732A91 - Bayesian learning

Lab 2 - "Bayesian Regression and Classification"

Alexander Karlsson (aleka769)
Thijs Quast (thiqu264)

24-04-2019

Contents

```r
knitr::opts_chunk$set(echo       = TRUE,
                      message    = FALSE,
                      warning    = FALSE,
                      fig.height = 3,
                      fig.keep   = TRUE,
                      fig.align  = 'c')

library(ggplot2)
library(dplyr)
library(tidyr)
library(mvtnorm)
library(lubridate)
library(gridExtra)

theme_set(theme_minimal())
```

Question 1 - Linear and polynomial regression

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

$time = \frac{the\ number\ of\ days\ since\ beginning\ of\ year}{366}$.

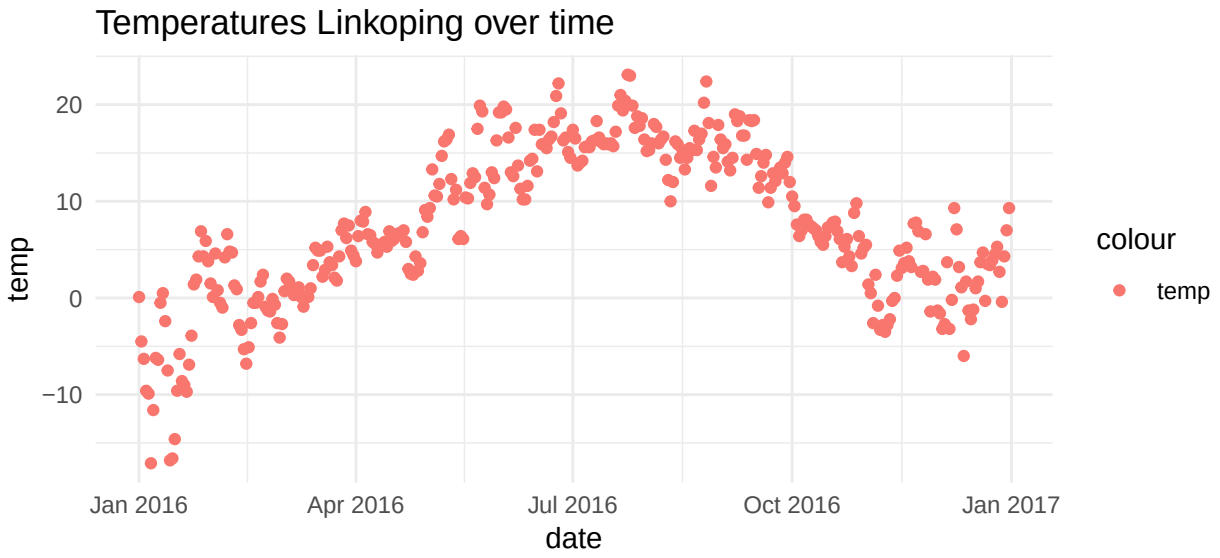
The task is to perform a Bayesian analysis of a quadratic regression:

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

1a) - Determining prior distribution of model parameters

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, 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-*χ^2 simulator from Lab 1.]

```
set.seed(12345)
temperatures <- read.delim("TempLinkoping.txt")
temperatures$date <- as_date(x = round(temperatures$time*366), origin="2015-12-31")
plot_temperatures <- ggplot(temperatures, aes(x = date, y = temp, col="temp")) + geom_point() +
  ggtitle("Temperatures Linkoping over time")
plot_temperatures
```



The conjugate priors for the linear regression model can be seen below:

$$\beta|\sigma^2 \sim N(\mu_0,\ \sigma^2\Omega_0^{-1})$$
$$\sigma^2 \sim Inv\text{-}\chi^2(\nu_0,\ \sigma_0^2)$$

These priors are neat because the posterior distribution(s) can be written as:

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

where the terms are calculated as:

$$\begin{aligned}
\mu_n &= (\mathbf{X}^{'}\mathbf{X})^{-1}(\mathbf{X}^{'}\mathbf{X}\hat{\beta} + \Omega_0\mu_0) \\
\Omega_n &= \mathbf{X}^{'}\mathbf{X} + \Omega_0 \\
\nu_n &= \nu_0 + n \\
\nu_n\sigma_n^2 &= \nu_0\sigma_0^2 + (\mathbf{y}^{'}\mathbf{y} + \mu_0^{'}\Omega_0\mu_0 - \mu_n^{'}\Omega_n\mu_n) \Rightarrow \\
&\Rightarrow \sigma_n^2 = \frac{\nu_0\sigma_0^2 + (\mathbf{y}^{'}\mathbf{y} + \mu_0^{'}\Omega_0\mu_0 - \mu_n^{'}\Omega_n\mu_n)}{\nu_n}
\end{aligned}$$

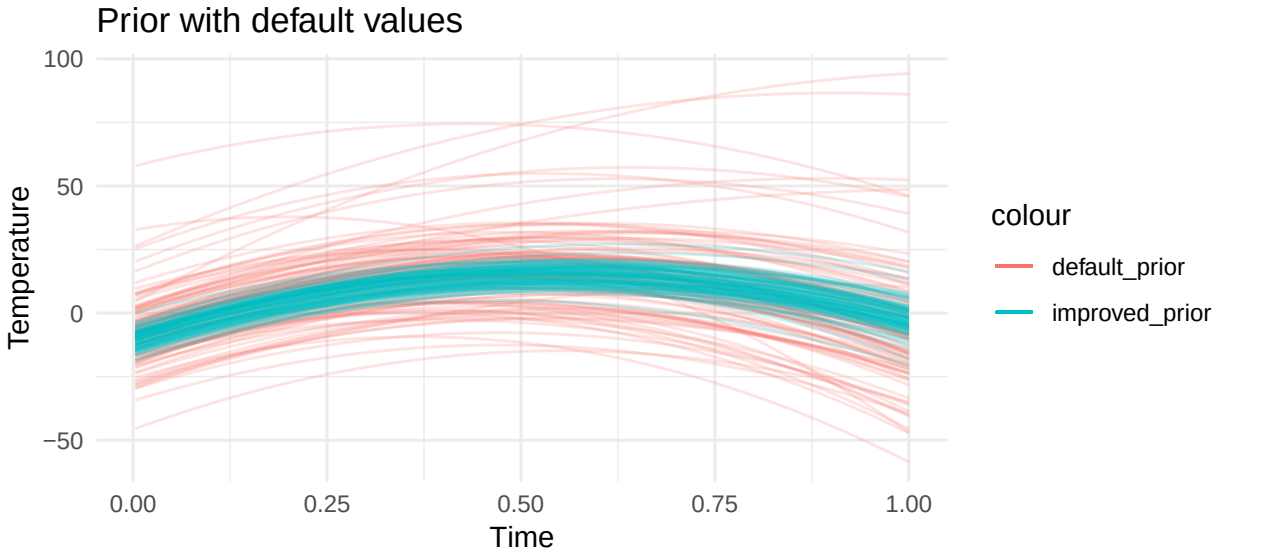
The term μ_n can with a little imagination (and cheating mathematics) be seen as the weighted mean of the prior beliefs and the actual data.

By sampling the sigma prior first, we can then later sample the beta according to this prior. The estimates for y are then computed by multiplying the prior betas with the X_matrix (the data).

```
X_matrix <- as.matrix(cbind(1, temperatures$time, (temperatures$time)^2))
mu_0 <- matrix(c(-10, 100, -100), nrow = 3, ncol = 1)
identity <- matrix(0, nrow = 3, ncol = 3)
diag(identity) <- 1
omega_0 <- 0.01 * identity
v_0 <- 4
sigma_0_2 <- 1

prior_default <- data.frame("time" = seq(1/366, 1, by = 1/366), "default_prior" = 0)

chi_squared <- rchisq(1, df = v_0)
sigma_2_prior <- (sigma_0_2 * v_0) / chi_squared
beta_prior <- rmvnorm(1, mean = mu_0, sigma = solve(omega_0)*sigma_2_prior)
prior_default$default_prior <- X_matrix%*%t(beta_prior)

plot <- ggplot(data = prior_default, aes(x = time, y = default_prior,
                                         col="default_prior")) + geom_line(alpha=0.2) +
  ggtitle("Prior with default values") + ylab("Temperature") + xlab("Time")
```

```
for (i in 1:100){
  chi_squared <- rchisq(1, df = v_0)
  sigma_2_prior <- (sigma_0_2 * v_0) / chi_squared
  beta_prior <- rmvnorm(1, mean = mu_0, sigma = solve(omega_0)*sigma_2_prior)
  df <- data.frame("time" = seq(1/366, 1, by = 1/366), "prior" = 0)
  df$prior <- X_matrix%*%t(beta_prior)
  plot <- plot + geom_line(data = df, aes(x = time, y = prior,
                                          col="default_prior"), alpha =0.2)
}

for (i in 1:100){
  mu_0 <- lm(temp ~ time + I(time^2), data = temperatures)$coefficients
  omega_0 <- identity
  v_0 <- 5
  sigma_0_2 <- 10
```

```r
  chi_squared <- rchisq(1, df = v_0)
  sigma_2_prior <- (sigma_0_2 * v_0) / chi_squared
  beta_prior <- rmvnorm(1, mean = mu_0, sigma = solve(omega_0)*sigma_2_prior)
  df <- data.frame("time" = seq(1/366, 1, by = 1/366),
                   "improved_prior" = 0)
  df$improved_prior <- X_matrix%*%t(beta_prior)
  plot <- plot + geom_line(data = df, aes(x = time,
                                          y = improved_prior, col="improved_prior"), alpha = 0.2)
}

plot
```



With the selected priors, we seem to catch the general trend swedish seasonal weather. However, some lines appear to be way of what can be seen as reasonable. We can check what parameters a frequentist polynomial model of order 2 would generate and use those parameters as a prior for μ_0. In addition, the parameter ν_0 is increased, as this gives more weight to the prior beliefs.

The curves seem to follow what resembles a normal year in sweden. It is not clear exactly what kind of measurements these temperatures are (average, at a specific time, or similar...) but it seems as the predicted temperatures for july are at least a little bit low. Obviously, these temperatures differ, but we don't want a model that overfits.

1b) - Simulation from joint posterior distribution of β_0

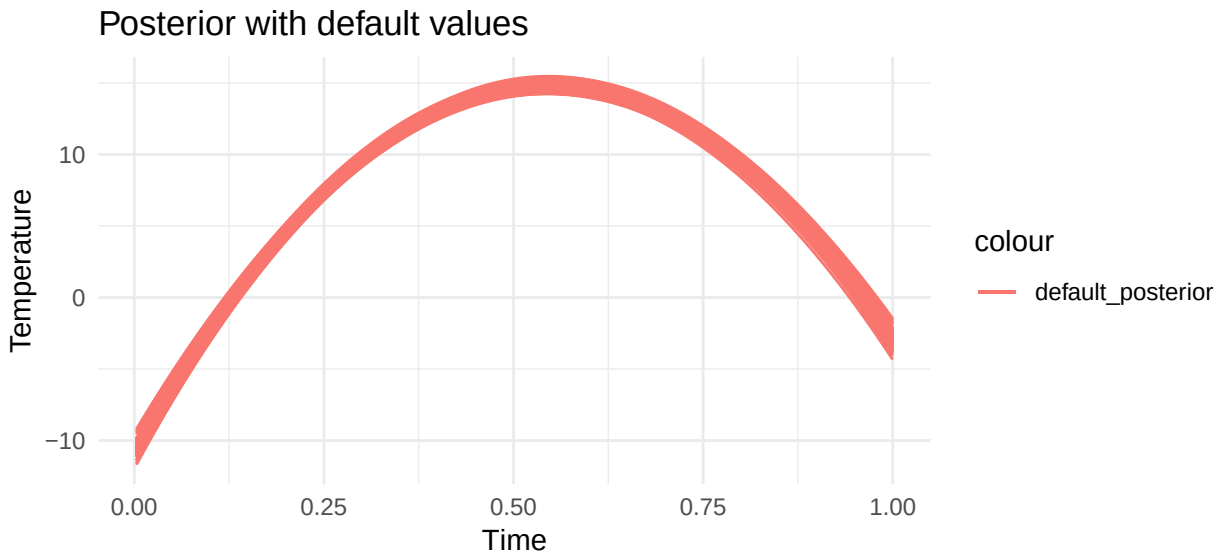
Write a program that *simulates from the posterior distribution* of β_0, β_1, β_2 and σ^2. Plot the marginal posteriors for each parameter as a histogram. Also produce another figure with a scatter plot of the temperature data and overlay a curve for the posterior median of the regression function $f(time) = \beta_0 + \beta_1 \cdot time + \beta_2 \cdot time^2$, computed for every value of *time*. Also overlay curves for the lower 2.5 % and upper 97.5 % posterior credible interval for $f(time)$. That is, compute the 95 % 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?

```
n <- nrow(temperatures)
y <- temperatures$temp
XtX <- t(X_matrix)%*%X_matrix
```

```
mu_n <- solve(XtX + omega_0) %*% (XtX%*%mu_0 + omega_0 %*% mu_0)
omega_n <- XtX + omega_0
v_n <- v_0 + n
sigma_n_2 <- ((v_0*sigma_0_2) + t(y)%*%y +
                t(mu_0)%*%omega_0%*%mu_0 -
                t(mu_n)%*%omega_n%*%mu_n) / v_n
```

```
# posterior draws
chi_squared <- rchisq(1, df = v_n)
sigma_2_posterior <- as.numeric(sigma_n_2 * v_n) / chi_squared
beta_posterior <- rmvnorm(1, mean = mu_n,
                          sigma = sigma_2_posterior*solve(omega_n))
```

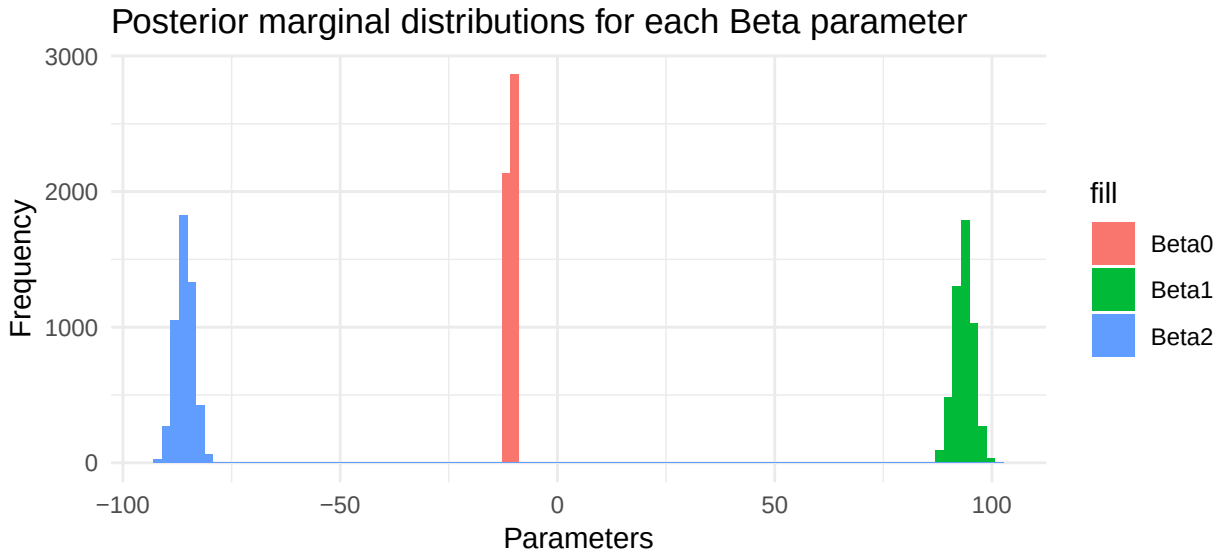
```
posterior_default <- data.frame(
  "time" = seq(1/366, 1, by = 1/366), "default_posterior" = 0)
posterior_default$default_posterior <- X_matrix%*%t(beta_posterior)

plot_posterior <- ggplot(data = posterior_default,
                         aes(x = time, y = default_posterior, col="default_posterior")) +
  geom_line() + ggtitle("Posterior with default values") + ylab("Temperature") + xlab("Time")


for (i in 1:100){
  chi_squared <- rchisq(1, df = v_n)
  sigma_2_posterior <- as.numeric(sigma_n_2 * v_n) / chi_squared
  beta_posterior <- rmvnorm(1, mean = mu_n,
                            sigma = sigma_2_posterior*solve(omega_n))
  df <- data.frame("time" = seq(1/366, 1, by = 1/366), "posterior" = 0)


  df$posterior <- X_matrix%*%t(beta_posterior)
  plot_posterior <- plot_posterior + geom_line(data = df,
                                               aes(x = time, y = posterior, col="default_posterior"))
}

plot_posterior
```



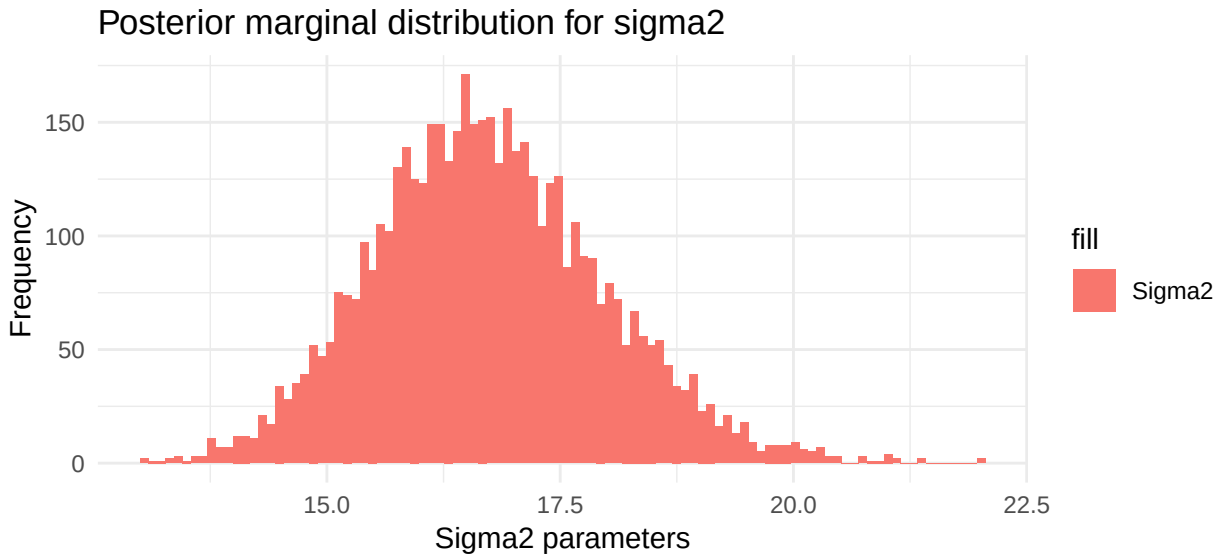
The posterior distribution seems a bit unsymmetric, probably this is caused by the data. The temperatures at the end of the year are slightly higher than at the beginning of the year, expected is that this raises the posterior curve at the end of the year.

```
df_parameters <- data.frame("Beta0"=0, "Beta1"=0, "Beta2"=0, "Sigma2"=0)

for (i in 1:5000){
  sigma_2_posterior <- (as.numeric(sigma_n_2) * v_n) / rchisq(1, df = v_n)
  beta_posterior <- rmvnorm(1, mean = mu_n, sigma = sigma_2_posterior*solve(omega_n))
  df_parameters[i,] <- cbind(beta_posterior, sigma_2_posterior)
}
```

```
plot_parameters <- ggplot(data = df_parameters,
                          aes(x = Beta0, fill="Beta0")) + geom_histogram(bins = 100)
plot_parameters <- plot_parameters +
  geom_histogram(aes(x = Beta1, fill="Beta1"), bins = 100)
plot_parameters <- plot_parameters +
  geom_histogram(aes(x = Beta2, fill="Beta2"), bins = 100)
plot_parameters <- plot_parameters +
  ylab("Frequency") + xlab("Parameters") +
  ggtitle("Posterior marginal distributions for each Beta parameter")
plot_parameters
```



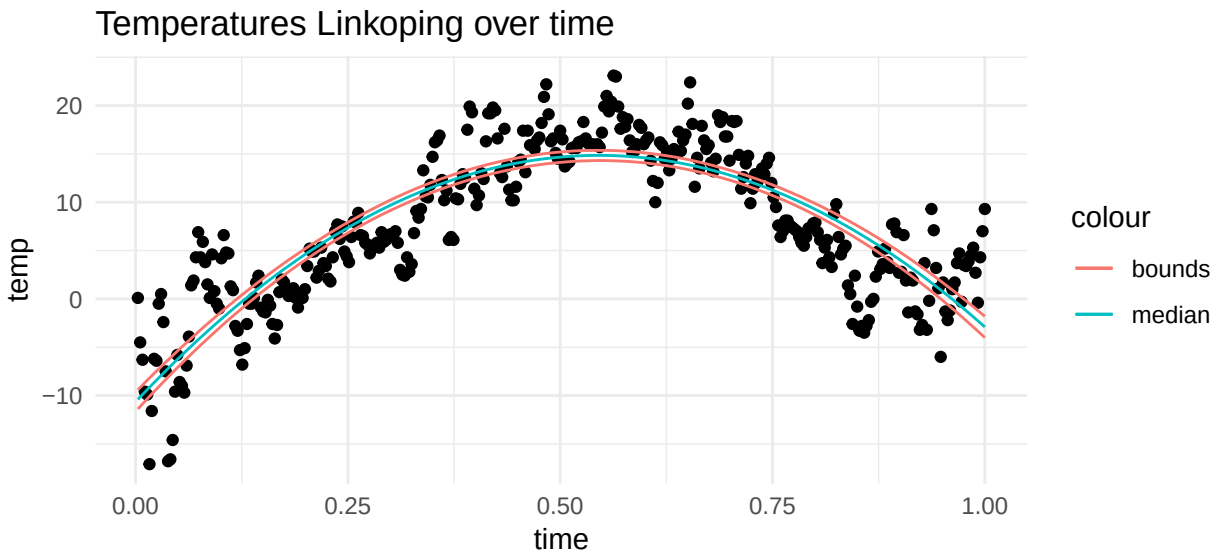

```
plot_sigma <- ggplot(data = df_parameters,
                     aes(x = Sigma2, fill="Sigma2")) + geom_histogram(bins = 100)
plot_sigma <- plot_sigma + ylab("Frequency") +
  xlab("Sigma2 parameters") + ggtitle("Posterior marginal distribution for sigma2")
plot_sigma
```



Calculations for the posterior parameters are done according to the formulas presented in 1a and the lecture slides.

```
temperatures2 <- temperatures
beta_median <- c(median(df_parameters$Beta0),
                 median(df_parameters$Beta1), median(df_parameters$Beta2))
beta_matrix <- as.matrix(df_parameters[,1:3])

temperatures_esimate <- X_matrix%*%t(beta_matrix)
temperatures2$median <- 0
temperatures2$bounds <- 0
temperatures$high <- 0
```

```r
for (i in 1:nrow(temperatures)){
  temperatures2$median[i] <- median(temperatures_esimate[i,])
}

for (i in 1:nrow(temperatures)){
  temperatures2$bounds[i] <- quantile(temperatures_esimate[i,], probs = 0.025)
}

for (i in 1:nrow(temperatures)){
  temperatures2$high[i] <- quantile(temperatures_esimate[i,], probs = 0.975)
}


plot_temperatures2 <- ggplot(temperatures2, aes(x = time, y = temp)) + geom_point() +
  ggtitle("Temperatures Linkoping over time")
plot_temperatures2 <- plot_temperatures2 + geom_line(aes(x = time, y = median, col="median"))
plot_temperatures2 <- plot_temperatures2 + geom_line(aes(x = time, y = bounds, col="bounds"))
plot_temperatures2 <- plot_temperatures2 + geom_line(aes(x = time, y = high, col="bounds"))
plot_temperatures2
```



The intervals are very tight around the median for each day. This is not strange since these intervals tell us with 95% certainty where the posterior regression line lies, not the actual points.

1c) - Simulation from posterior distribution of $\bar{x}$

> 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 $\tilde{x}$. Use the simulations in *b)* to simulate from the *posterior distribution of* $\tilde{x}$. [Hint: the regression curve is a quadratic. You can find a simple formula for $\tilde{x}$ given β_0, β_1 and β_2.]

The expression for a polynomial model of order 2 can be written as $f(x) = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot x^2$, where x is the time variable in oóur data. All densities plotted are based on this function. Thus, if this function is maximized, so is the mode. The first order derivative is shown below:

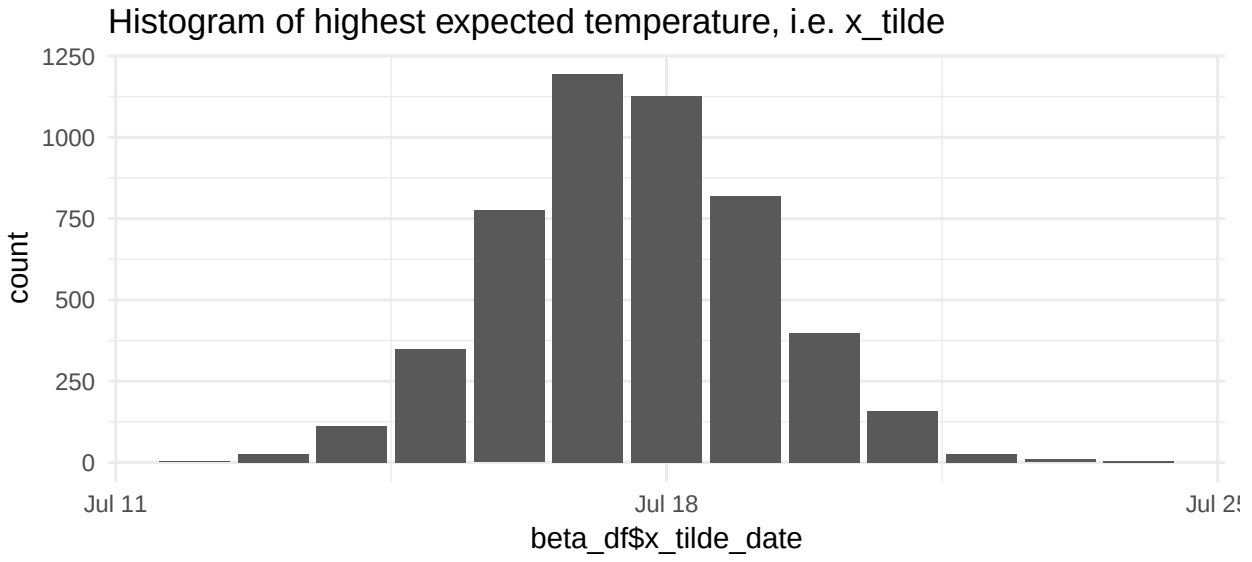
$$f(x) = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot x^2 \quad \Rightarrow \quad f^{'}(x) = \frac{\partial f}{\partial x} = \beta_1 + 2 \cdot \beta_2 \cdot x$$

Maximizing this concave function yields the maximum $\tilde{x}$:

$$f^{'}(x) = 0 \Leftrightarrow \beta_1 + 2 \cdot \beta_2 \cdot x = 0 \Leftrightarrow \tilde{x} = -\frac{\beta_1}{2 \cdot \beta_2}$$

This is relatively easy to program, all that is needed is the matrix with our beta coefficients from 1b. To make sense of this density, we transform the $\tilde{x}$ into dates from 2016.

```
beta_df <- as.data.frame(beta_matrix)

beta_df$x_tilde <- -(beta_df$Beta1)/(2*beta_df$Beta2)
beta_df$x_tilde_date <- as_date(x=round(beta_df$x_tilde*366), origin="2015-12-31")

x_tilde_plot <- ggplot(data = beta_df, aes(x = beta_df$x_tilde_date)) + geom_bar(stat = "count") +
  ggtitle("Histogram of highest expected temperature, i.e. x_tilde")
x_tilde_plot
```

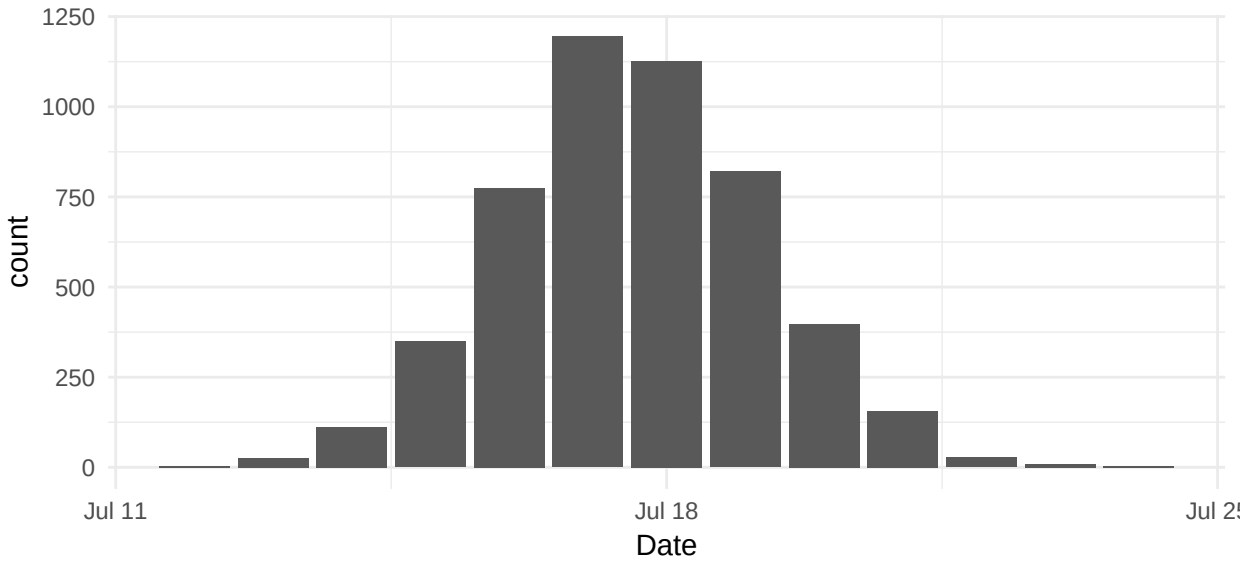


We observe that the highest temperatues occur in July.

```
# Simulation (bayesian) approach:
beta_df <- beta_df[, 1:3]
maximum_df <- as.data.frame(X_matrix%*%t(beta_df))

max_temperature <- apply(maximum_df, 2, which.max)
```

```
max_temperature <- as_date(x = max_temperature, origin="2015-12-31") %>% as.data.frame()
colnames(max_temperature) <- c("Date")
```

```
ggplot(max_temperature, aes(x = Date)) + geom_bar(stat = "count")
```



In answering this question we first did it analytically, however in the scope of this course the Bayesian way seems more appropriate. The highest expected temperatures center around the 18th of July, each year, when logically reasoning it, makes sense that the temperatures are generally the highest in summer.

1d) - Polynomial model of order 7

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.]

By combining the beta parameters in μ_0 to be close to 0 combined with a large λ_0 we would obtain a tight conditional prior distribution:

$$\beta|\sigma^2 \overset{iid}{\sim} N(0,\ \sigma^2[\lambda_0 \cdot \mathbf{I}]^{-1})$$

The idea is to use a bayesian approach to regularization. When doing so, the 'unimportant' beta coefficients will remain close to 0 and those that have a higher impact will 'stretch away' from 0.

Note: a large value of λ_0 yields a tight distribution because we take the inverse of $\lambda_0 \cdot \mathbf{I}$.

Question 2 - Posterior approximation for classification with logistic regression

The dataset `WomenWork.dat` contains $n = 200$ observations (i.e. women) on the following nine variables:

Variable	Data type	Meaning	Role
Work	Binary	Whether or not the woman works	Response
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
ExpYears2	Numeric	(Years of experience/10)^2	Feature
Age	Counts	Age	Feature
NSmallChild	Counts	Number of child leq 6 years in household	Feature
NBigChild	Counts	Number of child >6 years in household	Feature

2a) - Fit logistic regression model

Consider the logistic regression

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

where y is the binary variable with $y = 1$ if the woman works and $y = 0$ if she does not. $\mathbf{x}$ 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.

```
women <- read.csv("womenWork.dat", sep = "")
glmModel <- glm(Work ~ 0 + ., data = women, family = binomial)
summary(glmModel)
```

```
##
## Call:
## glm(formula = Work ~ 0 + ., family = binomial, data = women)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1662  -0.9299   0.4391   0.9494   2.0582
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## Constant     0.64430    1.52307   0.423 0.672274
## HusbandInc  -0.01977    0.01590  -1.243 0.213752
## EducYears    0.17988    0.07914   2.273 0.023024 *
## ExpYears     0.16751    0.06600   2.538 0.011144 *
## ExpYears2   -0.14436    0.23585  -0.612 0.540489
## Age         -0.08234    0.02699  -3.050 0.002285 **
## NSmallChild -1.36250    0.38996  -3.494 0.000476 ***
## NBigChild   -0.02543    0.14172  -0.179 0.857592
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 277.26  on 200  degrees of freedom
## Residual deviance: 222.73  on 192  degrees of freedom
## AIC: 238.73
##
## Number of Fisher Scoring iterations: 4
```

The approach above is a frequentist appraoch to a model estimation. HusbandInc, ExpYears2, Age, NSmallChild and NBigChild have a negative effect on the probability that the mother is working or not, although not all of them have significant p-values.

2b) - Approximate posterior 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

$$\beta|\mathbf{y},\mathbf{X} \sim N\Big(\tilde{\beta}, J_{\mathbf{y}}^{-1}(\tilde{\beta})\Big), \qquad J(\tilde{\beta}) = -\frac{\partial^2 ln[p(\beta|\mathbf{y})]}{\partial\beta\partial\beta^T}\Big|_{\beta=\tilde{\beta}}$$

where $\tilde{\beta}$ is the posterior mode and $J(\tilde{\beta})$ is the observed Hessian evaluated at the posterior mode. Note that $\frac{\partial^2 ln[p(\beta|\mathbf{y})]}{\partial\beta\partial\beta^T}$ is an 8×8 matrix with second derivatives on the diagonal and cross-derivatives $\frac{\partial^2 ln[p(\beta|\mathbf{y})]}{\partial\beta_i\partial\beta_j}$ 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 $\tilde{\beta}$ and $J(\tilde{\beta})$ 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 your 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 $\tilde{\beta}$ and $J_{\mathbf{y}}^{-1}(\tilde{\beta})$ for the `WomenWork` data. Compute an approximate 95 % credible interval for the variable NSmallChild. Would you say that this feature is an important determinant of the probability that a women works?

```
y <- as.vector(women[,1])
X <- as.matrix(women[, 2:ncol(women)])
covNames <- names(women)[2:ncol(women)]
nPara <- dim(X)[2]
```

```
# Prior:
tau <- 10
mu <- rep(0, nPara)
Sigma <- tau^2 * diag(nPara)  # diag(nPara) creates the identity matrix
```

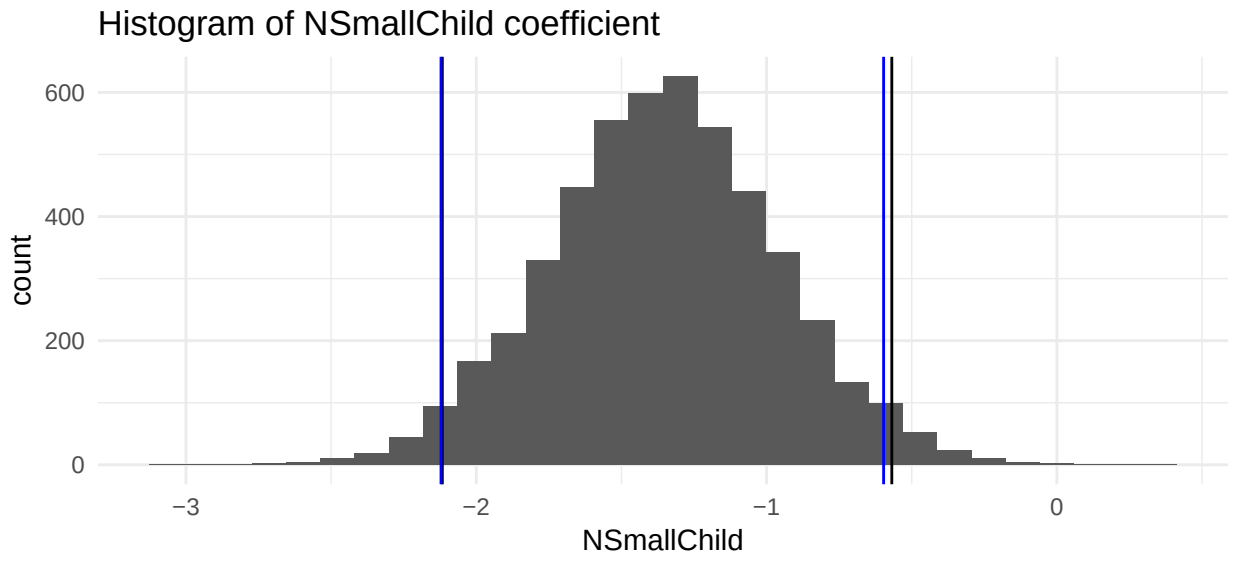
```
LogPostLogistic <- function(betaVect, y, X, mu, Sigma){
  nPara <- length(betaVect)
  linPred <- X%*%betaVect

  logLik <- sum(linPred*y - log(1+exp(linPred)))
  if (abs(logLik)==Inf) logLik = -20000

  logPrior <- dmvnorm(betaVect, matrix(0, nPara, 1), Sigma, log = TRUE)

  return(logLik + logPrior)
}
```

```
initVal <- as.vector(rep(0, dim(X)[2]))
logPost <- LogPostLogistic
```

```
OptimResults <- optim(initVal, logPost, gr = NULL, y, X, mu, Sigma, method = c("BFGS"),
                      control = list(fnscale=-1), hessian = TRUE)
```

```
PostMode <- OptimResults$par
postCov <- -solve(OptimResults$hessian)
names(PostMode) <- covNames
approxPostStd <- sqrt(diag(postCov))
names(approxPostStd) <- covNames
```

```r
NSmallChild <- data.frame("NSmallChild" = rnorm(5000,
                                                mean = PostMode["NSmallChild"],
                                                sd = approxPostStd["NSmallChild"]))

confidence_intervals <- c(PostMode["NSmallChild"] - 1.96*approxPostStd["NSmallChild"],
                          PostMode["NSmallChild"] + 1.96*approxPostStd["NSmallChild"])

NSmallChild_histogram <- ggplot(data = NSmallChild, aes(x = NSmallChild)) +
  geom_histogram() + ggtitle("Histogram of NSmallChild coefficient")
quantiles <- quantile(NSmallChild$NSmallChild, probs = c(0.025, 0.975))

#Simulated approach
NSmallChild_histogram <- NSmallChild_histogram + geom_vline(xintercept = quantiles[1])
NSmallChild_histogram <- NSmallChild_histogram + geom_vline(xintercept = quantiles[2])

# Theoretical approach
NSmallChild_histogram <- NSmallChild_histogram +
  geom_vline(xintercept = confidence_intervals[1], color="Blue")
NSmallChild_histogram <- NSmallChild_histogram +
  geom_vline(xintercept = confidence_intervals[2], color="Blue")

NSmallChild_histogram
```



2c) - Simulate from predictive distribution of the response variable

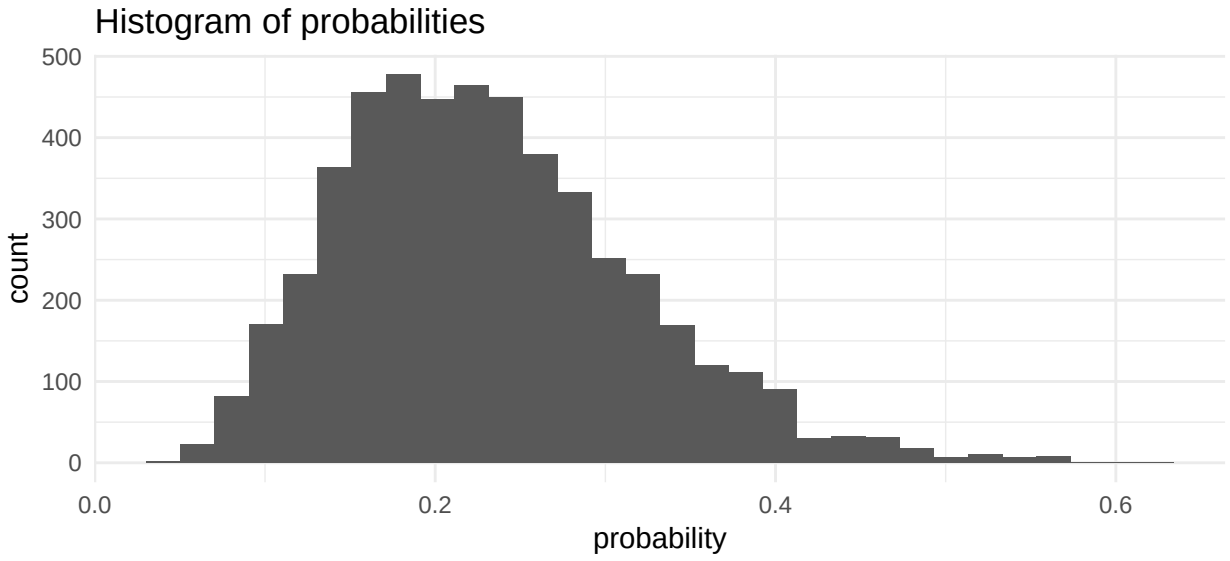
> Write a function that simulates from the predictive distribution of the response variable in a logistic regression. Use your normal approximation from 2(b). Use that function to simulate and plot the predictive distribution for the `Work` variable for a 40 year old woman, with two children (3 and 9 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 discus- sion on how Bayesian prediction can be done by simulation.]

```
n <- nrow(women)
predictive_y <- function(x_data){
  betas <- rmvnorm(1, mean = PostMode, sigma = postCov)
  betas <- matrix(betas, ncol = 1)
  numerator <- exp(t(example_c)%*%betas)
  denominator <- 1 + numerator
  probability <- numerator/denominator
  return(probability)
}
```

```
example_c <- matrix(c("Constant" = 1, "HusbandInc" = 10,
                      "EducYears"=8, "ExpYears"=10,
                "ExpYears2"=1, "Age"=40, "NSmallChild"=1,
                "NBigChild"=1), nrow=8, ncol = 1)

predictive_y(example_c)
```

```
##           [,1]
## [1,] 0.1474939
```

```
probability <- c()

for (i in 1:5000){
  probability[i] <- predictive_y(example_c)
}

probability <- as.data.frame(probability)
ggplot(data = probability, aes(x=probability))+
  geom_histogram() + ggtitle("Histogram of probabilities")
```



If we now say that if probability is smaller than 0.5, we predict she will not work, and if probability is greater than 0.5 means she will work we get the following predictions:

```
probability[probability > 0.5] <- 1
probability[probability <= 0.5] <- 0
colnames(probability) <- c("work")

ggplot(data = probability, aes(work)) + geom_bar() +
  ggtitle("Expected outcomes according to p<0.5 is not working")
```