# Bayesian Learning - Lab 02

*Lakshidaa Saigiridharan (laksa656) and Maximilian Pfundstein (maxpf364)*

*2019-04-29*

## Contents



## 1 Linear and Polynomial Regression

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

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

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

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

a) Determining the prior distribution of the model parameters. Use the conjugate prior for the linear regression model. Your task is to set the prior hyperparameters $\mu_0, \Omega_0, \nu_0$ and $\sigma_0^2$ to sensible values. Start with $\mu_0 = (-10, 100, -100)^T$ , $\Omega_0 = 0.01 \cdot I_3$, $\nu_0 = 4$ and $\sigma_0^2 = 1$. Check if this prior agrees with your prior opinions by simulating draws from the joint prior of all parameters and for every draw compute the regression curve. This gives a collection of regression curves, one for each draw from the prior. Do the collection of curves look reasonable? If not, change the prior hyperparameters until the collection of prior regression curves do agree with your prior beliefs about the regression curve. [Hint: the R package `mvtnorm` will be handy. And use your $Inv - \chi^2$ simulator from Lab 1.]

b) Write a program that simulates from the joint posterior distribution of $\beta_0, \beta_1, \beta_2$ and $\sigma^2$. 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?

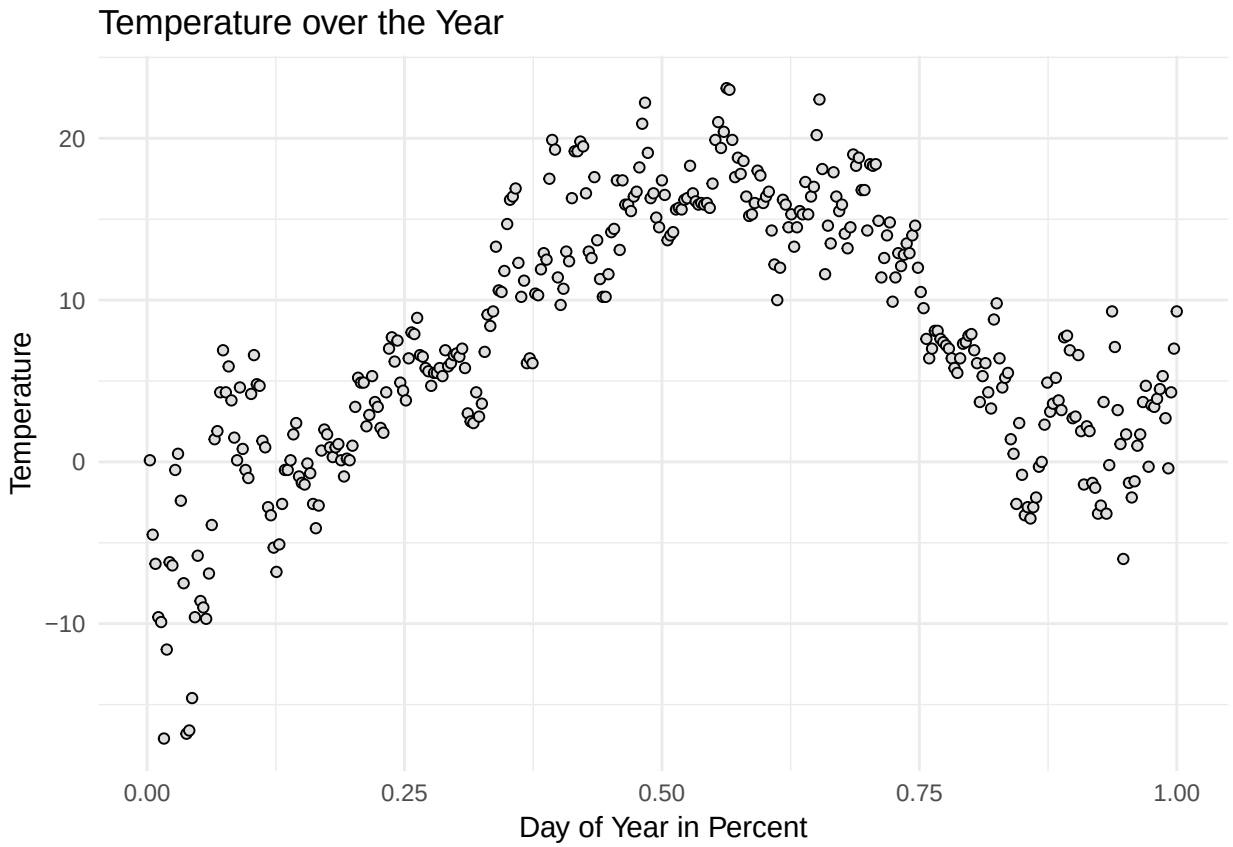
c) It is of interest to locate the time with the highest expected temperature (that is, the time where f(time) is maximal). Let???s call this value $\tilde{x}$ Use the simulations in b) to simulate from the *posterior distribution of* $\tilde{x}$. [Hint: the regression curve is a quadratic. You can find a simple formula for $\tilde{x}$ given $\beta_0, \beta_1$ and $\beta_2$.]

d) 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 $\mu_0$ and $\Omega_0$ in a smart way.]

## 1.1 Determining the prior Distribution of the Model Parameters

Let's first take a look at the data to get a feeling for it.

| time | temp |
|---|---|
| 0.0027322 | 0.1 |
| 0.0054645 | -4.5 |
| 0.0081967 | -6.3 |
| 0.0109290 | -9.6 |
| 0.0136612 | -9.9 |
| 0.0163934 | -17.1 |

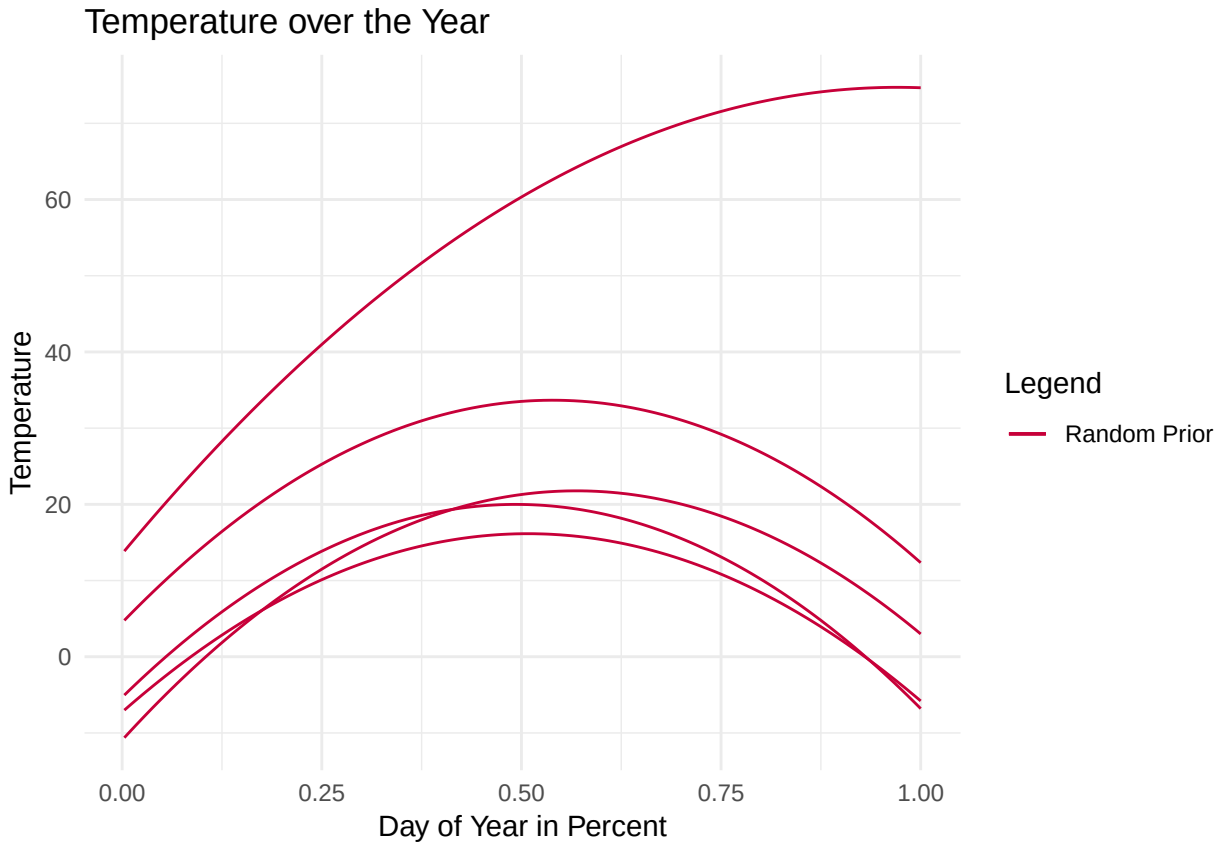



We save the initial parameters to variables and provide a function which calculates $\sigma^2$ and $\beta$ from the given parameters, which we use a the parameters for the prior. The calculation is done as follows:

1. First draw $\sigma^2$ from $Inv - \chi^2(\nu_0, \sigma_0^2)$

2. Then draw $\beta$ from $\mathcal{N}(\mu_0, \sigma^2\Omega_0^{-1})$
3. Calculate the prior according to $\hat{y} = X\beta$ where $X = (1, \text{temp}, \text{temp}^2)$. The first 1 is the intercept.

```r
# Initial Parameters
mu_0 = c(-10, 100, -100)
omega_0 = 0.01 * diag(3)
nu_0 = 4
sigma_sq_0 = 1

# Helper Variables
time = temperatures_linkoeping$time
temp = temperatures_linkoeping$temp
X = matrix(c(rep(1, length(time)), time, time^2), ncol = 3)
Y = matrix(temperatures_linkoeping$temp)
n = length(time)
m = 5 # Number of regression curves

sample_model_params = function(mu_0, omega_0, nu_0, sigma_sq_0) {

  # Taken from slides (set 5, page 7)
  sigma_sq = rinvchisq(n = 1, df = nu_0, scale = sigma_sq_0)
  beta = mvrnorm(n = 1, mu = mu_0, Sigma = sigma_sq * solve(omega_0))

  return(list(sigma_sq = sigma_sq, beta = beta))
}

predict_using_model_params = function(beta, X) {

  # Predicted temperatures
  return(X %*% matrix(beta))
}

# This matrix holds the true X and Y and then the simulated priors
priors = matrix(0, ncol = m + 2, nrow = n)
priors[,1] = time
priors[,2] = temp

# Simulate m times for plotting
for (i in 3:(m+2)) {
  model_params = sample_model_params(mu_0, omega_0, nu_0, sigma_sq_0)
  priors[,i] = predict_using_model_params(model_params$beta, X)
}
```




We can see that the chosen parameters do not seem optimal. Their variance seems to high and the temperatures should be lower at the end of the year. We therefore select the following new parameters:

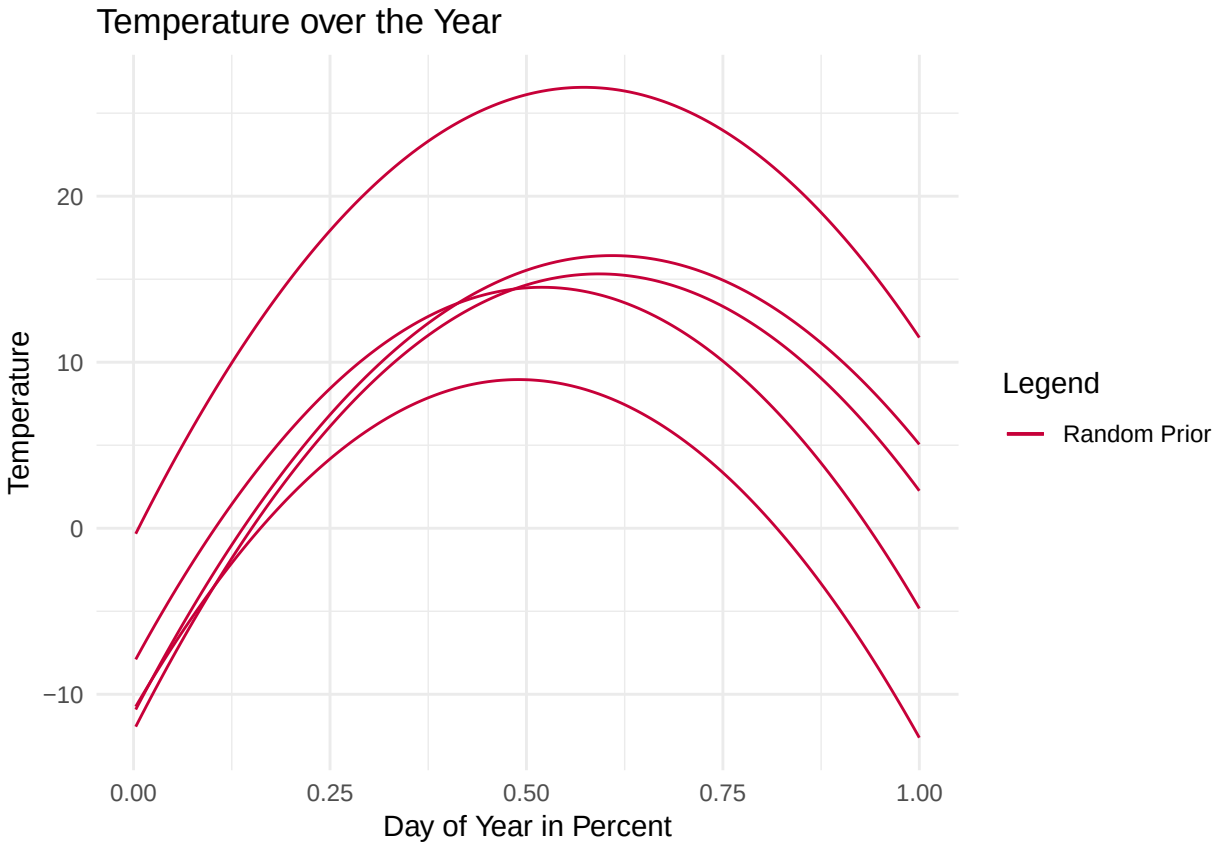
```
# Adjusted Parameters
mu_0 = c(-8, 90, -82)
omega_0 = 0.2 * diag(3)
nu_0 = 4
sigma_sq_0 = 2
```

$\mu_0$**:** We want to slightly adjust the intercept, so we set the first entry of $\mu_0$ to -8 (degrees). For the second linear regression coefficient we choose to set it to 90 as we think that the temperature is a little bit warmer in the second half of the year and to capture the warm spike in summer. For the last linear regression coefficient we chose a value that pulls the curve down again on the right side so that it has a smooth transition between December and January.

$\Omega_0$**:** The variance seems too high as we don't think the mean temperature will fluctuate that much, we thereby decrease it to 0.2 (decrease as the inverse is taken).

$\sigma^2$**:** We increase the variance of the error to allow for more fluctuation (and increasing uncertainty).

```
priors = matrix(0, ncol = m + 2, nrow = n)
priors[ ,1] = time
priors[ ,2] = temp

# Simulate m times for plotting
for (i in 3:(m+2)) {
  model_params = sample_model_params(mu_0, omega_0, nu_0, sigma_sq_0)
  priors[,i] = predict_using_model_params(model_params$beta, X)
}
```




We see that the parameters make the prior fit better to the observed (optical inspected) data.

## 1.2 Simulating from the Joint Posterior Distribution

For drawing $\beta$ and $\sigma$ from the posterior distribution we first have to calculate the different parameters:

- $\mu_n = (X^T X + \Omega_0)^{-1}(X^T X\hat{\beta} + \Omega_0\mu_0)$
- $\Omega_n = X^T X + \Omega_0$
- $\nu_n = \nu_0 + n$
- $\nu_n\sigma_n = \nu_0\sigma_0^2 + (y^T y + \mu_0^T\Omega_0\mu_0 - \mu_n^T\Omega_n\mu_n)$

We can then do:

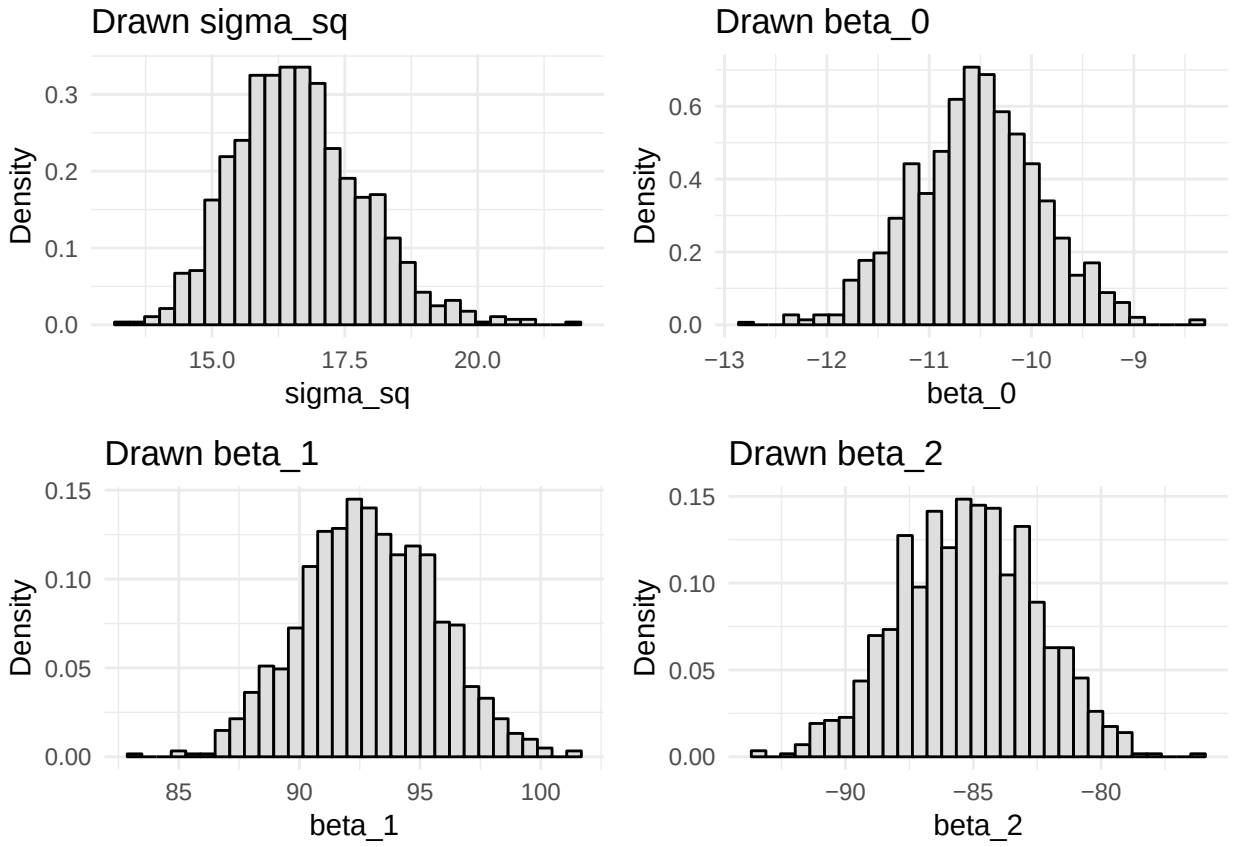
1. $\sigma^2 \sim Inv - \chi^2(\nu_n, \sigma_n^2)$
2. $\beta|\sigma^2 \sim \mathcal{N}(\mu_n, \sigma^2\Omega_n^{-1})$

We draw a new sample of parameters for the prior which we will use. We then define all the parameters used and create a function for sampling the posterior parameters.

```
##############################################################################
# Exercise 1.b)
##############################################################################

# Simulate prior
model_params = sample_model_params(mu_0, omega_0, nu_0, sigma_sq_0)

# Beta Hat
#beta_hat = model_params$beta
beta_hat = as.vector(lm(temp ~ time + I(time^2),
```

```
                    data = temperatures_linkoeping)$coefficients)

# Posterior parameters
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_and_sigma_sq_n = nu_0 * sigma_sq_0 +
  (t(Y) %*% Y + (t(mu_0) %*% omega_0 %*% mu_0) - (t(mu_n) %*% omega_n %*% mu_n))
sigma_sq_n = as.numeric(nu_n_and_sigma_sq_n / nu_n)
```

The following plots show the samples parameters and their corresponding histogram.



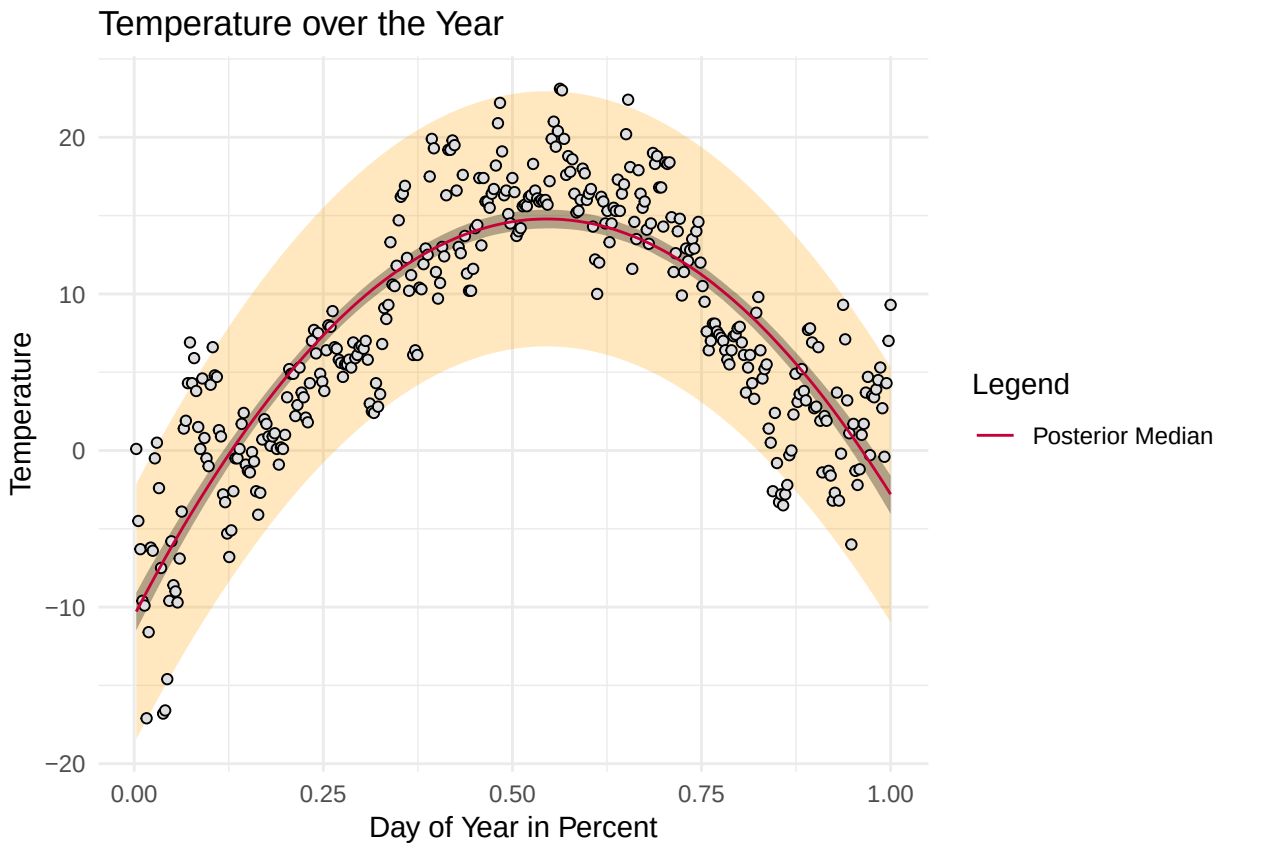

The following code predicts using the previously sampled parameters. It then calculates the median for each point of time in our grid.

```
posterior_predictions = matrix(0, ncol = n, nrow = simulations)

for (i in 1:simulations) {
  beta = unlist(df[i,2:4])
  posterior_predictions[i,] = predict_using_model_params(beta, X)
}

posterior_prediction_median = apply(posterior_predictions, 2, median)
posterior_prediction_mean = apply(posterior_predictions, 2, mean)
```

Looking at the plot we see that the posterior looks good. The requested credible interval can also be seen. The credible interval does not contain all of the data points as it just shows the interval for the median. If we

would like to have the credible interval to cover most of the points, thus work on the general prediction of the data points, we would have to count in the variance as well. $\sigma_n^2$ is equal to 16.5847736, so $\sigma = 4.0724407$. If we take $2\sigma$ we should be able to capture around 95% of the points. The orange area thus shows the 95% credible intercal for the whole data.

```
ci = apply(posterior_predictions, 2, quantile, probs = c(0.025, 0.975))
lower = ci[1,]
upper = ci[2,]

upper_two_sigma = posterior_prediction_mean + 2 * sqrt(sigma_sq_n)
lower_two_sigma = posterior_prediction_mean - 2 * sqrt(sigma_sq_n)
```




## 1.3 Simulating from the Posterior Distribution

We can use the previous data set and simply take the maximum of each. Then we can plot the histrogram.

We first take the derivate of the quadratic regression:

$$\frac{temp}{\partial time} = \beta_1 + 2\beta_2 time$$

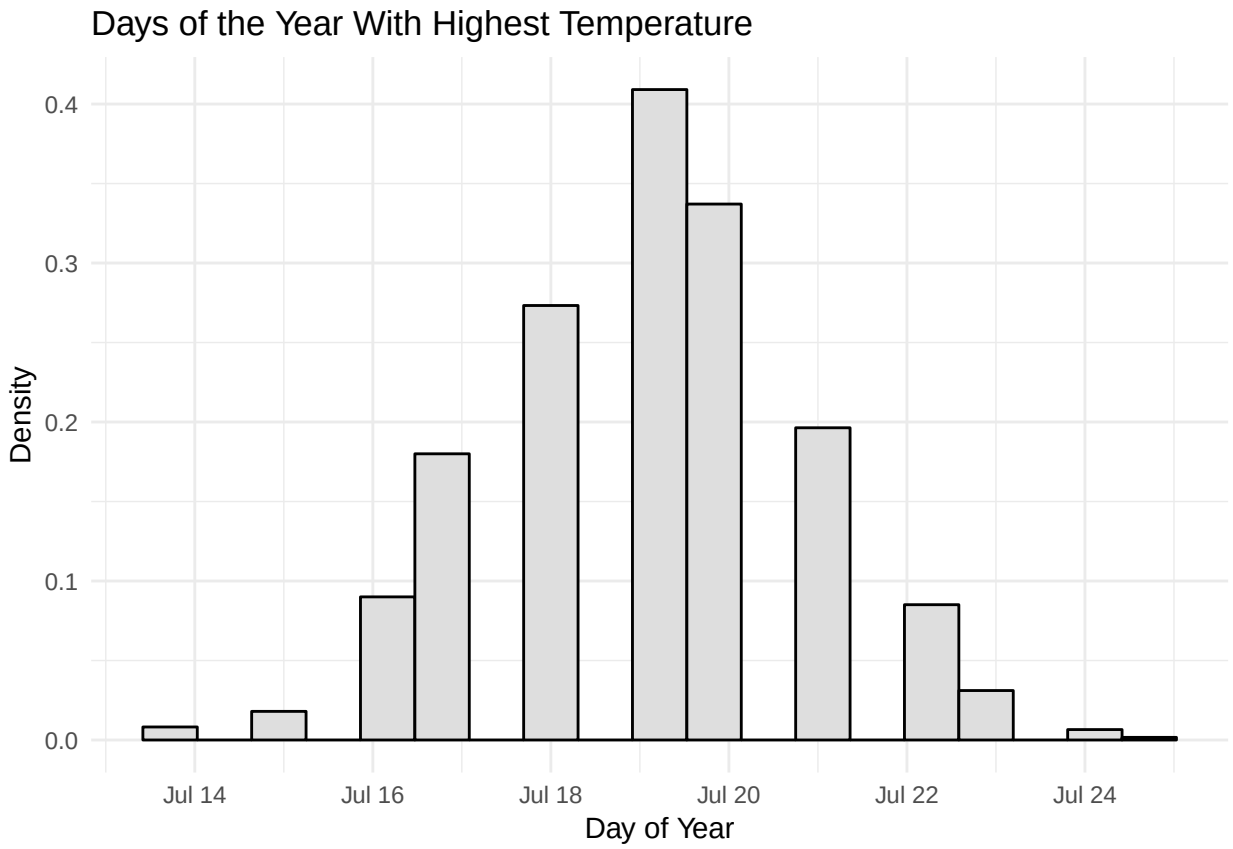
Setting the equation to 0 and solving for time gives us:

$$\tilde{x} = time = \frac{-\beta_1}{2\beta_2}$$

So given a time we can easily calculated $\tilde{x}$. As this is just the fraction of which day of the year it represents, we can easily get back the the day of year.

```
################################################################################
# Exercise 1.c)
################################################################################

x_tilde = - df_posterior_parameters$beta_1 / (2 * df_posterior_parameters$beta_2)

days = as.Date(trunc(366 * x_tilde), origin = "2014-01-01")
```

The following plot shows the histrogram of $\tilde{x}$ converted to date.




## 1.4 Suitable Prior for Preventing Overfitting

To not put to much emphasis on the higher order polynomials we need to specify a way to decrease the variance for the higher polynomials. If we take the previous $\sigma^2 = 16.5847736$ we could introduce a penalty factor depending on the height of the polynomial. We can use a common $\sigma^2$ and multiply it by a constant $\kappa^o$ where $\kappa$ is between 0 and 1 and o is the order of the polynomial minus 1. So if we take the previous $\sigma^2$ we will get the following covariance matrix $\Omega_0$:

```
################################################################################
# Exercise 1.d)
################################################################################

order = 7
kappa = 0.6
omega_0 = rep(0, order)
```

```
for (i in 1:order) {
  omega_0[i] = sigma_sq_n * kappa^(i-1)
}

omega_0 = omega_0 * diag(order)

omega_0
```

```
##           [,1]     [,2]     [,3]     [,4]     [,5]     [,6]      [,7]
## [1,] 16.58477 0.000000 0.000000 0.000000 0.000000 0.000000 0.0000000
## [2,]  0.00000 9.950864 0.000000 0.000000 0.000000 0.000000 0.0000000
## [3,]  0.00000 0.000000 5.970518 0.000000 0.000000 0.000000 0.0000000
## [4,]  0.00000 0.000000 0.000000 3.582311 0.000000 0.000000 0.0000000
## [5,]  0.00000 0.000000 0.000000 0.000000 2.149387 0.000000 0.0000000
## [6,]  0.00000 0.000000 0.000000 0.000000 0.000000 1.289632 0.0000000
## [7,]  0.00000 0.000000 0.000000 0.000000 0.000000 0.000000 0.7737792
```

For $\mu_0$ we'd take the previous posterior values and set all the other values to zero, so we get:

```
mu_0 = c(as.numeric(mu_n), rep(0, order - 3))

mu_0
```

```
## [1] -10.55190  92.92452 -85.18697   0.00000   0.00000   0.00000   0.00000
```

# 2 Posterior Approximation for Classification with Logistic Regression

**Exercise:** 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 |
| NSmallChildren | Counts | Number of child <= 6 years in household | Feature |
| NBigChildren | Counts | Number of child > 6 years in household | Feature |

a) Consider the logistic regression

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

where y is the binary variable with $y = 1$ if the woman works and $y = 0$ if she does not. x is a 8-dimensional vector containing the eight features (including a one for the constant term that models the intercept). Fit the logistic re- gression using maximum likelihood estimation by the command:

```
glmModel = glm(Work ~ 0 + ., data = WomanWork, 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.

b) Now the fun begins. Our goal is to approximate the posterior distribution of the 8-dim parameter vector $\beta$ with a multivariate normal distribution

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

where $\tilde{\beta}$ is the posterior mode and $J(\tilde{\beta}) = -\frac{\partial^2 lnp(\beta|y)}{\partial\beta\partial\beta^T}|_{\beta=\tilde{\beta}}$ is the observed Hessian evaluated at the posterior mode. Note that $\frac{\partial^2 lnp(\beta|y)}{\partial\beta\partial\beta^T}$ is an 8??8 matrix with second derivatives on the diagonal and cross-derivatives $\frac{\partial^2 lnp(\beta|y)}{\partial\beta_i\partial\beta_i^T}$ 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/blob/master/Code/MainOptimizeSpam.zip

where I have coded everything up for the spam prediction example (it also does probit regression, but that is not needed here). I want you to implement you own version of this. You can use my code as a template, but I want you to write your own file so that you understand every line of your code. Don???t just copy my code. Use the prior $\beta \sim \mathcal{N}(0, \tau^2 I)$, with $\tau = 10$. Your report should include your code as well as numerical values for $\tilde{\beta}$ and $J_y^{(-1)}\tilde{\beta}$ for the `WomanWork` 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?

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

## 2.1 Fitting the Logistic Model

Lets first read the data and take a glance at it:

```
##############################################################################
# Exercise 2.a)
##############################################################################

woman_work_data = read_table("data/WomenWork.dat")
kable(head(woman_work_data))
```

| Work | Constant | HusbandInc | EducYears | ExpYears | ExpYears2 | Age | NSmallChild | NBigChild |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 22.39494 | 12 | 7 | 0.49 | 43 | 0 | 3 |
| 0 | 1 | 7.23200 | 8 | 10 | 1.00 | 34 | 0 | 7 |
| 1 | 1 | 18.27199 | 12 | 4 | 0.16 | 41 | 1 | 5 |
| 0 | 1 | 28.06900 | 14 | 2 | 0.04 | 43 | 0 | 2 |
| 1 | 1 | 23.80000 | 12 | 24 | 5.76 | 45 | 0 | 1 |
| 0 | 1 | 96.00000 | 17 | 1 | 0.01 | 34 | 1 | 2 |

Then we can fit the model.

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

```
##
## Call:
## glm(formula = Work ~ 0 + ., family = binomial, data = woman_work_data)
##
## 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
```

## 2.2 Approximating the Posterior Distribution

```
################################################################################
# Exercise 2.b)
################################################################################

# Parameters
Y = as.matrix(woman_work_data[,1])
# We take all covariates
X = as.matrix(woman_work_data[,2:ncol(woman_work_data)])
# Feature names
feature_names =  colnames(woman_work_data[,2:ncol(woman_work_data)])
colnames(X) = feature_names

# Defining the prior parameters
tau_prior = 10
sigma_prior = tau_prior^2 * diag(ncol(woman_work_data) - 1)
mu_prior = rep(0, ncol(woman_work_data) - 1)
# An we need initial beta parameters
beta = rmvnorm(1, mu_prior, sigma_prior)

# Cost-Function
## We need cost function which will be optimized (e.g. the likelihood).
## The first parameters has to be the regression coefficient
```

```
## The function will calculate the log-likelihood and the log-prior for getting
## the posterior-log-likelihood which is to be optimized
## We use two sub-functions for the two logs

log_likelihood = function(beta, X ,Y) {
  llik = sum(-log(1 + exp((X %*% beta))) + Y * X %*% beta)
  if (abs(llik) == Inf) return(-20000)
  return(llik)
}

posterior_log_likelihood = function(beta, X, Y, sigma, mu) {
  return(dmvnorm(beta, mu, sigma, log = TRUE) + log_likelihood(beta, X, Y))
}

res = optim(beta, posterior_log_likelihood, method = "BFGS",
            control = list(fnscale = -1), hessian = TRUE,
            X = X, Y = Y, sigma = sigma_prior, mu = mu_prior)

# Now we use the results to extract the desired values and same them to
# variables with a speaking name
posterior_mode = as.vector(res$par)
names(posterior_mode) = feature_names
posterior_covariance = - solve(res$hessian)
posterior_sd = sqrt(diag(posterior_covariance))
names(posterior_sd) = feature_names
```

Now let us look at these results.

**Posterior Mode ($\tilde{\beta}$) for the optim approach:**

```
##    Constant  HusbandInc   EducYears    ExpYears   ExpYears2         Age
##  0.62669949 -0.01979037  0.18021619  0.16756707 -0.14460182 -0.08206416
## NSmallChild   NBigChild
## -1.35914328 -0.02468042
```

**Posterior Mode ($\tilde{\beta}$) for the glm model:**

```
##    Constant  HusbandInc   EducYears    ExpYears   ExpYears2         Age
##  0.64430363 -0.01977457  0.17988062  0.16751274 -0.14435946 -0.08234033
## NSmallChild   NBigChild
## -1.36250239 -0.02542986
```

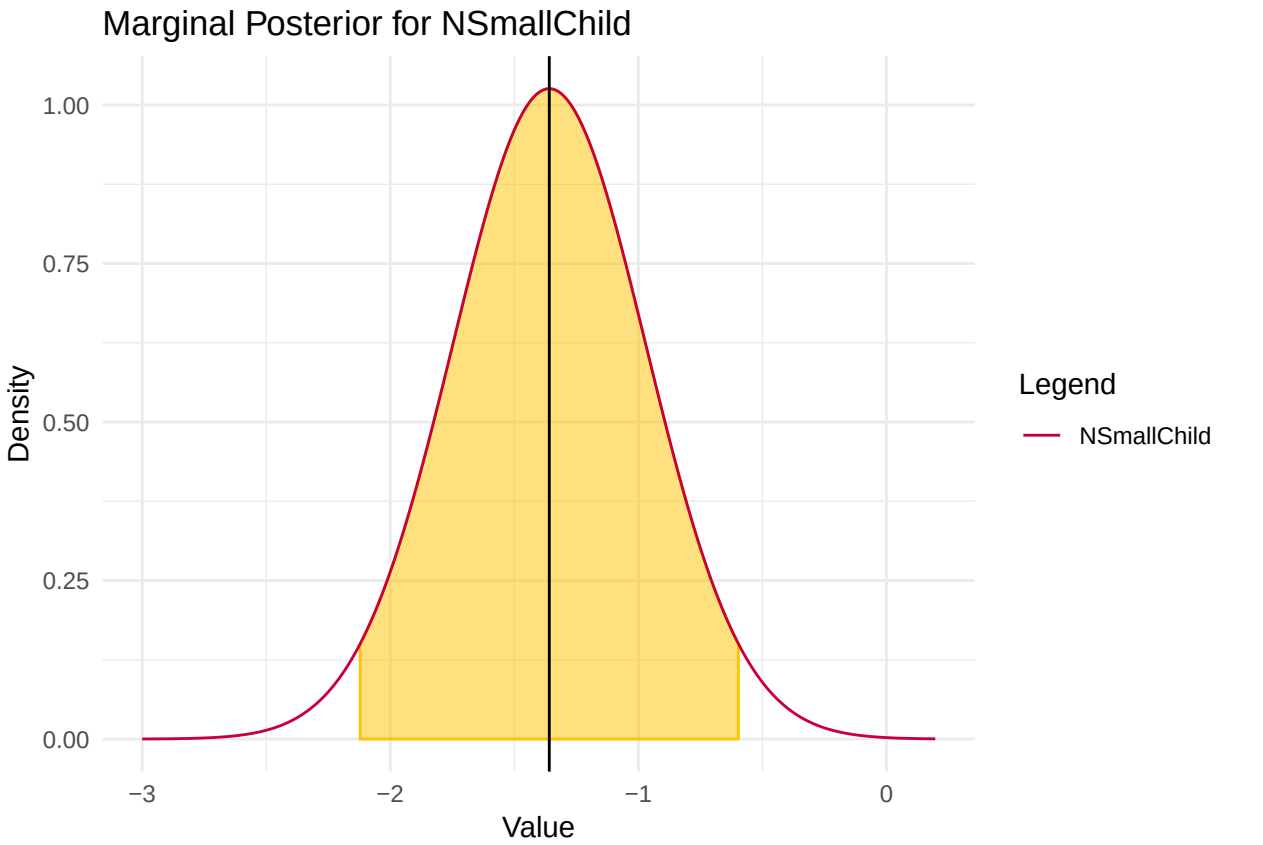
**Approximated Standard Deviation:**

```
##    Constant  HusbandInc   EducYears    ExpYears   ExpYears2         Age
##  1.50533049  0.01589979  0.07885542  0.06596735  0.23575055  0.02680406
## NSmallChild   NBigChild
##  0.38892467  0.14132315
```

The folliwing code procudes the data for the plot. It also includes the 95% credible interval, which is the yellow area. To compute this, we simply take the mean and sigma of our desireed feature and plug it into the `qnorm()` function.

```
# NSmallChild
X_bar = seq(-3, posterior_mode[7] + 4*posterior_sd[7], length = 1000)
Y_bar = dnorm(x = X_bar, mean = posterior_mode[7], sd = posterior_sd[7])

df = data.frame(X_bar, Y_bar)
```

```r
# Credible Interval
ci = qnorm(c(0.025, 0.975), mean = posterior_mode[7], sd = posterior_sd[7])
ci_x = seq(from = ci[1], to = ci[2], length.out = 1000)
ci_y = dnorm(ci_x, mean = posterior_mode[7], sd = posterior_sd[7])

df_ci = data.frame(ci_x, ci_y)
```




We think, as the whole interval is below 0, that this feature has an influence wether a woman is wokring or not. More specifically, the greater the number of small children, the less likely that the woman is working.

## 2.3 Simulating from the Predictive Distribution

We create our X which holds the given values, then we create a function that simulates the response variable for us.

```r
################################################################################
# Exercise 2.c)
################################################################################

# Parameters
X_woman = matrix(c(1, 10, 8, 10, (10/10)^2, 40, 1, 1))

# Function for simulation
simulate_posterior = function(n = 1000, X, mu_post, sigma_post) {

  # First we need to sample our betas
```

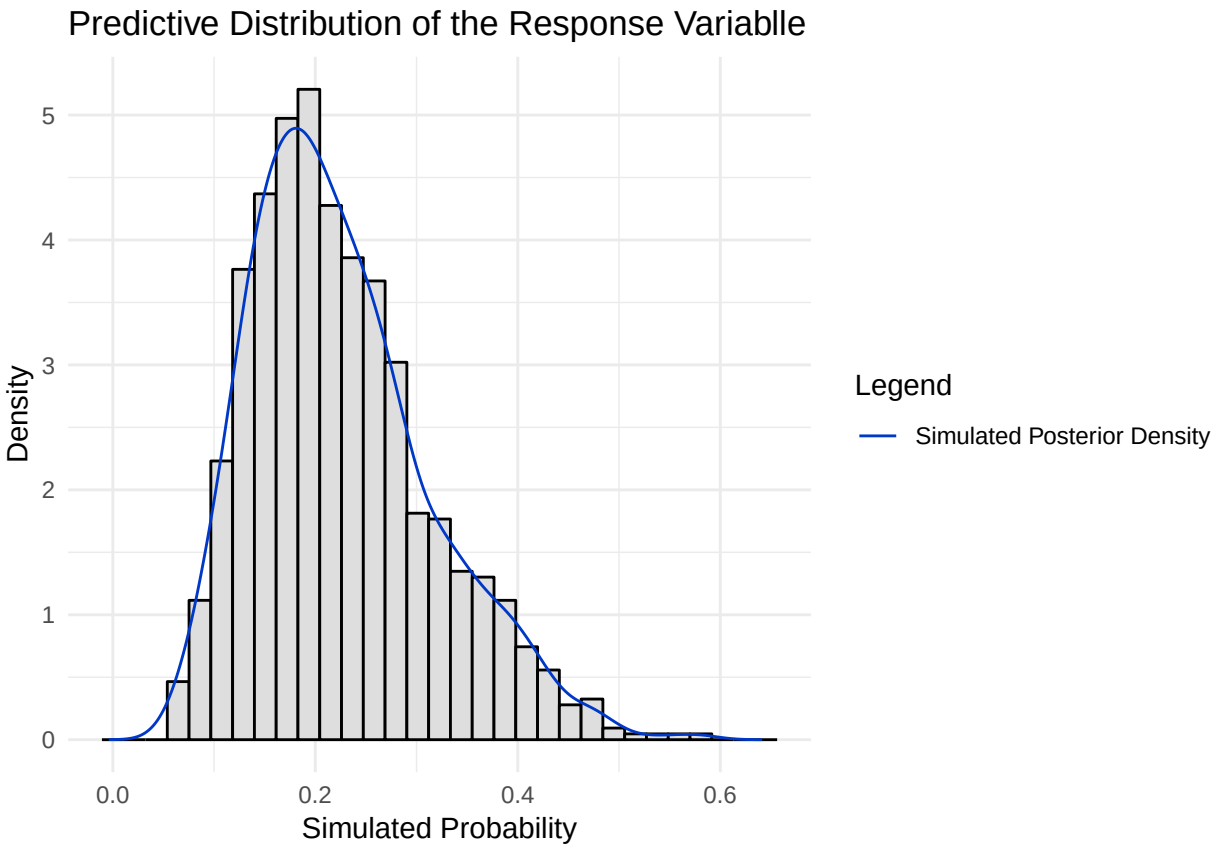
```
  beta = rmvnorm(n = n, mean = mu_post, sigma = sigma_post)

  # We then apply the logistic regression given in the exercise
  # This result will be the probability for y = 1 (woman works)
  return(plogis(beta %*% X))
}

# We use the posterior mean and the previously created covariance matrix
simulated_probabilities = simulate_posterior(n = 1000, X = X_woman,
                                             mu_post = posterior_mode,
                                             sigma_post = posterior_covariance)

# Estimate density
simulated_density = density(simulated_probabilities)
```

The following plot shows the histrogram and the estimated density of the predictive distribution. We can see that the woman is probably **not** working.

**Note:** To be completely precise we actually should not take the probabilities directly, but plug them into a Bernoulli distribution and look at these results.

```
predictive = rep(0, length(simulated_probabilities))

for (i in 1:length(simulated_probabilities)) {
  predictive[i] = rbinom(1, 1, simulated_probabilities[i])
}

probability_working = mean(predictive)
```

Therefore the women is working with a probability of 22.5%.




## 3 Source Code

```
knitr::opts_chunk$set(echo = TRUE)
library(kableExtra)
library(geoR)
library(MASS)
library(ggplot2)
library(gridExtra)
library(readr)
library(mvtnorm)
set.seed(42)

# Disable reard messages
options(readr.num_columns = 0)

################################################################################
# Exercise 1.a)
################################################################################

temperatures_linkoeping = read.table("data/TempLinkoping.txt", header=TRUE)


df = data.frame(head(temperatures_linkoeping))
```

```r
kable(df) %>%
  kable_styling(position = "center") %>%
  row_spec(0, bold=TRUE)

df = data.frame(matrix(c(
  temperatures_linkoeping$time,
  temperatures_linkoeping$temp),
  nrow = length(temperatures_linkoeping$time)))

plot = ggplot(df) +
  geom_point(aes(x = df[, 1], y = df[,2]), color = "black", fill = "#dedede",
             shape = 21)
  plot = plot + labs(title = "Temperature over the Year",
  y = "Temperature", x = "Day of Year in Percent", color = "Legend")
  plot = plot + scale_color_manual("Legend")
  plot = plot + theme_minimal()

plot


# Initial Parameters
mu_0 = c(-10, 100, -100)
omega_0 = 0.01 * diag(3)
nu_0 = 4
sigma_sq_0 = 1

# Helper Variables
time = temperatures_linkoeping$time
temp = temperatures_linkoeping$temp
X = matrix(c(rep(1, length(time)), time, time^2), ncol = 3)
Y = matrix(temperatures_linkoeping$temp)
n = length(time)
m = 5 # Number of regression curves

sample_model_params = function(mu_0, omega_0, nu_0, sigma_sq_0) {

  # Taken from slides (set 5, page 7)
  sigma_sq = rinvchisq(n = 1, df = nu_0, scale = sigma_sq_0)
  beta = mvrnorm(n = 1, mu = mu_0, Sigma = sigma_sq * solve(omega_0))

  return(list(sigma_sq = sigma_sq, beta = beta))
}

predict_using_model_params = function(beta, X) {

  # Predicted temperatures
  return(X %*% matrix(beta))
}

# This matrix holds the true X and Y and then the simulated priors
priors = matrix(0, ncol = m + 2, nrow = n)
priors[,1] = time
priors[,2] = temp
```

```
# Simulate m times for plotting
for (i in 3:(m+2)) {
  model_params = sample_model_params(mu_0, omega_0, nu_0, sigma_sq_0)
  priors[,i] = predict_using_model_params(model_params$beta, X)
}


df = data.frame(priors)

plot = ggplot()
  plot = plot + geom_line(aes(x = df[,1], y = df[,3], color = "Random Prior"))
  plot = plot + geom_line(aes(x = df[,1], y = df[,4], color = "Random Prior"))
  plot = plot + geom_line(aes(x = df[,1], y = df[,5], color = "Random Prior"))
  plot = plot + geom_line(aes(x = df[,1], y = df[,6], color = "Random Prior"))
  plot = plot + geom_line(aes(x = df[,1], y = df[,7], color = "Random Prior"))
  plot = plot + labs(title = "Temperature over the Year",
  y = "Temperature", x = "Day of Year in Percent", color = "Legend")
  plot = plot + scale_color_manual("Legend", values = rep("#C70039", n))
  plot = plot + theme_minimal()

plot

# Adjusted Parameters
mu_0 = c(-8, 90, -82)
omega_0 = 0.2 * diag(3)
nu_0 = 4
sigma_sq_0 = 2


priors = matrix(0, ncol = m + 2, nrow = n)
priors[ ,1] = time
priors[ ,2] = temp

# Simulate m times for plotting
for (i in 3:(m+2)) {
  model_params = sample_model_params(mu_0, omega_0, nu_0, sigma_sq_0)
  priors[,i] = predict_using_model_params(model_params$beta, X)
}


df = data.frame(priors)

plot = ggplot()
  plot = plot + geom_line(aes(x = df[,1], y = df[,3], color = "Random Prior"))
  plot = plot + geom_line(aes(x = df[,1], y = df[,4], color = "Random Prior"))
  plot = plot + geom_line(aes(x = df[,1], y = df[,5], color = "Random Prior"))
  plot = plot + geom_line(aes(x = df[,1], y = df[,6], color = "Random Prior"))
  plot = plot + geom_line(aes(x = df[,1], y = df[,7], color = "Random Prior"))
  plot = plot + labs(title = "Temperature over the Year",
  y = "Temperature", x = "Day of Year in Percent", color = "Legend")
  plot = plot + scale_color_manual("Legend", values = rep("#C70039", n))
  plot = plot + theme_minimal()
```

```r
plot

##############################################################################
# Exercise 1.b)
##############################################################################

# Simulate prior
model_params = sample_model_params(mu_0, omega_0, nu_0, sigma_sq_0)

# Beta Hat
#beta_hat = model_params$beta
beta_hat = as.vector(lm(temp ~ time + I(time^2),
                        data = temperatures_linkoeping)$coefficients)

# Posterior parameters
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_and_sigma_sq_n = nu_0 * sigma_sq_0 +
  (t(Y) %*% Y + (t(mu_0) %*% omega_0 %*% mu_0) - (t(mu_n) %*% omega_n %*% mu_n))
sigma_sq_n = as.numeric(nu_n_and_sigma_sq_n / nu_n)


df = data.frame()

simulations = 1000

for (i in 1:simulations) {
  simulated_params = sample_model_params(mu_n, omega_n, nu_n, sigma_sq_n)
  sample = list(sigma_sq = simulated_params$sigma_sq,
                beta_0 = simulated_params$beta[1],
                beta_1 = simulated_params$beta[2],
                beta_2 = simulated_params$beta[3])

  df = rbind(df, sample)
}

df_posterior_parameters = df # Saved for 1c)

p1 = ggplot(df) +
  geom_histogram(aes(x = sigma_sq, y=..density..),
                 bins = sqrt(nrow(df)), color = "black",
                 fill = "#DEDEDE") +
  labs(title = "Drawn sigma_sq",
       y = "Density", x = "sigma_sq") +
  scale_color_manual("Legend", values = c("#0039C7", "#000000")) +
  theme_minimal()

p2 = ggplot(df) +
  geom_histogram(aes(x = beta_0, y=..density..),
                 bins = sqrt(nrow(df)), color = "black",
                 fill = "#DEDEDE") +
```

```
  labs(title = "Drawn beta_0",
       y = "Density", x = "beta_0") +
  scale_color_manual("Legend", values = c("#0039C7", "#000000")) +
  theme_minimal()

p3 = ggplot(df) +
  geom_histogram(aes(x = beta_1, y=..density..),
                 bins = sqrt(nrow(df)), color = "black",
                 fill = "#DEDEDE") +
  labs(title = "Drawn beta_1",
       y = "Density", x = "beta_1") +
  scale_color_manual("Legend", values = c("#0039C7", "#000000")) +
  theme_minimal()

p4 = ggplot(df) +
  geom_histogram(aes(x = beta_2, y=..density..),
                 bins = sqrt(nrow(df)), color = "black",
                 fill = "#DEDEDE") +
  labs(title = "Drawn beta_2",
       y = "Density", x = "beta_2") +
  scale_color_manual("Legend", values = c("#0039C7", "#000000")) +
  theme_minimal()

grid.arrange(p1, p2, p3, p4, nrow = 2)


posterior_predictions = matrix(0, ncol = n, nrow = simulations)

for (i in 1:simulations) {
  beta = unlist(df[i,2:4])
  posterior_predictions[i,] = predict_using_model_params(beta, X)
}

posterior_prediction_median = apply(posterior_predictions, 2, median)
posterior_prediction_mean = apply(posterior_predictions, 2, mean)


ci = apply(posterior_predictions, 2, quantile, probs = c(0.025, 0.975))
lower = ci[1,]
upper = ci[2,]

upper_two_sigma = posterior_prediction_mean + 2 * sqrt(sigma_sq_n)
lower_two_sigma = posterior_prediction_mean - 2 * sqrt(sigma_sq_n)



df = data.frame(time, temp, posterior_prediction_median, lower, upper,
                upper_two_sigma, lower_two_sigma)

ggplot(df) +
  geom_ribbon(aes(x = time, ymin = lower_two_sigma, ymax = upper_two_sigma),
              fill = "orange", alpha = 0.25) +
  geom_ribbon(aes(x = time, ymin = lower, ymax = upper), fill = "black",
```

```
              alpha = 0.3) +
  geom_point(aes(x = time, y = temp), color = "black", fill = "#dedede",
             shape = 21) +
  geom_line(aes(x = time, y = posterior_prediction_median,
                color = "Posterior Median")) +
  labs(title = "Temperature over the Year",
  y = "Temperature", x = "Day of Year in Percent", color = "Legend") +
  scale_color_manual("Legend", values = c("#C70039", "blue", "blue")) +
  theme_minimal()

################################################################################
# Exercise 1.c)
################################################################################

x_tilde = - df_posterior_parameters$beta_1 / (2 * df_posterior_parameters$beta_2)

days = as.Date(trunc(366 * x_tilde), origin = "2014-01-01")


ggplot(as.data.frame(days)) +
  geom_histogram(aes(x = days, y=..density..),
                 bins = sqrt(nrow(df)), color = "black",
                 fill = "#DEDEDE") +
  labs(title = "Days of the Year With Highest Temperature",
       y = "Density", x = "Day of Year") +
  scale_color_manual("Legend", values = c("#0039C7", "#000000")) +
  theme_minimal()


################################################################################
# Exercise 1.d)
################################################################################

order = 7
kappa = 0.6
omega_0 = rep(0, order)

for (i in 1:order) {
  omega_0[i] = sigma_sq_n * kappa^(i-1)
}

omega_0 = omega_0 * diag(order)

omega_0


mu_0 = c(as.numeric(mu_n), rep(0, order - 3))

mu_0


variable = c("Work", "Constant", "HusbandInc", "EducYears", "ExpYears",
             "ExpYears2", "Age", "NSmallChildren", "NBigChildren")
```

```r
dataType = c("Binary", "1", "Numeric", "Counts", "Counts", "Numeric",
             "Counts", "Counts", "Counts")
meaning = c("Whether or not the woman works", "Constant to the intercept",
            "Husband???s income", "Years of education", "Years of experience",
            "(Years of experience/10)^2", "Age",
            "Number of child <= 6 years in household",
            "Number of child > 6 years in household")
feature = c("Response", "Feature", "Feature", "Feature", "Feature", "Feature",
            "Feature", "Feature", "Feature")

df = data.frame(variable, dataType, meaning, feature)
colnames(df) = c("Variable", "Data Type", "Meaning", "Role")

kable(df) %>%
  kable_styling(position = "center") %>%
  row_spec(0, bold=TRUE)


glmModel = glm(Work ~ 0 + ., data = WomanWork, family = binomial)


################################################################################
# Exercise 2.a)
################################################################################

woman_work_data = read_table("data/WomenWork.dat")
kable(head(woman_work_data))


glmModel = glm(Work ~ 0 + ., data = woman_work_data, family = binomial)
summary(glmModel)


################################################################################
# Exercise 2.b)
################################################################################

# Parameters
Y = as.matrix(woman_work_data[,1])
# We take all covariates
X = as.matrix(woman_work_data[,2:ncol(woman_work_data)])
# Feature names
feature_names =  colnames(woman_work_data[,2:ncol(woman_work_data)])
colnames(X) = feature_names

# Defining the prior parameters
tau_prior = 10
sigma_prior = tau_prior^2 * diag(ncol(woman_work_data) - 1)
mu_prior = rep(0, ncol(woman_work_data) - 1)
# An we need initial beta parameters
beta = rmvnorm(1, mu_prior, sigma_prior)

# Cost-Function
```

```
## We need cost function which will be optimized (e.g. the likelihood).
## The first parameters has to be the regression coefficient
## The function will calculate the log-likelihood and the log-prior for getting
## the posterior-log-likelihood which is to be optimized
## We use two sub-functions for the two logs

log_likelihood = function(beta, X ,Y) {
  llik = sum(-log(1 + exp((X %*% beta))) + Y * X %*% beta)
  if (abs(llik) == Inf) return(-20000)
  return(llik)
}

posterior_log_likelihood = function(beta, X, Y, sigma, mu) {
  return(dmvnorm(beta, mu, sigma, log = TRUE) + log_likelihood(beta, X, Y))
}

res = optim(beta, posterior_log_likelihood, method = "BFGS",
            control = list(fnscale = -1), hessian = TRUE,
            X = X, Y = Y, sigma = sigma_prior, mu = mu_prior)

# Now we use the results to extract the desired values and same them to
# variables with a speaking name
posterior_mode = as.vector(res$par)
names(posterior_mode) = feature_names
posterior_covariance = - solve(res$hessian)
posterior_sd = sqrt(diag(posterior_covariance))
names(posterior_sd) = feature_names

posterior_mode
glmModel$coefficients
posterior_sd

# NSmallChild
X_bar = seq(-3, posterior_mode[7] + 4*posterior_sd[7], length = 1000)
Y_bar = dnorm(x = X_bar, mean = posterior_mode[7], sd = posterior_sd[7])

df = data.frame(X_bar, Y_bar)

# Credible Interval
ci = qnorm(c(0.025, 0.975), mean = posterior_mode[7], sd = posterior_sd[7])
ci_x = seq(from = ci[1], to = ci[2], length.out = 1000)
ci_y = dnorm(ci_x, mean = posterior_mode[7], sd = posterior_sd[7])

df_ci = data.frame(ci_x, ci_y)


ggplot(df) +
  geom_ribbon(data = df_ci, aes(x = ci_x, ymin = 0, ymax = ci_y),
              alpha = 0.5, fill = "#FFC300", color = "#FFC300") +
  geom_line(aes(x = X_bar, y = Y_bar,
                color = "NSmallChild")) +
  geom_vline(xintercept = posterior_mode[7]) +
  labs(title = "Marginal Posterior for NSmallChild",
```

```
  y = "Density", x = "Value", color = "Legend") +
  scale_color_manual("Legend", values = c("#C70039")) +
  theme_minimal()


################################################################################
# Exercise 2.c)
################################################################################

# Parameters
X_woman = matrix(c(1, 10, 8, 10, (10/10)^2, 40, 1, 1))

# Function for simulation
simulate_posterior = function(n = 1000, X, mu_post, sigma_post) {

  # First we need to sample our betas
  beta = rmvnorm(n = n, mean = mu_post, sigma = sigma_post)

  # We then apply the logistic regression given in the exercise
  # This result will be the probability for y = 1 (woman works)
  return(plogis(beta %*% X))
}

# We use the posterior mean and the previously created covariance matrix
simulated_probabilities = simulate_posterior(n = 1000, X = X_woman,
                                             mu_post = posterior_mode,
                                             sigma_post = posterior_covariance)

# Estimate density
simulated_density = density(simulated_probabilities)


predictive = rep(0, length(simulated_probabilities))

for (i in 1:length(simulated_probabilities)) {
  predictive[i] = rbinom(1, 1, simulated_probabilities[i])
}

probability_working = mean(predictive)


df = data.frame(simulated_probabilities)
df_density = data.frame(simulated_density$x, simulated_density$y)

ggplot(df) +
  geom_histogram(aes(x = simulated_probabilities, y=..density..),
                 bins = sqrt(nrow(df)), color = "black",
                 fill = "#DEDEDE") +
  geom_line(data = df_density, aes(x = simulated_density.x,
                                   y = simulated_density.y,
                                   color = "Simulated Posterior Density")) +
  labs(title = "Predictive Distribution of the Response Variablle",
       y = "Density", x = "Simulated Probability") +
```

```
  scale_color_manual("Legend", values = c("#0039C7", "#000000")) +
  theme_minimal()

##############################################################################
# ALTERNATIVE METHOD
##############################################################################

# Task 1a

temp_data <- read.table("data/TempLinkoping.txt", header=TRUE, sep="\t")

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

model <- lm(temp ~ time + I(time^2))

mu_0 <- c(-10, 100, -100)
om_0 <- diag(c(0.01, 0.01, 0.01))
om_inv_0 <- solve(om_0)
v_0 <- 4
sigmasq_0 <- 1

nDraws <- 150

s2 <- (v_0 * sigmasq_0)/rchisq(nDraws, v_0)

plot(time, temp)
for(k in 1:nDraws){
  beta <- rmvnorm(1, mu_0, om_inv_0*s2[k])
  lines(time, beta[1] + beta[2] * time + beta[3] * time^2, col="blue")
}
points(time, temp)

mu0 <- model$coefficients
om0 <- diag(c(5, 5, 5))
ominv0 <- solve(om0)
v0 <- 10
sigmasq0 <- 16

s2 <- (v0 * sigmasq0)/rchisq(nDraws, v0)

plot(time, temp)
for(k in 1:nDraws){
  beta <- rmvnorm(1, mu0, ominv0*s2[k])
  lines(time, beta[1] + beta[2] * time + beta[3] * time^2, col="blue")
}
points(time, temp)

# Task 1b

X <- cbind(1, time, time^2)
y <- temp
XX <- t(X) %*% X
```

```
yy <- t(y) %*% y
n <- length(y)

beta_hat <- solve(XX) %*% t(X) %*% y
mu_n <- solve(XX + om0) %*% (XX %*% beta_hat + om0 %*% mu0)
om_n <- XX + om0
ominv_n <- solve(om_n)
v_n <- v0 + n
vn_sigmasq_n <- (v0 * sigmasq0) + (yy + (t(mu0) %*% om0 %*% mu0) -
                                     (t(mu_n) %*% om_n %*% mu_n)) /v_n
vn_sigmasq_n <- vn_sigmasq_n[1]

sigmasq <- (v_n * vn_sigmasq_n)/rchisq(nDraws, v_n)

b <- data.frame()
pDraws <- matrix(rep(0, n*nDraws), nDraws)
for(k in 1:nDraws){
  beta <- rmvnorm(1, mu_n, ominv_n*sigmasq[k])
  b <- rbind(b, data.frame(beta0=beta[1], beta1=beta[2], beta2=beta[3]))
  pDraws[k, 1:n] <- beta[1] + beta[2] * time + beta[3] * time^2
}

p1 <- ggplot()+
  geom_histogram(aes(x=sigmasq, color="red"), bins=30) +
  theme_bw()

p2 <- ggplot(b)+
  geom_histogram(aes(x=beta0, color="red"), bins=30) +
  theme_bw()

p3 <- ggplot(b)+
  geom_histogram(aes(x=beta1, color="red"), bins=30) +
  theme_bw()

p4 <- ggplot(b)+
  geom_histogram(aes(x=beta2, color="red"), bins=30) +
  theme_bw()

grid.arrange(p1, p2, p3, p4, nrow=2)

pMedian <- apply(pDraws, 2, median)
pDraws <- apply(pDraws, 2, sort)

# lower <- pDraws[floor(nDraws * 0.025) + 1, 1:n]
# upper <- pDraws[floor(nDraws * 0.975), 1:n]

ci = apply(pDraws, 2, quantile, probs = c(0.025, 0.975))
lower = ci[1,]
upper = ci[2,]

ggplot() +
  geom_point(aes(x=time, y=temp)) +
  geom_line(aes(x=time, y=pMedian, color="Median")) +
```

```
  geom_line(aes(x=time, y=lower, color="Lower interval")) +
  geom_line(aes(x=time, y=upper, color="Upper interval")) +
  theme_bw()

# Task 1c

set.seed(12345)
x_tilde <- c()
for(k in 1:150){
  beta <- rmvnorm(1, mu_n, ominv_n*sigmasq[k])
  pDraws[k, 1:n] <- beta[1] + beta[2] * time + beta[3] * time^2
  x_tilde <- c(x_tilde, -beta[2]/(2*beta[3]))
}

ggplot() +
  geom_histogram(aes(x=x_tilde), bins=30, color="white")

#------------------------------------------------------------------------------

# Task 2a

WomanWork <- read.table("data/WomenWork.dat", header = TRUE)
gmlModel = glm(Work ~ 0 + ., data = WomanWork, family = binomial)

# Task 2b

n = dim(WomanWork)
num = n[1]
n = n[2]
y = as.vector(WomanWork[, 1])
X = as.matrix(WomanWork[, 2:n])
cNames <- names(WomanWork)[2:n]
n = n - 1

tau = 10
pmB = as.vector(rep(0, n))
pJInv = diag(n) * tau * tau
init = as.vector(rnorm(dim(X)[2]))

logisticPostLoglikelihood = function(betas, y, X, pmB, pJInv){
    inp = X %*% betas

    ll = sum(y*inp - log(1 + exp(inp)))
    if(abs(ll) == Inf){
        ll = -99999
    }

    lp = dmvnorm(betas, pmB, pJInv, log=TRUE)

    return(ll + lp)
}

opt = optim(init, logisticPostLoglikelihood, gr=NULL, y, X,
```

```
            pmB, pJInv, method=c("BFGS"), control=list(fnscale=-1), hessian=TRUE)

mB = opt$par
JInv = -solve(opt$hessian)
names(mB) = cNames

# Posterior Mode:
print(mB)
# Inverse Negative Hessian:
print(JInv)

smallChildData = sort(WomanWork$NSmallChild)
cred = c(smallChildData[floor(num * 0.05)], smallChildData[floor(num * 0.95)+1])
cat("Credible Interval for number of small children: [", cred[1], ", ",
    cred[2], "]", sep="")

# Task 2c

nDraws = 100000
x = c(1, 10, 8, 10, (10/10)^2, 40, 1, 1)
betas = rmvnorm(nDraws, mB, JInv)
probWork = 1/(1+exp(-betas %*% x))
h=hist(probWork, breaks = 100)

##############################################################################
# ALTERNATIVE METHOD END
##############################################################################
```