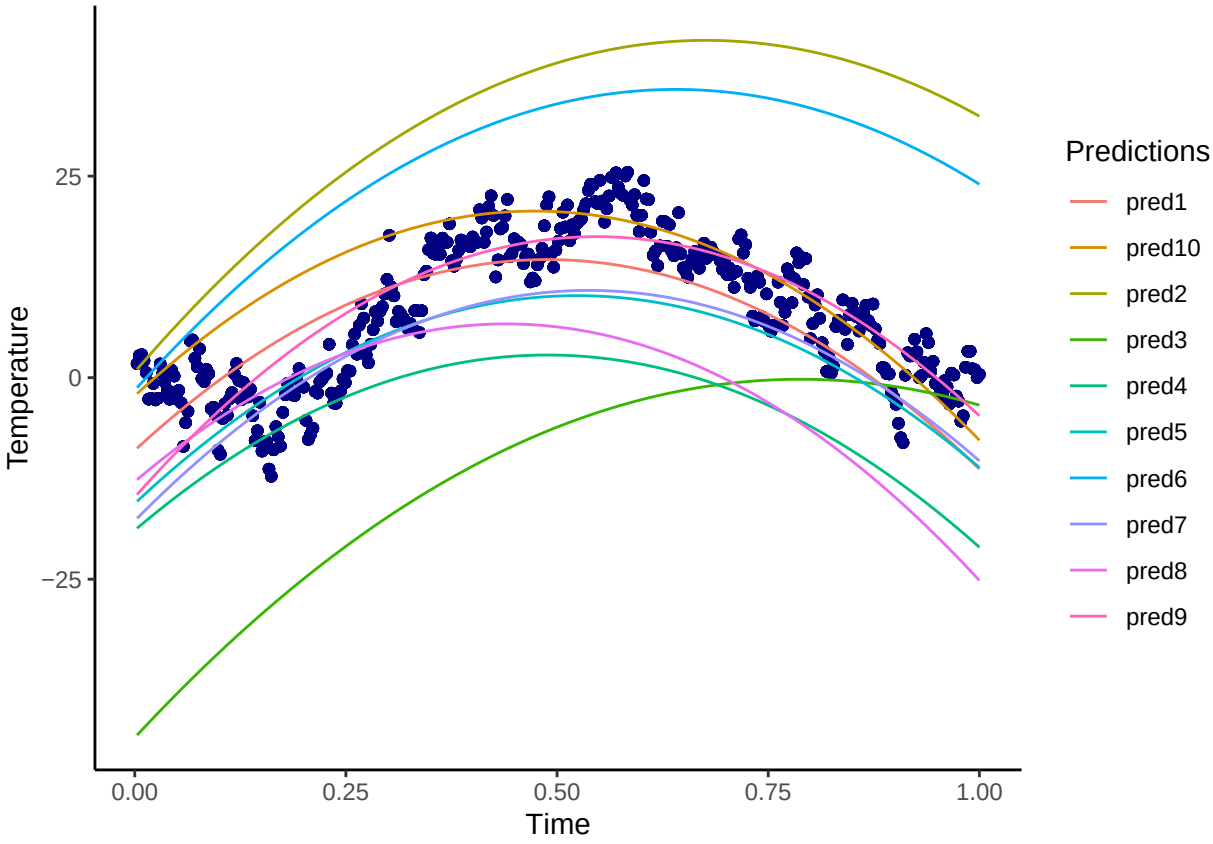
Bayesian Learning (732A91) Lab2 Report

Christoforos Spyretos (chrsp415) & Marketos Damigos (marda352)

2022-04-19

Assignment 1 *Linear and polynomial regression*

Task 1a



The above graph illustrates the actual temperatures (blue points) and a collection of regression curves; one for each draw from the prior. The graph provides mixed results as half of the curves are not in the region of the actual temperatures. Thus, the the prior hyperparameters must be changed.

Task 1b

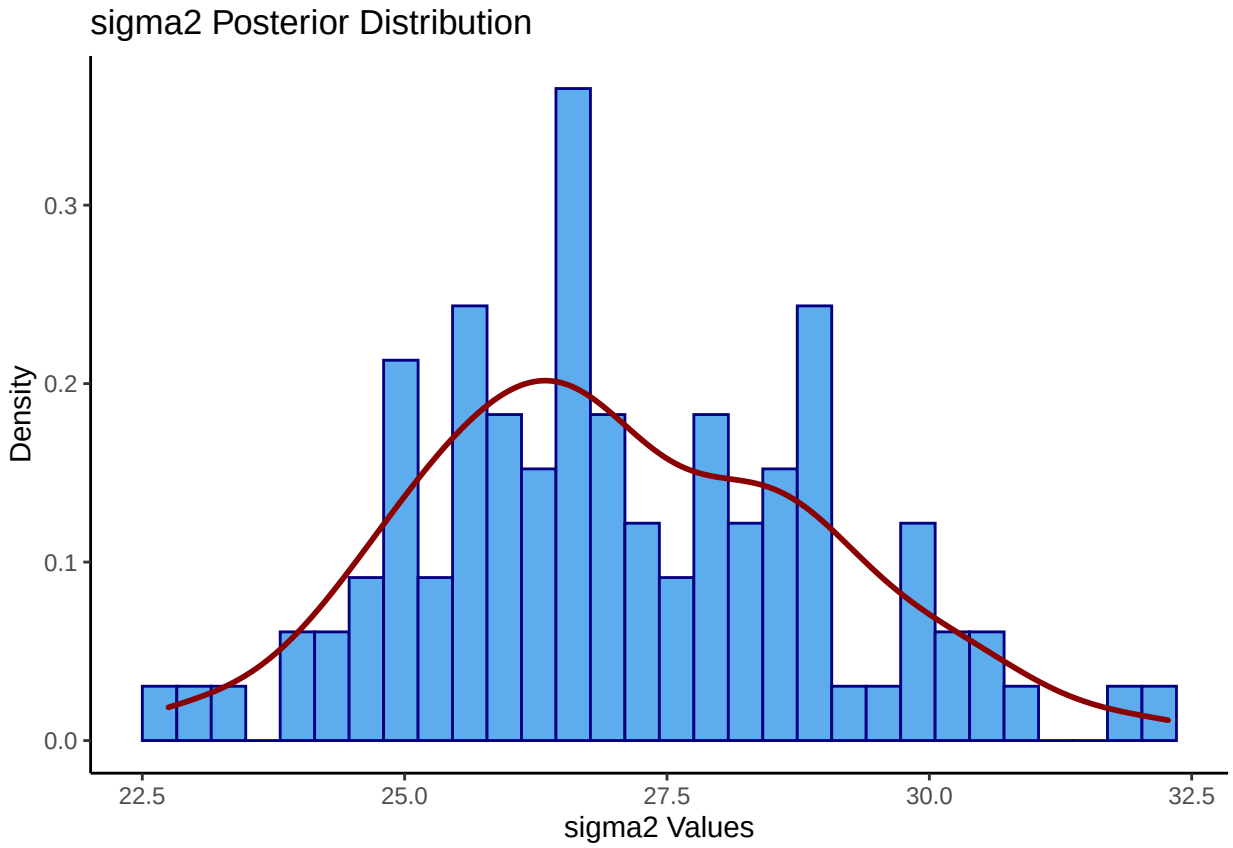
The joint posterior for β and σ^2 is given by:

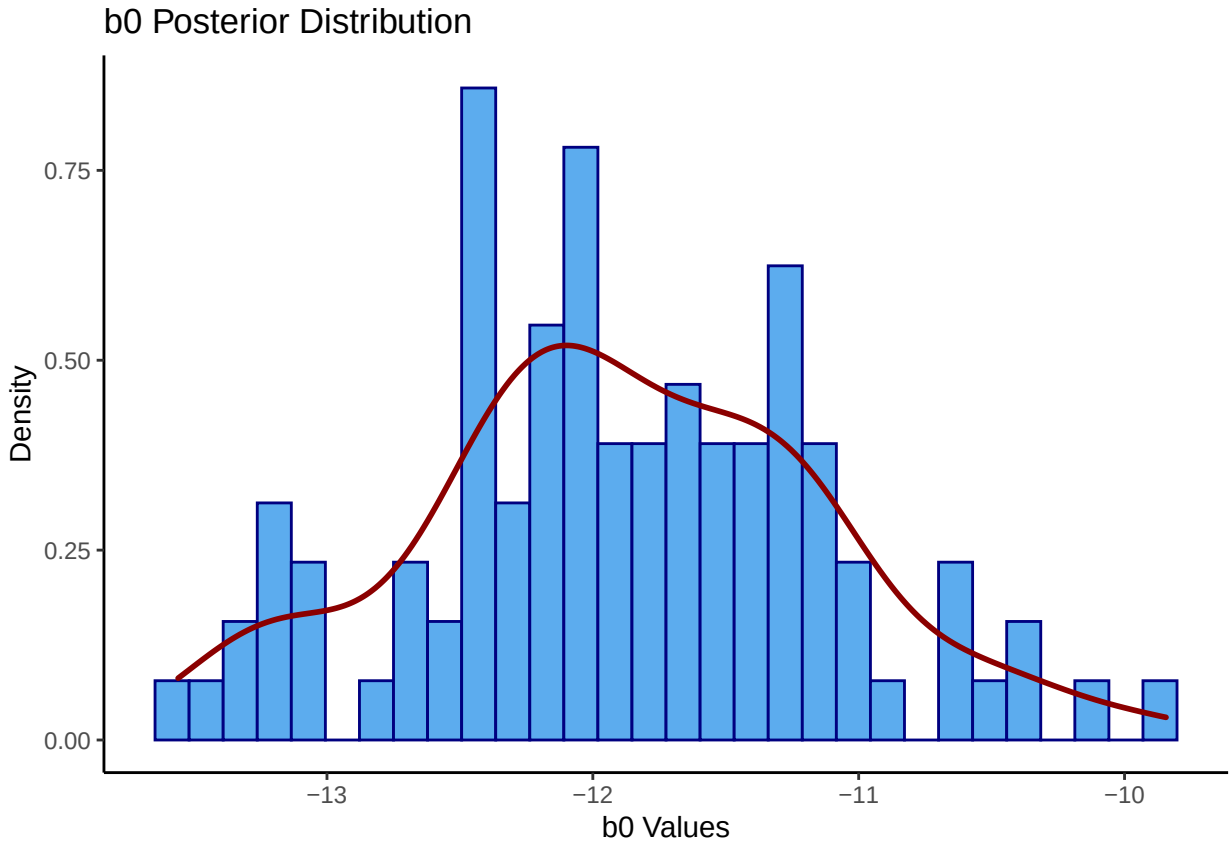
$\beta|\sigma^2, y \sim N(\mu_n, \sigma^2\Omega_n^{-1})$
$\sigma^2|y \sim Inv - \chi^2(\nu_n, \sigma_n^2)$

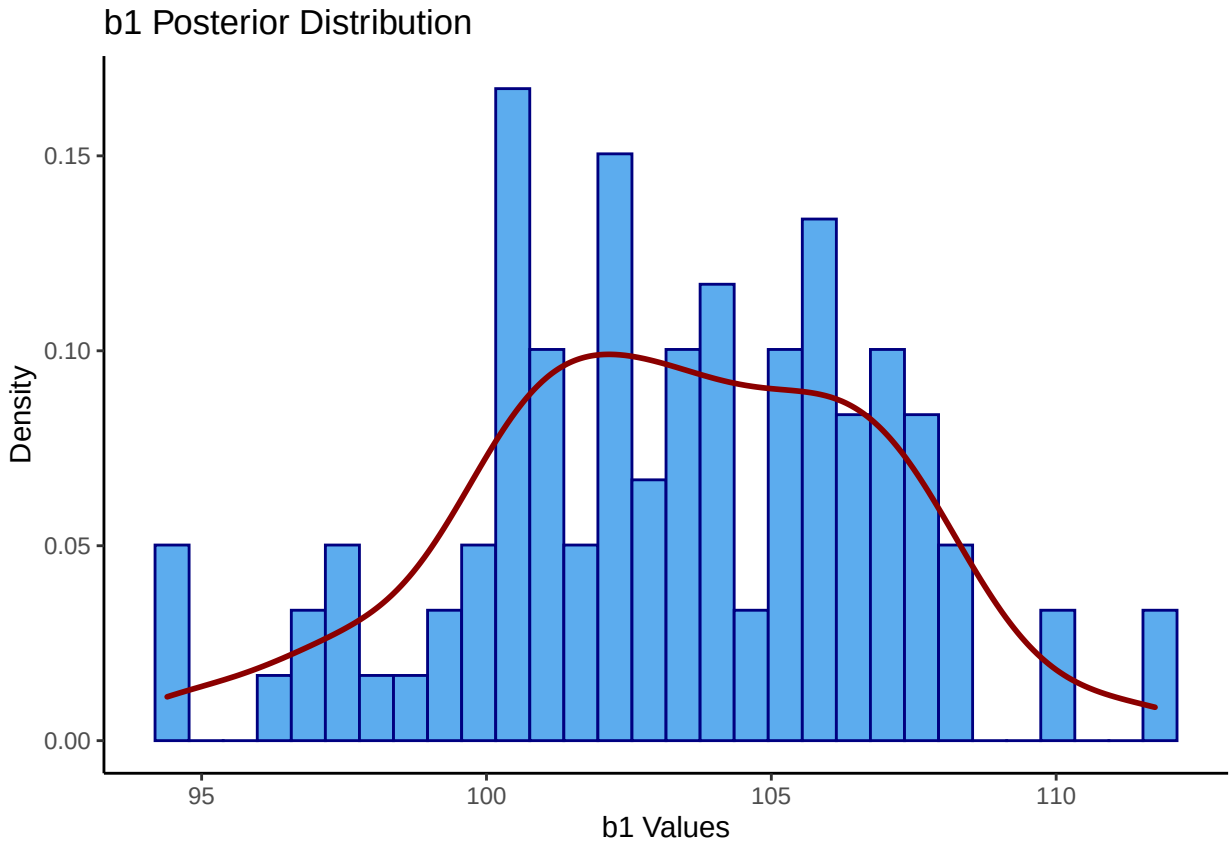
Where:

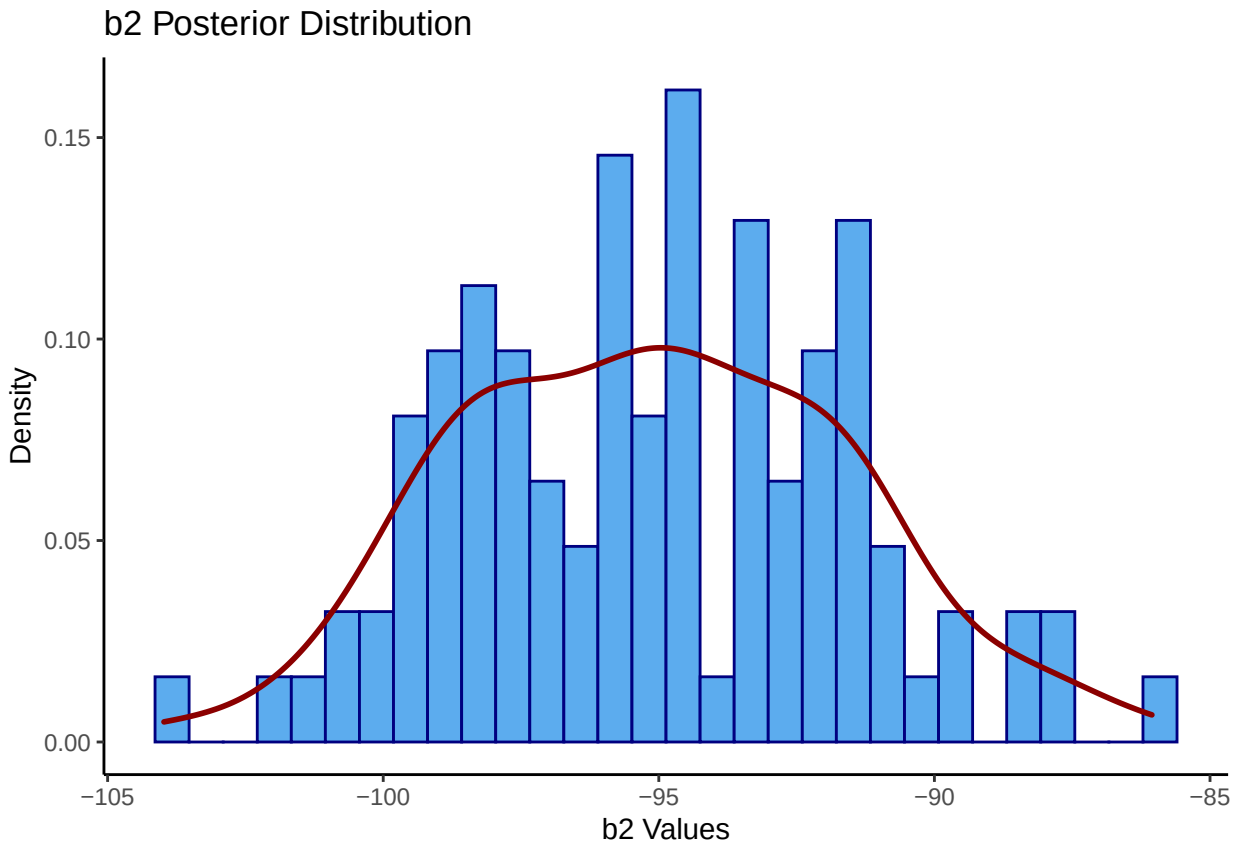
$\widehat{\beta} = (X'X)^{-1}X'y$
$\mu_n = (X'X + \Omega_0)^{-1}(X'X\widehat{\beta} + \Omega_0\mu_0)$
$\Omega_n = X'X + \Omega_0$
$\nu_n = \nu_0 + n$
$\sigma_n^2 = \frac{\nu_0\sigma_0^2 + (y'y + \mu_0'\Omega_0\mu_0 - \mu_n'\Omega_n\mu_n)}{\nu_n}$

i)

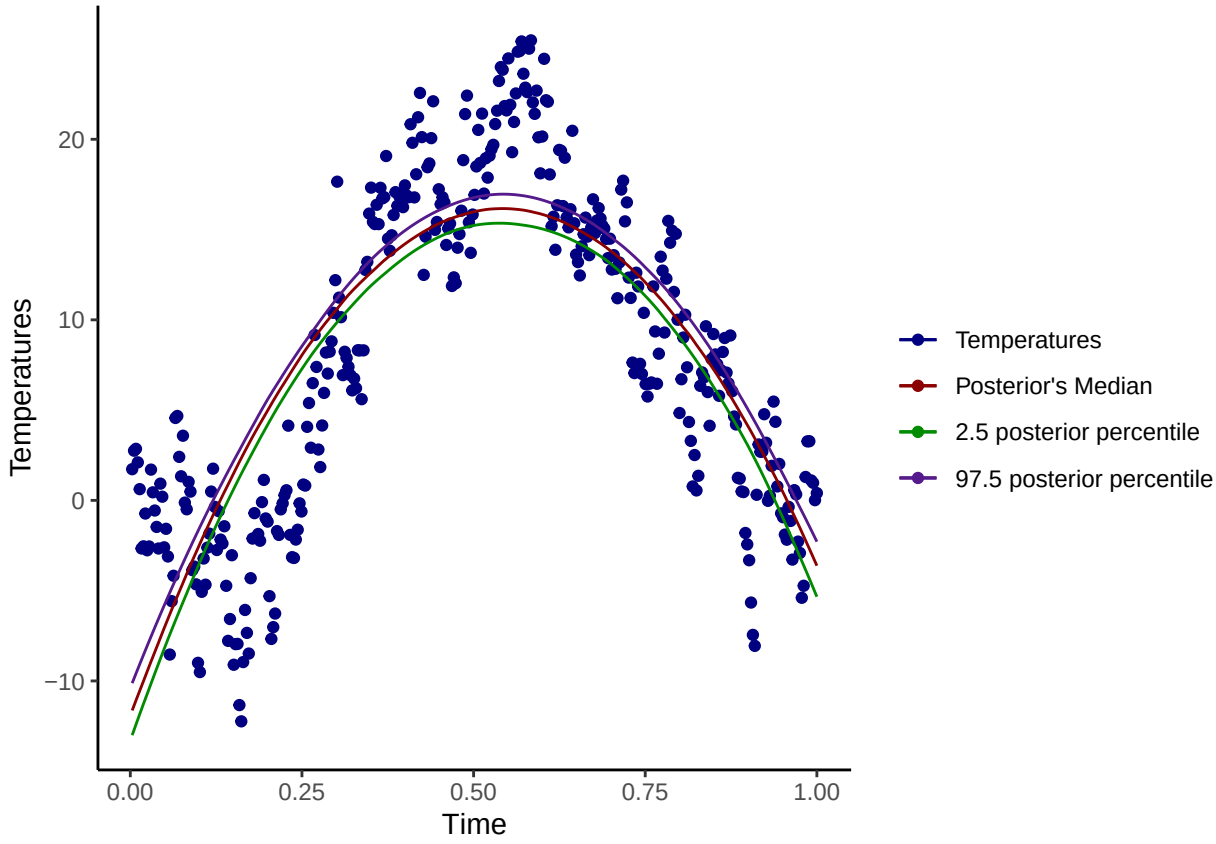



b0 Posterior Distribution
Density
0.75
0.50
0.25
0.00
−13
−12
−11
−10
b0 Values


b1 Posterior Distribution
Density
0.15
0.10
0.05
0.00
95
100
105
110
b1 Values


b2 Posterior Distribution
Density
0.15
0.10
0.05
0.00
−105
−100
−95
−90
−85
b2 Values

ii)



It is evident that 95% equal tail posterior probability intervals do not contain most of the data points. It should not contain most of the data points, because the interval illustrates the region of the beta parameter where the true parameter value lies with 95% certainty.

Task 1c

It is given that the regression function is $f(time) = \beta_0 + \beta_1 \cdot time + \beta_2 \cdot time^2$. In order to locate the time with the highest expected temperature; the time, where f(time) is maximal, is needed to be found. Thus, the derivative of $f(time)$ is needed to be found and set it equal to zero to find the maximal.

Set $time = x$; thus, $f(x) = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot x^2$

Calculate the derivative, $f'(x) = \beta_1 + 2 \cdot \beta_2 \cdot x$

Find the maximal: $f'(x) = 0 \Leftrightarrow \beta_1 + 2 \cdot \beta_2 \cdot x = 0 \Leftrightarrow x = \frac{-\beta_1}{2 \cdot \beta_2}$

```
# Task 1c

# getting the time with the highest expected temperature by
# using the above expression
max_temp <- max(-b[, 2]/(2 * b[, 3]))
max_temp
```

```
## [1] 0.5601627
```

The time with the highest expected temperature is approximately 0.56.

Task 1d

Estimating a polynomial regression of order 8 with the suspicion that higher order terms may not be needed, might lead to overfitting the data. The main problem that could lead to overfitting is the number of knots. A solution to this problem is to introduce a regularised prior, $\beta_j|\sigma^2 \sim N(\mu_0, \frac{\sigma^2}{\lambda})$, where $Omega_0 = \lambda \cdot I$.

Assignment 2 *Posterior approximation for classification with logistic regression*

```
# ___________________Assignment 2_____________________#

# reading data
women <- read.table("WomenAtWork.dat", header = T)
```

```
# Task 2a

# given parameteres
tau <- 5
```

Task 2a

```
# target value
y = women$Work

# prior inputs Select which covariates/features to include
X <- as.matrix(women[2:8])
Xnames <- colnames(X)

Npar <- dim(X)[2]

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

LogPostLogistic <- function(betas, y, X, mu, Sigma) {
    linPred <- X %*% betas
    logLik <- sum(linPred * y - log(1 + exp(linPred)))
    if (abs(logLik) == Inf)
        logLik = -20000  # Likelihood is not finite, stear the optimizer away from here!
    logPrior <- dmvnorm(betas, mu, Sigma, log = TRUE)

    return(logLik + logPrior)
}

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

# The argument control is a list of options to the
```

```r
# 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)
```

```r
names(OptimRes$par) <- Xnames  # Naming the coefficient by covariates
approxPostStd <- sqrt(diag(-solve(OptimRes$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,]  0.99295290
## [2,] -0.03435024
## [3,]  0.17941763
## [4,]  0.12306284
## [5,] -0.07279229
## [6,] -1.62277354
## [7,] -0.08388832
## 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
approxPostStd <- sqrt(diag(-solve(OptimRes$hessian)))
print(approxPostStd)
```

```
## [1] 1.59036578 0.01978569 0.08568259 0.03000973 0.02841075 0.43799331 0.14905137
```

```r
interval <- c(OptimRes$par[6] - 1.96 * approxPostStd[6], OptimRes$par[6] +
    1.96 * approxPostStd[6])
interval
```

```
## NSmallChild NSmallChild
##  -2.4812404  -0.7643067
```

Appendix

```r
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_chunk$set(tidy.opts = list(width.cutoff = 60), tidy = TRUE)
library(mvtnorm)
library(ggplot2)
library(tidyr)

# _____________________Assignment 1_____________________#

# reading data
temperatures <- read.table("TempLambohov.txt", header = TRUE)
```

```r
# given parameters
m_0 <- c(-10, 100, -100)
omega_0 <- 0.02 * diag(3)
n_0 <- 3
sigma2_0 <- 2

# Task 1a

set.seed(123456)

# number of draws
N <- 10

# matrix to save b0, b1, b2
b <- matrix(NA, nrow = N, ncol = length(m_0))

# matrix to save the predicted temperatures
pred_temp <- matrix(NA, nrow = N, ncol = nrow(temperatures))

for (i in 1:N) {

    # Inv-x simulator from lab 1
    sigma2 <- n_0 * sigma2_0/rchisq(1, n_0)

    # generate b0,b1,b2
    b[i, ] <- rmvnorm(1, mean = m_0, sigma = sigma2 * solve(omega_0))

    # quadratic regression model
    pred_temp[i, ] <- b[i, 1] + b[i, 2] * temperatures$time +
        b[i, 3] * (temperatures$time^2) + rnorm(1, 0, sigma2)
}

# preparing data for plot
rownames(pred_temp) <- c("pred1", "pred2", "pred3", "pred4",
    "pred5", "pred6", "pred7", "pred8", "pred9", "pred10")
pred_temp <- t(pred_temp)

plot_data <- cbind(temperatures, pred_temp)

plot_data <- pivot_longer(plot_data, c(3:12))

ggplot(plot_data) + geom_point(aes(x = time, y = temp), color = "navy") +
    geom_line(aes(x = time, y = value, color = name)) + theme(legend.position = "right") +
    guides(color = guide_legend("Predictions")) + xlab("Time") +
    ylab("Temperature") + theme_classic()

# Task 1b i

set.seed(123456)

# matrix with the times (1st collum with 1 because of the
# intercept)
X <- cbind(rep(1, nrow(temperatures)), temperatures$time, temperatures$time^2)
```

```r
# vector with target values
y <- temperatures$temp
# calculate b hat
b_hat <- solve(t(X) %*% X) %*% t(X) %*% y

# number of samples
N <- 100

# matrix to store the beta values
b <- matrix(0, N, length(m_0))
# vector to store sigma2 values
sigma2 <- c()

for (i in 1:N) {

    # calculate mu_n
    m_n <- solve(t(X) %*% X + omega_0) %*% (t(X) %*% X %*% b_hat +
        omega_0 %*% m_0)

    # calculate omega_n
    omega_n <- t(X) %*% X + omega_0

    # calculate n_n
    n_n <- n_0 + nrow(temperatures)

    # calculate sigma_n
    sigma2_n <- (n_0 * sigma2_0 + (t(y) %*% y + t(m_0) %*% omega_0 %*%
        m_0 - t(m_n) %*% omega_n %*% m_n))/n_n

    # Inv-x simulator from lab 1
    sigma2[i] <- n_n * sigma2_n/rchisq(1, n_n)

    # generate b0,b1,b2
    b[i, ] <- rmvnorm(1, mean = m_n, sigma = sigma2[i] * solve(omega_n))

}

# df of sigma2 for plot
plot_df_sigma2 <- data.frame(sigma2 = sigma2)

# histogram of sigma2
ggplot(plot_df_sigma2, aes(x = sigma2)) + geom_histogram(bins = 30,
    color = "navy", fill = "steelblue2", aes(y = ..density..)) +
    geom_density(colour = "red4", size = 1) + ggtitle("sigma2 Posterior Distribution") +
    xlab("sigma2 Values") + ylab("Density") + theme_classic()

# df of betas for plot
plot_df_b <- data.frame(b0 = b[, 1], b1 = b[, 2], b2 = b[, 3])

# histogram of sigma2
ggplot(plot_df_b, aes(x = b0)) + geom_histogram(bins = 30, color = "navy",
    fill = "steelblue2", aes(y = ..density..)) + geom_density(colour = "red4",
    size = 1) + ggtitle("b0 Posterior Distribution") + xlab("b0 Values") +
```

```
    ylab("Density") + theme_classic()

# histogram of sigma2
ggplot(plot_df_b, aes(x = b1)) + geom_histogram(bins = 30, color = "navy",
    fill = "steelblue2", aes(y = ..density..)) + geom_density(colour = "red4",
    size = 1) + ggtitle("b1 Posterior Distribution") + xlab("b1 Values") +
    ylab("Density") + theme_classic()

# histogram of sigma2
ggplot(plot_df_b, aes(x = b2)) + geom_histogram(bins = 30, color = "navy",
    fill = "steelblue2", aes(y = ..density..)) + geom_density(colour = "red4",
    size = 1) + ggtitle("b2 Posterior Distribution") + xlab("b2 Values") +
    ylab("Density") + theme_classic()

# Task 1b ii

# matrix to store the predicted values
pred_temp2 <- matrix(NA, nrow(b), nrow(temperatures))

# calculate the predicted values
for (i in 1:nrow(b)) {
    pred_temp2[i, ] <- b[i, 1] + b[i, 2] * temperatures$time +
        b[i, 3] * (temperatures$time^2)
}

# calculate posterior's median for every value of time
post_median <- c()
for (i in 1:ncol(pred_temp2)) {
    post_median[i] <- median(pred_temp2[, i])
}

# calculate the 2.5 and 97.5 posterior percentiles of f
# (time)
intervals <- matrix(NA, nrow = 2, ncol = ncol(pred_temp2))
for (i in 1:ncol(pred_temp2)) {
    intervals[, i] <- quantile(pred_temp2[, i], probs = c(0.025,
        0.975))
}

# df for plot
plot_df_1b2 <- data.frame(temperatures = temperatures$temp, time = temperatures$time,
    interval1 = intervals[1, ], interval2 = intervals[2, ], medians = post_median)

# scatterplot of the temperatures & curve of the posterior
# median & curves for the 95% equal tail posterior
# probability intervals
ggplot(data = plot_df_1b2) + geom_point(aes(x = time, y = temperatures,
    color = "navy")) + geom_line(aes(x = time, y = medians, color = "red4")) +
    geom_line(aes(x = time, y = interval1, color = "green4")) +
    geom_line(aes(x = time, y = interval2, color = "purple4")) +
    theme(legend.position = "right") + scale_color_identity(guide = "legend",
    name = "", breaks = c("navy", "red4", "green4", "purple4"),
    labels = c("Temperatures", "Posterior's Median", "2.5 posterior percentile",
```

```
        "97.5 posterior percentile")) + xlab("Time") + ylab("Temperatures") +
    theme_classic()

# Task 1c

# getting the time with the highest expected temperature by
# using the above expression
max_temp <- max(-b[, 2]/(2 * b[, 3]))
max_temp

# __________________Assignment 2_________________#

# reading data
women <- read.table("WomenAtWork.dat", header = T)

# Task 2a

# given parameteres
tau <- 5

# target value
y = women$Work

# prior inputs Select which covariates/features to include
X <- as.matrix(women[2:8])
Xnames <- colnames(X)

Npar <- dim(X)[2]

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

LogPostLogistic <- function(betas, y, X, mu, Sigma) {
    linPred <- X %*% betas
    logLik <- sum(linPred * y - log(1 + exp(linPred)))
    if (abs(logLik) == Inf)
        logLik = -20000  # Likelihood is not finite, stear the optimizer away from here!
    logPrior <- dmvnorm(betas, mu, Sigma, log = TRUE)

    return(logLik + logPrior)
}

# 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
```

```r
# 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)
names(OptimRes$par) <- Xnames  # Naming the coefficient by covariates
approxPostStd <- sqrt(diag(-solve(OptimRes$hessian)))  # Computing approximate standard deviations.
names(approxPostStd) <- Xnames  # Naming the coefficient by covariates
print("The posterior mode is:")
print(OptimRes$par)
print("The approximate posterior standard deviation is:")
approxPostStd <- sqrt(diag(-solve(OptimRes$hessian)))
print(approxPostStd)
interval <- c(OptimRes$par[6] - 1.96 * approxPostStd[6], OptimRes$par[6] +
    1.96 * approxPostStd[6])
interval
```