Bayesian Learning

Computer Lab 2

1. Linear and polynomial regression

A dataset containing daily temperatures (in celsius degree) at Malmslätt, Linköping, was used to perform a Bayesian analysis of the following quadratic regression:

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

where the response variable is time and the covariate is

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

a) The following priors to the linear regression was used:

- $\beta|\sigma^2 \sim N(\mu_o, \sigma^2\Omega_o^{-1})$
- $\sigma^2 \sim Inv-\chi^2(v_o, \sigma_o^2)$

μ_0 (the beta-values) was chosen as follows:

- $\beta_0 = -5$
- $\beta_1 = 20$
- $\beta_2 = 60$

β_0 was set to -5 as it seems reasonable that the temperature at time = 0 (January 1st) is around -5 degrees.
β_1 and β_2 was chosen to generate a linear regression model that estimate the temperature for 30th of June to 20 degrees (time = 0.5).

v_0 is representing the degrees of freedom in the prior, thus how sure we are about our prior knowledge. The initial value for v_0 was set to 60.

It seems possible that the temperature will differ with 7 degrees in 95 % of the cases. σ_o^2 was set to:

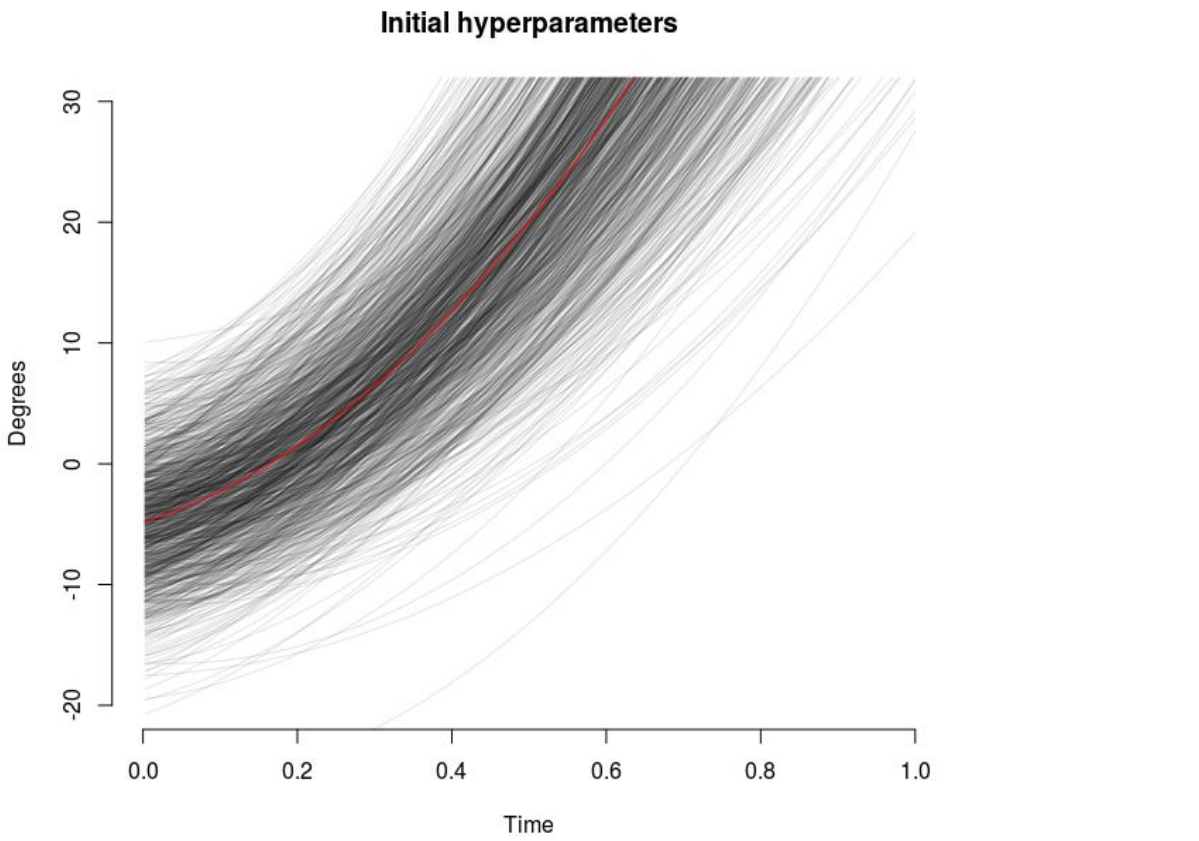
- $\sigma_o^2 = (\frac{7}{1.96})^2 = 12.5771$

Ω_0 is a vector describing how sure we are about our betas. As we are more certain about β_0 than β_1 and β_2, Ω_0 was set to:

- Ω_0 = (0.5, 0.1, 0.1)

b)
To find out if our chosen hyper parameters and priors are sensible, 1000 draws of beta was done from the joint prior of all parameters. A regression curve was then computed and plotted for each draw. A read mean-curve of all regression lines is visible in the middle of the plot.

```
# simulation
chi2 = rchisq(v0, n=nDraws)
sigma2_draws = v0*sigma2_0/chi2
omega0_inv = solve(omega0)
B_draws = matrix(0, nDraws, 3)

# start new plot
plot.new()
plot.window(xlim=c(0,1), ylim=c(-20,30))
axis(side=1)
axis(side=2)

# simulate draws
for(i in 1:nDraws){
  B_draws[i,] = mvrnorm(n = 1, mu = my0, Sigma=sigma2_draws[i]*omega0_inv)
  lines(data$time, B_draws[i,1]+B_draws[i,2]*data$time+B_draws[i,3]*data$time^2, col=rgb(0,0,0,0.1))
}
lines(data$time, mean(B_draws[,1])+mean(B_draws[,2])*data$time+mean(B_draws[,3])*data$time^2, col=rgb(1,0,0,1))
title(main="Initial hyperparameters", xlab="Time", ylab="Degrees")
```

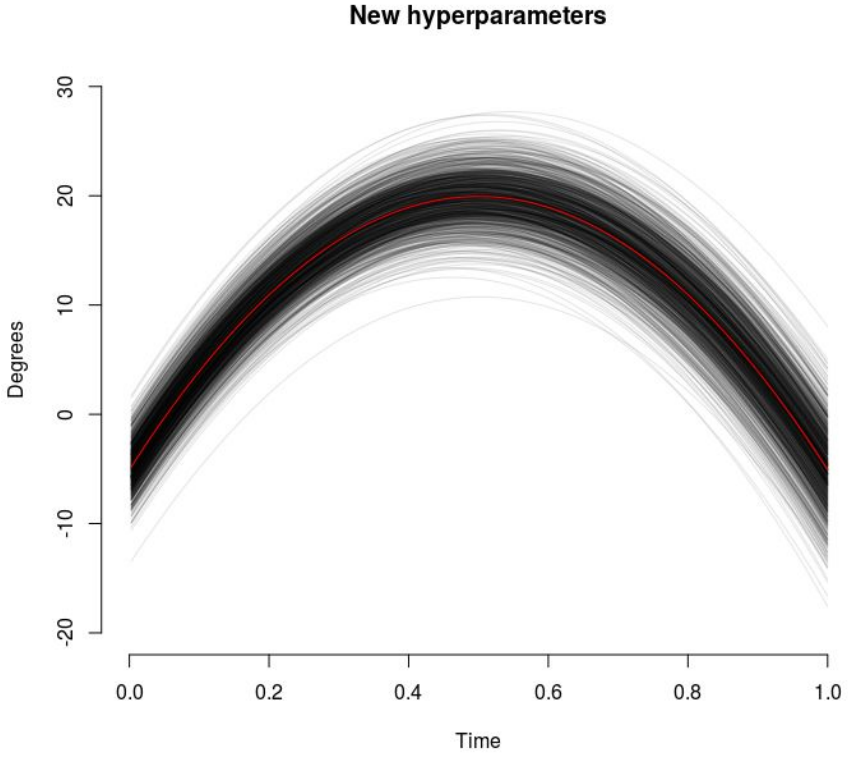


One reasonable pattern that should appear is that the regression lines should be at almost the same degree at time = 0 and time = 1, which is not the case in the curve above. It's also seems like the regression lines are exponentially increasing, a pattern that doesn't represent the temperature during a year. Thus the chosen hyper parameters are not sensible.

New hyper parameters was chosen:

- $\beta_0 = -5$
- $\beta_1 = 100$
- $\beta_2 = -100$
- $\sigma_o^2 = 2$
- 0= (0.5, 0.5, 0.5)

New draws of beta was done and regression lines was computed and plotted:



The plot above has the same temperature at time = 0 and time = 1. It also has the lowest temperature in the beginning/end of the year, it increases during the spring, peaks in the middle of the summer end dips towards the fall. The newly chosen hyper parameters seams reasonable.

c)
The same hyper parameters as in b) was used. To calculate μ_n, beta-hat was calculated.

```
B0 = -5
B1 = 100
B2 = -100
my0 = c(B0, B1, B2)
sigma2_0 = 2
v0 = 100
omega0 = diag(c(0.5, 0.5, 0.5))
X = cbind(1, data$time, I(data$time)^2)
Y = data$temp
B_hat = solve(t(X)%*%X)%*%t(X)%*%Y
```

The rest of the parameters needed to calculate the posterior was calculated:

```
myn = solve((t(X)%*%X+omega0)) %*% (t(X)%*%X%*%B_hat + omega0 %*% my0)
omegan = t(X)%*%X + omega0
omegan_inv = solve(omegan)
vn = v0 + dim(X)[1]
sigma2_n = ((v0*sigma2_0 + t(Y)%*%Y + t(my0)%*%omega0%*%my0 - t(myn)%*%omegan%*%myn)/vn)[1,1]
```

The posterior σ^2 was drawn as before:

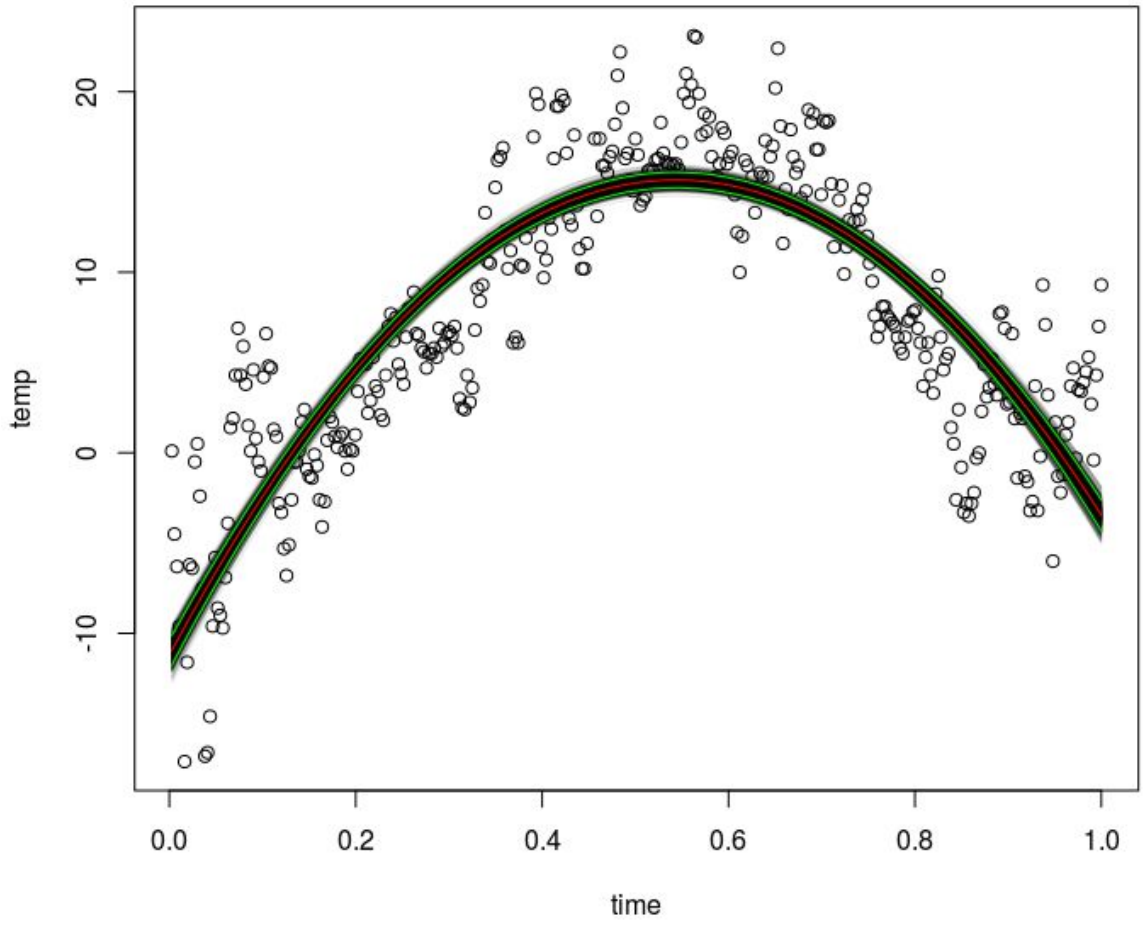
```
chi2_n = rchisq(vn, n=nDraws)
sigma2_draws = vn*sigma2_n/chi2_n
```

The data was then plotted and nDraws of beta was done. For each draw of beta, a regression line was calculated and plotted.

```
plot(data)
beta_draws = matrix(0, nDraws, 3)
posterior_y = matrix(0, nDraws, 366)
for (i in 1:nDraws) {
  beta_draws[i,] = mvrnorm(n = 1, mu = myn, Sigma = sigma2_draws[i]*omegan_inv) # Draw betas
  lines(data$time, beta_draws[i,1] + beta_draws[i,2]*data$time + beta_draws[i,3]*data$time^2, col=rgb(0,0,0,0.1)) # Plot regression line
  posterior_y[i,] = beta_draws[i,1] + beta_draws[i,2]*data$time + beta_draws[i,3]*data$time^2
}
lines(data$time, mean(beta_draws[,1])+mean(beta_draws[,2])*data$time+mean(beta_draws[,3])*data$time^2, col=rgb(1,0,0,1))
title(main="New hyperparameters", xlab="Time", ylab="Degrees")
```

Using the calculated posterior times, a 90% credible interval (lower 5% and upper 95%) was calculated and plotted ontop if the temperature data.

```
posterior_CI = apply(X = posterior_y, MARGIN=2, FUN=function(x) quantile(x,c(0.05, 0.95), na.rm=T))
lines(data$time, posterior_CI[2,], col="green")
lines(data$time, posterior_CI[1,], col="green")
```

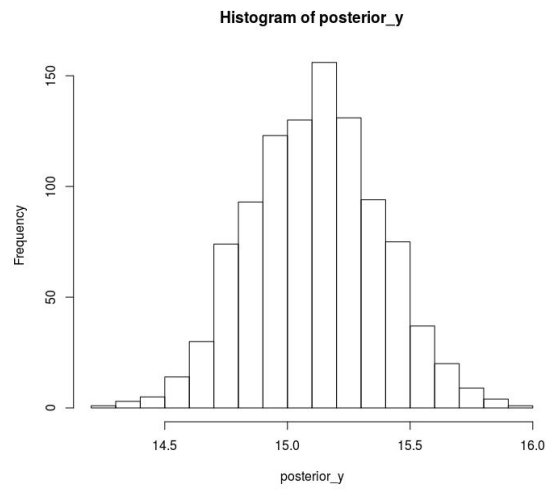


Looking at the interval above, it's seems like very few of the data points is contained by the interval. This is reasonable as the interval is calculated of the regression line and not from data directly.

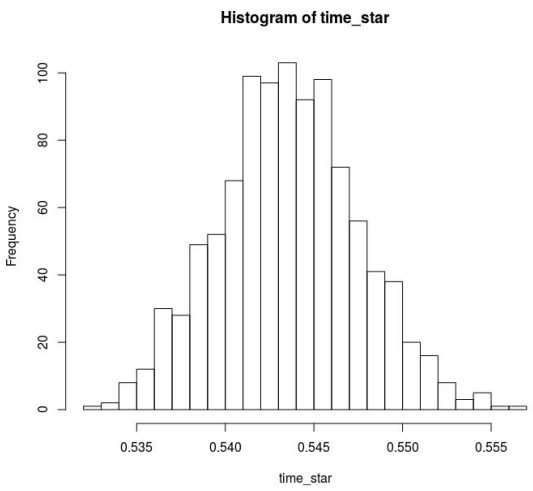
d)
To locate the time with the highest expected temperature, the first derivative of the quadratic regression function was calculated. By setting it equal to 0, an equation to calculate the time with the maximum temperature was obtained.

The beta simulations from c) was used to calculate times with the maximum temperatures.

```
# f'(time) = B_1 + 2*B_2*time = 0
# time_* = (-B_1)/(2*B_2)
plot(data)
time_star = numeric()
posterior_y = numeric()
for (i in 1:nDraws) {
  beta_draw = mvrnorm(n = 1, mu = myn, Sigma = sigma2_draws[i]*omegan_inv)
  time_star[i] = (-beta_draw[2])/(2*beta_draw[3])
  posterior_y[i] = beta_draw[1] + beta_draw[2]*time_star + beta_draw[3]*time_star^2
}
plot(time_star, posterior_y)
hist(posterior_y, breaks=20) # around 15 degrees
```



Histogram of the frequency of the maximum temperatures.



Histogram of the frequency of times with maximum temperature

From a visual analysis of the histograms above, it seems like the maximum temperature is happening around time = 0.544.

e)
To mitigate the problem of overfitting when using high order polynomial models, it's reasonable to replace the prior with a ridge regression, where $\mu_o = 0$ and Ω_0 is a diagonal matrix with lambda values. To limit the higher orders, larger values for lambdas corresponding to the higher orders should be chosen:

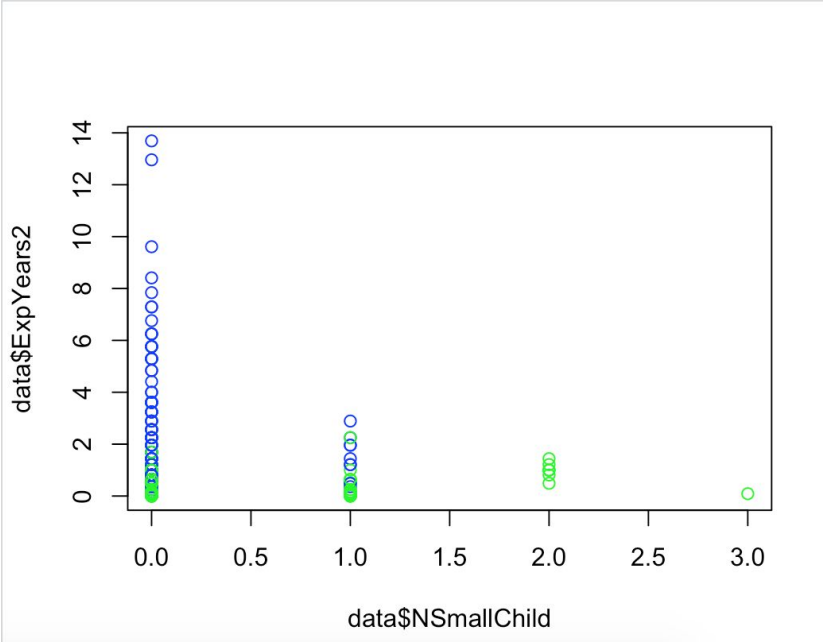
1						
	1					
		2				
			5			
				5		
					10	
						10

2. Posterior approximation for classification with logistic regression

a) Fitting a regular logistic regression using maximum likelihood estimation with glm:

```
# libraries
library("mvtnorm")

# data
data = read.table("WomenWork.dat", header=TRUE)

# a)
# fit model
glm.model <- glm(Work ~ 0 + ., data = data, family = binomial)

# make predictions
pred = glm.model$fitted.values > 0.5

# depict predictions based on # of small children and experience years^2
plot(data$NSmallChild, data$ExpYears2, col=rgb(0,1-pred,pred))
```

Gives the resulting plot:



We can see that many small children usually results in a prediction of the woman not working whereas having long work experience results in a prediction of the woman working.

b) Approximating the posterior distribution of the parameters in the logistic regression. Then, estimating the parameters using the mode of the posterior.
Setting up data and prior of parameters:

```
# setup data
y = data[,1]
X = data[,-1]
names = names(X)
p = dim(X)[2]
X = as.matrix(X)

# setup prior
my = rep(0, p)
tau = 10
Sigma = tau^2*diag(p)
```

Optimizing the posterior via the log

```
# use optim to find max
beta.init = rep(0, p)
optim.res = optim(beta.init, LogPostLogistic
beta.mode = optim.res$par
beta.invhessian = -solve(optim.res$hessian)
```

Where LogPostLogistic is

```
# log posterior of a logistic regression
logPostLogistic = function(beta, y, X, my, Sigma) {
  p = length(beta)
  ypred = X %*% beta # make prediction

  # Logarithm of the likelihood seen on page 5 in lecture 6.
  log.likelihood = sum(ypred * y - log(1 + exp(ypred)))

  # If likelihood is very large or very small, the abs(log.likelihood)
  # will be close to infinity. In those cases we set
  # log.likelihood to -20000
  if (abs(log.likelihood) == Inf) log.likelihood = -20000;

  # Logarithm of the prior
  log.prior <- dmvnorm(beta, matrix(0, p, 1), Sigma, log=TRUE);

  return(log.likelihood + log.prior)
}
```

Results in Beta mode and :

```
beta.mode = optim.res$par #0.62672884 -0.01979113  0.18021897  0.16756670 -0.14459669 -0.08206561 -1.35913317 -0.02468351
```

```
beta.invhessian = -solve(optim.res$hessian)
#  2.266022568  3.338861e-03 -6.545121e-02 -1.179140e-02  0.0457807243 -3.029345e-02 -0.1887483542 -0.0980239285
#  0.003338861  2.528045e-04 -5.610225e-04 -3.125413e-05  0.0001414915 -3.588562e-05  0.0005066847 -0.0001444223
# -0.065451206 -5.610225e-04  6.218199e-03 -3.558209e-04  0.0018962893 -3.240448e-06 -0.0061345645  0.0017527317
# -0.011791404 -3.125413e-05 -3.558209e-04  4.351716e-03 -0.0142490853 -1.340888e-04 -0.0014689508  0.0005437105
#  0.045780724  1.414915e-04  1.896289e-03 -1.424909e-02  0.0555786706 -3.299398e-04  0.0032082535  0.0005120144
# -0.030293450 -3.588562e-05 -3.240448e-06 -1.340888e-04 -0.0003299398  7.184611e-04  0.0051841611  0.0010952903
# -0.188748354  5.066847e-04 -6.134564e-03 -1.468951e-03  0.0032082535  5.184161e-03  0.1512621814  0.0067688739
# -0.098023929 -1.444223e-04  1.752732e-03  5.437105e-04  0.0005120144  1.095290e-03  0.0067688739  0.0199722657
```

Simulating draws from the posterior to calculate a CI for NSmallChild:

```
# simulate draws
nDraws = 10000
beta.draws = rmvnorm(n = nDraws, mean = beta.mode, sigma = beta.invhessian) # draws from posterior

# CI
beta.NSmallChild.CI = apply(X = beta.draws, MARGIN=2, FUN=function(x) quantile(x,c(0.025, 0.975), na.rm=T))[,7]
```

Results in a 9% CI.NSmallChild= [-2.1315425 -0.6067602] which does not contain 0 and we would then say that this feature is an important determinant of whether a woman works.

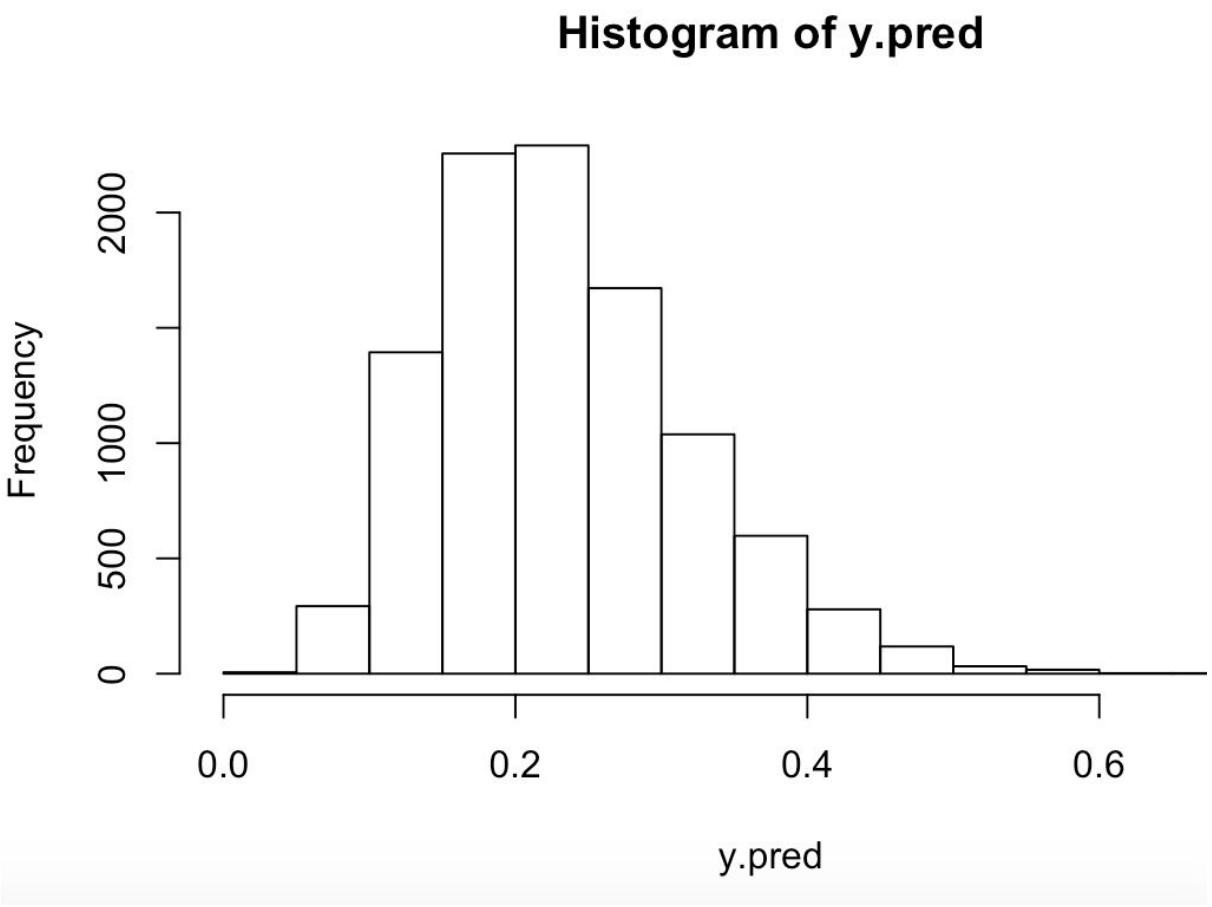
c)
Function simulates from the predictive distribution:

```
logReg = function(woman, mean, Sigma, nDraws) {
  beta.draws = rmvnorm(n = nDraws, mean = mean, sigma = Sigma)
  pred = beta.draws %*% woman
  y.pred = sigmoid(pred)
  return(y.pred)
}
```

Making predictions of a woman:

```
woman = c(1, 10, 8, 10, (10/10)^2, 40, 1, 1)
y.pred = logReg(woman, beta.mode, beta.invhessian, 10000)
hist(y.pred)
print(mean(ifelse(y.pred>0.5, 1, 0))) #0.0055 ~0.5%
```

Results in histogram:



Where 0.55% of all predictions resulted in prediction of her working, where these are uncertain with a probability slightly over 0.5 as can be seen in the histogram.