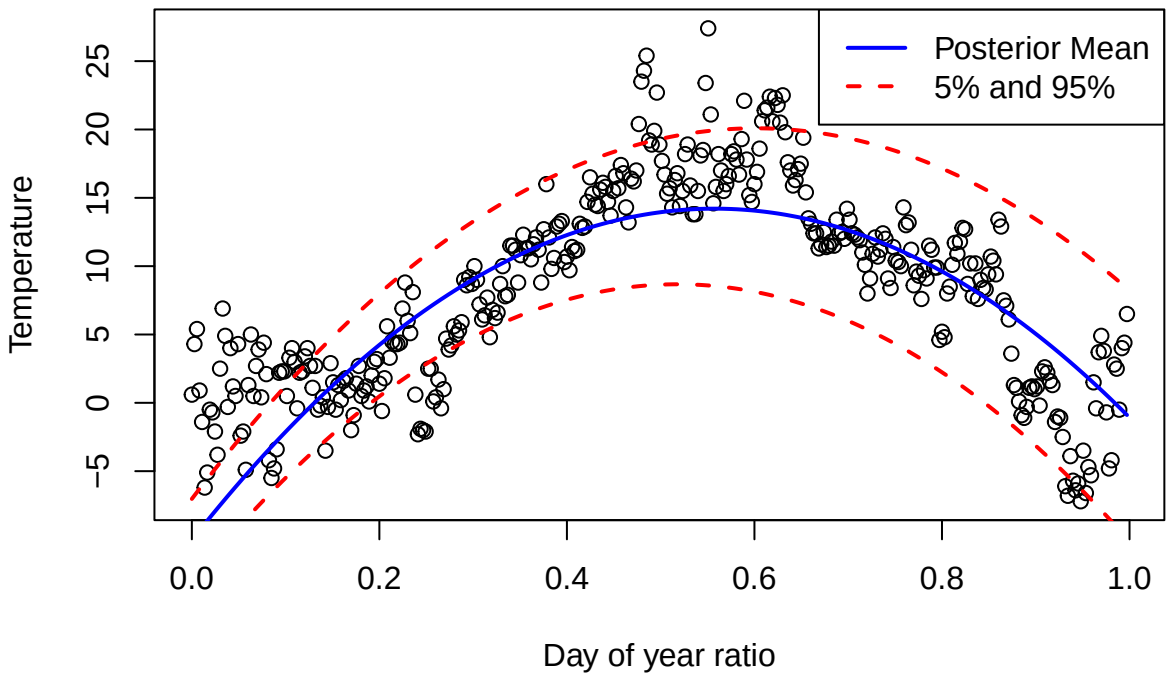



```
hist(sigsq_draw_post, main="Sigma^2", xlab="Value")
```



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 90% equal tail posterior probability intervals of f (time), i.e. the 5 and 95 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?

```
#Scatter plot of the temperature data + curve for the posterior median of the regression function
plot(time, data$temp, type = 'p', xlab="Day of year ratio", ylab = "Temperature")
#Mean
betas_mean = colMeans(beta_draws_post)
lines(time, betas_mean[1] + betas_mean[2]*time + betas_mean[3]*time^2, type = 'l', col='blue', lwd=2)

#5 and 95 percentiles for each beta for each time
betas_5 = c(quantile(beta_draws_post[,1], 0.05),
            quantile(beta_draws_post[,2], 0.05),
            quantile(beta_draws_post[,3], 0.05))

betas_95 = c(quantile(beta_draws_post[,1], 0.95),
             quantile(beta_draws_post[,2], 0.95),
             quantile(beta_draws_post[,3], 0.95))

lines(time, betas_5[1] + betas_5[2]*time + betas_5[3]*time^2, type = 'l', col='red', lty=2, lwd=2)
lines(time, betas_95[1] + betas_95[2]*time + betas_95[3]*time^2, type = 'l', col='red', lty=2, lwd=2)
legend("topright", legend=c("Posterior Mean", "5% and 95%"), col=c("blue", "red"), lty=c(1,2),
       lwd=c(2,2))
```



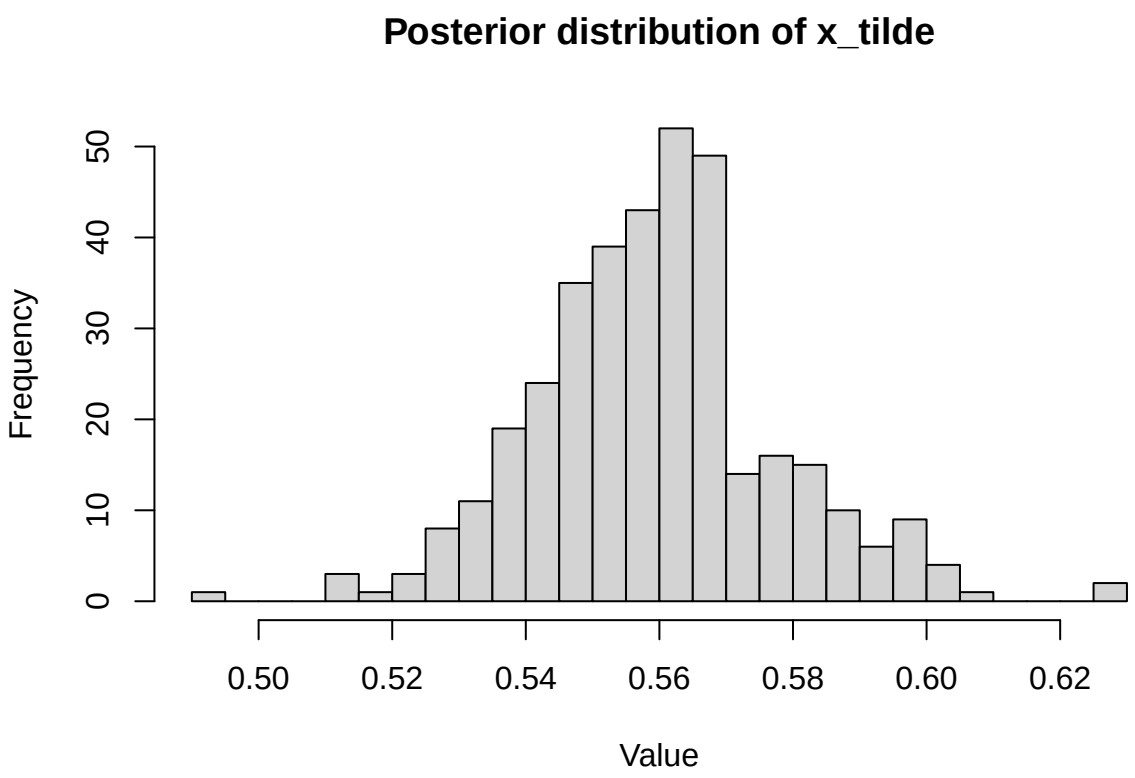
The posterior probability intervals contain most of the data points, but they are not perfect. They should contain most of the data points if the temperature in a year followed our model, which is a quadratic polynomial. However, the reality is different so it is not surprising that the intervals do not contain all the data points, especially where the data is noisy.

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 $\tilde{x}$. Use the simulated draws in (b) to simulate from the posterior distribution of $\tilde{x}$. [Hint: the regression curve is a quadratic polynomial. Given each posterior draw of β_0, β_1 and β_2 , you can find a simple formula for $\tilde{x}$.]

To find the maximum of a quadratic polynomial, we can use the formula $x = -\frac{\beta_1}{2\beta_2}$ which is the maximum of the quadratic polynomial $f(x) = \beta_0 + \beta_1 x + \beta_2 x^2$ (derivative of f(x) equal to 0). We can use this formula to find the maximum temperature for each draw of β_0, β_1 and β_2 and simulate the posterior distribution of $\tilde{x}$, and also calculate the mean of these maximum temperatures.

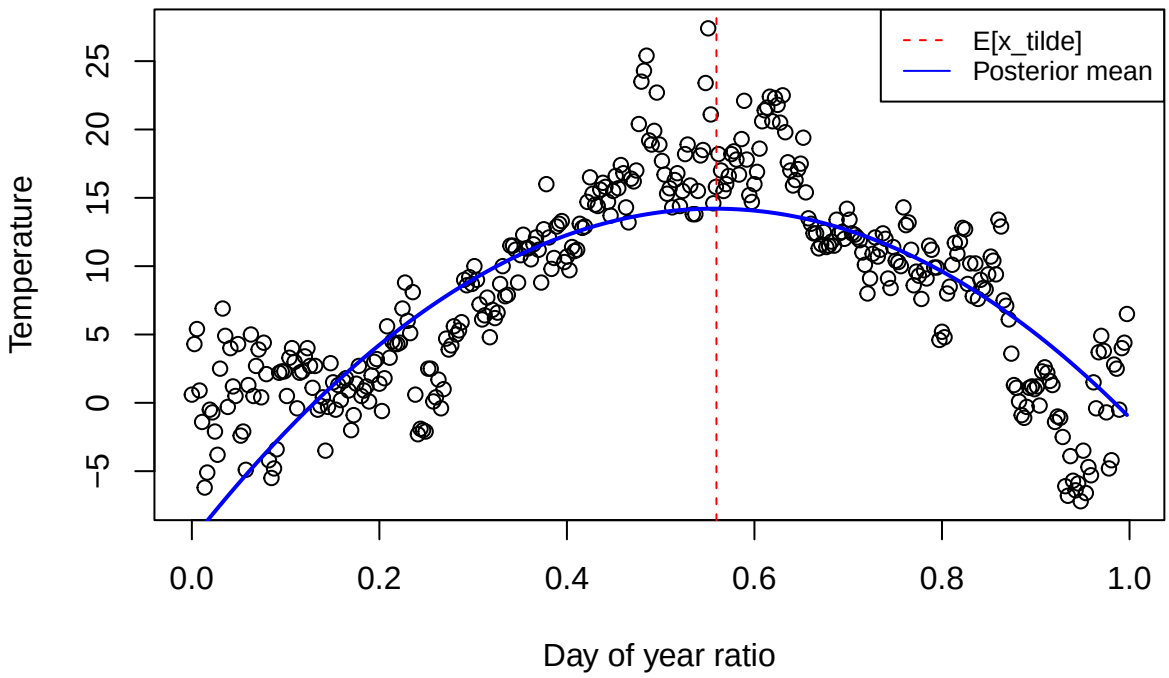
```
max_temp <- function(betas){
  (-betas[2])/(2*betas[3])
}
x_maxs = matrix(0,nrow=0, ncol=1)
for(i in 1:365){
  betas = beta_draws_post[i,]
  x_maxs = rbind(x_maxs, max_temp(betas))
}
hist(x_maxs, main="Posterior distribution of x_tilde", xlab="Value", ylab="Frequency", breaks=30)
```




```
x_tilde = mean(x_maxs)
print(paste("E[x_tilde] is",x_tilde))
```

```
## [1] "E[x_tilde] is 0.559465882922543"
```

```
plot(time, data$temp, type = 'p', xlab="Day of year ratio", ylab = "Temperature")
abline(v=x_tilde, col='red', lty=2)
legend("topright", legend=c("E[x_tilde]", "Posterior mean"), col=c("red","blue"),
       lty=c(2,1), cex=c(0.8,0.8))
lines(time, betas_mean[1] + betas_mean[2]*time + betas_mean[3]*time^2, type = 'l', col='blue', lwd=2)
```



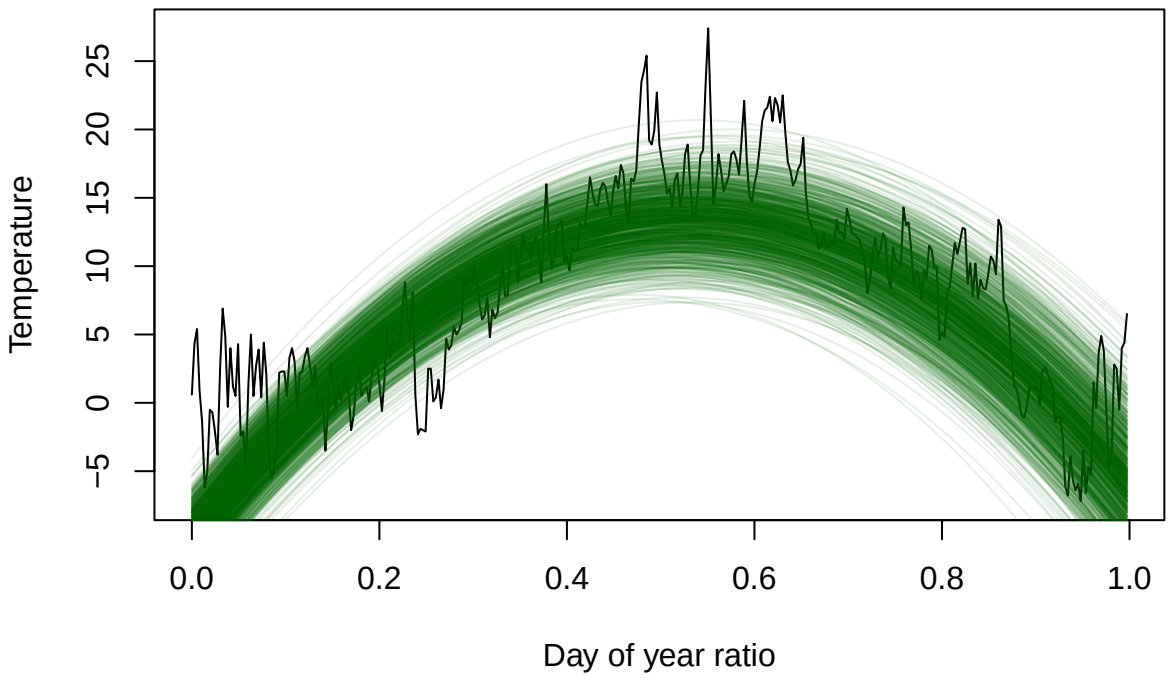
d)

Say now that you want to estimate a polynomial regression of order 10, 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.]

Because we suspect that higher order terms may not be needed (the data seems to follow a quadratic path), and it might cause overfitting, a prior that has null/close to null mean for higher order terms, and a small variance (high omegas) is suggested. This way, the higher order terms will not have a big impact on the model. We keep the mean of the first three terms the same as before, as the previous questions showed that the quadratic polynomial is a good fit for most of the data. The posterior is recalculated for sanity check.

```
# Initializing the parameters

##### MODIFIED #######
mu_0 = c(-10,87,-78, -5, 0, 0, 0, 0, 0, 0, 0) #(y, beta_0, ..., beta_9)
Omega_0 = diag(c(rep(0.01,4), rep(100,7))) # Small variance (high omega) for higher order terms
#######################

nu_0 = 3 #Vertical thickness (higher = thicker)
sigma_sq_0 = 4 #vertical thickness (higher = thicker)
n = 365

X = matrix(c(time*0+1, time, time*time), ncol=3)
y = data$temp
```

```r
#Prior
sigsq_draws_prior = as.matrix((nu_0*sigma_sq_0) / rchisq(1000, n-1))

########## MODIFIED ##########
beta_draws_prior = matrix(NA,nrow=0, ncol=11)
##############################

for(i in (1:length(sigsq_draws_prior))){
  sig = sigsq_draws_prior[i]
  beta_draws_prior = rbind(beta_draws_prior,rmvnorm(n = 1, mean = mu_0, sigma = sig*solve(Omega_0)))
}

#Plot
########## MODIFIED ##########
f = function(x,betas,sigsq){
  betas[1] + betas[2]*x + betas[3]*x^2 + betas[4]*x^3 + betas[5]*x^4 + betas[6]*x^5 + betas[7]*x^6 + be
}
##############################

x = time
betas = beta_draws_prior[1,]
plot(x, y, type = 'l', xlab="Day of year ratio", ylab = "Temperature")
for(i in 1:1000){
  x = time
  betas = beta_draws_prior[i,]
  sigsq = sigsq_draws_prior[i,]
  lines(x, f(x, betas, sigsq), type = 'l', col=alpha('darkgreen', 0.1))
}
```



2. Posterior approximation for classification with logistic regression

The dataset WomenAtWork.dat contains n = 132 observations on the following eight variables related to women (see table in lab compendium).

a)

Consider the logistic regression model:

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

where y equals 1 if the woman works and 0 if she does not. x is a 7-dimensional vector containing the seven features (including a 1 to model the intercept). The goal is to approximate the posterior distribution of the parameter vector β with a multivariate normal distribution

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

where $\tilde{\beta}$ is the posterior mode and $J_y^{-1} = -\frac{complete}{later}$ is the negative of the observed Hessian evaluated at the posterior mode. Note that $\frac{complete}{later}$ is a 7*7 matrix with second derivatives on the diagonal and cross derivatives *inserteq* on the off-diagonal. You can compute this derivative by hand, but we will let the computer do it numerically for you. Calculate both $\tilde{\beta}$ and $J(\tilde{\beta})$ by using the optim function in R. [Hint: You may use code snippets from my demo of logistic regression in Lecture 6.] Use the prior $beta \sim N(0, \tau^2 I)$, where $\tau = 2$.

Present the numerical values of $\tilde{\beta}$ and $J_y^{-1}(\tilde{\beta})$ 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: r glmModel<- glm(Work ~ 0 + ., data = WomenAtWork, family = binomial).]

```r
# Template code from Lecture material, modified for the WomenAtWork data set:

Covs <- c(2:8) # Selects which covariates/features to include
standardize <- FALSE # If TRUE, covariates/features are standardized to mean 0 and variance 1
lambda <- 1 # scaling factor for the prior of beta
tau <- 2

WomenData <- read.table("WomenAtWork.dat", header = T) # read data from .dat file
Nobs <- dim(WomenData)[1] # number of observations
y <- WomenData$Work

X <- as.matrix(WomenData[,Covs]);
Xnames <- colnames(X)
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 <- (1/lambda)*(tau^2)*diag(Npar) # Prior covariance matrix

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

# Log posterior for logistic regression
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);
  #posterior = likelihood*prior, log posterior = log likelihood + log prior
  return(logLik + logPrior)
}

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

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

# Printing the results to the screen
names(OptimRes$par) <- Xnames # Naming the coefficient by covariates
```

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

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

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

```
##             [,1]
## [1,] -0.04036943
## [2,] -0.03730689
## [3,]  0.17868950
## [4,]  0.12073637
## [5,] -0.04618995
## [6,] -1.47248930
## [7,] -0.02014458
## attr(,"names")
## [1] "Constant"    "HusbandInc"  "EducYears"   "ExpYears"    "Age"
## [6] "NSmallChild" "NBigChild"
```

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

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

```r
print(approxPostStd)
```

```
##    Constant  HusbandInc   EducYears    ExpYears         Age NSmallChild
##  1.38198486  0.02198474  0.08920960  0.03335982  0.02747315  0.47746764
##   NBigChild
##  0.16401959
```

```r
glmModel <- glm(Work ~ 0 + ., data = WomenData, family = binomial)
print("---- comparing with GLM ----")
```

```
## [1] "---- comparing with GLM ----"
```

```r
print(glmModel)
```

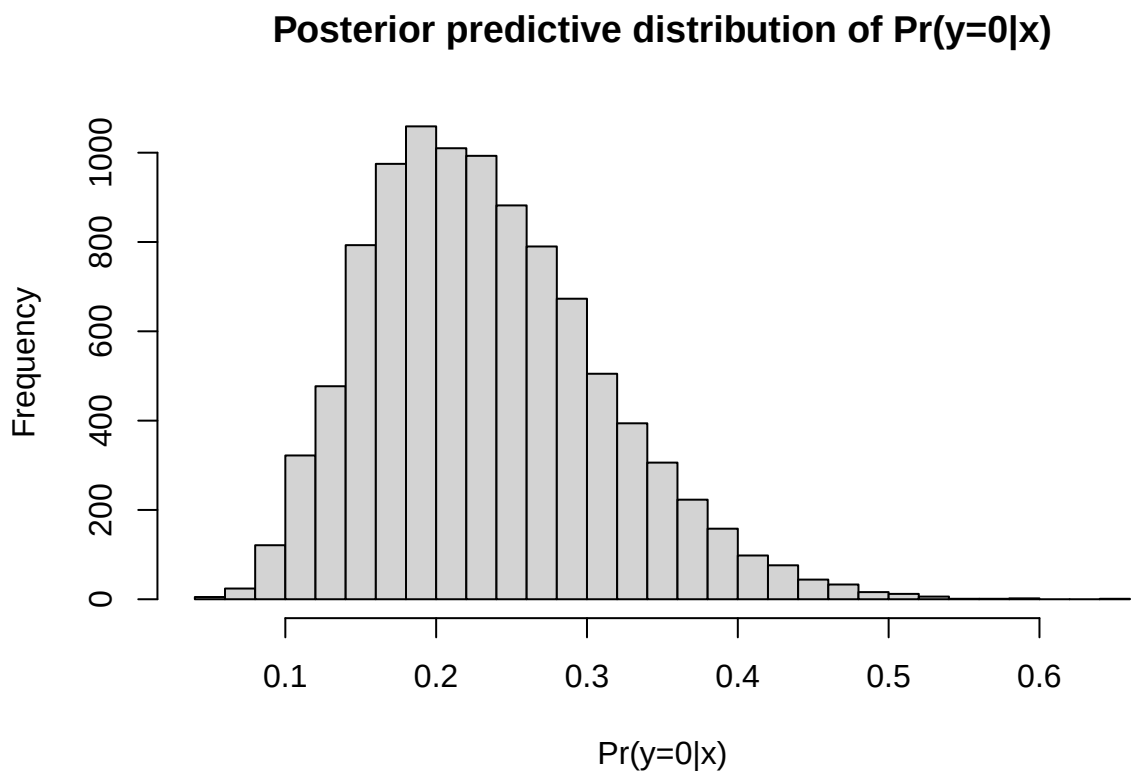
```
##
## Call:  glm(formula = Work ~ 0 + ., family = binomial, data = WomenData)
##
## Coefficients:
##    Constant   HusbandInc    EducYears     ExpYears          Age  NSmallChild
##     0.02263     -0.03796      0.18447      0.12132     -0.04858     -1.56485
##   NBigChild
##    -0.02526
##
## Degrees of Freedom: 132 Total (i.e. Null);  125 Residual
## Null Deviance:       183
## Residual Deviance: 146.7     AIC: 160.7
```

NSmallChild has the biggest absolute value out of all the posterior means, hence it is the feature with the most importance for the probability that a woman works.

b)

Use your normal approximation to the posterior from (a). 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. [Hints: The R package mvtnorm will be useful. Remember that $Pr(y = 0|x)$ can be calculated for each posterior draw of β .]

```
SimulatePostPred <- function(Nsim,OptimRes,X){
  Npar <- dim(X)[2]
  postMean <- OptimRes$par #Posterior mean
  postCov <- solve(-OptimRes$hessian) #Posterior covariance matrix
  postDraws <- rmvnorm(Nsim, postMean, postCov) #Draws from the posterior
  postPred <- rep(0,Nsim)
  for (i in 1:Nsim){
    postPred[i] <- 1/(1+exp(-X%*%postDraws[i,])) # Pr(y=0|x)
  }
  return(postPred)
}

Nsim <- 10000
x <- c(1, 18, 11, 7, 40, 1, 1) #cst, HusbandIncome, Education, Experience, Age, NSmallChild, NBigChild
postPred <- SimulatePostPred(Nsim,OptimRes,x)
hist(postPred,main="Posterior predictive distribution of Pr(y=0|x)",xlab="Pr(y=0|x)",ylab="Frequency",
     breaks=30)
```



c)

Now, 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. [Hint: Simulate from the binomial distribution, which is the distribution for a sum of Bernoulli random variables.]

```
SimulatePostPredBin <- function(Nsim,OptimRes,X,Nwomen){
  Npar <- dim(X)[2]
  postMean <- OptimRes$par #Posterior mean
  postCov <- solve(-OptimRes$hessian) #Posterior covariance matrix
  postDraws <- rmvnorm(Nsim, postMean, postCov) #Draws from the posterior
  postPred <- rep(0,Nsim)
  for (i in 1:Nsim){
    postPred[i] <- rbinom(1, Nwomen, 1 -1/(1+exp(-X%*%postDraws[i,]))) # Pr(y=0|x) = 1 - Pr(y=1|x)
  }
  return(postPred)
}

Nsim <- 10000
Nwomen <- 13
postPred <- SimulatePostPredBin(Nsim,OptimRes,x,Nwomen)
hist(postPred,main="Posterior predictive distribution of number of women not working",
     xlab="Number of women",ylab="Frequency")
```