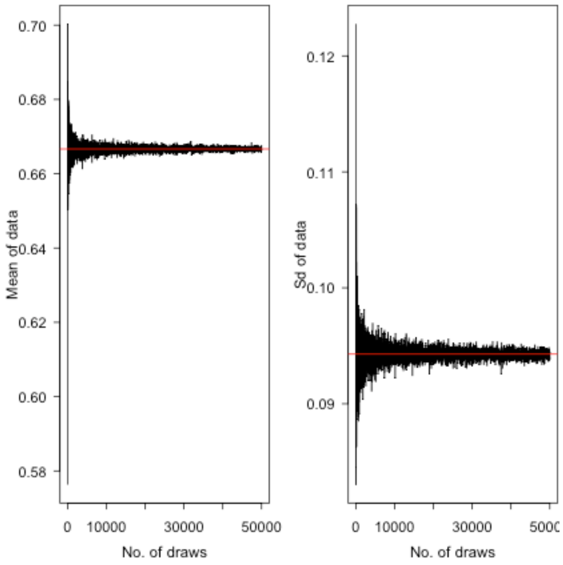
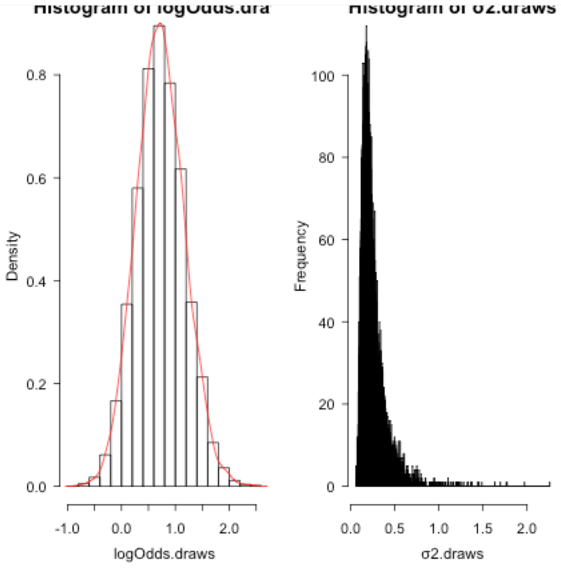
```
######################### Lab1 #########################
### Meta-info
## Beta distribution
## Simulations
## Gini coefficient
## Credible interval
## Highest Posterior Density HPD

# Task 1: Bernoulli ... again
# a) Draw random numbers from Beta distribution and graphical verification
# of posterior
# b) Simulate to compute posterior prob of Pr(theta < 0.4)
# c) Compute log-oods posterior distribution

# Task 2: Log-normal distribution and the Gini coefficient
# a) Simulate 1000 draws from posterior or theta2. Compare with real value
# b) Compute posterior distribution of Gini coefficient G
# c) Compute 95% equal tail credible interval of Gini coefficient G.
# Doing a kernal density estimate
# Compute 95% Highest Posterior Density interval (HPD) of G

# Task 3: Bayesian inference for the concentration parameter in the von Mises distributio
# a) Plot posterior distribution of kappa for wind direction data
# b) Find approximate posterior mode of kappa

############# Task 1 #############
# Bernoulli ... again
#a)
# Instrucitons
# Likelihood: y_1, ..., y_n | θ ~ Bern(θ)
# Prior: θ ~ Beta(alpha_0, beta_0), alpha_0 = beta_0 = 2
# Posterior: θ|y_1, ..., y_n ~ Beta(alpha_0 + s, beta_0 + f)
# s = 14
# n = 20
# f = 6

# Setup
n = 20
s = 14
f = n - s
nDraws = 50000
drawsInterval = 10
intervalVec <- seq(10, nDraws, drawsInterval)

# Prior
prior.alpha = 2
prior.beta = 2

# Posterior
posterior.alpha <- prior.alpha + s
posterior.beta <- prior.beta + f
posterior.draws_means = numeric()
posterior.draws_sd = numeric()

# For-loop - (10, 20, 30, ..., 49980, 49990)
for (i in intervalVec) {
  posterior.draws <- rbeta(n = i,
                           shape1 = posterior.alpha,
                           shape2 = posterior.beta) # Draw from beta

  posterior.draws_means <- c(posterior.draws_means, mean(posterior.draws)) # Add mean to vector of means
  posterior.draws_sd <- c(posterior.draws_sd, sd(posterior.draws)) # Add sd to vector of sd
}

# True values
posterior.true_mean <- posterior.alpha/(posterior.alpha + posterior.beta)
posterior.true_sd <- sqrt((posterior.alpha*posterior.beta)/((posterior.alpha + posterior.beta)^2 * (posterior.alpha + posterior.beta + 

# Plot
par(mfrow = c(1, 2))
plot(x = intervalVec,
     y = posterior.draws_means,
     type = 'l',
     xlab = 'No. of draws',
     ylab = 'Mean of data') # Plot means
abline(h = posterior.true_mean, col = 'red') # Add line of real mean to plot

plot(x = intervalVec,
     y = posterior.draws_sd,
     type = 'l',
     xlab = 'No. of draws',
     ylab = 'Sd of data') # Plot sd's
abline(h = posterior.true_sd, col = 'red')
```

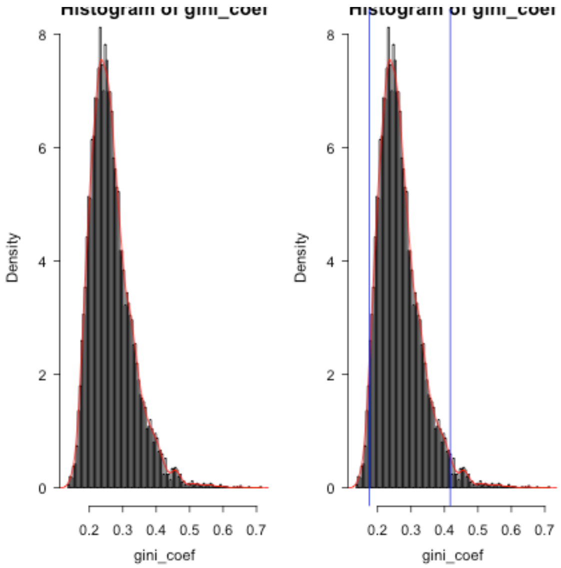



```
# b)
# Setup
n = 20
s = 14
f = n - s
nDraws = 10000

# Prior
prior.alpha = 2
prior.beta = 2

# Posterior
posterior.alpha <- prior.alpha + s
posterior.beta <- prior.beta + f
posterior.draws_means = numeric()
posterior.draws_sd = numeric()

# Draws
posterior.draws_10000 <- rbeta(n = nDraws,
                               shape1 = posterior.alpha,
                               shape2 = posterior.beta)

# Calculate probability
posterior.prob_0.4 <- length(which(posterior.draws_10000 < 0.4))/length(posterior.draws_10000)
posterior.prob_real<- pbeta(q = 0.4,
                            shape1 = posterior.alpha,
                            shape2 = posterior.beta)

# c)
# Functions
logOdds <- function(theta) {
  return (log(theta/(1-theta)))
}

# Setup
n = 20
s = 14
f = n - s
nDraws = 10000

# Prior
prior.alpha = 2
prior.beta = 2

# Posterior
posterior.alpha <- prior.alpha + s
posterior.beta <- prior.beta + f

# Draws
posterior.draws_10000 <- rbeta(n = nDraws,
                               shape1 = posterior.alpha,
                               shape2 = posterior.beta)

# Log-odds the draws
logOdds.draws <-logOdds(posterior.draws_10000)

# Hist and plot the density function of the log-odds draws
hist(logOdds.draws, probability = TRUE)
lines(density(logOdds.draws), col = 'red')

############# Task 2 #############
# Log-normal distribution and the Gini coefficient
# Likelihood: y_1, ..., y_n | μ, σ2 ~ log[ N(μ, σ2) ], μ known, σ2 unknown
# Prior: p(σ2) ∝ 1/σ2

# Posterior of σ2: Inv-X(n, tao^2)
# Tao^2 - The sample variance. Calculated as following:
# sum[ (log(y_i) - μ)^2 ]/n

# If X ~ N(0, 1)
# Y = exp(X) ~ log[ N(0,1) ] (lognormal)

# Setup
y <- c(14, 25, 45, 25, 30, 33, 19, 50, 34, 67)
```

```
nDraws = 10000
mu = 3.5

# a)
# Functions
scaled_rchisq <- function(Y, mu, nDraws) {
  n <- length(Y) # Length of data
  Z <- rchisq(n = nDraws, df = n) # Draw from Chi-squared distribution
  tao2 <- sum((log(Y) - mu)^2)/n # Calculate tau2 (sample standard diviation)

  return (n*tao2/Z)
}

σ2.draws <- scaled_rchisq(y, mu, nDraws) # Draw from Scaled inverse chi-squared distribution
hist(σ2.draws, breaks=1000)
```



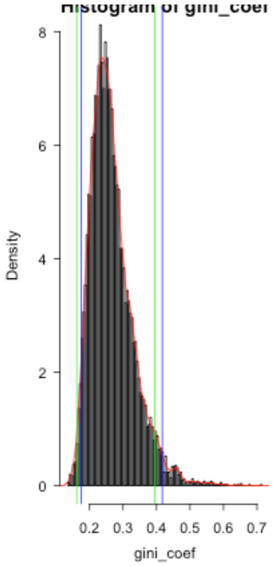

```
# b)
# Gini-coefficient measuers inequality (0: equal, 1: unequal)
# Uses the Lorenz curve where:
# x-axis: Cumulative share of people from low to high income
# y-axis: Cumulative share of income earned
#
# If a straight line, it's 100% equal
#
# The Gini-coefficient is the ratio between the area between the straight line and the Lorenz curve
# divided by the total area
#
# If the data follows an lognormal distribution (e.g wealth), the Gini-coef is calculated as follows:
# G = 2 * Φ(σ/√2) - 1

σ <- sqrt(σ2.draws) # Square σ2 to get σ
gini_coef <- 2 * pnorm(σ/sqrt(2), mean = 0, sd = 1) - 1 # Calculate the Gini-coefficients for each σ
hist(gini_coef, breaks=200, probability = TRUE) # Hist Gini-coefficients
lines(density(gini_coef), col='red') # Plot density curve

# c)
# Functions
eqtail_CI <- function(..X, interval) {
  lower <- (1-interval)/2
  upper <- 1 - lower
  n <- length(..X)
  X <- sort(..X) # Sort from smallest to largest value

  return (list(lower=X[n*lower], upper=X[n*upper]))
}

HPD <- function(density, interval) {
  gini_df <- data.frame(x = density$x, y = density$y)
  gini_df <- gini_df[with(gini_df, order(y)),]
  gini_df <- data.frame(x = gini_df$x, y = gini_df$y)
  n <- dim(gini_df)[1]
  lower <- 1 - interval
  print(lower)
  HPD_cumsum <- cumsum(gini_df$y)/sum(gini_df$y)
  HPD_lower <- which(HPD_cumsum >= lower)[1]
  gini_df_95 <- gini_df[(HPD_lower + 1):n, ]
  HPD_interval <- c(min(gini_df_95$x), max(gini_df_95$x))
  return (list(lower = HPD_interval[1], upper = HPD_interval[2]))
}

# 95% equal tail credible interval
gini_coef_CI <- eqtail_CI(gini_coef, 0.95)

hist(gini_coef, breaks=200, probability = TRUE) # Hist Gini-coefficients
lines(density(gini_coef), col='red') # Plot density curve
abline(v = gini_coef_CI$lower, col='blue') # Plot lower CI line
abline(v = gini_coef_CI$upper, col='blue') # Plot upper CI line
```

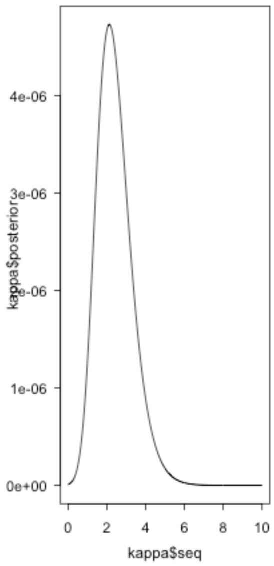



```
# Highest Posterior Density Interval
gini_density <- density(gini_coef)
HPD_interval <- HPD(gini_density, 0.95)
```

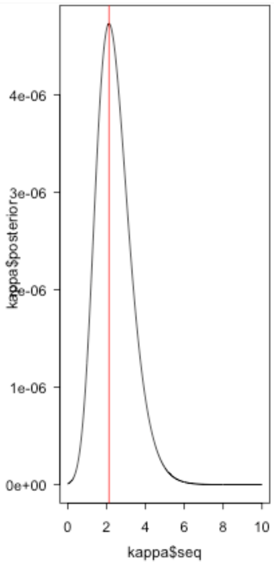
```
## [1] 0.05
```

```
# Plot histogram of Gini coefficients, 95% credible interval (blue) and 95% HPD interval (green)
hist(gini_coef, breaks=200, probability = TRUE) # Hist Gini-coefficients
lines(density(gini_coef), col='red') # Plot density curve
abline(v = gini_coef_CI$lower, col='blue') # Plot lower CI line
abline(v = gini_coef_CI$upper, col='blue') # Plot upper CI line
abline(v = HPD_interval$lower, col='green') # Plot lower HPD interval
abline(v = HPD_interval$upper, col='green') # Plot upper HPD interval

############# Task 3 #############
# von Mises distribution looks like a normal distribution with a spiky top and
# is a continues probability distribution on the circle, where theta is an angle.
# Kappa (ϰ): Concentration parameter. Large ϰ gives small variance around μ.
# Likelihood: p(y_1, ..., y_n | μ, ϰ) = exp[ ϰ * cos(y - μ) ]/(2πIo(ϰ))
# Prior: ϰ ~ Exp(λ = 1), mean = 1/λ

# Setup
# Wind-angles in degrees on 10 different days
# North is zero
#
y.degrees <- c(40, 303, 326, 285, 296, 314, 20, 308, 299, 296)
y.radians <- c(-2.44, 2.14, 2.54, 1.83, 2.02, 2.33, -2.79, 2.23, 2.07, 2.02)
mu <- 2.39 # Mean directon

# a)
# Prior
kappa <- data.frame(seq = seq(0, 10, 0.01),
                    posterior = 0
)

for (i in 1:dim(kappa)[1]) { # Loop over every kappa
  k <- kappa$seq[i] # Extract current kappa
  prior <- exp(-k) # Calculate prior with current kappa
  bessel <- besselI(x = k,
                    nu = 0) # Bessel-function
  likelihood <- prod(exp(k * cos(y.radians - mu))/(2*pi*bessel)) # Calculate von Mises probability
  kappa$posterior[i] <- likelihood * prior # Calculate posterior with current kappa
}

# Plot posterior for different kappas
plot(kappa$seq, kappa$posterior, type='l')
```

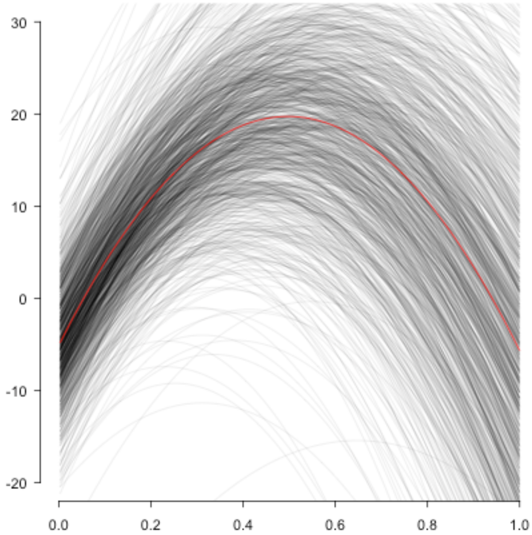





```
# b)
index <- which.max(kappa$posterior) # Finds index with maximum posterior
kappa.mode <- kappa$seq[which.max(kappa$posterior)] # Extract kappa with maximum posterior

plot(kappa$seq, kappa$posterior, type='l')
abline(v = kappa.mode, col='red')
```



```
########################## Lab 2 ##########################
## Meta-info
# Linear regression
# Polynomial regression
# Logistic regressions
# Credible interval
# Maximum likelihood
# Optim
# Hessian
# Mode of beta
# Predictive distributions

# Task 1: Linear and polynomial regression
# a) Set the prior hyperparameters μ0, Ω0, ν0 and σ2 to sensible values
# b) Check if your prior from a) is sensible
# Simulate draws from joint prior and compute regression curve of each draw
# c) Simulates from the joint posterior distribution of β0, β1,β2 and σ2
# Plot:
# • Posterior mean of simulations
# • Curve of lower and upper credible interval of f(time)
# d) Simulate highest expected temperatures
# Simulate from posterior distribution of time with highest expected temperatures
# What to do to mitigate risk of overfitting high order polynomial regression?

# Task 2: Posterior approximation for classification with logistic regression
# a) Fit logistic regression using maximum likelihood estimations
# b) Approximate the posterior distribution of the 8-dim parameter vector β with a multivariate normal distribution
# c) Simulates from the predictive distribution of the response variable in a logistic regression

####################################################################################################

# Functions
scal_inv_schsq <- function(v, σ_2, nDraws) {
  X <- rchisq(n = nDraws, df = v)
  return (v*σ_2/X)
}

############# Task 1 #############
# Respons variable: temp = β0 + β1*time + β2 * time^2 + ε, ε ~ N(0, σ2)
# Covariate: time = No. of days since beginning of year/366

# a)
# Conjugate priors
# β | μ ~ N(μ0, σ2*Ω0_(-1))
# σ2 ~ Inv-X(ν0, σ0_2)
#
# Set prior hyperparameters:
# μ0: The expected value of the betas [vector]
# Ω0: How sure we are about the betas (scales the variance) [matrix]
# ν0: How sure we are about our prior knowledge of the sigmas [scalar]
# σ0_2: The variance of the betas [vector]

dataset <- read.table("TempLinkoping.txt", header = TRUE)

prior.mu0 <- c(-5, 100, -100)
prior.v0 <- 10
prior.σ0_2 <- (7/1.96)^2 # 10 = 1.95 * σ -> 10 degrees are in the confidence interval 95% of the times
prior.Ω0 <- matrix(c(0.5, 0, 0, 0, 0.1, 0, 0, 0, 0.1),
                   nrow = 3,
                   ncol = 3)
prior.inv_Ω0 <- solve(prior.Ω0)

# b)
nDraws = 1000
beta_draws <- matrix(nrow = nDraws, ncol = 3)

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

# Sample betas & plot regression curves
for (i in 1:nDraws) {
  sigma2 <- scal_inv_schsq(prior.v0, prior.σ0_2, 1) # Sample sigma2 from scaled inverse chi squared
  beta_draws[i,] <- rmvnorm(n = 1, mean = prior.mu0, sigma = sigma2*prior.inv_Ω0) # Sample betas from Multinormal

  # Generates regression point for each dataset$time. All of them are then plotted as a line
  lines(dataset$time,
        beta_draws[i,1] + beta_draws[i,2]*dataset$time + beta_draws[i,3]*dataset$time^2,
        col=rgb(0, 0, 0, 0.1))
}

lines(dataset$time,
      mean(beta_draws[,1]) + mean(beta_draws[,2])*dataset$time + mean(beta_draws[,3])*dataset$time^2,
      col=rgb(1, 0, 0, 1))
```




```
# Priors not sensible. New priors
prior.mu0 <- c(-5, 100, -100)
prior.v0 <- 10
prior.σ0_2 <- (7/1.96)^2 # 7 = 1.95 * σ -> 7 degrees are in the confidence interval 95% of the times
prior.Ω0 <- matrix(c(0.5, 0, 0, 0, 0.1, 0, 0, 0, 0.1),
                   nrow = 3,
                   ncol = 3)
prior.inv_Ω0 <- solve(prior.Ω0)
n <- dim(dataset)[1]

# c)

# Setup
X <- cbind(1, dataset$time, dataset$time^2) # Create X with constant row for beta_0
Y <- dataset$temp
n <- dim(dataset)[1]
CI <- 0.90
CI_lower <- (1-CI)/2 # 0.05
CI_upper <- 1-(1-CI)/2 # 0.95

# Posterior
beta_hat <- solve(t(X)%*%X)%*%t(X)%*%Y # Beta_hat by classic by OLS (Ordinary Least Square)
posterior.mun <- solve(t(X) %*% X + prior.Ω0) %*% (t(X) %*% X %*% beta_hat + prior.Ω0 %*% prior.mu0) # Mu_n
posterior.Ωn <- t(X)%*%X + prior.Ω0 # Omega_n
posterior.inv_Ωn <- solve(posterior.Ωn) # Inverse Omega_n
posterior.vn <- prior.v0 + n # v_n
posterior.sigman_2 <- (t(Y) %*% Y + t(prior.mu0) %*% prior.Ω0 %*% prior.mu0 - t(posterior.mun) %*% posterior.Ωn %*% posterior.mun)/post

nDraws <- 10000
beta_post_draws <- matrix(nrow = nDraws,
                          ncol = 3)
posterior_y <- matrix(nrow = nDraws,
                      ncol = n)

# Draws of sigma2 and beta posteriors
for (i in 1:nDraws) {
  sigma2 <- as.vector(scal_inv_schsq(posterior.vn, posterior.sigman_2, 1))
  beta_post_draws[i,] <- rmvnorm(n = 1,
                             mean = posterior.mun,
                             sigma = sigma2*posterior.inv_Ωn)
}

# Generates a large matrix (10000 x 366), For each time/column (366) -> 10000 predicted temps/rows, one for each beta-draw
temp_pred_each_time <-  t(X %*% t(beta_post_draws))
temp_pred_each_time_sorted = apply(X = temp_pred_each_time,
                                   MARGIN = 2,
                                   sort) # Sort each time/row

# Extract lower and upper
temp_pred_CI90 <- rbind(temp_pred_each_time_sorted[round(nDraws*CI_lower),],
                        temp_pred_each_time_sorted[round(nDraws*CI_upper),])

# Plot data and mean regression line
plot(dataset)
lines(dataset$time, mean(beta_post_draws[,1]) + mean(beta_post_draws[,2]) * dataset$time + mean(beta_post_draws[,3]) * dataset$time^2,
        col=rgb(1, 0, 0, 1))
lines(dataset$time, temp_pred_CI90[1,], col=rgb(0, 1, 0, 1))
lines(dataset$time, temp_pred_CI90[2,], col=rgb(0, 1, 0, 1))

# d)
# Time with highest expected temperature: time = -B1/2B2
# Calculated from the derivation of f(time) set to 0.

posterior_max_time <- -beta_post_draws[,2]/(2*beta_post_draws[,3])
abline(v = mean(posterior_max_time), col=rgb(0, 0, 1, 1))

# e)
# To mitigate the risk of overfitting due to higher order polynomials we want to have a small
# mu_n for larger betas. This will lead to a smaller coefficient for the higher polynomials,
# thus making them affect the end result less.
# 1) Set my_0 corresponding to higher polynomials low
# 2) Set the diagonal indices of Ometa_0 corresponding higher polynomial high

############# Task 2 #############
```

```r
# Posterior approximation for classification with logistic regression
# Dataset:
# Response variable: Work
# Covariates: Constant  HustbandInc EducYear  ExpYear   ExpYear2  Age   NSmallChild   NBigChild

dataset <- read.table("WomenWork.dat", header=TRUE)

# a)

logRegFit <- glm(formula = Work ~.-Constant, data = dataset, family = "binomial")

# b)
# Approx posterior distribution of 8-dim parameter vector β
# Posterior: β | y, X ~ N(Beta_mode, Inv_Hessian_At_Beta_bode)
# Likelihood: y | β, X = exp(x_i*β)^(y_i)/[ 1+exp(x_i*β) ]
# Prior: β ~ N(0, τ_2*I), τ = 10

# Functions
# !!!!! Important to remember !!!!!
## 1) Always use log posterior as it's more stable and avoids problems with to small or large numbers
## 2) Don't forget that in log -> posterior = log.likelihood + log.prior
## 3) Don't forget to handle Infinity
postLogReg <- function(β, mu_0, X, Y, tau) {
  no_of_betas <- length(β) # Number of covariates
  sigma <- tau^2 * diag(no_of_betas) # Calculate sigma tau^2 * I

  # Likelihood
  # Logarithm of prod[ exp(x*β)^Y / (1 + exp(x*β))]
  log.likelihood <- sum(t(Y) %*% X %*% β) - log(prod(1 + exp(X %*% β)))

  if (abs(log.likelihood) == Inf) log.likelihood = -20000;

  # Prior
  log.prior <- dmvnorm(β, mean = mu_0, sigma = sigma, log = TRUE)

  return (log.likelihood + log.prior)
}

# Setup
n_parameters <- dim(dataset[,-1])[2] # No. of covariates
X <- as.matrix(dataset[, -1])
Y <- as.matrix(dataset[, 1])
nDraws = 10000
CI_interval = c(0.025, 0.975)

# Prior
β0 <- as.matrix(rep(0, n_parameters)) # Initial beta-values
beta.prior.mu_0 <- rep(0, n_parameters) # Mu_0
beta.prior.tau <- 10 # Tau: Given in the task

# Optim
# par: Initial values for parameter to me optimized
# fn: Function to be minimized/maximized
# Variables: Pass all variables except the one being maximized
optim.res <- optim(par = β0,
                   fn = postLogReg,
                   gr = NULL,
                   mu_0 = beta.prior.mu_0,
                   X = X,
                   Y = Y,
                   tau = beta.prior.tau,
                   method = "BFGS",
                   control = list(fnscale=-1),
                   hessian = TRUE
                   )

β.mode <- optim.res$par # Mode of beta
β.hessian.neg.inv <- -solve(optim.res$hessian) # Negative inverse hessian of beta

# Beta draws
β.draws <- matrix(nrow = nDraws,
                  ncol = n_parameters)

for (i in 1:nDraws) {
  β.draws[i, ] <- mvrnorm(n = 1, mu = β.mode, Sigma = β.hessian.inv)
}
```

```
## Error in mvrnorm(n = 1, mu = β.mode, Sigma = β.hessian.inv): could not find function "mvrnorm"
```

```r
# Calculate Credible Interval of NSmallChild
NSmallChild.draws <- sort(β.draws[, 7]) # Sort all draws in ascending order
NSmallChild.CI <- c(NSmallChild.draws[round(nDraws*CI_interval[1])],
                    NSmallChild.draws[round(nDraws*CI_interval[2])]) # Extract Credible intervals

# Hist of draws of NSmallChild
breaks <- 200
h <- hist(β.draws[,7], breaks = breaks, plot = FALSE)
```

```
## Error in hist.default(β.draws[, 7], breaks = breaks, plot = FALSE): 'x' must be numeric
```

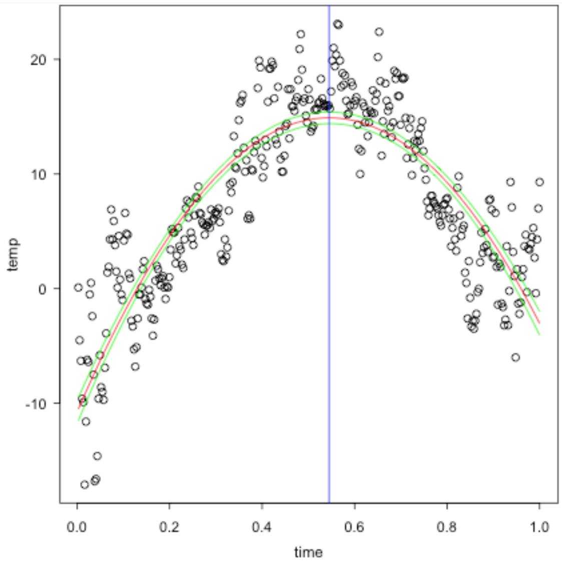
```r
cut <- cut(h$breaks, c(NSmallChild.CI[1], NSmallChild.CI[2]))
```

```
## Error in cut(h$breaks, c(NSmallChild.CI[1], NSmallChild.CI[2])): object 'h' not found
```

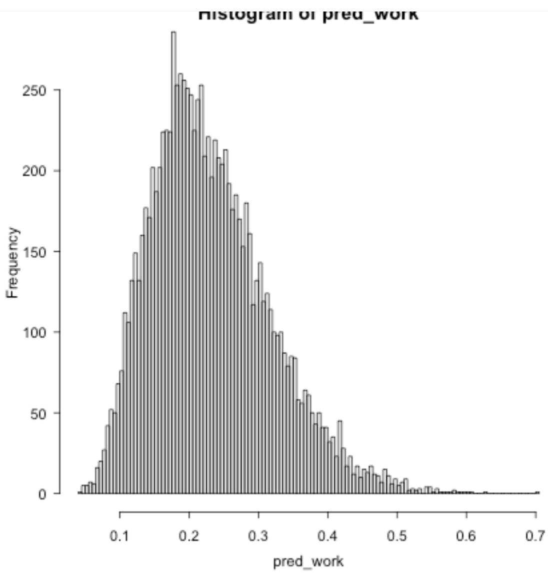
```r
plot(h,
     col = cut,
     main = "Draws of NSmallChild",
     xlab = "Value")
```

```
## Error in plot(h, col = cut, main = "Draws of NSmallChild", xlab = "Value"): object 'h' not found
```

```
abline(v = NSmallChild.CI[1], col = rgb(1, 0, 0, 1))
abline(v = NSmallChild.CI[2], col = rgb(1, 0, 0, 1))
```




```
# c)

# Functions
sigmoid <- function(x) {
  return (exp(x) / (1 + exp(x)))
}

drawPredDist <- function(βmode, negInvHess, y, nDraws) {
  beta_draws <- rmvnorm(n = nDraws,
                        mean = βmode,
                        sigma = negInvHess) # Draw from posterior beta distribution

  return (sigmoid(beta_draws %*% y))
}

# Setup
target <- c(1, 10, 8, 10, (10/10)^2, 40, 1, 1) # Target woman covariates
pred <- numeric() # Vector to collect results
nDraws <- 10000 # No. of draws

# Distribution of logistic regression of target
pred_work <- drawPredDist(βmode = β.mode,
                          negInvHess = β.hessian.neg.inv,
                          y = target,
                          nDraws = nDraws)

# Histogram of logistic regression
hist(pred_work, breaks=100)
```




```
# ~99.5% of the values are below 0.5.
# The data is indicating that the target woman doesn't work.
percent_below_05 <- sum(ifelse(pred_work < 0.5, 1, 0))/length(pred_work)
```