Lab 2 report

Christian Kammerer(chrka821), Jakob Lindner (jakli758)

Lab 2

Exercise 1 Linear and polynomial regression

The dataset temp_linkoping.csv contains daily average temperatures (in degree Celcius) in Linköping between July 2023 and June 2024. Import the dataset in R. The response variable is temp and the covariate time, which is defined as

$$time = \frac{\textbf{the number of days since the beginning of the observation period}}{365}$$

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

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

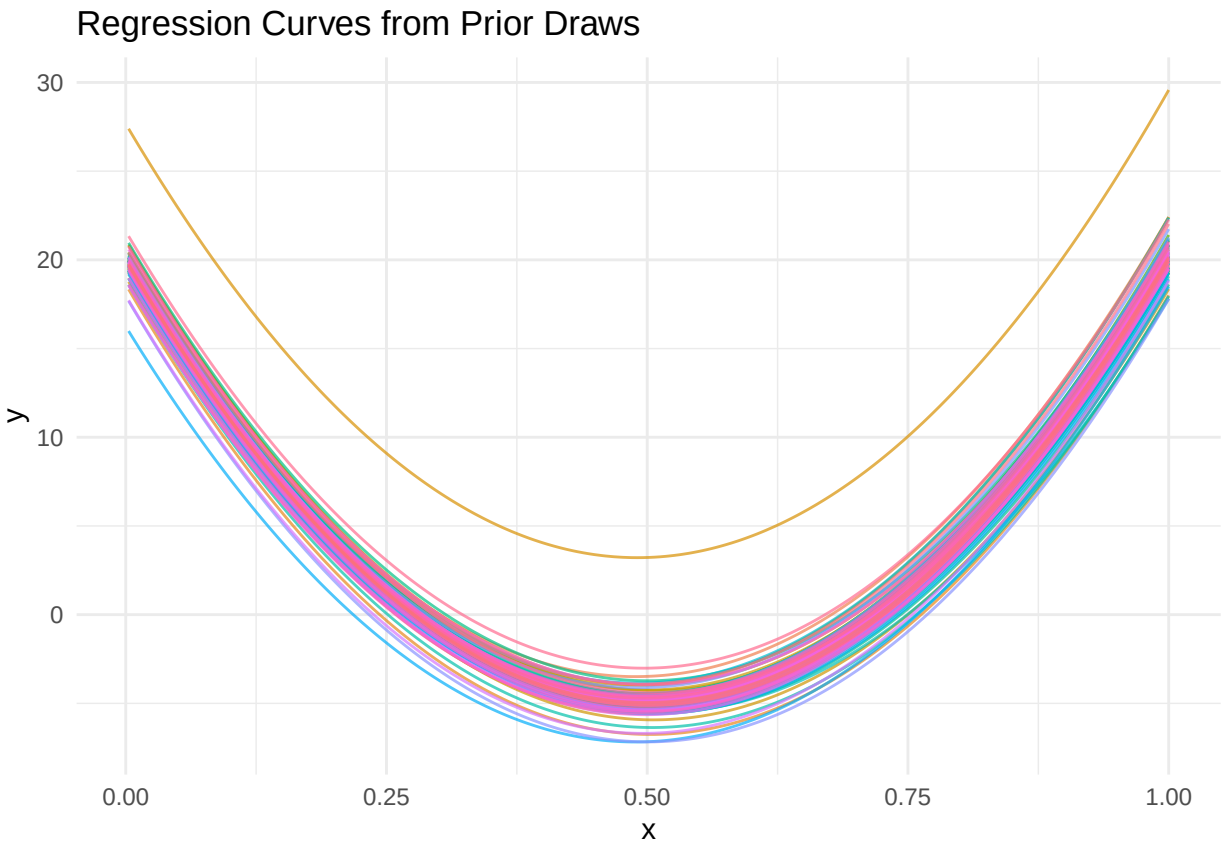
.

Part a)

Use the conjugate prior for the linear regression model. The prior hyper parameters μ_0, $\Omega 0$, ν_0 and σ_o^2 0 shall be set to sensible values. Start with μ_0 = (0, -100, 100)T , $\Omega 0 = 0.01 \cdot$ I3, $\nu_0 = 1$ 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. 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 $Scaled - Inv - \chi^2$ simulator of random draws from Lab 1.]

```
data <- read.csv("temp_linkoping.csv")

beta_prior_f <- function(n, mu0, sigma_sq, omega_0_inv){
  return(rmvnorm(n, mean = mu0, sigma = sqrt(sigma_sq * omega_0_inv)))
}

sigma_prior_f <- function(n, v0, sigma_sq_0){
  rinvchisq(n, df = v0, scale = sigma_sq_0)
}

joint_prior <- function(n=1, mu0 = c(20, -100, 100),
                        omega_0_inv = 0.01 * diag(3),
                        v0=1, sigma_sq_0=1){
  sigma_sq <- sigma_prior_f(1, v0, sigma_sq_0)
  beta <- beta_prior_f(1, mu0, sigma_sq, omega_0_inv)
  return(beta)
}
```

```r
beta_draws <- sapply(1:100, joint_prior)



# Transpose to make each row one set of coefficients
beta_draws_t <- t(beta_draws)

x_vals <- seq(1, 365, 1)/365

# Build data frame of predictions
curve_data <- lapply(1:nrow(beta_draws_t), function(i) {
  beta <- beta_draws_t[i, ]
  data.frame(
    x = x_vals,
    y = beta[1] + beta[2]*x_vals + beta[3]*x_vals^2,
    curve = paste0("Draw_", i)
  )
}) %>% bind_rows()

p <- ggplot(curve_data, aes(x = x, y = y, group = curve, color = curve)) +
  geom_line(alpha = 0.7) +
  labs(title = "Regression Curves from Prior Draws",
       x = "x", y = "y") +
  theme_minimal() +
  theme(legend.position = "none")

print(p)
```



The resulting regression curves from the initial values for the prior already showed the expected shape with the lowest temperature around half a year after the start and then going up towards a year later again. Only the values were a bit too low, so we set $mu_0 = (20, -100, 100)^T$ instead, to move to curve up a bit. This way it pretty much matches our prior believes.

Part b)

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

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

```
# Returns posterior parameters (mu_n, omega_n, sigma_sq_n, vn)
get_posterior_parameters <- function(mu0, omega0, sigma_sq_0, v0, y, X) {
  vn <- v0 + length(y)
  XtX <- t(X) %*% X
  omega_n <- XtX + omega0
  beta_hat <- solve(XtX) %*% t(X) %*% y
  mu_n <- solve(omega_n) %*% (XtX %*% beta_hat + omega0 %*% mu0)
  sigma_sq_n <- as.numeric(
    (v0 * sigma_sq_0 + t(y) %*% y + t(mu0) %*% omega0 %*% mu0 - t(mu_n) %*% omega_n %*% mu_n) / vn
  )

  return(list(
    mu_n = mu_n,
    omega_n = omega_n,
    sigma_sq_n = sigma_sq_n,
    vn = vn
  ))
```

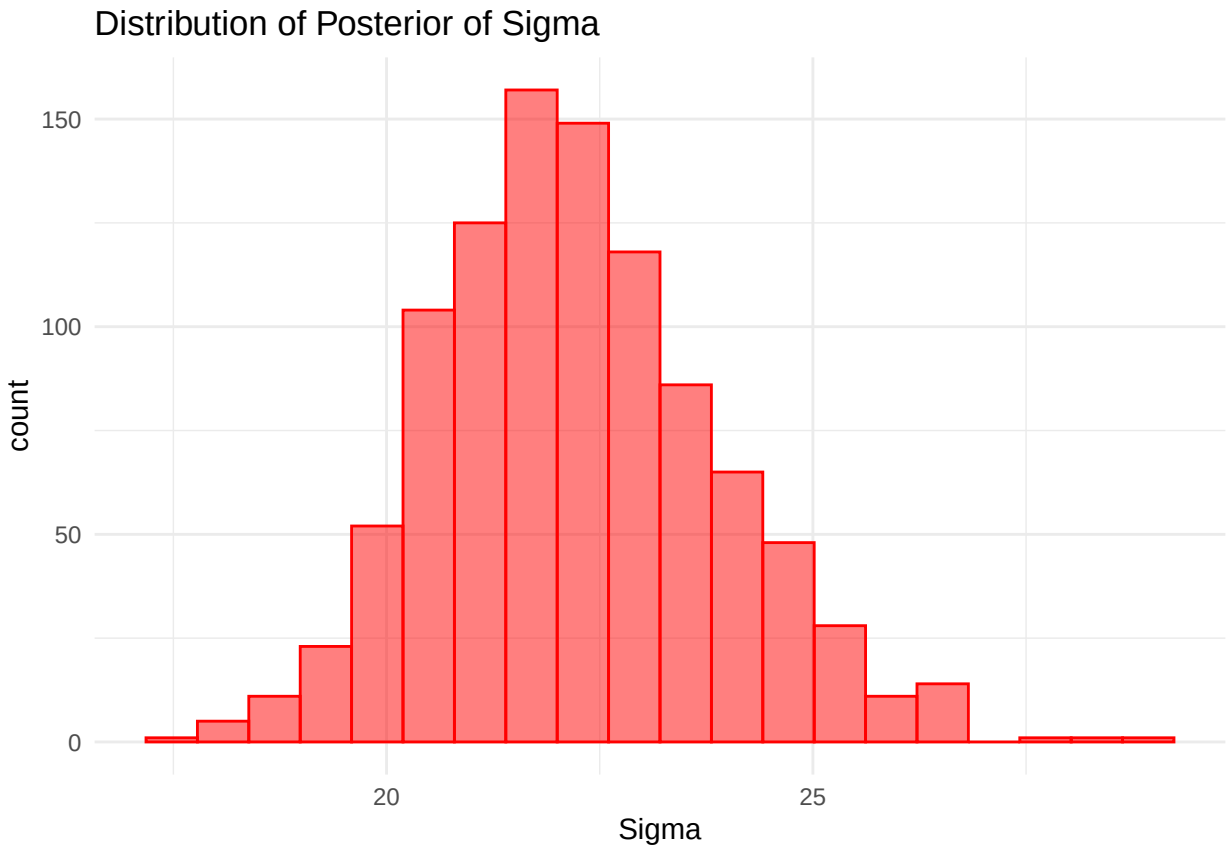
```r
}

sigma_posterior_f <- function(n = 1, posterior_params){
  with(posterior_params, {
    return(rinvchisq(n, df = vn, scale = sigma_sq_n))
  })
}

beta_posterior_f <- function(n = 1, posterior_params) {
  with(posterior_params, {
    # Sample sigma
    sigma_sq_samples <- sigma_posterior_f(n = n, posterior_params)

    # Sample beta conditional for each sigma
    beta_samples <- t(sapply(sigma_sq_samples, function(sigma_sq) {
      rmvnorm(1, mean = mu_n, sigma = sigma_sq * solve(omega_n))
    }))

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

marginal_beta <- function(n, mu, Sigma, df) {
  p <- length(mu)
  chi_samples <- rchisq(n, df)
  Z <- matrix(rnorm(n * p), nrow = n)  # standard normal
  t_samples <- sweep(Z, 1, sqrt(chi_samples / df), FUN = "/") %*% chol(Sigma)
  sweep(t_samples, 2, mu, FUN = "+")
}

X <- data$time
y <- data$temp

X <- cbind(1, X, X^2)
posterior_params <- get_posterior_parameters(mu0 = c(20, -100, 100),
                                             omega0 = 0.01 * diag(3),
                                             sigma_sq_0 = 1, v0 = 1,
                                             y, X)

n_post = 1000
posterior_draws <- beta_posterior_f(n = n_post, posterior_params)
beta_posterior_draws <- posterior_draws$beta
sigma_posterior_draws <- posterior_draws$sigma

beta_df <- as.data.frame(beta_posterior_draws)
beta_df$beta0 <- beta_df$V1
beta_df$beta1 <- beta_df$V2
beta_df$beta2 <- beta_df$V3

beta_long <- tidyr::gather(beta_df, key = "Beta", value = "Value", beta0, beta1, beta2)

# Plot overlapping histograms
p <- ggplot(beta_long, aes(x = Value, fill = Beta)) +
```

```
  geom_histogram(alpha = 0.5, position = "identity", bins = 200) +
  scale_fill_manual(values = c("red", "blue", "green")) +
  labs(title = "Overlapping Histograms of Betas",
       x = "Beta values",
       y = "Frequency") +
  theme_minimal() +
  theme(legend.title = element_blank(), legend.position = c(0.8, 0.8))
```

```
## Warning: A numeric `legend.position` argument in `theme()` was deprecated in ggplot2
## 3.5.0.
## i Please use the `legend.position.inside` argument of `theme()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
print(p)
```

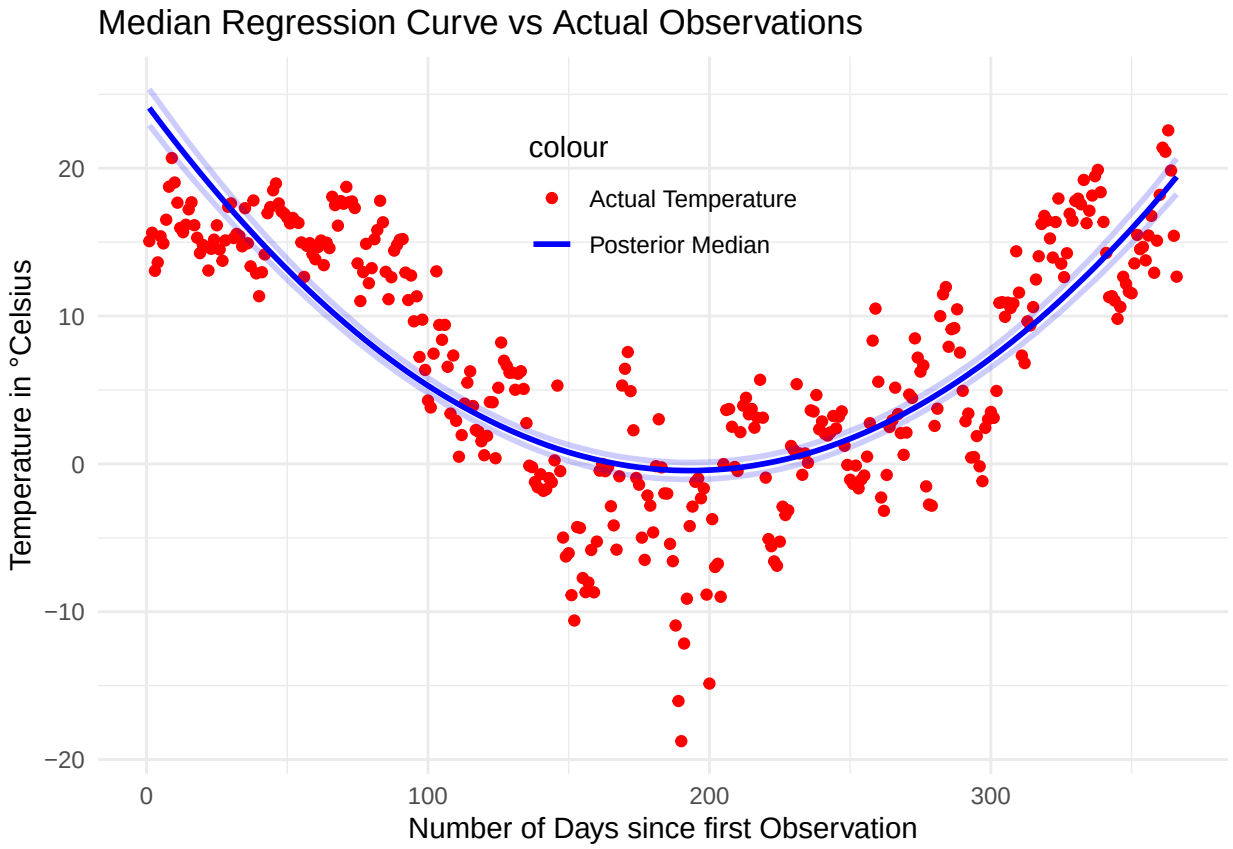

```
p <- ggplot() +
  geom_histogram(aes(x = sigma_posterior_draws), alpha = 0.5, position = "identity", bins = 20, col = "
  xlab("Sigma") +
  ggtitle("Distribution of Posterior of Sigma") +
  theme_minimal()

print(p)
```



Subpart ii) Make a scatter plot of the temperature data and overlay a curve for the posterior median of the regression function $f(time) = E[temp|time] = \beta_0 + \beta_1 \cdot time + \beta_2 \cdot time^2$, 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?

```
apply_regression <- function(beta_values, X){
  return(beta_values[1] + beta_values[2] * X[2] + beta_values[3] * X[3])
}

temp_matrix <- matrix(ncol = nrow(beta_posterior_draws), nrow = nrow(X))

for(i in 1:nrow(temp_matrix)){
  temp_matrix[i, ] <- apply(beta_posterior_draws, 1, function(betas) apply_regression(betas, X[i, ]))
}

median_temp <- apply(temp_matrix, 1, median)
fifth_perc <- apply(temp_matrix, 1, function(row) quantile(row, 0.05))
ninetyfifth_perc <- apply(temp_matrix, 1, function(row) quantile(row, 0.95))

temperature_frame <- data.frame(actual = y, median_posterior_temp = median_temp,
                                lower_bound = fifth_perc, upper_bound = ninetyfifth_perc)

p <- ggplot(temperature_frame) +
  aes(x = 1:366) +
  geom_point(aes(y = actual, color = "Actual Temperature"), shape = 21, fill = "red") +
```

```
  geom_line(aes(y = median_posterior_temp, color = "Posterior Median"), linewidth = 1) +
  geom_line(aes(y = lower_bound), color = "blue", alpha = 0.2, linewidth = 1, show.legend = FALSE) +
  geom_line(aes(y = upper_bound), color = "blue", alpha = 0.2, linewidth = 1, show.legend = FALSE) +
  scale_color_manual(values = c("Actual Temperature" = "red",
                                "Posterior Median" = "blue")) +
  xlab("Number of Days since first Observation") +
  ylab("Temperature in °Celsius") +
  ggtitle("Median Regression Curve vs Actual Observations") +
  theme_minimal() +
  theme(legend.position = c(0.5, 0.8))

print(p)
```



The plot shows the actual data points in red and the curve for the posterior median in blue. The upper and lower bound for the 5 and 95 posterior intervals are displayed in lighter blue. It shows, that only a minority of data points are contained in this interval. This is not unexpected behavior, as the interval only shows the uncertainty of the regression curve but not for single data points. We can only say, that with 90% probability, the regression curve is in this interval.

Part c)

It is of interest to locate the time with the lowest expected temperature (i.e. the time where $f(time)$ is minimal). Let's call this value $\tilde{x}$. Use the simulated draws in (b) to simulate from the posterior distribution of $\tilde{x}$. You can solve this analytical or numerical. [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 $\tilde{x}$.]

Solving it analytically gives that $\tilde{x}$ can be computed as

$$-\frac{1}{2} \cdot \frac{\beta_1}{\beta_2}$$

```
# Posterior distribution of value that minimizes regression curve
posterior_x <- function(betas){
  return(-1/2 * (betas[2]/betas[3]))
}
x_s <- apply(beta_posterior_draws, 1, posterior_x) * 365
```

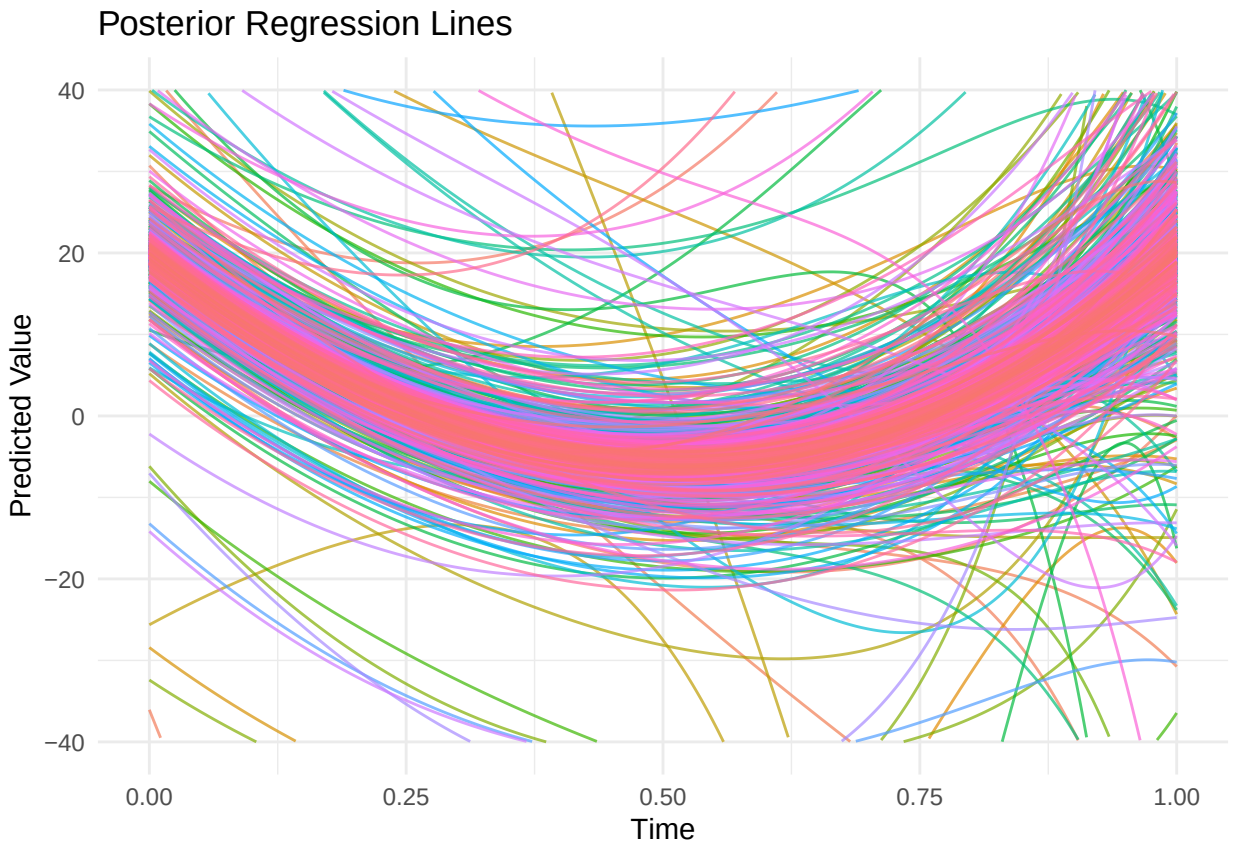
The mean of the simulated values is 192.0393801, which matches the impression we got from the graph.

Part 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 and motivate your choice. Repeat task (a) for the selected prior and the higher order polynomial to see if it behaves as expected. [Hint: the task is to specify μ_0 and Ω_0 in a suitable way.]

```
X <- data$time
X <- cbind(1, X, X^2, X^3, X^4, X^5, X^6, X^7, X^8, X^9, X^10)

m <- 10
n <- 1000
eta0 <- 10
lambda0 <- 1
mu0 <- c(20, -100, 100, -10, 10, -1, 1, -0.1, 0.1, -0.01, 0.01)
sigma_sq_0 <- 1
v0 <- 1
lambda_prior_draws <- rinvchisq(n, eta0, lambda0) # prior of lambda
sigma_prior_draws <- rinvchisq(n, v0, sigma_sq_0)
beta_prior_draws <- t(sapply((1:n), function(i) rmvnorm(n = 1, mean = mu0,
                                    sigma = (sigma_prior_draws[i] * 1/lambda_prior_draws[i] * diag(m + 1
temp_reg_values <- apply(beta_prior_draws, 1, function(betas) X %*% betas)

time_values <- data$time
temp_reg_df <- as.data.frame(temp_reg_values)
temp_reg_df$time <- time_values

# Convert to long format
long_df <- pivot_longer(temp_reg_df, cols = -time, names_to = "draw", values_to = "y")

# Assuming 'long_df' is structured like curve_data:
# long_df has columns: time (x-axis), y (y-axis), draw (equivalent to "curve")

p <- ggplot(long_df, aes(x = time, y = y, group = draw, color = draw)) +
  geom_line(alpha = 0.7) +
  labs(title = "Posterior Regression Lines",
       x = "Time", y = "Predicted Value") +
  ylim(-40, 40) +
  theme_minimal() +
  theme(legend.position = "none")

print(p)
```

```
## Warning: Removed 5415 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



As prior μ_0 we chose similar values as for the quadratic regression with alternating positive and negative values that get smaller. Depending on the η_0 and λ_0, the curves are more or less similar. We decided to allow some uncertainty with some curves to be off the prior believes with e.g. a max value, where the temperature should be the lowest, but still the most curves match the prior believe.

Exercise 2: Posterior approximation for classification with logistic regression

The dataset Disease.xlsx contains n = 313 observations on seven variables related to a certain disease.

Part 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 person has a disease and 0 if not. x is a 7-dimensional vector containing six features and a column of 1's to include an intercept in the model. 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 $J(\tilde{beta}) = -\frac{\delta^2 \ln p(\beta|y)}{\delta\beta\delta\beta^T}$ is the negative of the observed Hessian evaluated at the posterior mode. Note that $\frac{\delta^2 \ln p(\beta|y)}{\delta\beta\delta\beta^T}$ is a 7 × 7 matrix with second derivatives

on the diagonal and cross-derivatives $\frac{\delta^2 \ln p(\beta|y)}{\delta\beta\delta\beta^T}$ 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 $\beta \sim N(0, \tau 2I)$, where $\tau = 2$. Present the numerical values of $\tilde{\beta}$ and $J_y^-1(\tilde{\beta})$ for the Disease data. Compute an approximate 95% equal tail posterior probability interval for the regression coefficient to the variable Age. Would you say that this feature is of importance for the probability that a person has the disease? [Hint: You can verify that your estimation results are reasonable by comparing the posterior means to the maximum likelihood estimates, given by: glmModel <- glm(Class_of_diagnosis ~ 0 + ., data = Disease, family = binomial).]

```
library(mvtnorm)
data <- read.csv("Disease.csv")

Covs <- c(1:6)
Nobs <- nrow(data)
Npar <- ncol(data) -1
tau <- 2
mu <- as.matrix(rep(0,Npar)) # Prior mean vector
Sigma <- tau^2*diag(Npar) # Prior covariance matrix

y <- data$class_of_diagnosis
X <- data[,1:6]
Xnames <- colnames(X)
X <- as.matrix(X)
# 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 h
  logPrior <- dmvnorm(betas, mu, Sigma, log=TRUE);

  return(logLik + logPrior)
}

initVal <- matrix(0,Npar,1)

OptimRes <- optim(initVal,LogPostLogistic,gr=NULL,y,X,mu,Sigma,method=c("BFGS"),control=list(fnscale=-1)

approxPostMode <- matrix(OptimRes$par,1, length(Covs)) # beta-tilde (MLE)
colnames(approxPostMode) <- Xnames
approxPostStd <- sqrt(diag(solve(-OptimRes$hessian))) # Computing approximate standard deviations.
names(approxPostStd) <- Xnames

Cred_int <- matrix(0,2,length(Covs)) # Create 95 % approximate credibility intervals for each coefficie
Cred_int[1,] <- approxPostMode - 1.96*approxPostStd # LCI
Cred_int[2,] <- approxPostMode + 1.96*approxPostStd # UCI
colnames(Cred_int) <- Xnames
rownames(Cred_int) <- c("LCI","UCI") # CI
j_theta <- -OptimRes$hessian
j_theta_inv <- solve(j_theta)
```

```
print(OptimRes$hessian) # Mode of Posterior of Beta
```

```
##               [,1]         [,2]         [,3]         [,4]          [,5]
## [1,]     -63.59379    -3454.879    -36.27311    -342.2145     -51.35340
## [2,]   -3454.87892  -196215.169  -1967.87484  -18671.2186   -2800.39527
## [3,]     -36.27311    -1967.875    -36.52311    -178.9516     -28.90764
## [4,]    -342.21451   -18671.219   -178.95157   -3259.2906    -274.49968
## [5,]     -51.35340    -2800.395    -28.90764    -274.4997     -51.60340
## [6,] -155574.99997 -8498888.456 -89582.09807 -843336.7710 -126157.00890
##               [,6]
## [1,]     -155575.0
## [2,]    -8498888.5
## [3,]      -89582.1
## [4,]     -843336.8
## [5,]     -126157.0
## [6,] -1343492362.7
```

```
print(j_theta_inv) # J(theta)^-1
```

```
##               [,1]          [,2]          [,3]          [,4]          [,5]
## [1,]  4.674305e-01 -6.247225e-03 -4.295278e-02 -4.025898e-03 -7.545537e-02
## [2,] -6.247225e-03  1.176055e-04  1.094136e-05 -6.614883e-06 -1.223894e-04
## [3,] -4.295278e-02  1.094136e-05  6.397978e-02  7.360040e-04  2.548374e-03
## [4,] -4.025898e-03 -6.614883e-06  7.360040e-04  7.143854e-04  1.651182e-04
## [5,] -7.545537e-02 -1.223894e-04  2.548374e-03  1.651182e-04  9.893004e-02
## [6,] -2.131620e-06 -5.630896e-09 -6.270818e-08 -4.974235e-09 -5.144255e-08
##               [,6]
## [1,] -2.131620e-06
## [2,] -5.630896e-09
## [3,] -6.270818e-08
## [4,] -4.974235e-09
## [5,] -5.144255e-08
## [6,]  1.038923e-09
```

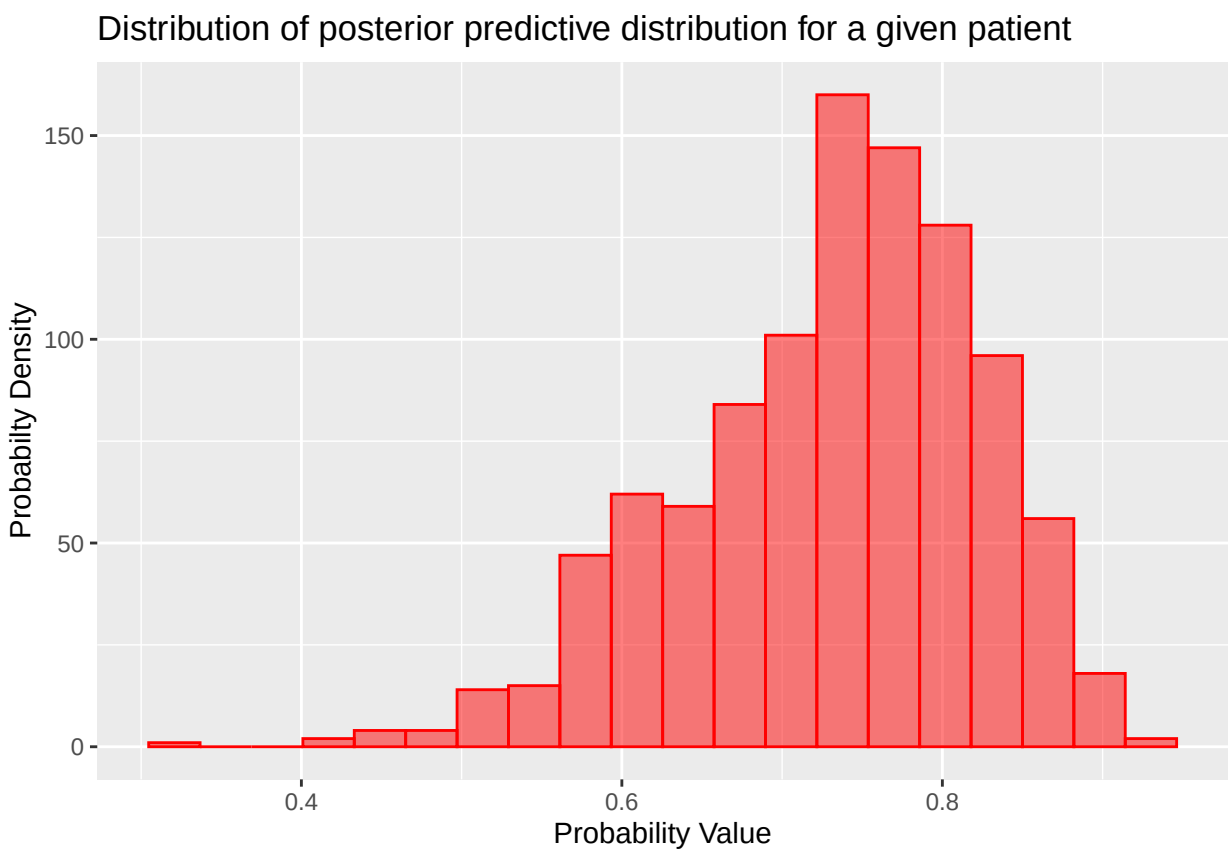
The numerical values lower bound of the confidence interval for the regression coefficient of the variable age is -0.0213 and the upper bound is 0.0213 . Since both the lower and upper bound are very close to zero, and in fact one is negative and the other is positive, it is indicative of no significant relation between the age variable and our disease.

Part b)

Use your normal approximation to the posterior from (a). Write a function that simulate draws from the posterior predictive distribution of $Pr(y = 1|x)$, where the certain diagnosis method was used and where the values of x corresponds to a 38-year-old woman, with 10 days of symptoms, no labored breathing and 11000 white blood cells per microliter. Plot the posterior predictive distribution of $Pr(y = 1|x)$ for this person. [Hints: The R package mvtnorm will be useful. Remember that $Pr(y = 1|x)$ can be calculated for each posterior draw of β.]

```
draw_betas <- function(n, mus, cov_matrix){
  return(rmvnorm(n, mean = mus, sigma = cov_matrix))
  }

beta_draws <- draw_betas(1000, approxPostMode, solve((-OptimRes$hessian)))
test_subject <- c(1, 38, 1, 10, 0, 11000)

posterior_predictive_distribution <- 1/(1 + exp(beta_draws %*% test_subject))
```

```r
p <- ggplot() +
  geom_histogram(aes(x = posterior_predictive_distribution), alpha = 0.5, position = "identity", bins =
  xlab("Probability Value") +
  ylab("Probabilty Density") +
  ggtitle("Distribution of posterior predictive distribution for a given patient")

print(p)
```



The plot of the posterior predictive distribution shows that the probability that the person has the disease it quite high.