TDDE07 Bayesian Learning lab2

William Bergekrans & Shahin Salehi

2021-04-22

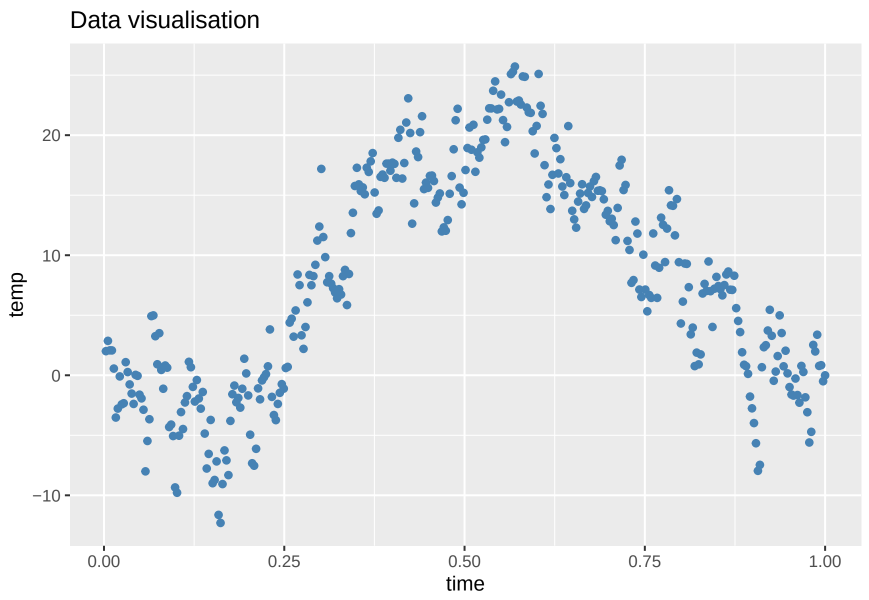
1. Linear and polynomial regression

The quadratic regression model to be used is

$$temp = \beta_0 + \beta_1 * time + \beta_2 * time^2 + \varepsilon,\ \varepsilon \sim N(0, \sigma^2)$$

.

The temperature data looks as follows:

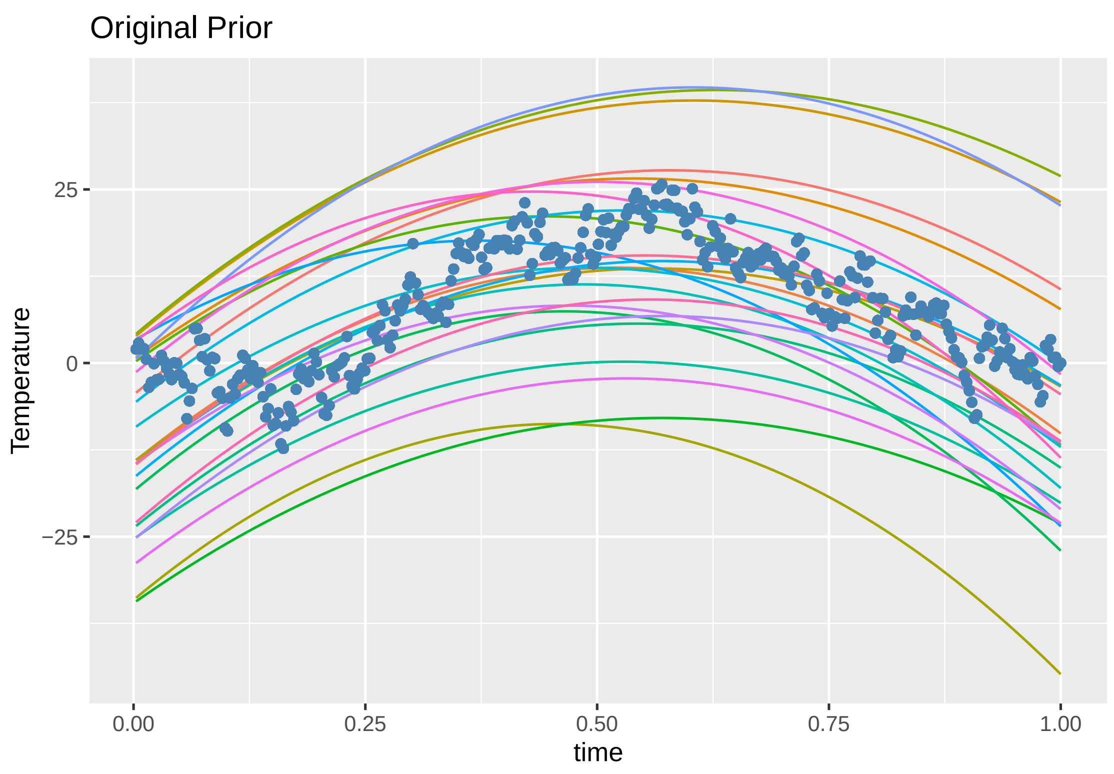


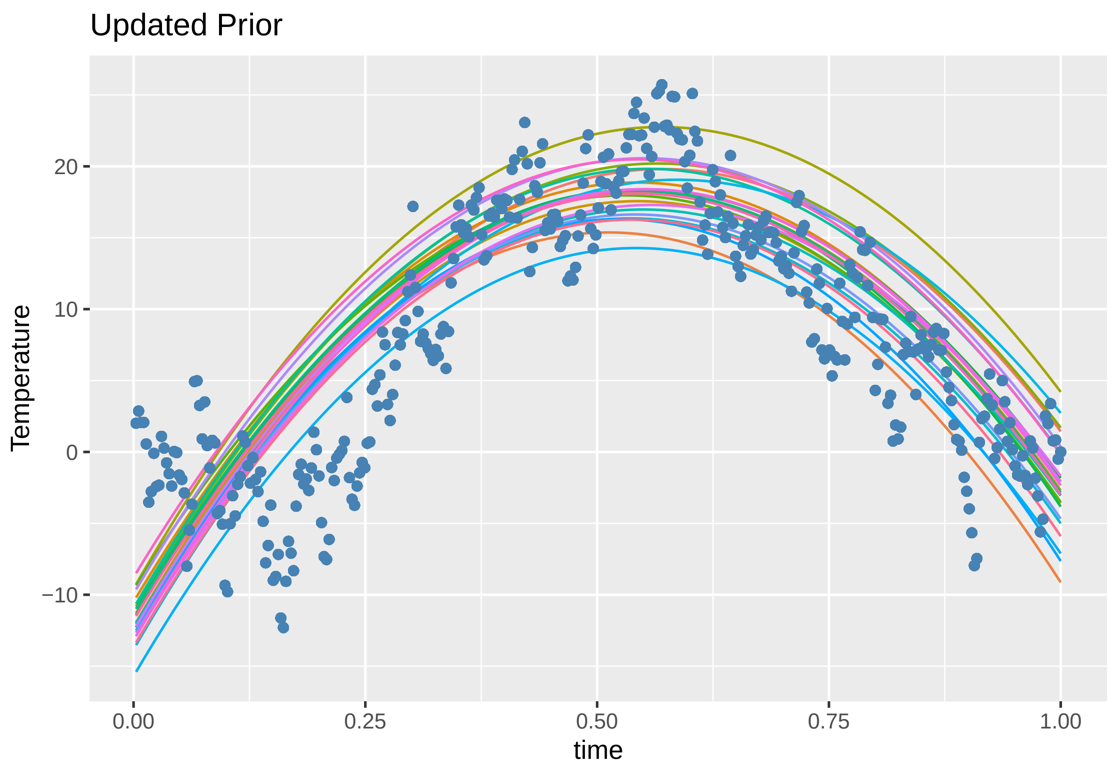
1a) The joint conjugate prior for β and σ^2 is:

$$\beta|\sigma^2 \sim N(\mu_0, \sigma^2\Omega_0^{-1})$$

$$\sigma^2 \sim Inv - \chi^2(v_0, \sigma_0^2)$$

We start with the values $\mu_0 = (-10, 100, -100)^T$, $\Omega_0 = 0.01 * I_3$, $v_0 = 6$ and $\sigma_0^2 = 1$. In the first graph random draws using this prior is shown. In the second graph this prior is updated with new values in order to fit the data (our belief) better.



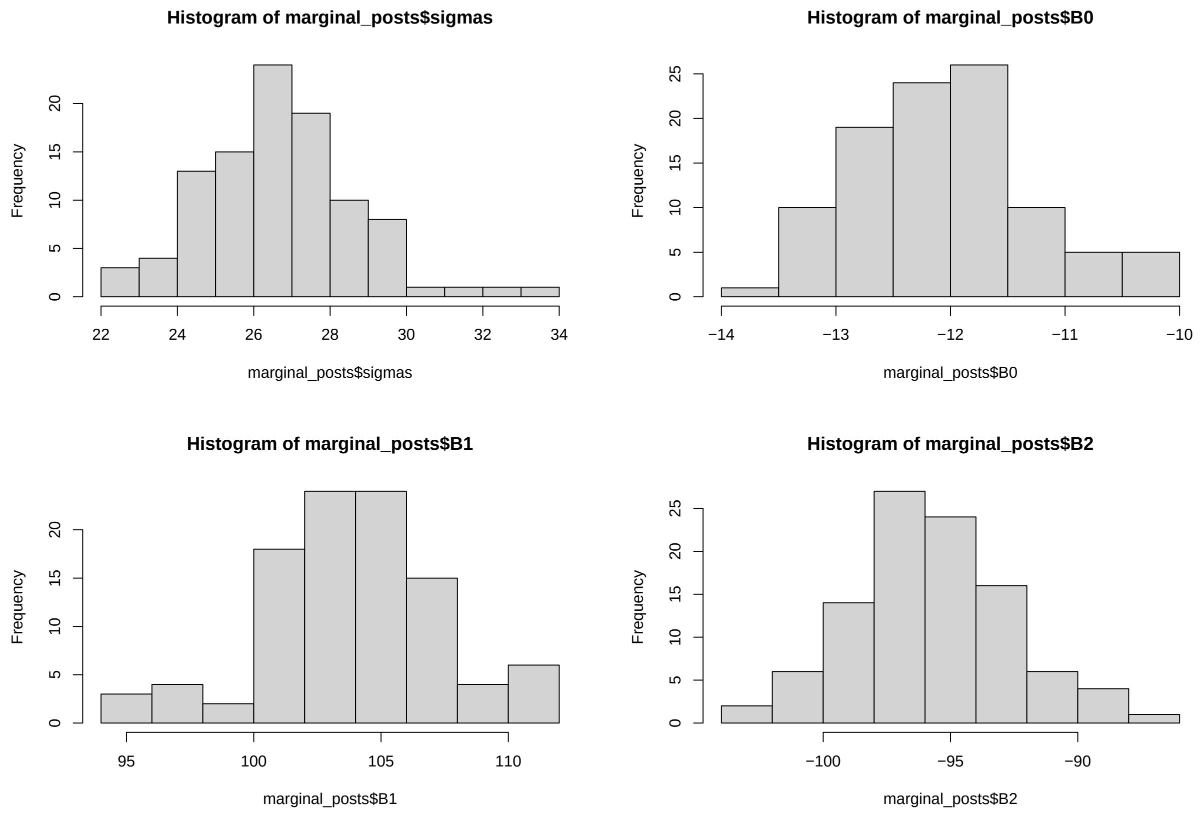


The updated prior has the following values: $\mu_0 = (-12, 110, -100)^T$, $\Omega_0 = 0.05 * I_3$, $v_0 = 6$ and $\sigma_0^2 = 0.2$.

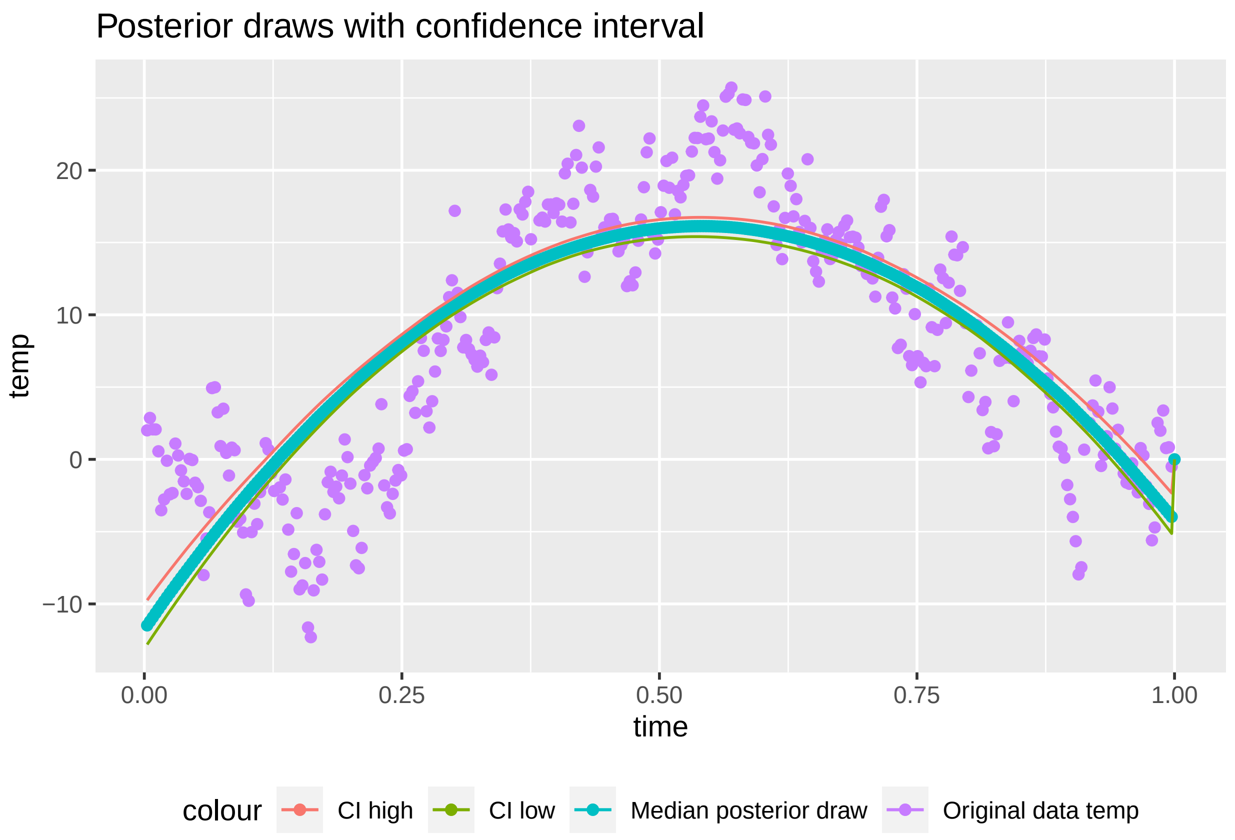
The code for 1a:

```r
# Prior values
u0 <- t(c(-10, 100, -100))
omega0 <- diag(x = 0.01, 3,3)
v0 <- 6
s_square0 <- 1
draws <- 25

# Function for generating sigma^2 draws from a inverse chi prior
sigma_draw <- function(draws, v0, s0){
  x <- rchisq(draws, v0)
  sigmas <- (v0 * s0) / x
  return(sigmas)
}

# Function for generating beta draws from a multinormal distribution
beta_draw <- function(draws, u0, s0, omega0){
  betas <- matrix(0, draws, 3) # Matrix to store betas in
  i = 1
  for (s in s0) {
    betas[i,] <- rmvnorm(1,u0, sigma = s*solve(omega0))
    i = i+1
  }
  return(betas)
}

draw_prior <- function() {
  # Draw sigma and beta values from the prior distribution
  sigmas <- sigma_draw(draws, v0, s_square0)
  betas <- beta_draw(draws, u0, sigmas, omega0)

  # Matrix for X (covariates)
  n <- length(temps$time)
  x <- matrix(c(rep(1, n), temps$time, temps$time ^ 2),
          nrow = n,
          ncol = 3)

  test_prior <- data.frame("time" = temps$time, "temp" = temps$temp)

  # Draw temperature samples from the prior
  for (i in 1:draws) {
    eps <- rnorm(n = 1,
                 mean = 0,
                 sd = sqrt(sigmas[i]))
    y <- x %*% betas[i, ] + eps
    test_prior[paste0("model", i)] <- drop(y)
  }
  test_prior <- melt(test_prior, id.vars = c("time", "temp"), variable.name ="models")
  return(test_prior)
}

test_prior <- draw_prior()

ggplot(test_prior, aes(x=time, y=value, group=models, color=models)) +
  geom_line(show.legend = FALSE) +
  geom_point(aes(y=temp), color="steelblue") +
  ggtitle("Original Prior") +
  ylab("Temperature")

# Now we want to obtain a prior that better matches our beliefs about the data.
u0 <- t(c(-12, 110, -100))
omega0 <- diag(x = 0.05, 3,3)
v0 <- 6
s_square0 <- 0.2
draws <- 25

test_prior <- draw_prior()

ggplot(test_prior, aes(x=time, y=value, group=models, color=models)) +
  geom_line(show.legend = FALSE) +
  geom_point(aes(y=temp), color="steelblue") +
  ggtitle("Updated Prior") +
  ylab("Temperature")
```

1b) Now we shall simulate draws from the joint posterior distribution of β_0, β_1, β_2 and σ^2. In the following plots we see the marginal posterior of the different parameters.



Now we use the posterior draws to generate a new curve for the posterior median of the regression function f(time).



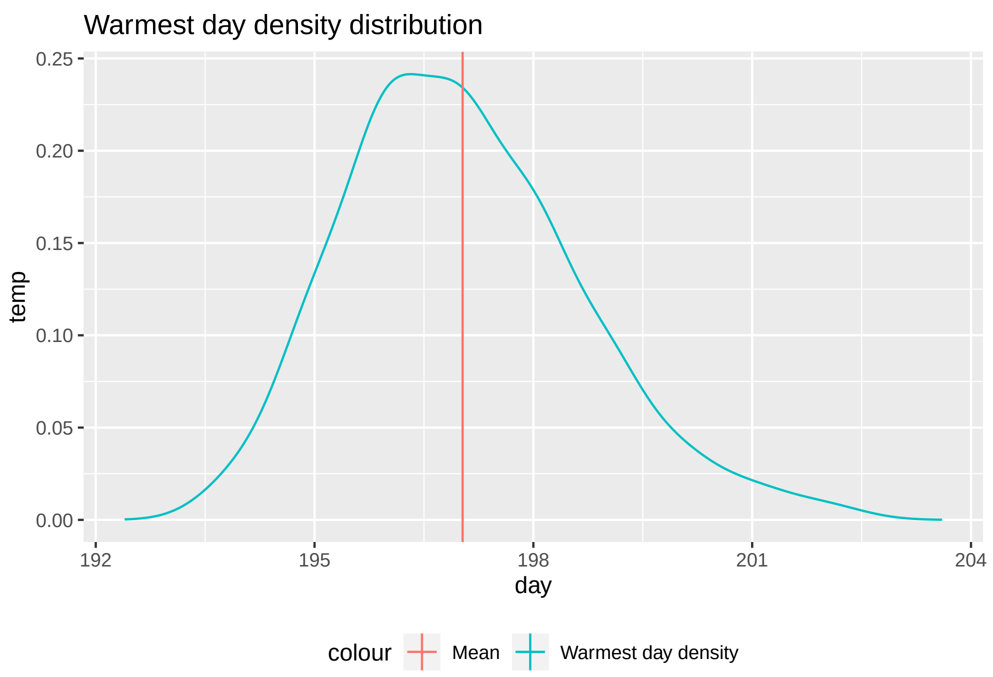
As we can see in the plot the confidence interval for the generated points from the posterior is very narrow and does not include many of the points from the original data set. A better model should probably include more of the data points in the confidence interval as the current model will very likely give a bad prediction for certain times of the year. However, the current model is of the second degree and cannot fit the data perfectly. The confidence interval reflects that our model have quite low variance.

Code for 1b:

```
n <- length(temps$time)
x <- matrix(c(rep(1, n), temps$time, temps$time ^ 2), nrow = n, ncol = 3)
sigmas <- sigma_draw(draws, v0, s_square0)
betas <- beta_draw(draws, u0, sigmas, omega0)
y <- temps$temp

draw_posterior <- function(y, n, x, u0, omega0, v0, s_square0) {
  # Calculate the parameters for the posterior
  b_hat <- solve(t(x) %*% x) %*% t(x) %*% y
  un <- solve(t(x) %*% x + omega0) %*% (t(x) %*% x %*% b_hat + omega0 %*% t(u0))
  omega_n <- t(x) %*% x + omega0
  vn <- v0 + n
  vnsigma2 <- v0 %*% s_square0 + (t(y)%*%y + u0%*%omega0%*%t(u0) - t(un)%*%omega_n%*%un)
  sigma_n <- vnsigma2 / vn

  # Draw sigma and beta values from the prior distribution
  sigmas <- sigma_draw(draws, vn, sigma_n)
  betas <- beta_draw(draws, un, sigmas, omega_n)
  res <- data.frame(sigmas, betas)
```

```r
  res <- rename(res, B0 = X1, B1 = X2, B2 = X3)
  return(res)
}

draws <- 100
marginal_posts <- draw_posterior(y, n, x, u0, omega0, v0, s_square0)

# Plot the histograms of the marginal posteriors of the parameters
hist(marginal_posts$sigmas)
hist(marginal_posts$B0)
hist(marginal_posts$B1)
hist(marginal_posts$B2)

# 1b: ii
f_res <- integer(365)
high <- integer(365)
low <- integer(365)
preds <-  matrix(0,365,draws)
i = 0
for (time in temps$time) {
  fval <- marginal_posts$B0 + marginal_posts$B1 %*% t(time) + marginal_posts$B2 %*% t((time^2))
  preds[i,] <- fval
  f_res[i] <- median(fval)
  # confidence interval
  interval <- ci(fval, ci=0.95, method="ETI")
  high[i] <- interval$CI_high
  low[i] <- interval$CI_low
  i = i+1
}

df <- (data.frame("y" = f_res, "CI_high" = high, "CI_low" = low, "time" = temps$time, "temp" =temps$tem

ggplot(df, aes(x=time, y=temp, color="Original data temp")) +
  geom_point() +
  geom_point(aes(y=f_res, color="Median posterior draw")) +
  geom_line(aes(y=high, color="CI high")) +
  geom_line(aes(y=low, color="CI low")) +
  ggtitle("Posterior draws with confidence interval") +
  theme(legend.position = "bottom")
```

1c) Now we want to simulate the posterior distribution of $\tilde{x}$.



The mean for the warmest day is 197.03.

The code for 1c:

```r
# 1c,
# Get the day of the max temp for each draw
# Based on that the days are in order (pos in list is day)
max_days <- apply(preds,2,which.max)
days_densiity <- density(max_days)

df1 <- data.frame(
  day = days_densiity$x,
  temp = days_densiity$y
)

ggplot(df1, aes(x=day, y=temp, color="Warmest day density")) +
  geom_line() +
  geom_vline(aes(xintercept = mean(max_days), color="Mean")) +
  ggtitle("Warmest day density distribution") +
  theme(legend.position = "bottom")
```

1d) Now we shall define μ_0 and Ω_0 to mitigate the problem of over-fitting for a polynomial regression model of order 7.

The problem is that too many knots in spline-regression can lead to over-fitting. The solution is to add a shrinkage prior. In such a prior $\mu_0 = 0$ and $\Omega_0 = \lambda I$. A larger λ gives a smoother fit, we therefore need to choose a λ that gives a smooth enough fit. Based on slide 10 from lecture 5 a lambda of around 2 should be sufficient for making sure that the model is not over-fitted. This can be tested with cross-validation on the data to see how the model fits the data but as the tasks says we do not carry out any calculations.

2. Posterior approximation for classification with logistic regression

The dataset contains 200 observations with data related to women. There is 1 target variable and 8 features.

2a) The logistic regression model to be used is: $Pr(y=1|x) = \frac{exp(x^T\beta)}{1+exp(x^T\beta)}$. y is 1 if the woman works and 0 if she does not. We want to approximate the posterior distribution of β with the multivariate normal distribution.

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

The posterior mode is $\tilde{\beta}$ and $J(\tilde{\beta})$ is the negative of the observed Hessian evaluated at the posterior mode. We use the optim-function to calculate both of these values.

For the calculations we use the prior $\beta \sim N(0, \tau^2 I)$ and $\tau = 10$.

The numerical values calculated for $\tilde{\beta}$ and $J_y^{-1}(\tilde{\beta})$ are:

The beta values are

Beta	Value
B0	0.6267288
B1	-0.0197911
B2	0.1802190
B3	0.1675667
B4	-0.1445967
B5	-0.0820656
B6	-1.3591332
B7	-0.0246835

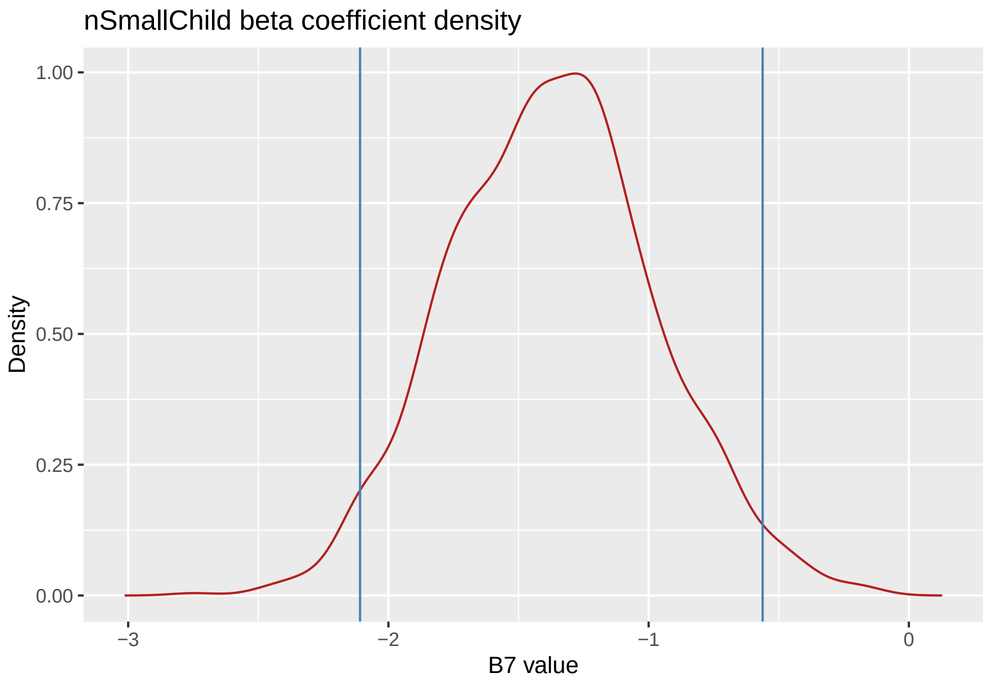
and the inverse hessian is

2.2660226	0.0033389	-0.0654512	-0.0117914	0.0457807	-0.0302934	-0.1887484	-0.0980239
0.0033389	0.0002528	-0.0005610	-0.0000313	0.0001415	-0.0000359	0.0005067	-0.0001444
-0.0654512	-0.0005610	0.0062182	-0.0003558	0.0018963	-0.0000032	-0.0061346	0.0017527
-0.0117914	-0.0000313	-0.0003558	0.0043517	-0.0142491	-0.0001341	-0.0014690	0.0005437
0.0457807	0.0001415	0.0018963	-0.0142491	0.0555787	-0.0003299	0.0032083	0.0005120
-0.0302934	-0.0000359	-0.0000032	-0.0001341	-0.0003299	0.0007185	0.0051842	0.0010953
-0.1887484	0.0005067	-0.0061346	-0.0014690	0.0032083	0.0051842	0.1512622	0.0067689
-0.0980239	-0.0001444	0.0017527	0.0005437	0.0005120	0.0010953	0.0067689	0.0199723

The coefficients calculated using the glm-function is:

	x
(Intercept)	0.6443036
HusbandInc	-0.0197746
EducYears	0.1798806
ExpYears	0.1675127
ExpYears2	-0.1443595
Age	-0.0823403
NSmallChild	-1.3625024
NBigChild	-0.0254299

We see that the beta values above are very similar to the coefficients calculated with the glm-function. Next we draw from the posterior distribution for the beta coefficients and calculate a 95% equal tailed probability interval for the coefficient for the variable NSmallChild.



As we can see the confidence interval does not include 0 which indicates that this variable is of importance for the probability that a women works. The magnitude of the coefficient for NSmallChild is also large in comparison to other coefficients, strengthening the belief that this variable is important.

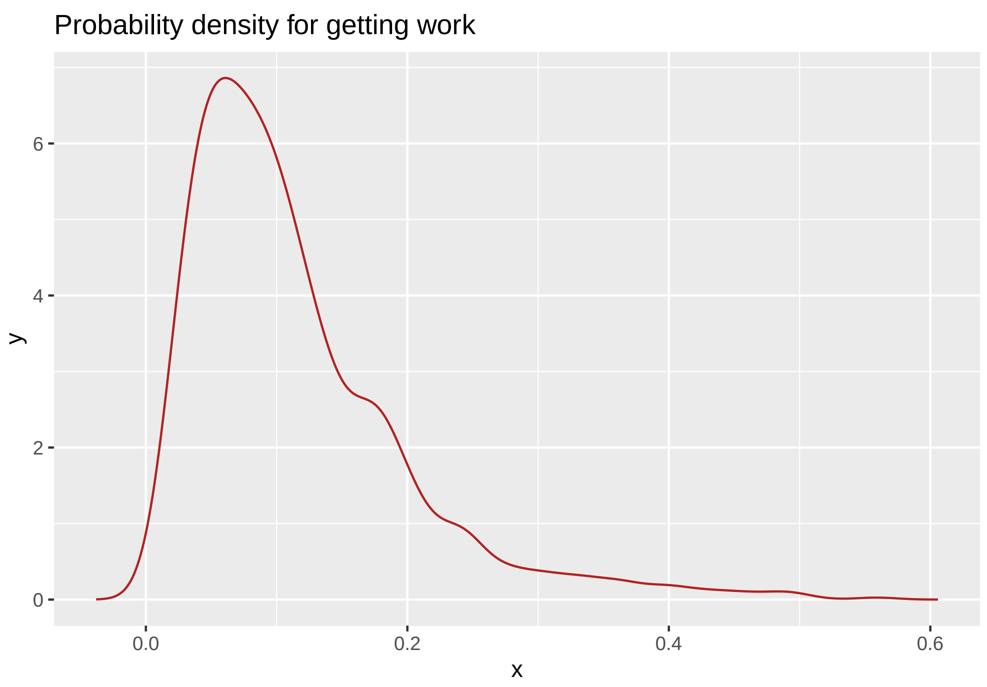
```r
# Setting up the model
x <- as.matrix(data[,2:9])
y <- as.vector(data[,1])
tau <- 10
Npar <- dim(x)[2]

# Setting up the prior
u0 <- as.matrix(rep(0, Npar)) # prior mean vector
sigma <- (tau^2)*diag(Npar) # prior covariance vector

# Function for calculating posterior, Taken from lecture 6 code example.
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 he

  logPrior <- dmvnorm(betas, mu, Sigma, log=TRUE);

  return(logLik + logPrior)
}

init_values <- as.matrix(rep(0,Npar)) # Initial beta values
# Use optim to calculate the best beta values for the model
opt_res <- optim(init_values, fn=LogPostLogistic, gr = NULL, y, x, u0, sigma, method=c("BFGS"), control=

# Calculate the inverse of the hessian
invJ <- -solve(opt_res$hessian)
```

```r
beta <- data.frame(
  Beta = c("B0", "B1", "B2", "B3", "B4", "B5", "B6", "B7"),
  Value = opt_res$par
)

# Check if the results are similar so far
dataEx <- data[,-2]
dataEx$Work <- as.factor(dataEx$Work)
lm = glm(Work ~ ., data = dataEx, family = binomial)

# Draw from the posterior
post_betas = rmvnorm(1000, opt_res$par, invJ)
nSmallChild_betas <- post_betas[,7]
nSmallChild_density <- density(nSmallChild_betas)
ci <- ci(nSmallChild_betas, ci = 0.95, method="ETI")

df2 <- data.frame(
  x <- nSmallChild_density$x,
  y <- nSmallChild_density$y,
  ci_high <- ci$CI_high,
  ci_low <- ci$CI_low
)

ggplot(df2, aes(x, y)) +
  geom_line(color="firebrick") +
  geom_vline(xintercept = ci_high, color="steelblue") +
  geom_vline(xintercept = ci_low, color="steelblue") +
  ggtitle("nSmallChild beta coefficient density") +
  xlab("B7 value") +
  ylab("Density")
```

2b) Now we use our normal approximation from 2a to write a function that generates draws from the posterior predictive distribution of $Pr(y = 1|x)$. The x values shall be fore a 37-years old woman, with two children (3 and 6 years old), 8 years of education, 11 years of experience, and a husband with an income of 13.

The probability of work is calculated with the formula:

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

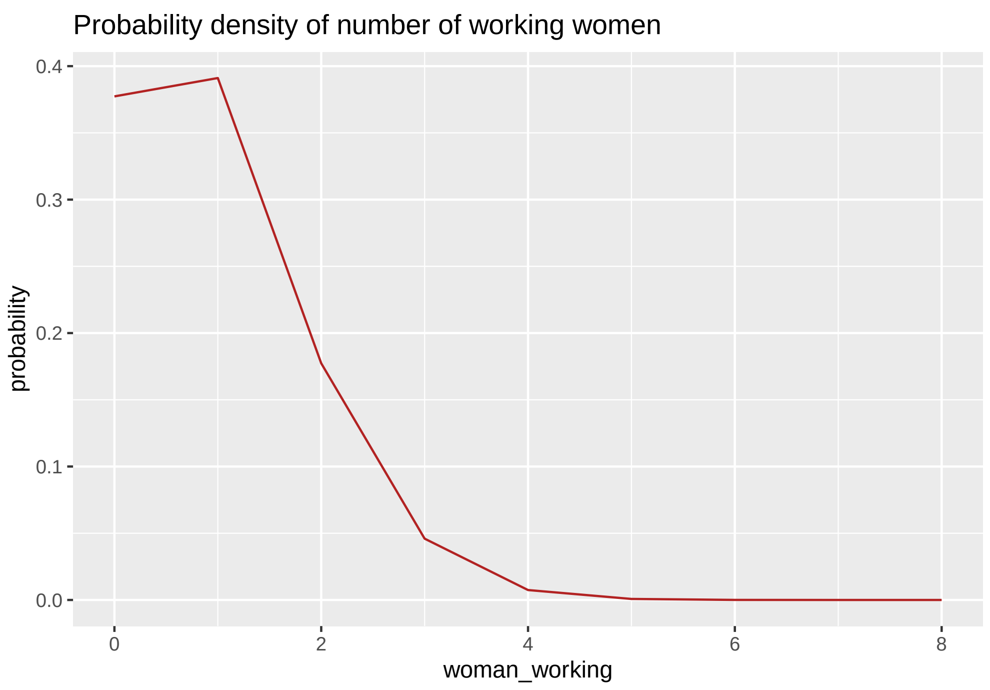


From the graph we see that the expected probability of working for a women with the described features is very low.

The code for 2b is:

```
# Code for 2b
woman <- as.matrix(c(1, 13, 8, 11, (11/10)^2, 37, 2, 0))
post_betas <- t(post_betas)
# Generate draws for the woman based on our betas from 2b
pr_woman_work <- exp(t(woman)%*%post_betas) / (1+exp(t(woman)%*%post_betas))

den <- density(pr_woman_work) # Density for the probabilites of work
df3 <- data.frame(
  x <- den$x,
  y <- den$y
)

ggplot(df3, aes(x, y)) +
  geom_line(color="firebrick") +
  ggtitle("Probability density for getting work")
```

2c) Now we consider 8 women with the same features as the women in 2b. We write a new function for how many out of these 8 women are working. The distribution of this function is seen in the following graph:



The result is consistent with the earlier results, we expect few of a sample of 8 to work.

The code for 2c:

```r
# Code for 2c
p <- mean(pr_woman_work) # The probability that 1 woman work.
x <- c(0,1,2,3,4,5,6,7,8)

post_dist <- dbinom(x, 8, p)
df4 <- data.frame(
  woman_working <- x,
  probability <- post_dist
)
ggplot(df4, aes(woman_working, probability)) +
  geom_line(color="firebrick") +
  ggtitle("Probability density of number of working women")
```