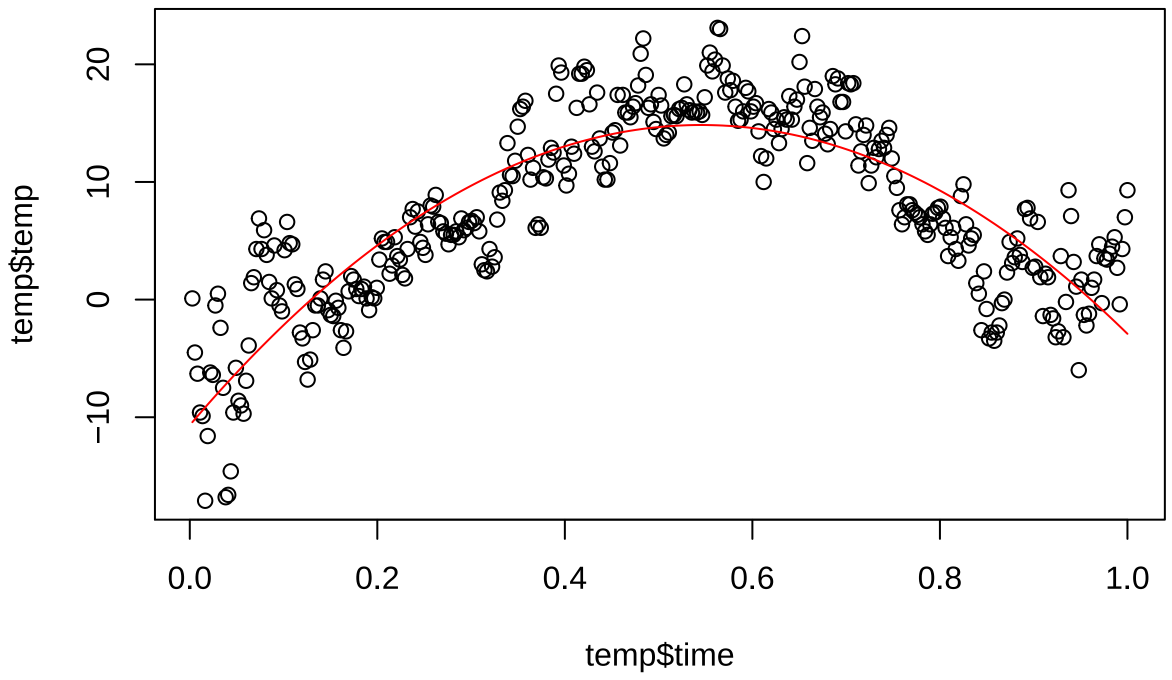
Bayesian Learning Lab2

Farhana Chowdhury Tondra(farch587),Syeda Farha Shazmeen(syesh076)

26 April 2018

1.Linear and polynomial regression

The dataset TempLinkoping is considered which contains daily temperatures (in Celcius degrees)at Malmslätt, Linköping . The response variable is temp and the covariate is time. The task is to perform a Bayesian analysis of a quadratic regression. Below shows the plot for temp v/s time and fitted quadratic regression.



1.(a) Determining the prior distribution of the model parameters.

In this task we need to set the prior hyper parameters $\mu_o, \Omega_0, v_0, \sigma_0^2$ Your task is to set the prior hyperparameters.

Following is the selected hyper parameters.

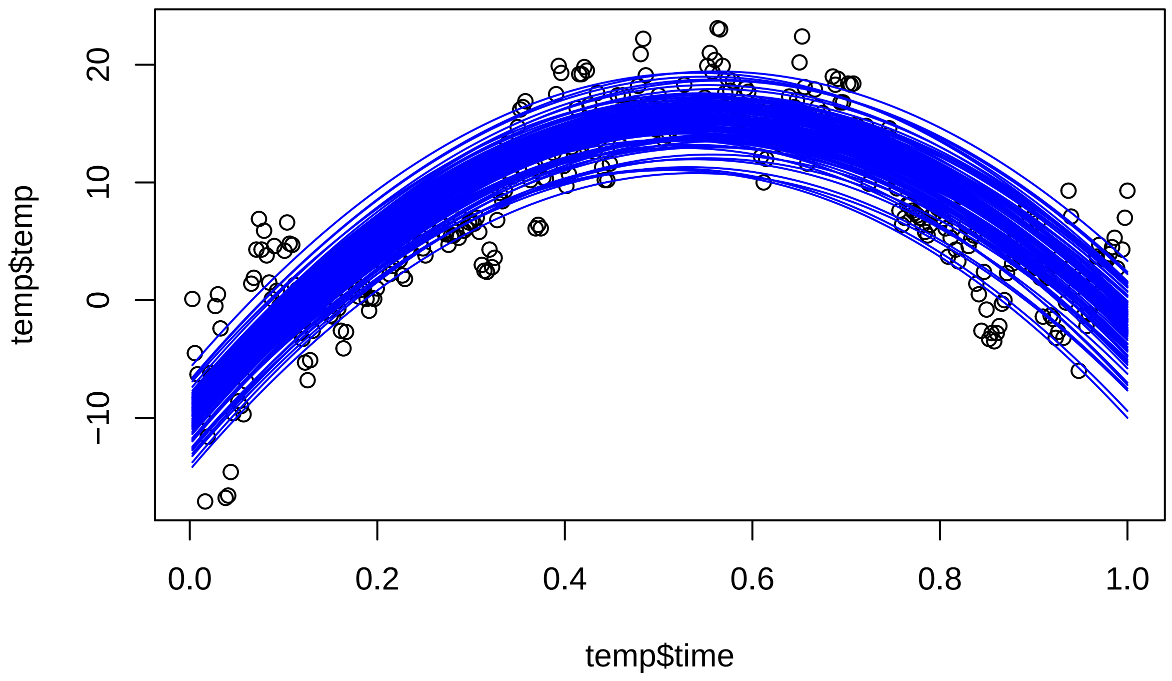
1. b)Check if your prior from a) is sensible

We need to check if a suggested prior is reasonable. Hence we need to simulate draws from the joint prior of all parameters.

$$\sigma^2 = Inv - \chi^2(v_o, \sigma_0^2)$$

Variance of the prior is $\sigma^2 \Omega_0^- 1$

for every draw we need to compute the regression curve.

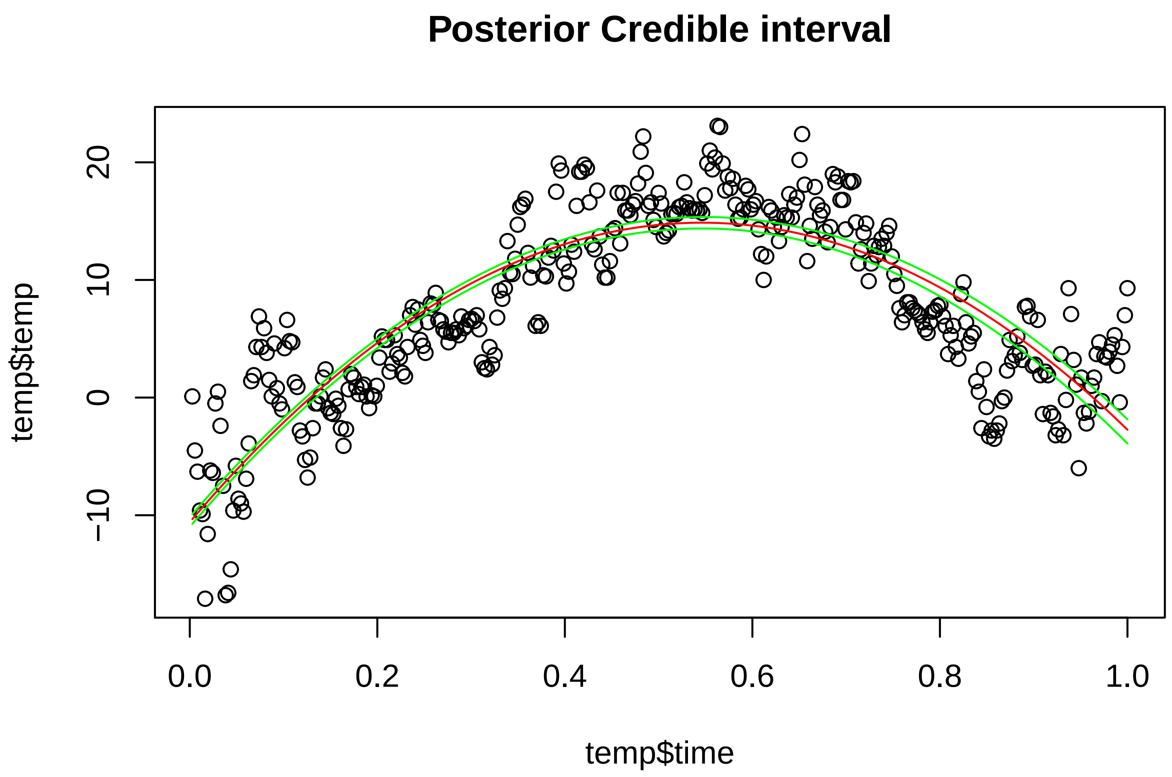


1.(c) Write a program that simulates from the joint posterior distribution

The posterior hyper parameters are :

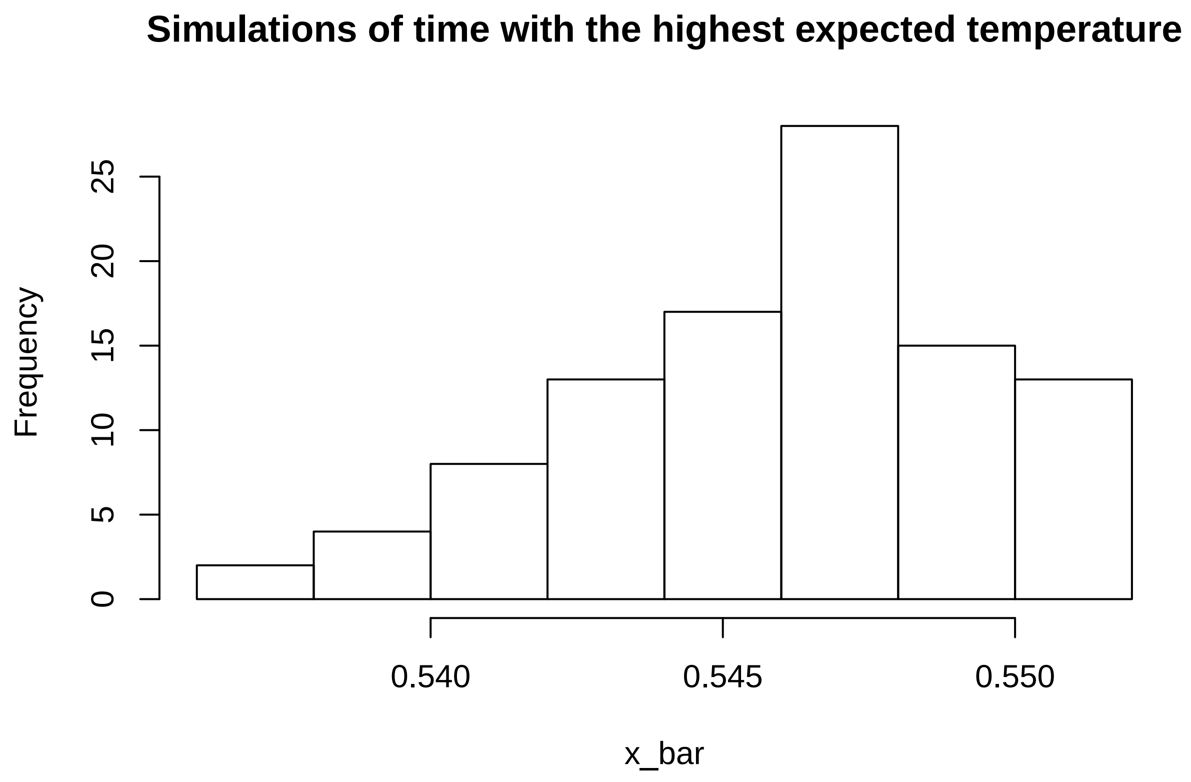
$$\mu_n = (X^{`}X + \Omega_0)^{-1}(X^{`}X\hat{\beta} + \Omega_0\mu_0)$$
$$\omega_n = X^{`}X + \Omega_0$$

$$v_n = v_0 + n$$
$$\sigma^2 = \frac{1}{v_n} * (v_0\sigma_0^2 + (y^{`}y\hat{\beta} + \Omega_0\mu_0))$$



1.d)

```
## The time with the highest expected temperature is : 0.5517646
```



1.e)

To eliminate higher order parameters it would be helpful to set μ_0 to a vector of numbers where higher order terms are close to zero and Ω_0 to a diagonal matrix of large numbers.

```r
omega_0 = diag(c(1,1,1,999,999,999,999,999))
mu_0 = c(-10,93,-85,0,0,0,0,0)
```

Instead of using a polynomial model one could use splines for approximation for stable piecewise definition of the regression curve.

2.Posterior approximation for classification with logistic regression

2.a)

```r
#Make the logistic regression model
glmModel<-glm(Work ~ 0 + ., data = WomenWork, family = binomial)
```

2.b)

Approximate the posterior distribution of the 8-dim parameter vector β with a multivariate normal distribution.

```
## Optimal values for the beta vector are:
```

```
##    Constant  HusbandInc   EducYears    ExpYears   ExpYears2         Age
##  0.62672884 -0.01979113  0.18021897  0.16756670 -0.14459669 -0.08206561
## NSmallChild   NBigChild
## -1.35913317 -0.02468351
```

```
##Posterior covariance matrix is -inv(Hessian)
cat("\n The posterior covariance matrix is:")
```

```
##
##  The posterior covariance matrix is:
```

```
post_Cov <- - solve(Optim_res$hessian)
print(post_Cov)
```

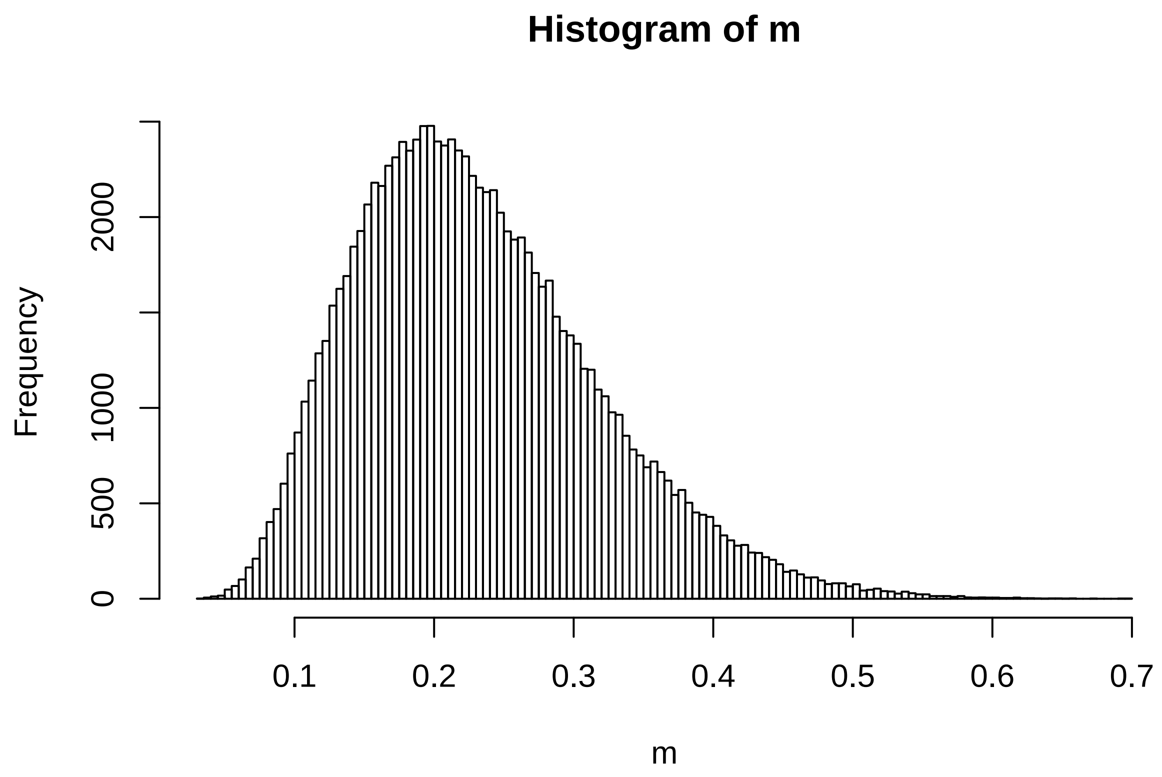
```
##              [,1]          [,2]          [,3]          [,4]          [,5]
## [1,]  2.266022568  3.338861e-03 -6.545121e-02 -1.179140e-02  0.0457807243
## [2,]  0.003338861  2.528045e-04 -5.610225e-04 -3.125413e-05  0.0001414915
## [3,] -0.065451206 -5.610225e-04  6.218199e-03 -3.558209e-04  0.0018962893
## [4,] -0.011791404 -3.125413e-05 -3.558209e-04  4.351716e-03 -0.0142490853
## [5,]  0.045780724  1.414915e-04  1.896289e-03 -1.424909e-02  0.0555786706
## [6,] -0.030293450 -3.588562e-05 -3.240448e-06 -1.340888e-04 -0.0003299398
## [7,] -0.188748354  5.066847e-04 -6.134564e-03 -1.468951e-03  0.0032082535
## [8,] -0.098023929 -1.444223e-04  1.752732e-03  5.437105e-04  0.0005120144
##               [,6]          [,7]          [,8]
## [1,] -3.029345e-02 -0.1887483542 -0.0980239285
## [2,] -3.588562e-05  0.0005066847 -0.0001444223
## [3,] -3.240448e-06 -0.0061345645  0.0017527317
## [4,] -1.340888e-04 -0.0014689508  0.0005437105
## [5,] -3.299398e-04  0.0032082535  0.0005120144
## [6,]  7.184611e-04  0.0051841611  0.0010952903
## [7,]  5.184161e-03  0.1512621814  0.0067688739
## [8,]  1.095290e-03  0.0067688739  0.0199722657
```

```
## Credible Interval for number of small children: [0, 1]
```

2.c)

We need to write a function that simulates from the predictive distribution of the response variable in a logistic regression.Plot the predictive distribution for the Work variablefor a 40 year old woman, with two children (3 and 9 years old), 8 years of education, 10 years of experience. and a husband with an income of 10. Hence the selected data is

```
data <- c(1, 10, 8, 10, (10/10)^2, 40, 1, 1)
nDraws=100000
```



From the plot we can we can say it's not normal and chances for 40 years old women working is less than 25%.

Appendix

```
temp<-read.table("TempLinkoping2016.dat",header=TRUE)

temp.lm<-lm(temp$temp ~ temp$time + I(temp$time^2))
summary(temp.lm)
plot(temp$time,temp$temp)
lines(temp$time,fitted(temp.lm),col="red")

library(mvtnorm)
library(MASS)

#a)
sigma_sq_0 = 2
X<-cbind(1,temp$time,temp$time^2)
Y<-temp$temp
beta_hat<- solve(t(X)%*%(X)) %*% (t(X)%*%(Y))
omega_0 = diag(3)
v_0 <- 25
mu_0 = c(-10,93,-85)
prior <-c()
n=366
library(geoR)

#b)
nDraws = 100
resultlis <- list()
for (i in 1:nDraws)
{
  sigma_sqr_prior <-  rinvchisq(1, df=v_0,  scale=sigma_sq_0)
  variance_prior = sigma_sqr_prior * solve(omega_0)
  prior = mvrnorm(1,mu = mu_0 , Sigma = variance_prior)
  resultlis[[i]] = prior
  prior_model = prior[1] + prior[2]*temp$time  + prior[3]*temp$time^2
  lines(temp$time,prior_model,col="blue")
}

#c) posterior

nDraws = 100
v_n<-v_0+n
mu_n<-solve((t(X)%*%X)+omega_0)%*%(((t(X)%*%X)%*%beta_hat)+
        (omega_0 %*%mu_0))
omega_n<-(t(X)%*%X)+omega_0
sigma_sq_n<-(1/v_n)*((v_0*sigma_sq_0)+((t(Y)%*%Y)+(t(mu_0)
           %*%omega_0%*%mu_0)-(t(mu_n)%*%omega_n%*%mu_n)))

diagnal_elements = as.numeric(diag(1/omega_n))
omega_inv = diag(diagnal_elements)
posterior <- list()
x_bar<-c()
sigma_sqr_post<-c()
plot(temp$time,temp$temp)
```

```r
for (i in 1:nDraws)
{
  sigma_sqr_post[i] <- rinvchisq(1, df=v_n,  scale=sigma_sq_n)
  variance_post = sigma_sqr_post[i] * omega_inv
  post = mvrnorm(1,mu = mu_n , Sigma =variance_post)
  post_model = post[1] + post[2]*temp$time  + post[3]*temp$time^2
  x_bar[i]<- -(post[2]/(2*post[3]))
  posterior[[i]] = post_model
}

posterior_beta  <- as.data.frame(do.call("rbind", posterior))
lower<-c()
upper<-c()
for (i in 1:n){
lower[i] <- quantile(as.vector(posterior_beta[,i]), probs = c(0.025, 0.975))[1]
upper[i] <- quantile(as.vector(posterior_beta[,i]), probs = c(0.025, 0.975))[2]
}
posterior_mean = colMeans(posterior_beta)
plot(x=temp$time,y=temp$temp, main = "Posterior Credible interval" )
lines(x=temp$time,y = posterior_mean,col =  "red")
lines(x=temp$time,y = upper, col =  "green")
lines(x=temp$time,y = lower,col =  "green")

#d)
cat("The time with the highest expected temperature is :",(max(x_bar)))
hist(x_bar,main = "Simulations of time with the highest expected temperature")

#e)
omega_0 = diag(c(1,1,1,999,999,999,999,999))
mu_0 = c(-10,93,-85,0,0,0,0,0)

#2.a
WomenWork=read.table("WomenWork.dat",header=TRUE)
library(mvtnorm)
####2.2
# y is response variable which is work
y<-as.vector(WomenWork[,1])
# other 8 variables are in X
X<-as.matrix(WomenWork[,2:9])
no_of_Para<-ncol(X)
#given that tau =10
tau<-10


mu_0<-as.vector(rep(0,no_of_Para))
sigma_0 <- tau^2*diag(no_of_Para)

##prior function
prior_prob <- function(beta_values,sigma_0) {
  mean1=matrix(rep(0,8),ncol = 1)
  prior <- dmvnorm(beta_values, mean= mean1, sigma=sigma_0, log=TRUE)
  return(prior)
}
```

```
##likelyhood function
like_prob <- function(beta_values, y, X) {
  Logist_Par <- X%*%beta_values
  exp_part<-1 + exp(Logist_Par)
  log_liklihood <- sum(Logist_Par*y - log(exp_part))
  if (abs(log_liklihood) == Inf) {
    log_liklihood = -20000
  }
  return(log_liklihood)
}

##posterior function

post_prob<-function(beta_values, y, X, mu_0, sigma_0) {
  param <- as.vector(beta_values)
  ##posterior is the sum of prior and likelyhood
  logposter <- prior_prob(beta_values, sigma_0) + like_prob(beta_values, y, X)
  return(logposter)
}

##Optim function
intital_value=rep(0,8)
Optim_res <- optim(intital_value, fn=post_prob,
                      y = y, X = X, mu_0 = mu_0, sigma_0 = sigma_0,
                      method="BFGS", control=list(fnscale=-1), hessian=TRUE)

##Optimal values for param are
n = dim(WomenWork)
num = n[1]
n = n[2]
cNames <- names(WomenWork)[2:n]
mu=Optim_res$par
names(mu) = cNames
cat("Optimal values for the beta vector are: \n")
print(mu)
##Posterior covariance matrix is -inv(Hessian)
cat("\n The posterior covariance matrix is:")
post_Cov <- - solve(Optim_res$hessian)
print(post_Cov)
n_SmallChild = sort(WomenWork$NSmallChild)

cred_int = c(n_SmallChild[floor(nrow(WomenWork) * 0.05)],
            n_SmallChild[floor(nrow(WomenWork) * 0.95)+1])
cat("Credible Interval for number of small children:
   [", cred_int[1], ", ", cred_int[2], "]", sep="")
data <- c(1, 10, 8, 10, (10/10)^2, 40, 1, 1)
nDraws=100000
prediction<-function() {
  sigma <-post_Cov
  mu <-Optim_res$par
  betas = rmvnorm(nDraws, mu, sigma)
  probWork<-1/(1+exp(-betas %*% data))
  return(probWork)
}

m<-prediction()
h=hist(m, breaks = 100)
```

Lab4

Priya Kurian Pullolickal(priku577) and Thi Pham (thiph169)

23 May 2018

1 The maximum likelihood estimator of beta in the Poisson regression model

```
poi <- read.table("eBayNumberOfBidderData.dat", header = TRUE)
model <- glm(nBids ~ . -Const, data = poi)
```

The features that are significant are:

VerifyID

Sealed

Minblem

MinBidShare

Minblem

```
features <- summary(model)$coef[-1, 4] <= 0.05
features
```

```
## PowerSeller    VerifyID      Sealed     Minblem     MajBlem     LargNeg
##       FALSE        TRUE        TRUE        TRUE        TRUE       FALSE
##     LogBook MinBidShare
##       FALSE        TRUE
```

2.Bayesian analysis of the Poisson regression

```
library(mvtnorm)
```

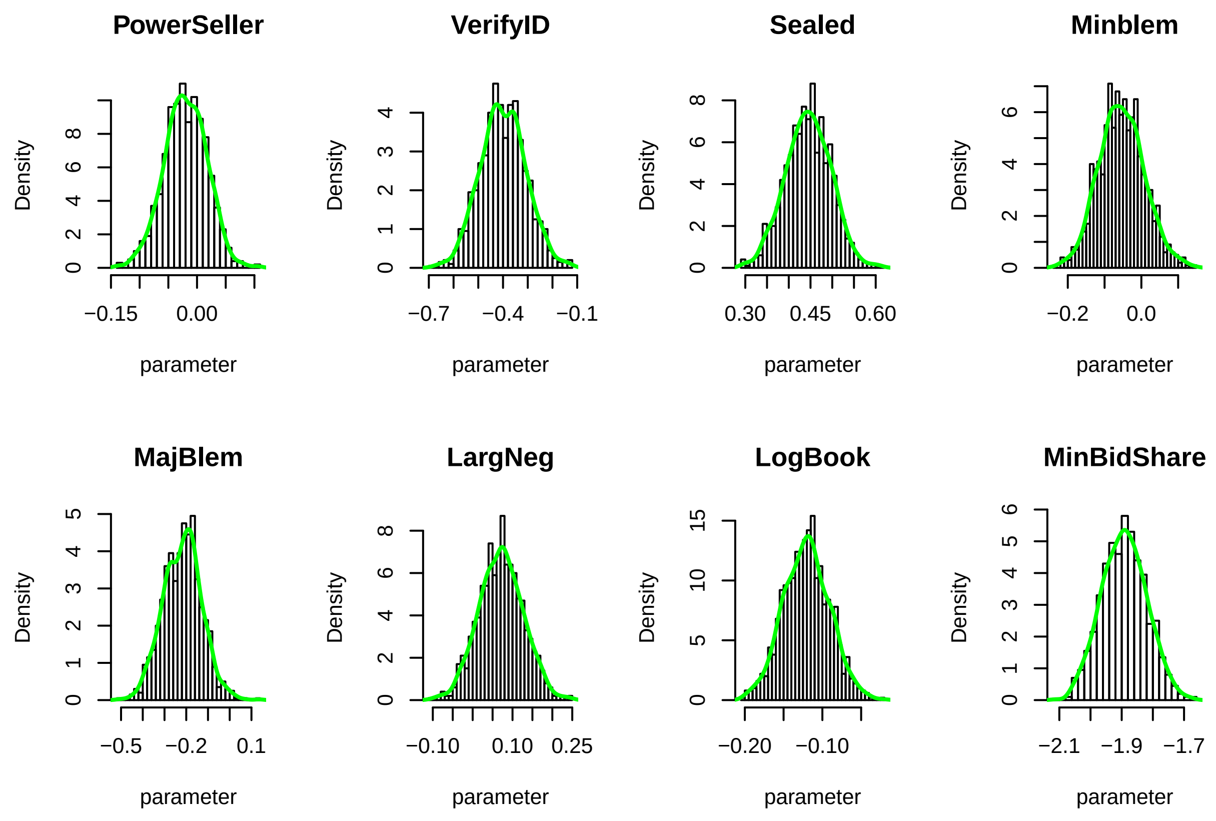
```
## Warning: package 'mvtnorm' was built under R version 3.4.3
```

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.4.3
```

```
y <- as.vector(poi[,1])
X <- as.matrix(poi[,2:10])
npara <- ncol(X)
mu0 <- rep(0, npara)
tau <- 10
sigma0 <- (tau^2) * (solve(t(X)%*%X))
```

```r
logPostFunc<-function(betas,y,X,mu0,sigma0) {
  betas<-as.vector(betas)
  lambda<-exp(X%*%betas)
  mu <- matrix(rep(0,9),ncol = 1)
  logposter<- dmvnorm(betas, mean=mu, sigma=sigma0, log=TRUE) + sum(dpois(y,lambda,log = TRUE))
  return(logposter)
}

initVal <- rep(0, npara)
OptimResults <- optim(initVal,logPostFunc,
                      y = y, X = X, mu0 = mu0, sigma0 = sigma0,
                      method="BFGS", control=list(fnscale=-1), hessian=TRUE)
parmts<-mvrnorm(1000,OptimResults$par,-(solve(OptimResults$hessian)))
par(mfrow=c(2,4))
for(i in 2:9) {
  hist(parmts[,i],breaks = 30,freq = FALSE, xlab = "parameter",main = colnames(X)[i])
  lines(density(parmts[,i]),col="green",lwd=2)
}
```



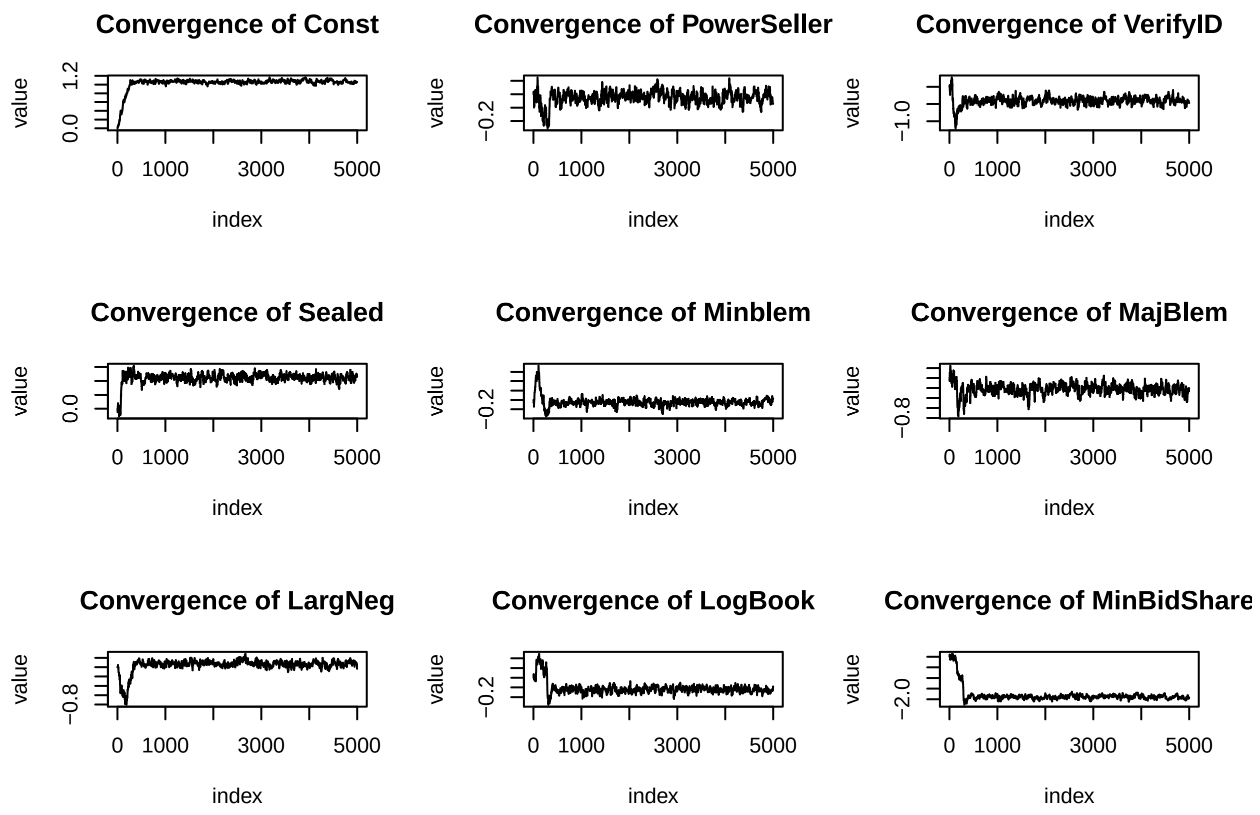
```r
hessian <- -(solve(OptimResults$hessian))
```

3.simulate from the actual posterior of beta using the Metropolis algorithm and compare

```r
posterr <- function(first_fun, x0, iter, z, ...) {
  xval1<-x0
  thetas <- matrix(NA,nrow = iter, ncol = 9)
  thetas[1,] <- xval1
  for(i in 2:iter) {
    val_y <- mvrnorm(1,xval1,z)
    ph1 <- first_fun(val_y, ...)
    ph2 <- first_fun(xval1, ...)
    ph3 <- exp(ph1-ph2)
    alpha <- min(1,ph3)
    dist<- runif(1)
    if(dist< alpha) {
      xval1 <- val_y
    }
    thetas[i,] <- xval1
  }
  return(thetas)
}

result <- posterr(logPostFunc, x0=rep(0,9), iter=5000, z=0.5*hessian, y=y, X=X, mu0, sigma0)
```

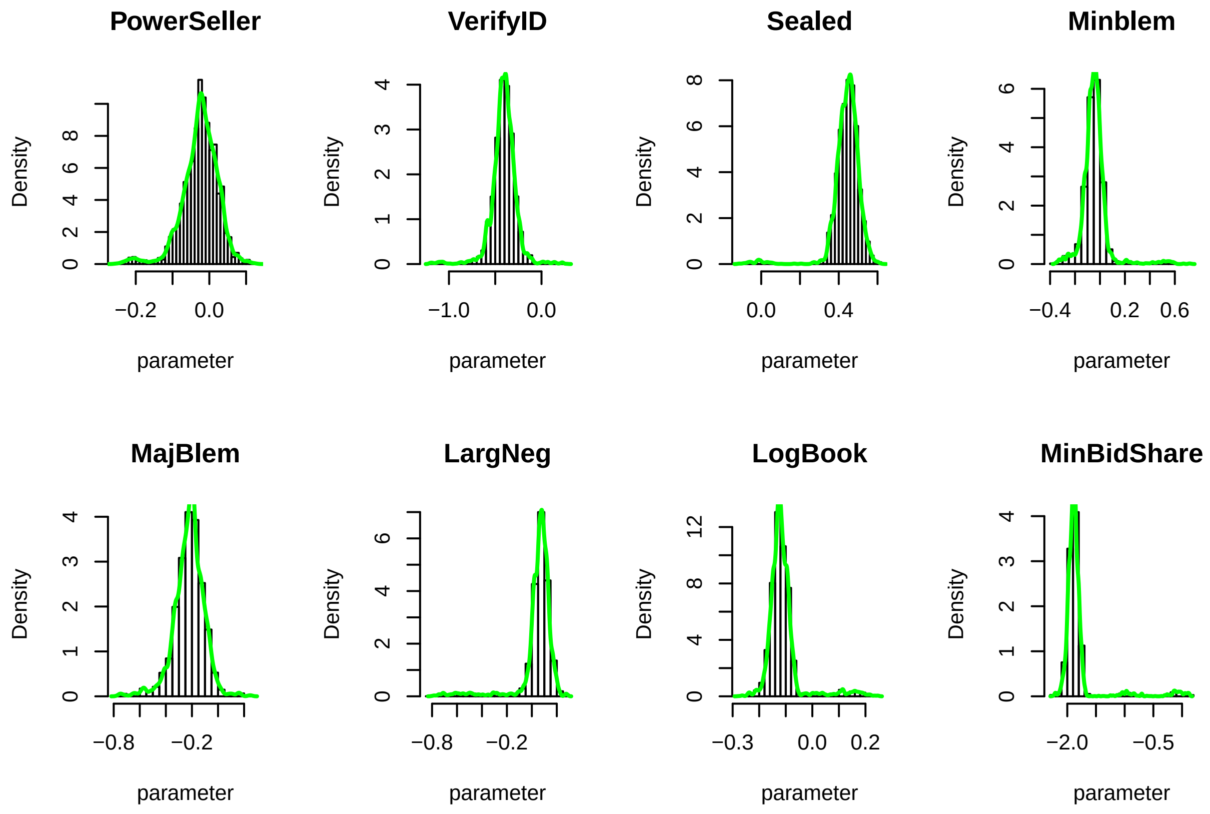
The convergence can be seen from the below plot

```r
par(mfrow=c(3, 3))
for (i in 1:9) {
  plot(result[, i],type = "l",
       main = paste("Convergence of", colnames(X)[i]),
       xlab = "index", ylab = "value")
}
```



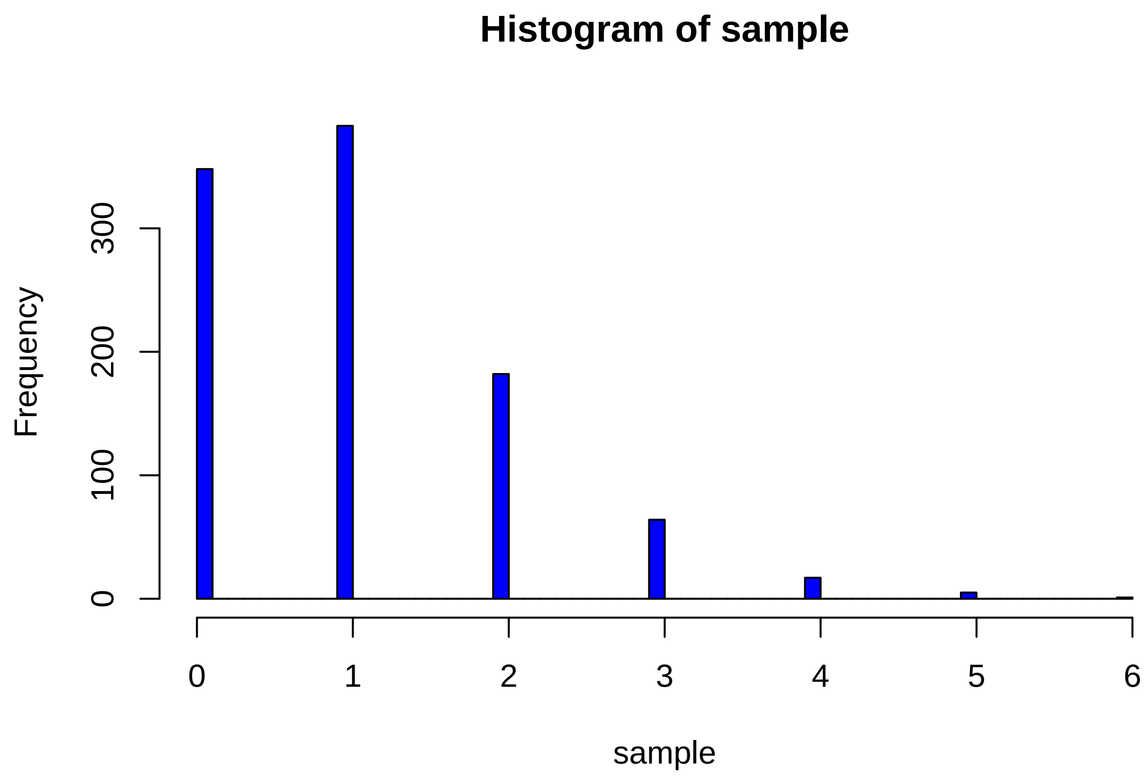
We then compute the posterior

```r
par(mfrow=c(2,4))
for(i in 2:9) {
  hist(result[,i],breaks = 30,freq = FALSE, xlab = "parameter",main = colnames(X)[i])
  lines(density(result[,i]),col="green",lwd=2)
}
```



4.Plot the predictive distribution and probability of no bidders in this new auction.

```r
val_coeff <- colMeans(result)
val_observed <- matrix(c(1, 1, 1, 1, 0, 0, 0, 1, 0.5), nrow = 1)
lambda <- exp(val_observed %*% val_coeff)
sample <- rpois(1000, lambda)
hist(sample, breaks = 50, col = "blue")
```



The probability is :

```
probability <- dpois(0, lambda)
sum(sample == 0 + 1) / (length(sample) + 1)
```

```
## [1] 0.3826174
```

732A91 Bayesian Learning
Report Lab 1

Thi Pham (thiph169) and Priya Kurian Pullolickal (priku577)

10 April 2018

Question 1 - Bernoulli. . . again

Let $y_1, ..., y_n|\theta$ $Bern(\theta)$, and assume that you have obtained a sample with $s = 14$ successed in $n = 20$ trials. Assume a $Beta(\alpha_0, \beta_0)$ prior for θ and let $\alpha_0 = \beta_0 = 2$.

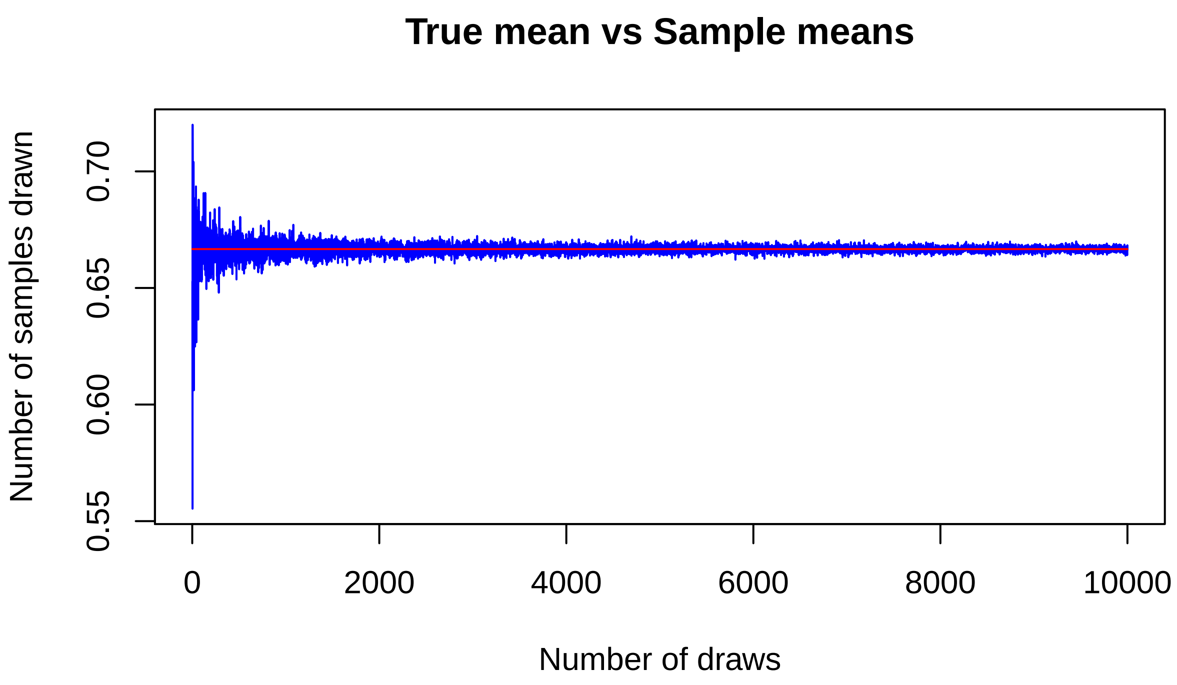
(a)

Draw random numbers from the posterior $\theta|y \sim Beta(\alpha_0 + s, \beta_0 + f), y = y_1, ..., y_n$, and verify graphically that the posterior mean and standard deviation converges to the true values as the number of random draws grows large.

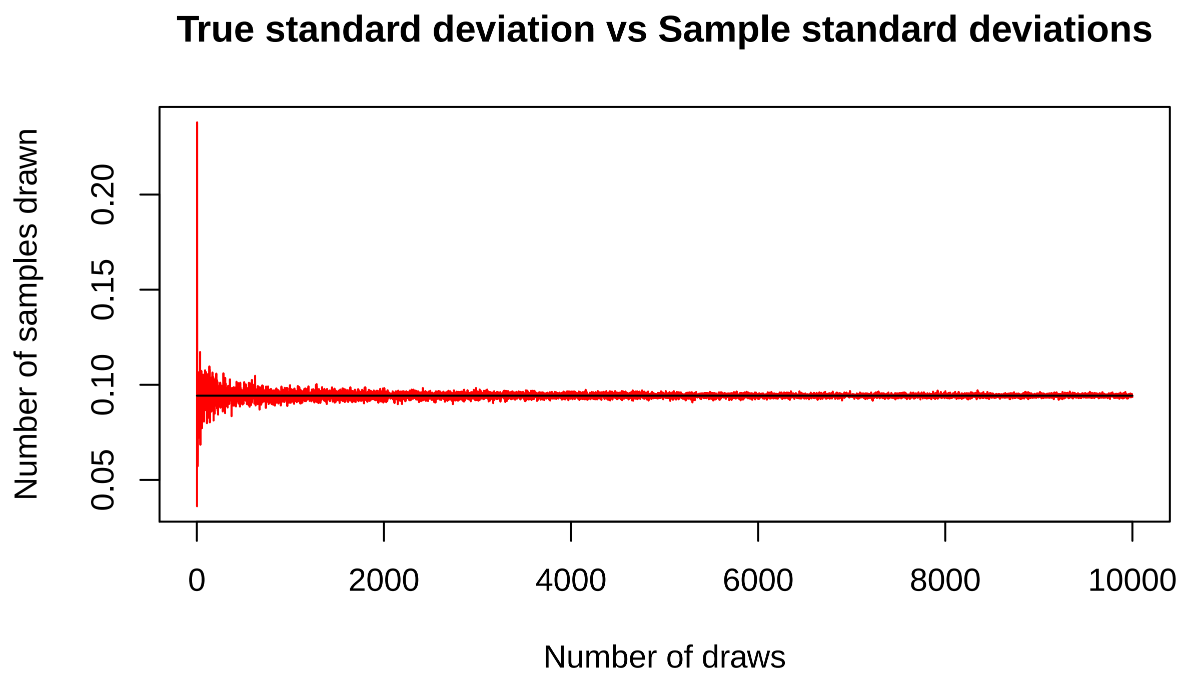
The true mean and standard deviation are shown below:

```
## The true mean: 0.6666667
```

```
## The standard deviation: 0.0942809
```



It can be seen from the plot above that the sample mean converges to 0.6667 (which is the true mean value) when number of samples grow large.



Similarly, when the numbers of random draws grow large, the sample standard deviation converges to 0.094 (which is the true standard deviation) as shown in plot above.

(b)

Use simulation(nDraws = 1000) to compute the posterior probability $Pr(\theta < 0.4|y)$ and compare with the exact value.

```
## The simulated posterior probability: 0.0031
```

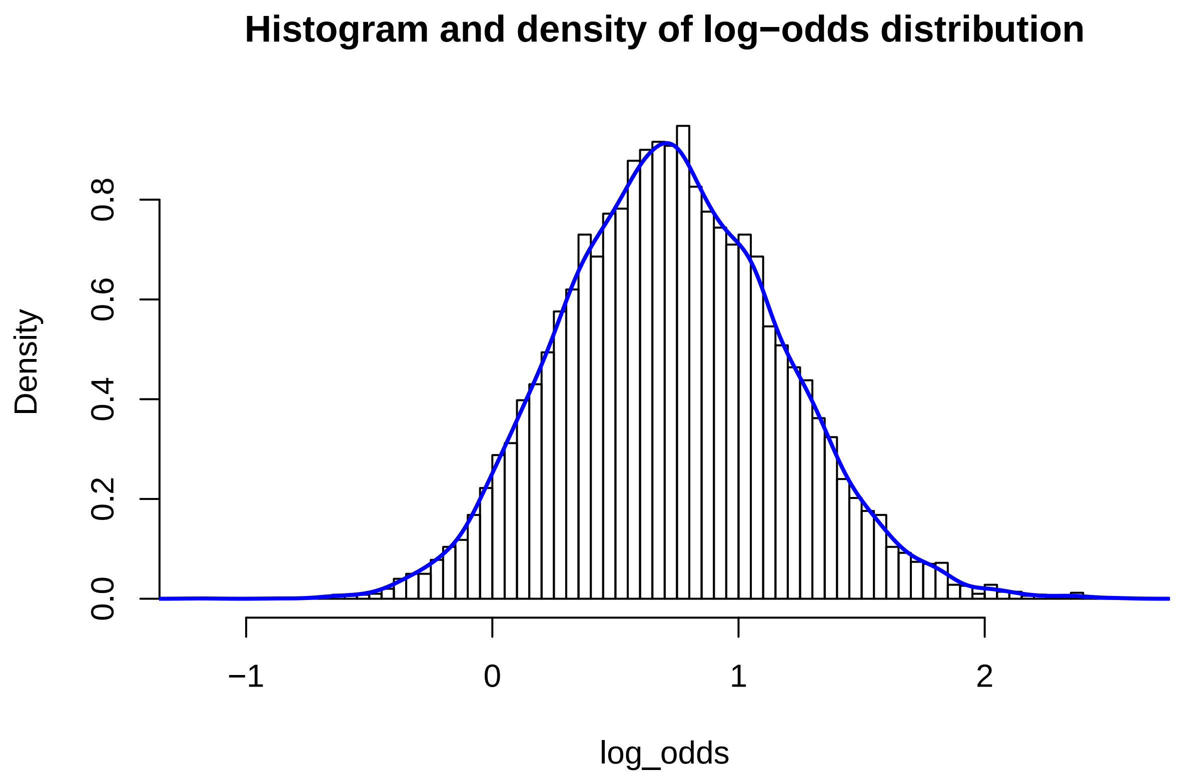
```
## The exact probability: 0.003972681
```

The simulated posterior probability $Pr(\theta < 0.4|y)$ obtained is about 0.0036 which is very close to the exact value.

(c)

Compute the posterior distribution of the log-odds $\phi = log\frac{\theta}{1-\theta}$ by simulation (nDraws = 1000).

The posterior distribution of the log-odds is showed below.



Question 2 - Log-normal distribution and the Gini coefficient.

Given 10 observations : 14, 25, 45, 25, 30, 33, 19, 50, 34 and 67.

Model: log-normal distribution $logN(\mu, \sigma^2)$.

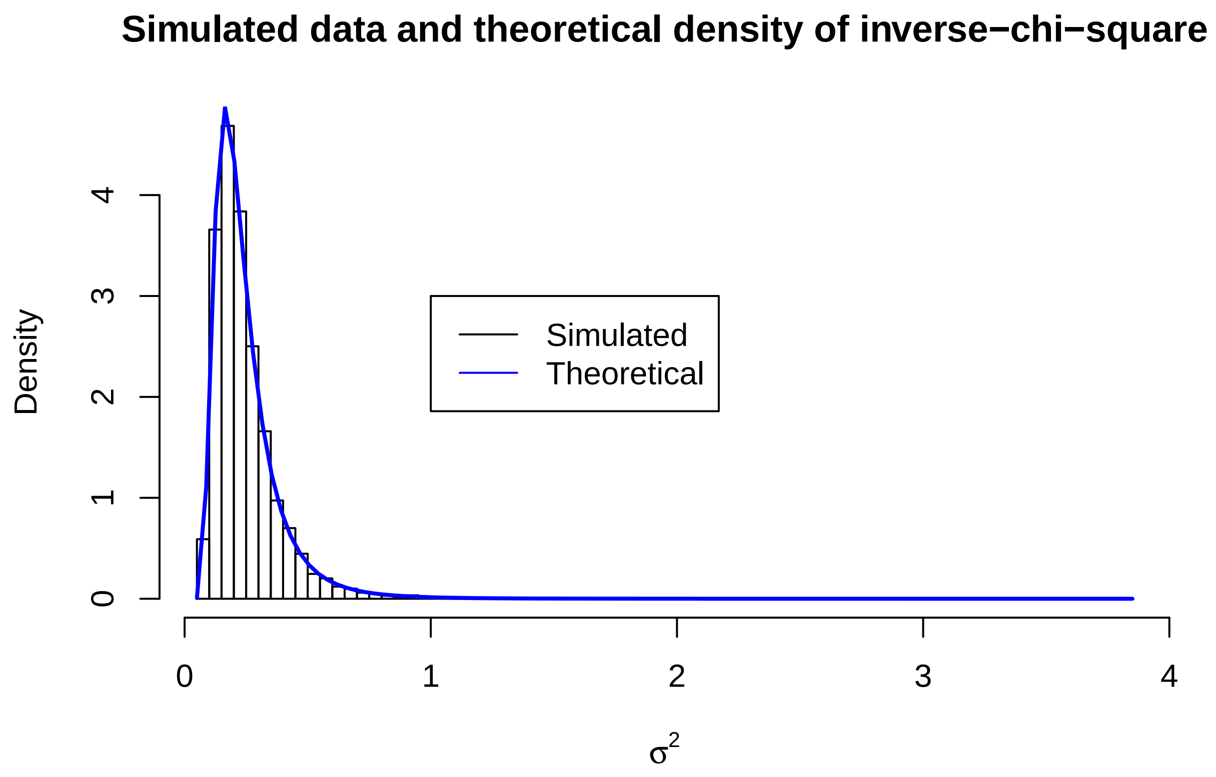
Parameters: $\mu = 3.5$; σ^2 is unknown

Prior: Non-informative $p(\sigma^2) \propto 1/\sigma^2$

Posterior: $Inv - \chi^2(n, \tau^2)$

(a)

Simulate 10,000 draws from the posterior of σ^2 (assuming $\mu = 3.5$) and compare with the theoretical $Inv - \chi^2(n, \tau^2)$ posterior distribution.

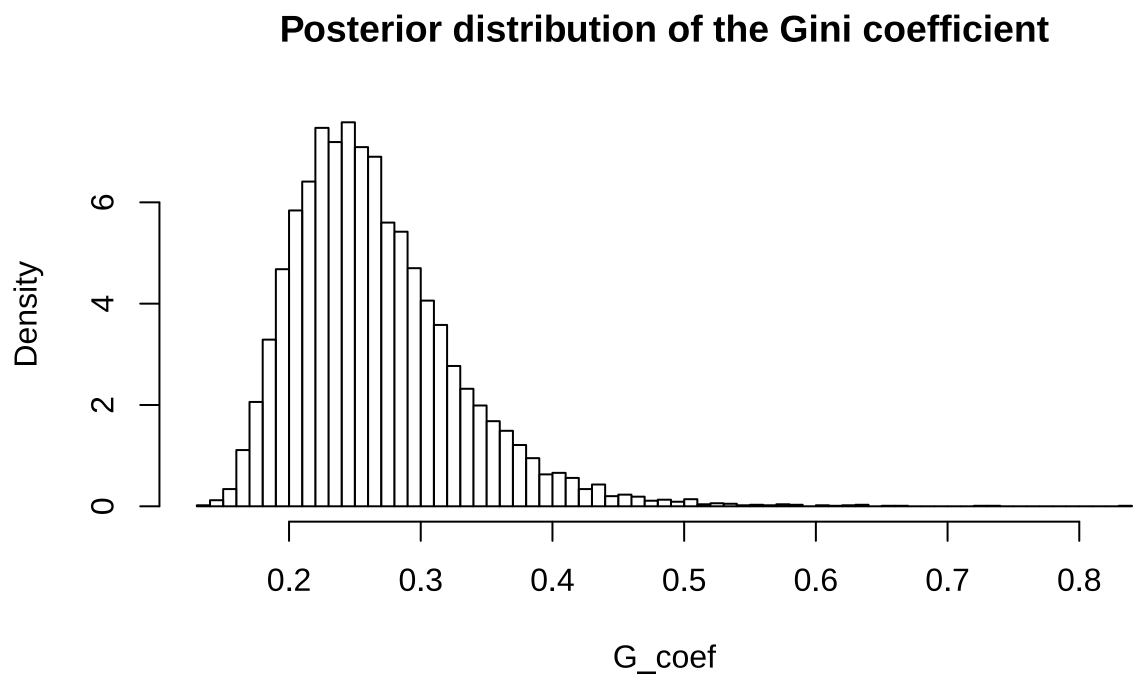


It can be seen that the shape of simulated distribution fits quite well with the theoretical distribution.

(b)

The most common measure of income inequality is the Gini coefficient, G, where $0 \leq G \leq 1$. $G = 0$ means a completely equal income distribution, whereas $G = 1$ means complete income inequality. SIt can be shown that $G = 2\Phi(\sigma^2/\sqrt{2}) - 1$ when incomes follow a $logN(\mu, \sigma^2)$ distribution. $\Phi(z)$ is the cumulative distribution function (CDF) for the standard normal distribution with mean zero and unit variance.

Use the posterior draws in (a) to compute the posterior distribution of the Gini coefficient G for the current data set.



A histogram above shows the posterior distribution of the Gini coefficent. The values show the G coefficients are closer to 0 than 1 which indicate the equal income distribution.

(c)

Use the posterior draws from (b) to compute a 95% equal tail credible interval for G. An 95% equal tail interval (a, b) cuts off 2.5% percent of the posterior probability mass to the left of a, and 97.5% to the right of b.

Also, do a kernel density estimate of the posterior of G using the density function in R with default settings, and use that kernel density estimate to compute a 95% Highest Posterior Density interval for G. Compare the two intervals.

The 95% equal tail credible interval for G and the 95% Highest Posterior Density(HPD) are shown below.

```
## A 95% credible interval for G:
```

```
##      2.5%     97.5%
## 0.1750479 0.4158995
```

```
## 95% Highest Posterior Density set:
```

```
##     lower     upper
## 0.1605326 0.3894407
## attr(,"credMass")
## [1] 0.95
## attr(,"height")
## [1] 0.8186877
```

The highest density interval slightly skewed to the lelf but the intervals are quite similar for both cases.

Question 3 - von Mises distribution (Work with directional data)

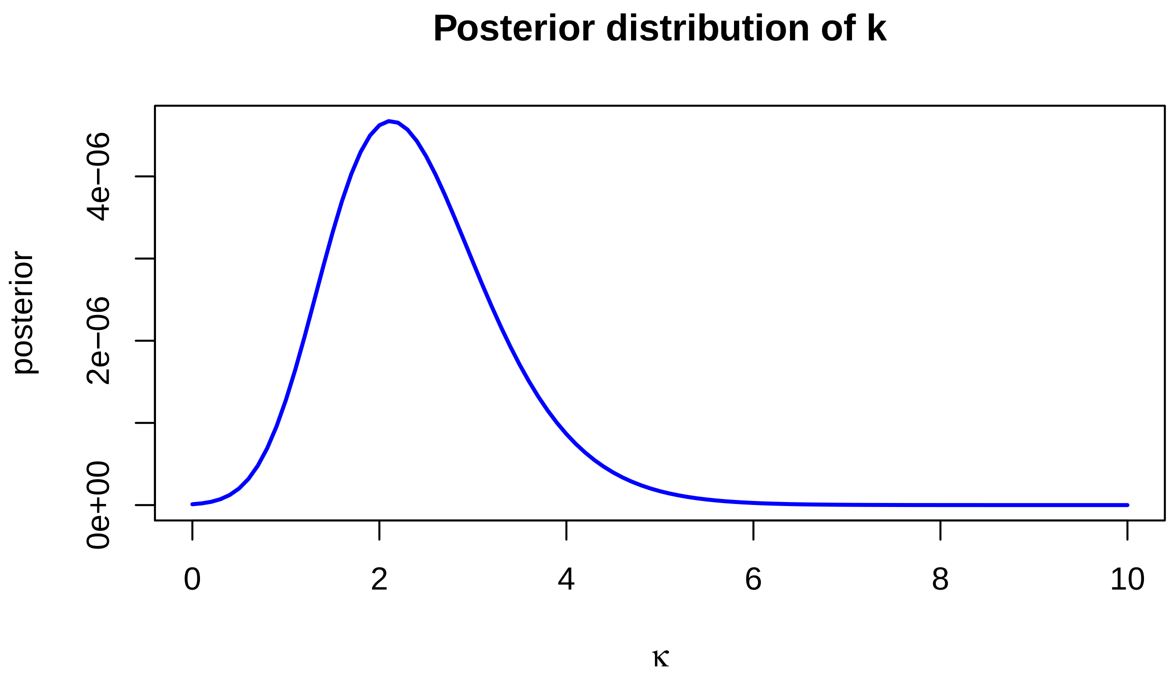
The data points are observed wind directions at a given location on ten different days. The data are recorded in degrees: (40, 303, 326, 285, 296, 314, 20, 308, 299, 296) which converted to radians:(-2.44, 2.14, 2.54, 1.82, 2.02, 2.33, -2.79, 2.23, 2.07, 2.02).

Model: Von Mises distribution
Parameters: $\mu = 2.39$; concentration parameter κ where $\kappa \sim exp(\lambda = 1)$
Prior: $p(k) \sim exp(-\lambda)$
Posterior: $p(\kappa|\mu, y) = p(y|\mu, \kappa) * p(\kappa)$

$$p(\kappa|\mu, y) = (\prod_{i=1}^{10} \frac{exp(\kappa * cos(y-\mu))}{2\pi I_0(\kappa)}) * (exp(-\kappa))$$

(a)

Plot the posterior distribution of κ for the wind direction data over a fine grid of κ values.



(b)

Find the (approximate) posterior mode of κ from the information in (a).

```
## Approximate posterior mode of k: 2.1
```

Contributions

Results and comments presented in this report have been developed and discussed together by the members of the group.

Appendix

```
#####Question 1#####
#(a)

n = 20; s = 14; f = 6
alpha0 = 2; beta0 = 2

drawMean = c()
drawSd = c()

for (i in 1:10000){
   draw <- rbeta(i, shape1 = alpha0 + s, shape2 = beta0 + f)
   drawMean[i] <- mean(draw)
   drawSd[i] <- sd(draw)
}

#true mean and sd
trueMean <- (alpha0+s)/(alpha0+s+beta0+f)
trueSd <- sqrt(((alpha0+s)*(beta0+f))/(((alpha0+s+beta0+f)^2)*(alpha0+s+beta0+f+1)))
cat("The true mean:", trueMean, "\n")
cat("The standard deviation:", trueSd, "\n")

#Plot of posterior mean
n = length(drawMean)
plot(x = 1:n,y = drawMean , type = "l", col = "blue",
     xlab = "Number of draws", ylab = "Number of samples drawn",
     main = "True mean vs Sample means")
lines(x = 1:n, y = rep(trueMean,n), col = "red")

#Plot of posterior sd
n = length(drawSd)
plot(x = 1:n,y = drawSd , type = "l", col = "red",
     xlab = "Number of draws", ylab = "Number of samples drawn",
     main = "True standard deviation vs Sample standard deviations")
lines(x = 1:n, y = rep(trueSd,n), col = "black")


#(b)

#Draw 10000 samples
simulated_data <- rbeta(10000, shape1 = alpha0 + s, shape2 = beta0 + f)

#Calculate probability p(theta < 0.4|y) which is 0.0047
prob_lessthan_0.4 <- simulated_data[simulated_data < 0.4]
require_prob <- length(prob_lessthan_0.4)/length(simulated_data)
require_prob
```

```
#Exact value which is 0.003972681
exact_prob <- pbeta(0.4, shape1 = alpha0 + s, shape2 = beta0 + f)
exact_prob

#(c)

log_odds <- log(simulated_data/(1 - simulated_data))
hist(log_odds, breaks = 100, probability = TRUE, main = "Histogram and density of log-odds distribution
lines(density(log_odds), col = "blue", lwd = 2)

#####Question 2#####
library(geoR)

#(a)
data <- c(14, 25, 45, 25, 30, 33, 19, 50, 34, 67)
dataMean <- mean(data)
dataVar <- var(data)
n = length(data)
df = 10
x= seq(0, 10000, 1)
#Simulate inverse chisquare
tau_sq <- sum((log(data)-3.5)^2)/n
inv_chi_sq <- rinvchisq(10000, df = 10, scale = tau_sq) ##Simulated signma^2

#Plot simulated with theoretical inverse chisquare
hist(inv_chi_sq, breaks = 100, probability = TRUE,
     xlab = expression(paste(sigma^2)),
     main = "Simulated data and theoretical density of inverse-chi-square")
curve(dinvchisq(x, df=10, scale = tau_sq),
      add = TRUE,col = "blue", lwd = 2) ##Theoretical inverse chisquare
legend(1,3, legend = c("Simulated", "Theoretical"), col = c("black", "blue"), lty = 1:1)

#(b)
sigma_sq <- inv_chi_sq
sigma <- sqrt(sigma_sq)
phi <- pnorm((sigma/sqrt(2)), 0, 1)
G_coef <- 2*phi-1

hist(G_coef, breaks = 100, probability = TRUE,
     main = "Posterior distribution of the Gini coefficient")
lines(density(G_coef), col = "blue", lwd = 2)

#(c)

#The HPD has the nice property that any point **within**
#the interval has a higher density than any other point outside.
#Thus, HPD interval is **the** collection of most likely values of
#the parameters.
#Ref https://www.r-bloggers.com/probable-points-and-credible-intervals-part-1/

#Compute a 95% equal tail credible interval for G.
credible_interval <- quantile(G_coef, probs = c(0.025, 0.975) )
credible_interval
```

```r
#  2.5%     97.5%
#0.1748642 0.4220841

#compute a 95% Highest Posterior Density interval for G.
library(HDInterval)
kernel_density <- density(G_coef)
highest_density_interval <- hdi(kernel_density, credMass = 0.95)
highest_density_interval

#lower     upper
#0.1606569 0.3963481

#Comparing two intervals
#The highest density interval slightly skewed to the lelf, the intervals
#quite similar

#####Question 3#####
#(a)

#record_degree <- c(40, 303, 326, 285, 296, 314, 20, 308, 299, 296)
record_radian <- c(-2.44, 2.14, 2.54, 1.82, 2.02, 2.33, -2.79, 2.23, 2.07, 2.02)

posterior <- function(k){
  bessel_0 <- besselI(k, nu = 0)
  loglike <- prod(exp(k*cos(record_radian - 2.39)) / (2*pi*bessel_0))
  prior <- exp(-k)

  pos <- loglike * prior
  return(pos)
}

#Apply posterior to vector of k
k = seq(0, 10, 0.1)

res <- sapply(seq(0, 10, 0.1), posterior)

plot(x=k, y=res, type = "l", col = "blue", lwd = 2,
     xlab = expression(paste(kappa)), ylab = "posterior",
     main = "Posterior distribution of k")

#(b)
k[which.max(res)]
#Approximate posterior mode of k is 2.1
```

732A91 Bayesian Learning
Report Lab 2

Thi Pham (thiph169) and Priya Kurian Pullolickal (priku577)

26 April 2018

Question 1: Linear and polynomial regression

The dataset **TempLinkoing.txt** is given, it contains daily temperature at Malmslatt (in Celcius degrees) of the year 2016. The response variable is *temp* and covariate is *time*. The task is to perform a Bayesian analysis of a quadratic regression.

Below is an extract of the data.

```
##          time  temp
## 1 0.002732240   0.1
## 2 0.005464481  -4.5
## 3 0.008196721  -6.3
## 4 0.010928962  -9.6
## 5 0.013661202  -9.9
## 6 0.016393443 -17.1
```

(a) Determining the prior distribution of the model parameters

This task is to set the prior hyperparameters μ_0, Ω_0, ν_0 and σ_0^2 to sensible values, using the conjugate prior for the linear regression model, assuming Ω_0 is diagonal matrix.

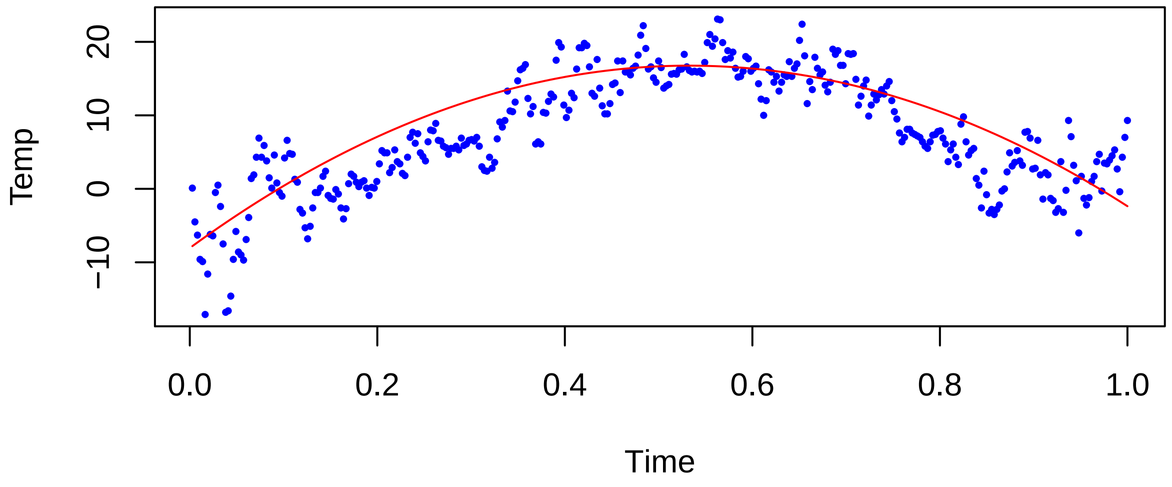
Given that the likelihood is a quadratic regression, we fit our model as below.

```
##
## Call:
## lm(formula = temp ~ time + I(time^2), data = temp_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.0408  -2.6971  -0.1414   2.5157  12.2085
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -10.6754     0.6475  -16.49   <2e-16 ***
## time         93.5980     2.9822   31.39   <2e-16 ***
## I(time^2)   -85.8311     2.8801  -29.80   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.107 on 363 degrees of freedom
## Multiple R-squared:  0.7318, Adjusted R-squared:  0.7304
## F-statistic: 495.3 on 2 and 363 DF,  p-value: < 2.2e-16
```

The prior hyperparameters are chosen as follow.

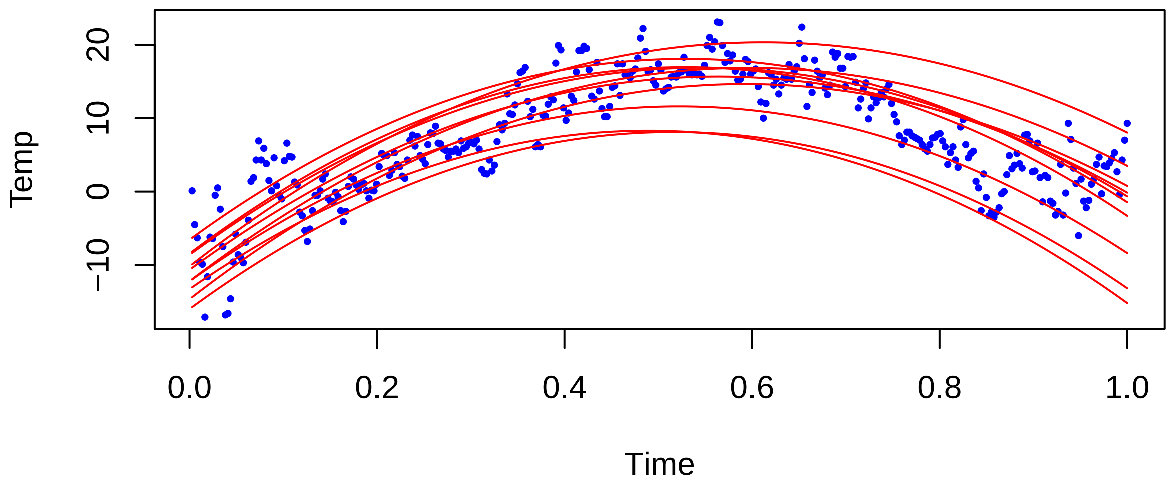
- μ_0: coefficients of *lm* model, which are -10.67535, 93.59796 and -85.83109
- Ω_0: a diagonal matrix of 1s with 3 dimenstions as we assume all features with the same weight (importance)
- ν_0: 6
- σ_0^2: 16 which is a square of resudual standard error of our model

The plot below with the chosen values of parameters.



(b) Check if our prior from part (a) is sensible.

To check for the sensiblility of our prior parameters, we computed two tasks, the first one is regression curves for 10 draws and the second one is the parameters mean of 1000 draws.



```
## Estimate of beta prior:  -10.84455 93.8767 -85.89584
```

```
## Estimate of sigma prior:  14.69923
```

The values obtained are quite similar to the values we set in part (a).

(c)

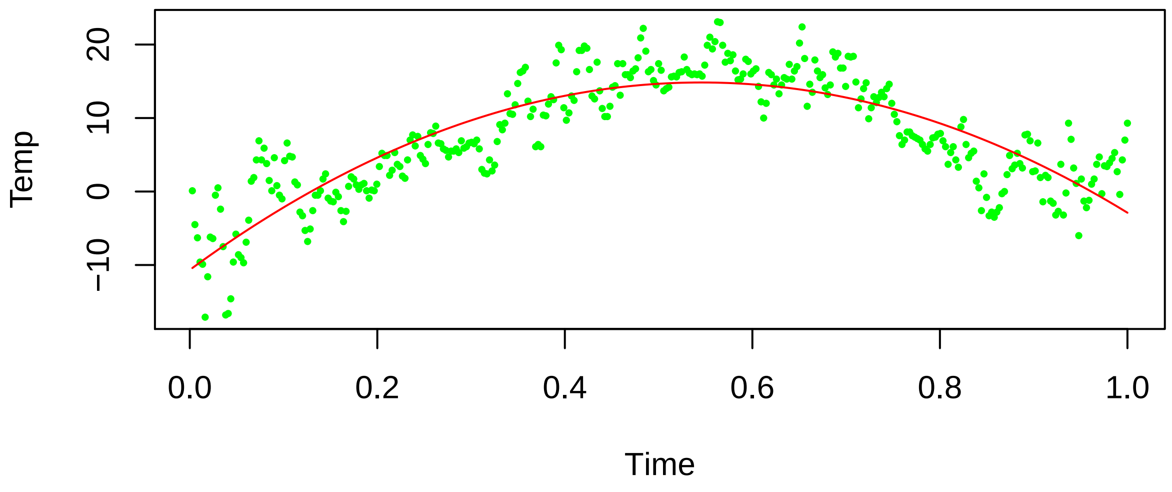
Write a program that simulates from the joint posterior distribution of $\beta_0, \beta_1, \beta_2$ and σ^2.

```
X <- cbind(rep(1, nrow(temp_data)), temp_data$time, temp_data$time^2)
y <- temp_data$temp
n = nrow(temp_data)
beta_hat <- solve(t(X)%*%X) %*% t(X) %*% y
mu_n <- solve(t(X)%*%X+omega_0)%*%(t(X)%*%X%*%beta_hat+omega_0%*%mu_0 )
omega_n <- t(X) %*% X + omega_0
nu_n <- nu_0 + n
term <- as.vector(t(y)%*%y+t(mu_0)%*%omega_0%*%mu_0)-(t(mu_n)%*%omega_n%*%mu_n)
sigmasq_n <- as.vector((1/nu_n)*(nu_0*sigmasq_0 + term))
#betas_posterior of 1000 draws
myposterior <- suggest_prior(n=1000,nu=nu_n,sigmasq=sigmasq_n,mu=mu_n,omega=omega_n)
beta_pos_n <- colMeans(myposterior$beta)
sigma2_pos_n <- mean(myposterior$sigma2)
```

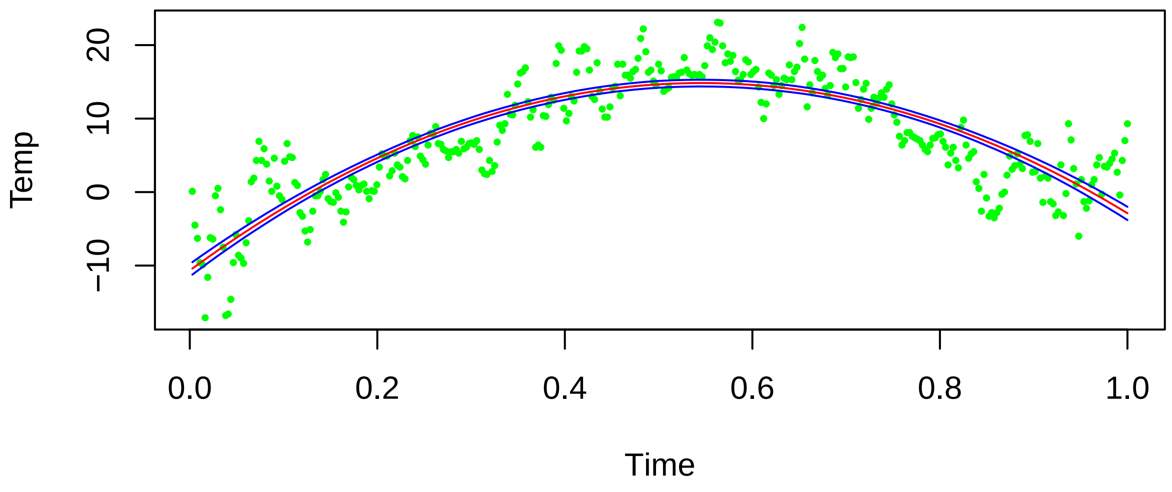
```
## Estimate of beta posterior:  -10.6654 93.52609 -85.74625
```

```
## Estimate of sigma posterior:  15.97545
```

Produce a scatter plot of the temperature data and overlay a curve for the posterior mean of the regression function $f(time) = \beta_0 + \beta_1.time + \beta_2.time^2$ computed for every value of time.



Also overlay the lower 5% and upper 95% posterior credible interval for $f(time)$.



Does the interval contain most of the data points? should it?

The intervals does not cover many of the data points, due to the fact that it does not capture the uncertainty from the noise, but captures the uncertainty about the trend curve instead.

(d)

Use the simulations in part (c) to simulate from the posterior distribution of the time with the highest temperature.

To find the highest temperature, we compute the following.

Let

$$\frac{\partial f(time)}{\partial time} = 0$$

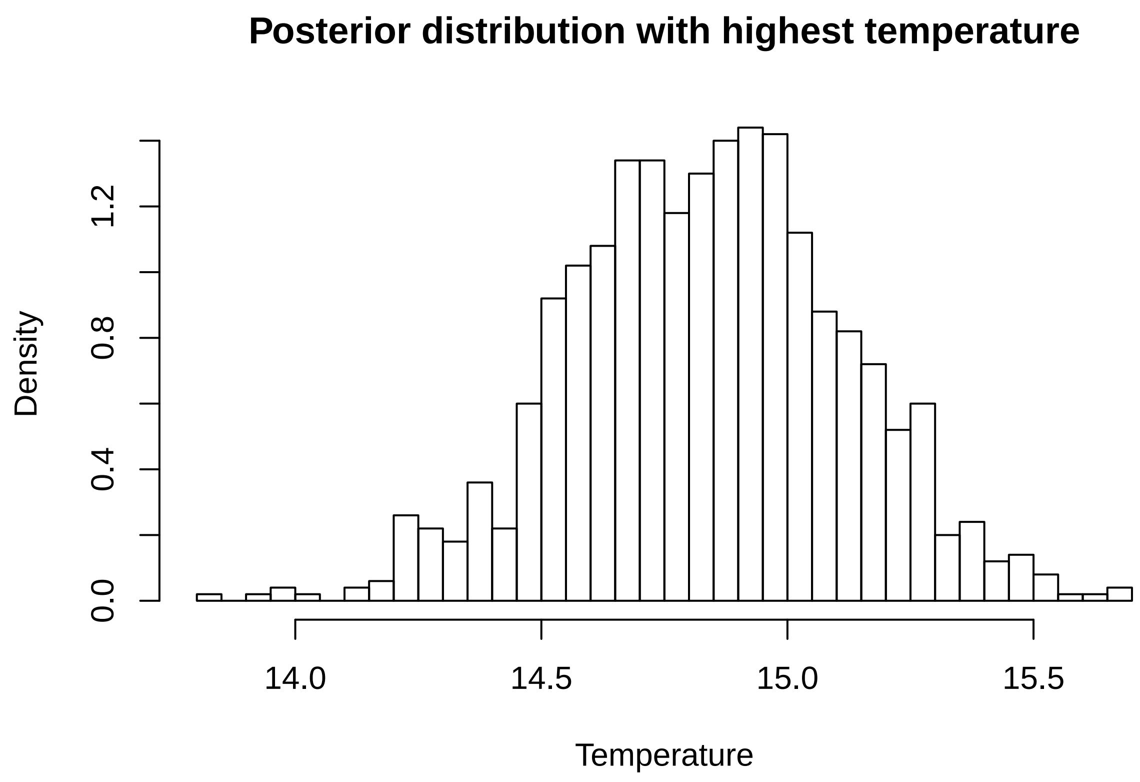
Therefore,

$$\beta_1 + 2.\beta_2.time = 0$$

so,

$$time = -\frac{\beta_1}{2.\beta_2}$$

This formula give the *time* where *f(time)* is maximal.



It can be seen from a histogram that the mean of the highest temperature is around 14.5 to 15.2.

(e)

Suggest a suitable prior that estimate a polynomial model of order 7 where the higher order terms may not be needed to avoid overfitting.

We suggest the following parameters:

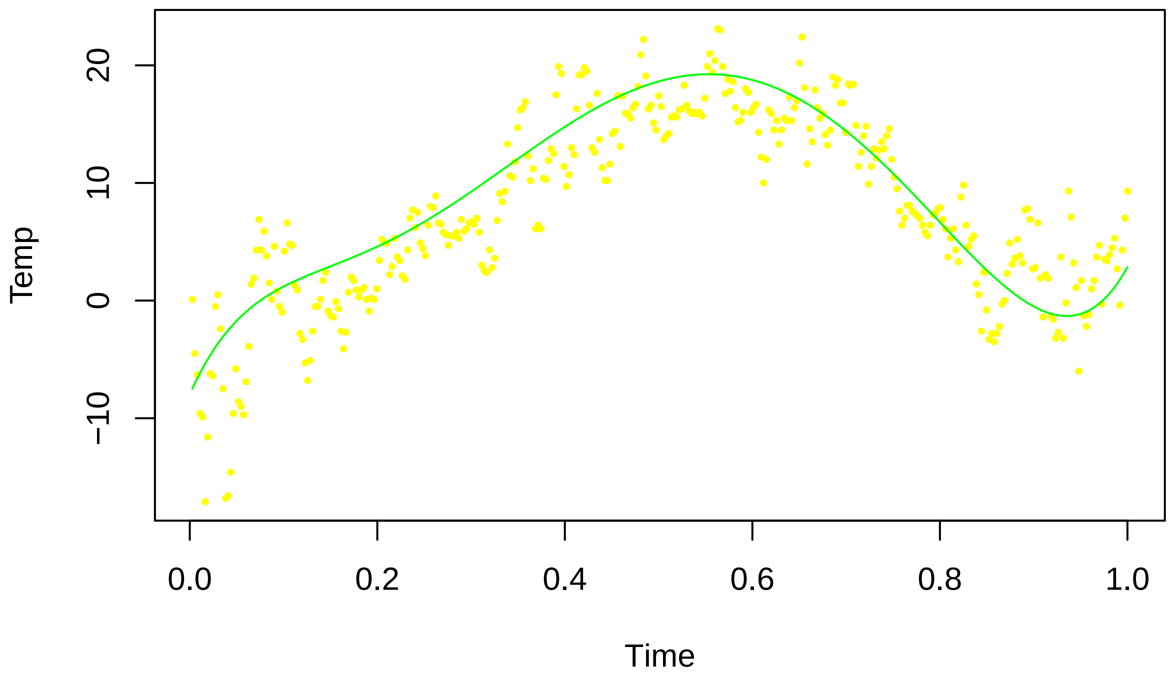
- μ_0: coefficients of polynomial model of order 7, which are -10.2781,179.6519, -1346.0931, 5819.9955, -12498.3189, 14252.8719, -8690.8270, 2299.0738

- Ω_0: a diagonal matrix of 1s with 8 dimensions
- ν_0: 366
- σ_0^2: 11 which is a square of residual standard error of our model

The polynomial model of order 7 is fitted below.

```
##
## Call:
## lm(formula = temp ~ time + I(time^2) + I(time^3) + I(time^4) +
##     I(time^5) + I(time^6) + I(time^7), data = temp_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.7907  -1.8389  -0.2099   1.8929   9.8972
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -10.278      1.471  -6.985 1.39e-11 ***
## time           179.652     52.861   3.399 0.000754 ***
## I(time^2)    -1346.093    608.165  -2.213 0.027502 *
## I(time^3)     5819.995   3120.584   1.865 0.062995 .
## I(time^4)   -12498.319   8217.029  -1.521 0.129136
## I(time^5)    14252.872  11551.406   1.234 0.218063
## I(time^6)    -8690.827   8238.680  -1.055 0.292191
## I(time^7)     2299.074   2341.533   0.982 0.326829
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.368 on 358 degrees of freedom
## Multiple R-squared:  0.8221, Adjusted R-squared:  0.8186
## F-statistic: 236.3 on 7 and 358 DF,  p-value: < 2.2e-16
```

To check the sensibility of suggested parameters, a plot with a regression curve is shown below.



It can be seen that the choice of parameters is reasonable.

Question 2: Posterior approximation for classification with logistic regression

In this task, the dataset *WomenWork.dat* is given, it contains $n = 200$ observations and 9 variables. An extract is below.

```
##   Work Constant HusbandInc EducYears ExpYears ExpYears2 Age NSmallChild
## 1    1        1   22.39494        12        7      0.49  43           0
## 2    0        1    7.23200         8       10      1.00  34           0
## 3    1        1   18.27199        12        4      0.16  41           1
## 4    0        1   28.06900        14        2      0.04  43           0
## 5    1        1   23.80000        12       24      5.76  45           0
## 6    0        1   96.00000        17        1      0.01  34           1
##   NBigChild
## 1         3
## 2         7
## 3         5
## 4         2
## 5         1
## 6         2
```

(a)

The glm model is as below.

```
##
## Call:  glm(formula = Work ~ 0 + ., family = binomial, data = WomenWork)
##
## Coefficients:
##    Constant   HusbandInc    EducYears     ExpYears    ExpYears2
##     0.64430     -0.01977      0.17988      0.16751     -0.14436
##         Age  NSmallChild    NBigChild
##    -0.08234     -1.36250     -0.02543
##
## Degrees of Freedom: 200 Total (i.e. Null);  192 Residual
## Null Deviance:       277.3
## Residual Deviance: 222.7     AIC: 238.7
```

(b)

The code for the prior,likelihood and posterior as follows.

```
#response variable is the work
y<-as.vector(WomenWork[,1])
#all the other 8 variables comes in X
X<-as.matrix(WomenWork[,2:9])
nPara<-ncol(X)
#given that tau =10
tau<-10
mu_zero<-as.vector(rep(0,nPara))
sigma0 <- tau^2*diag(nPara)

##prior function
priorprob <- function(beta,sigma0) {
  prior <- dmvnorm(beta, mean=matrix(rep(0,8),ncol = 1), sigma=sigma0, log=TRUE)
  return(prior)
}

##likelihood function
likeprob <- function(beta, y, X) {
  LP <- X%*%beta
  nw<-1 + exp(LP)
  logliklihood <- sum(LP*y - log(nw))
  if (abs(logliklihood) == Inf) {
    logliklihood = -20000
  }
  return(logliklihood)
}

##posterior function
postprob<-function(beta, y, X, mu0, sigma0) {
  beta <- as.vector(beta)
  ##posterior is sum of prior and likelyhood
  logposter <- priorprob(beta, sigma0) + likeprob(beta, y, X)
  return(logposter)
}

##Optim function

OptimResults <- optim(rep(0,8), postprob,
                      y = y, X = X, mu0 = mu0, sigma0 = sigma0,
                      method="BFGS", control=list(fnscale=-1), hessian=TRUE)
```

The values of beta are:

```
## [1]  0.62672884 -0.01979113  0.18021897  0.16756670 -0.14459669 -0.08206561
## [7] -1.35913317 -0.02468351
```

The negative inverse hessian is:

```
##              [,1]          [,2]          [,3]          [,4]          [,5]
## [1,]  2.266022568  3.338861e-03 -6.545121e-02 -1.179140e-02  0.0457807243
## [2,]  0.003338861  2.528045e-04 -5.610225e-04 -3.125413e-05  0.0001414915
## [3,] -0.065451206 -5.610225e-04  6.218199e-03 -3.558209e-04  0.0018962893
## [4,] -0.011791404 -3.125413e-05 -3.558209e-04  4.351716e-03 -0.0142490853
## [5,]  0.045780724  1.414915e-04  1.896289e-03 -1.424909e-02  0.0555786706
## [6,] -0.030293450 -3.588562e-05 -3.240448e-06 -1.340888e-04 -0.0003299398
## [7,] -0.188748354  5.066847e-04 -6.134564e-03 -1.468951e-03  0.0032082535
## [8,] -0.098023929 -1.444223e-04  1.752732e-03  5.437105e-04  0.0005120144
##               [,6]          [,7]          [,8]
## [1,] -3.029345e-02 -0.1887483542 -0.0980239285
## [2,] -3.588562e-05  0.0005066847 -0.0001444223
## [3,] -3.240448e-06 -0.0061345645  0.0017527317
## [4,] -1.340888e-04 -0.0014689508  0.0005437105
## [5,] -3.299398e-04  0.0032082535  0.0005120144
## [6,]  7.184611e-04  0.0051841611  0.0010952903
## [7,]  5.184161e-03  0.1512621814  0.0067688739
## [8,]  1.095290e-03  0.0067688739  0.0199722657
```

95% credible interval for the variable *NSmallChild* is:

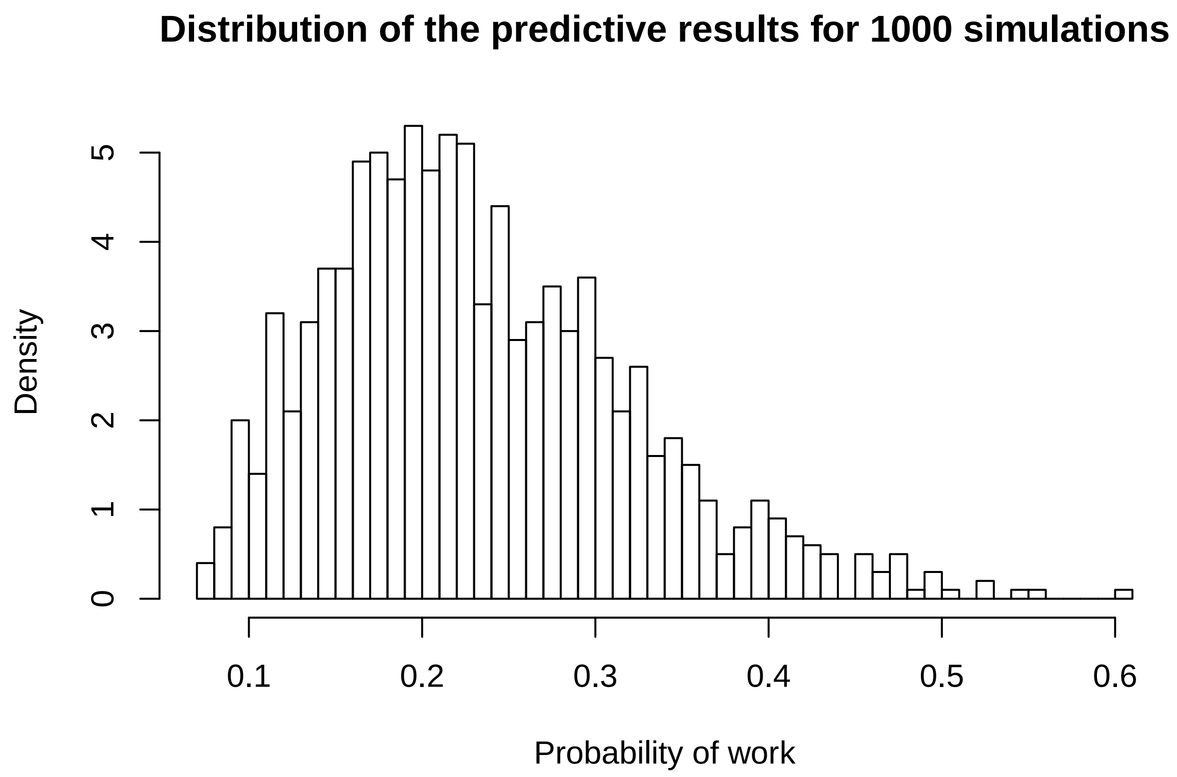
```
##      lower      upper
## -2.1214250 -0.5968414
```

Would you say that this feature is an inportant determinant of the probability that a women works?

The credible interval is far away from zero, which indicates that the variable is an important features.

(c)

Simulate the predictive distribution for the *Work* variable for a 40 year old woman, with two children (3 and 9 years old), 8 years of education, 10 years of experience, and a husband with an income of 10.



Since the histogram is left skewed we can say that the prbability of a women to work is less.

Contribution

Results and comments presented in this report have been developed and discussed together by the members of the group.

Appendix

```
###Question 1###
#(a)
library(geoR) #for rinvchisq
library(mvtnorm) #mvtnorm

#Load and read data
temp_data <- read.table("TempLinkoping2016.txt", header = TRUE, dec = ".")
head(temp_data)

#preliminary steps
lm <- lm(temp ~ time + I(time^2), data = temp_data)
summary(lm)

#Hyperparameters:
```

```r
mu_0 <- lm$coefficients #coefficents of the modelwith 3vars:intercepts,time,time^2
omega_0 <- diag(3) #diagonal matrix of 1
nu_0 <- 6 #start with 3
sigmasq_0 <- 16 #residual standard error(standard deviation)^2 (4.017^2 =16.1362)

#(b)
#betas_prior of 1 draw
set.seed(12345)
sigma_prior <- rinvchisq(1, df = nu_0, scale = sigmasq_0)
beta_prior <- rmvnorm(1, mean = mu_0, sigma = sigma_prior*solve(omega_0))
reg_curve <- beta_prior[1] + beta_prior[2]*temp_data$time + beta_prior[3]*temp_data$time^2
plot(x=temp_data$time, y=temp_data$temp,pch = 16,cex=0.5, col = "blue",
     xlab = "Time", ylab = "Temp", main = "")
lines(x=temp_data$time, y=reg_curve, col = "red")

#betas_prior of 10 draws with regression curve
set.seed(1234)
plot(x=temp_data$time, y=temp_data$temp,pch = 16, cex=0.5, col = "blue",
     xlab = "Time", ylab = "Temp", main = "")

res <- list()
for(i in 1:10){
  sigma_prior_n <- rinvchisq(1, df = nu_0, scale = sigmasq_0)
  beta_prior_n <- rmvnorm(1, mean = mu_0, sigma = sigma_prior_n*solve(omega_0))
  res[[i]] <- beta_prior_n
  reg_curve_n <- beta_prior_n[1]+beta_prior_n[2]*temp_data$time
                + beta_prior_n[3]*temp_data$time^2
  lines(x=temp_data$time, y=reg_curve_n, col = "red")
}

#betas_prior of 10000 draws with average
set.seed(1234567)
suggest_prior <- function(n, nu, sigmasq, mu, omega){
  res <- matrix(ncol = length(mu), nrow = n)
  for(i in 1:n){
    sigma_prior_n <- rinvchisq(1, df = nu, scale = sigmasq)
    res[i,] <- rmvnorm(1, mean = mu, sigma = sigma_prior_n*solve(omega))
  }
  return(list(beta = res, sigma2 = sigma_prior_n))
}
myprior <- suggest_prior(n=1000, nu=nu_0, sigmasq=sigmasq_0, mu=mu_0, omega=omega_0)
beta_prior_avg <- colMeans(myprior$beta)
sigma2_prior_avg <- mean(myprior$sigma2)
cat("Estimate of beta prior: ", beta_prior_avg, "\n")
cat("Estimate of sigma prior: ", sigma2_prior_avg, "\n")

#Betas distribution, good if it is normal distributed
# par(mfrow = c(1,3))
# hist(myprior$beta[,1],breaks=50,probability=TRUE,xlab="beta1",main="Distribution of beta1")
# hist(myprior$beta[,2],breaks=50,probability=TRUE,xlab="beta2",main ="Distribution of beta2")
# hist(myprior$beta[,3],breaks=50,probability=TRUE,xlab="beta3",main="Distribution of beta3")

#(c)
```

```
#joint posterior distributions
#X is dataframe of intercept, time, time^2
X <- cbind(rep(1, nrow(temp_data)), temp_data$time, temp_data$time^2)
y <- temp_data$temp
n = nrow(temp_data)

beta_hat <- solve(t(X)%*%X) %*% t(X) %*% y
mu_n <- solve(t(X)%*%X+omega_0)%*%(t(X)%*%X%*%beta_hat+omega_0%*%mu_0 )
omega_n <- t(X) %*% X + omega_0
nu_n <- nu_0 + n
term <- as.vector(t(y)%*%y+t(mu_0)%*%omega_0%*%mu_0)-(t(mu_n)%*%omega_n%*%mu_n)
sigmasq_n <- as.vector((1/nu_n)*(nu_0*sigmasq_0 + term))

#betas_posterior of single draw
# sigma_pos <- rinvchisq(1, df = nu_n, scale = sigmasq_n)
# beta_pos <- rmvnorm(1, mean = mu_n, sigma = sigma_pos*solve(omega_n))

#betas_posterior of 1000 draws
myposterior <- suggest_prior(n=1000,nu=nu_n,sigmasq=sigmasq_n,mu=mu_n,omega=omega_n)
beta_pos_n <- colMeans(myposterior$beta)
sigma2_pos_n <- mean(myposterior$sigma2)

#A curve for the posterior mean of the regression function
reg_curve_pos <- beta_pos_n[1]+beta_pos_n[2]*temp_data$time+beta_pos_n[3]*temp_data$time^2

#scatter plot
par(mfrow = c(1,1))
plot(x=temp_data$time, y=temp_data$temp,pch = 16, cex=0.5, col = "green",
     xlab = "Time", ylab = "Temp", main = "")
#regression curve
lines(x=temp_data$time, y=reg_curve_pos, col = "red")

#Posterior credible interval
b0 <- myposterior$beta[,1]
b1 <- myposterior$beta[,2]
b2 <- myposterior$beta[,3]

temp_posterior <- function(){
  n = 1000
  k = 366
  temp_pos <- matrix(data= NA, nrow = n, ncol = k)
  for(j in 1:k){
    for(i in 1:n){
      temp_pos[i,j] <- b0[i] + b1[i]*temp_data$time[j] + b2[i]*(temp_data$time[j])^2
    }
  }
  return(temp_pos)
}
pos <- temp_posterior(k = 366, iter= 1000)

interval <- function(){
  k = 366
  qlower <- numeric(k)
```

```r
  qupper <- numeric(k)
  for(i in 1:k){
    qlower[i] <- quantile(pos[,i], probs = c(0.05, 0.95))[1] #[1] is 5%
    qupper[i] <- quantile(pos[,i], probs = c(0.05, 0.95))[2] #[2] is 95%
  }
  return(data.frame(qlower, qupper))
}
credibleI <- interval()

#5% and 95% interval curve
lines(x=temp_data$time, y=credibleI[,1], col = "blue")
lines(x=temp_data$time,y=credibleI[,2], col = "blue")

#d
#max_time = df/dt=0; so time=-beta1/2beta2
max_time <- -(beta_pos_n[2])/(2*beta_pos_n[3])
temp_pos_max = numeric(1000)
for(i in 1:1000){
  temp_pos_max[i] <- myposterior$beta[i,1]+(myposterior$beta[i,2])*max_time
                     +(myposterior$beta[i,3])*max_time^2
}
hist(temp_pos_max, breaks = 50, probability = TRUE,
     xlab = "Temperature",
     main = "Posterior distribution with highest temperature")

#(e)
lm7 <- lm(temp~time+I(time^2)+I(time^3)+I(time^4)+I(time^5)+I(time^6)+I(time^7),
          data=temp_data)
summary(lm7)
mu7 <- lm7$coefficients
nu7 <- 366
sigmasq7 <- 11 #3.368^2
omega7 <- diag(8)

set.seed(12345)
sigma_prior7 <- rinvchisq(1, df = nu7, scale = sigmasq7)
beta_prior7 <- rmvnorm(1, mean = mu7, sigma = sigma_prior7*solve(omega7))
reg_curve7 <- (beta_prior7[1]+beta_prior7[2]*temp_data$time+beta_prior7[3]*temp_data$time^2
                   + beta_prior7[4]*temp_data$time^3+beta_prior7[5]*temp_data$time^4
                   + beta_prior7[6]*temp_data$time^5+beta_prior7[7]*temp_data$time^6
                   + beta_prior7[8]*temp_data$time^7)
plot(x=temp_data$time, y=temp_data$temp,pch = 16, cex=0.5, col = "yellow",
     xlab = "Time", ylab = "Temp", main = "")
lines(x=temp_data$time, y=reg_curve7, col = "black")

############################
#####Question 2#####
###2.a
library(mvtnorm) #use for dmvnorm()

##Read the data
WomenWork <-read.table("WomenWork.dat",header=TRUE)
head(WomanWork)
```

```r
#Make the logistic regression model
glmModel<-glm(Work ~ 0 + ., data = WomenWork, family = binomial)
glmModel

###2.b
#response variable is the work
y<-as.vector(WomenWork[,1])
#all the other 8 variables comes in X
X<-as.matrix(WomenWork[,2:9])
nPara<-ncol(X)
#given that tau =10
tau<-10

mu_zero<-as.vector(rep(0,nPara))
sigma0 <- tau^2*diag(nPara)

##prior function
priorprob <- function(param,sigma0) {
  prior <- dmvnorm(param, mean=matrix(rep(0,8),ncol = 1), sigma=sigma0, log=TRUE)
  return(prior)
}

##likelihood function
likeprob <- function(param, y, X) {
  LP <- X%*%param
  nw<-1 + exp(LP)
  logliklihood <- sum(LP*y - log(nw))
  if (abs(logliklihood) == Inf) {
    logliklihood = -20000
  }
  return(logliklihood)
}

##posterior function
postprob<-function(param, y, X, mu0, sigma0) {
  param <- as.vector(param)
  ##posterior is sum of prior and likelyhood
  logposter <- priorprob(param, sigma0) + likeprob(param, y, X)
  return(logposter)
}

##Optim function
OptimResults <- optim(rep(0,8), postprob,
                      y = y, X = X, mu0 = mu0, sigma0 = sigma0,
                      method="BFGS", control=list(fnscale=-1), hessian=TRUE)

##Optimal values for param are
beta_tilde <- OptimResults$par

##Posterior covariance matrix is -inv(Hessian)
postCov <- -solve(OptimResults$hessian)

##Finding the 95% credibility interval for nSmallChild
```

```r
sd <- sqrt(abs(postCov[7,7]))
credibleI <- c(lower=beta_tilde[7]-1.96*sd, upper = beta_tilde[7]+1.96*sd)

###2.c
prediction_value<-function(n) {
  data <- c(1, 10, 8, 10, 1, 40, 1, 1)
  sigma <-postCov
  mu <-OptimResults$par
  probWork <- numeric(n)
  for(i in 1:n){
    beta <- as.vector(rmvnorm(1, mu, sigma))
    probWork[i] <- (exp(t(data)%*%beta))/(1+exp(t(data)%*%beta))
  }
  return(probWork)
}
work_pred<-prediction_value(1000)
hist(work_pred, probability = TRUE, breaks = 50,
     xlab = "Probability of work", main = "Distribution of the predictive results for 1000 simulation")
```

732A91 Bayesian Learning
Report Lab 3

Thi Pham (thiph169) and Priya Kurian Pullolickal (priku577)

12 May 2018

1. Normal model, mixture and normal model with semi-conjugate prior.

The data *rainfall.dat* is used in this assignment. It consists of daily records of precipitation at Snoqualmie Falls between 1948 to 1983. An extract of data below:

```
##    V1
## 1 136
## 2  17
## 3   1
## 4  34
## 5   2
## 6  59
```

(a) Normal model

(i) Implement Gibbs sampler

The Gibbs sampler is implemented below which simulates from the joint posteriors whith semi-conjugate prior. The formulae of posterior are (lexture 1 and 7):

$$\mu|\sigma^2, y \sim N(\mu_n, \tau_n^2)$$
$$\sigma^2|\mu, y \sim Inv - \chi^2(\nu_n, \frac{\nu_0\sigma_0^2 + \sum_{i=1}^n (x_i - \mu)^2}{n + \nu_0})$$

where

$$\mu_n = w\bar{x} + (1 - w)\mu_0$$
$$w = \frac{\frac{n}{\sigma^2}}{\frac{n}{\sigma^2} + \frac{1}{\tau_0^2}}$$

and

$$\frac{1}{\tau_n^2} = \frac{n}{\sigma^2} + \frac{1}{\tau_0^2}$$

```r
#Initial parameters
mu0 <- 1
sigmasq0 <- 1
v0 <- 1
tausq0 <- 10
nDraws = 1000
```

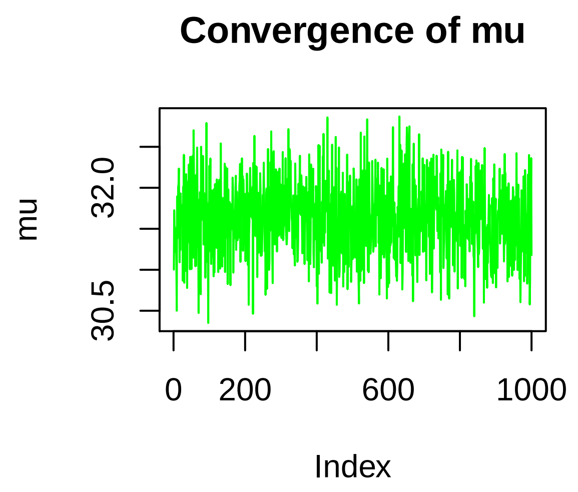
```r
gibbSampler <- function(iter, data, mu0, tausq0, sigmasq0){
  results <- matrix(0, nrow = nDraws, ncol = 2)
  colnames(results) <- c("mu", "sigmasq")
  n = length(data)
  mu = mu0
  vn = v0 + n
  for(i in 1:nDraws){
    #compute sigmasq
    sigmasq <- rinvchisq(1, df = vn, scale = (v0*sigmasq0 + sum((data-mu)^2))/vn)
    results[i,2] <- sigmasq
    #compute mu
    tausqInv <- (n/sigmasq)+(1/tausq0)
    tausqn <- 1/tausqInv
    w <- (n/sigmasq)/tausqInv
    mun <- w*mean(data) + (1-w)*mu0
    mu <- rnorm(1, mean = mun, sd = sqrt(tausqn))
    results[i,1] <- mu
  }
  return(results)
}
gibbs <- gibbSampler(iter=nDraws, data=data, mu0=mu0, tausq0=tausq0, sigmasq0=sigmasq0)
```

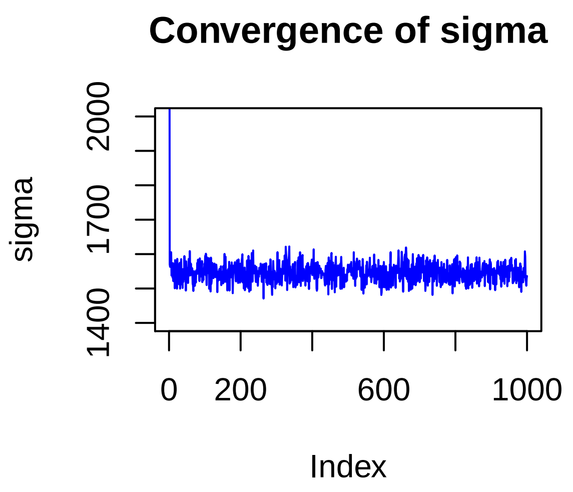
The estimation of μ and σ^2 are:

```
##         mu    sigmasq
##   31.62337 1547.65165
```

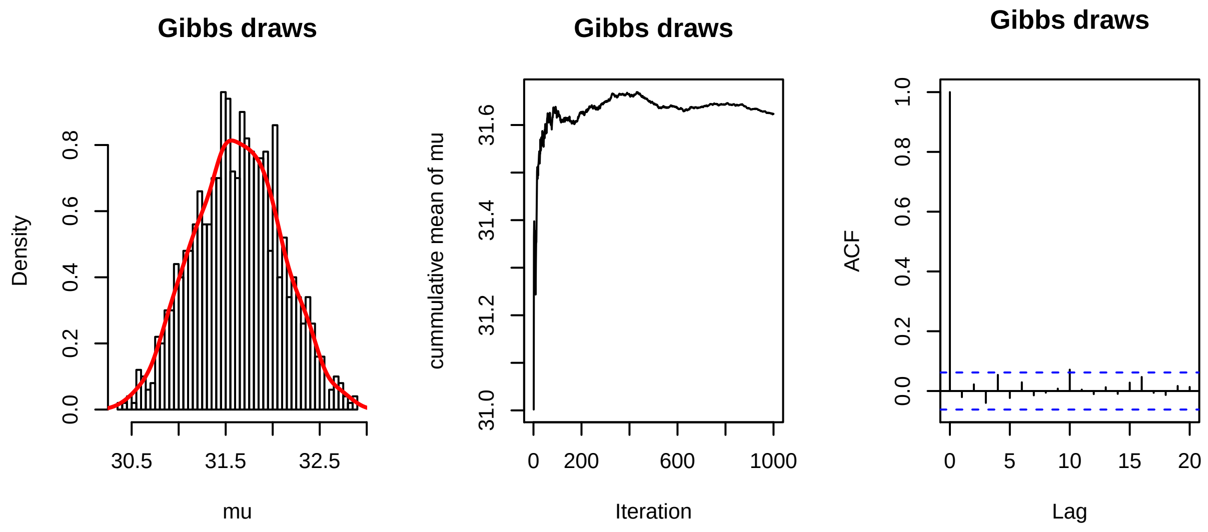
(ii) Graphical methods

Plots below show the convergence of μ and σ^2. It can be seen that μ converges to the a value between 31-32 and σ^2 converges to 1550.





More graphs below which show how μ is distributed. It can be seen that the data is corelated.



(b) Mixture model

In this part, the mixture of normal model is executed (codes given in lecture 7). The convergence of mixture density model after 100 iterations shows below.

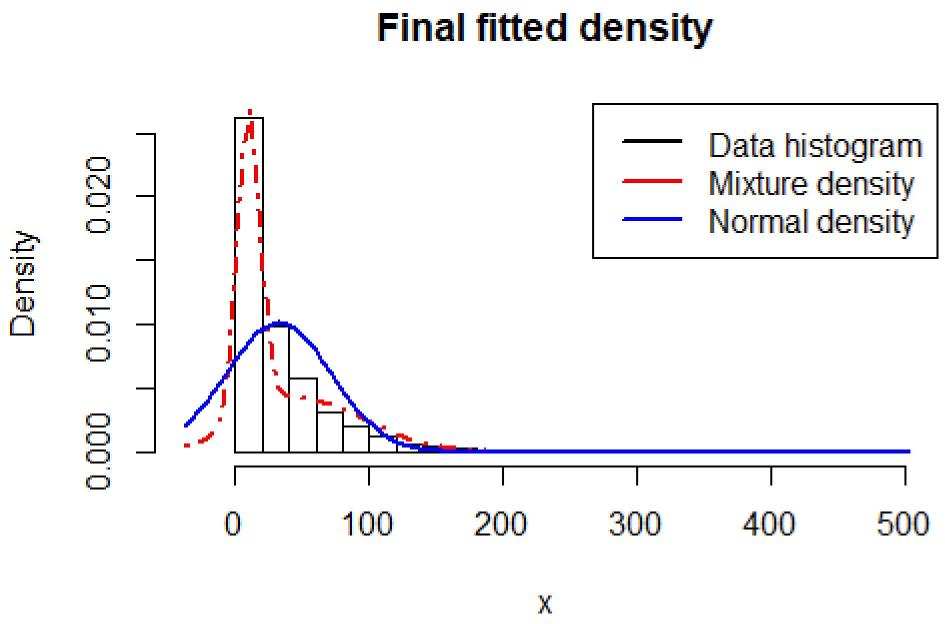


Figure 1:

(c) Graphical comparison

Plot the following densities in one figuure:
1. A histogram or kernel density estimate of the data.

2. Normal density $N(\mu, \sigma^2)$ in part(a).
   The normal density of 1000 random samples with $\mu = 31.56618$ and $\sigma^2 = 1549.10161$, which obtain from part(a).

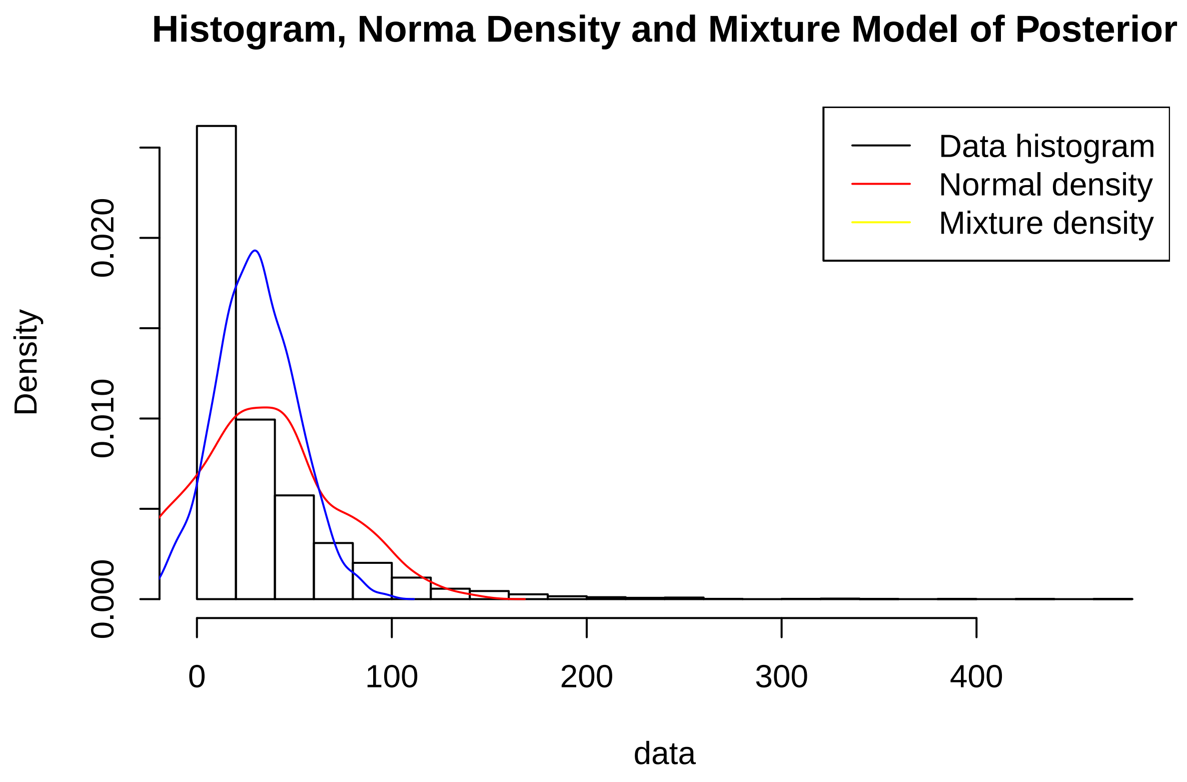
3. Mixture of normal density $p(y_i|\mu, \sigma^2, \pi) in part(b)$.

The mixture of normal density of 1000 random samples obtained from part(b) with the following formula:

$$\pi * N(\mu_1, \sigma_1^2 + (1 - \pi) * N(\mu_2, \sigma_2^2))$$

Use the posterior mean value for all the parameters.

```
#Parameters from part (a) and (b)
muA = meanGibbs[1]
sigma2A = meanGibbs[2]
muB=mu
sigma2B=sigma2
piB=pi

normalDen<-rnorm(1000, mean = muA, sd = sqrt(sigma2A))
mixtureDen<- (pi[1])*rnorm(1000, mean = muB[1], sd = sqrt(sigma2B[1]))+
  (pi[2])*rnorm(1000, mean = muB[2], sd = sqrt(sigma2B[2]))

par(mfrow = c(1, 1))
hist(data, freq = FALSE, main='Histogram, Norma Density and Mixture Model of Posterior',  breaks = 30)
lines(density(normalDen), col = "red", lwd = 1)
lines(density(mixtureDen), col = "blue", lwd = 1)
legend("topright", box.lty = 1, legend = c("Data histogram","Normal density", "Mixture density"), col=c
```



It can be seen that both of the distributions are not good for modelling the data.

2. Time series models in Stan

(a)

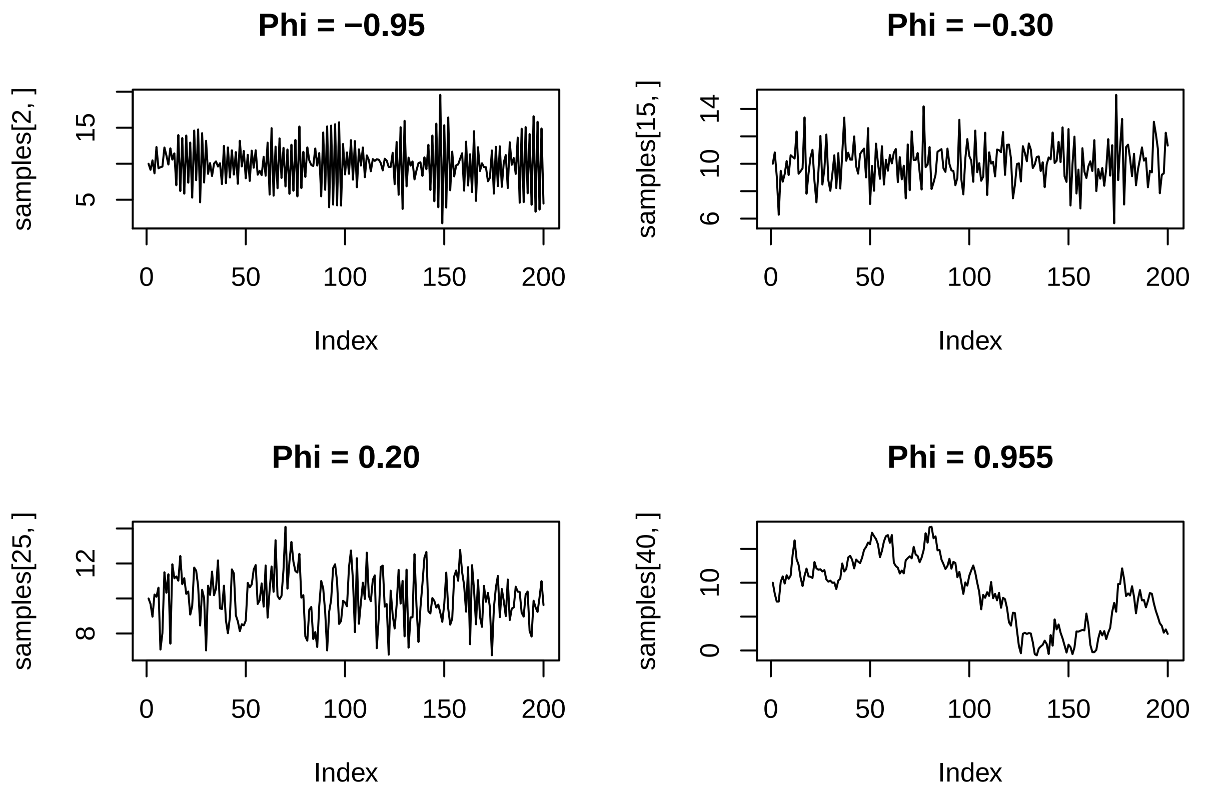
Write a function in R that simulates data from the AR(1) process

$$x_t = \mu + \phi(x_{t-1} - \mu) = \epsilon_t, \epsilon_t \sim N(0, \sigma^2)$$

```
# Requirement
rm(list=ls())
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())

# Prior
T = 200
mu = 10
sigma = sqrt(2)
phi = 0.5

# Model
arStanModel = '
```

```
data{
int<lower=0> N;
vector[N] x;
}
parameters{
real mu;
real<lower=0> sigma;
real phi;
}
model{
for(t in 2:N)
x[t] ~ normal(mu + phi*(x[t-1]-mu), sigma);
}
'
#AR process
arProcess <- function(phi, init, sigma, T){
  y <- rep(0, T)
  y[1] <- init
  for(t in 2:T){
    y[t] <- mu + phi*(y[t-1]-mu) + rnorm(1, 0, sigma)
  }
  return(y)
}

#Simulation of AR process
phis = seq(-1, 1, 0.05)
samples <- matrix(0, nrow = length(phis), ncol = T)
for(i in 1:length(phis)){
  samples[i,] <- arProcess(phis[i], mu, sigma, T)
}
```

Below are plots of 4 different values of ϕ between[-1, 1]; $\phi = -0.95, -0.3, 0.2, 0.955$. It can be seen that the curve is smoother when ϕ value is high.



(b)

(i)

Implementation of Stan-code in part(a) for $\phi = 0.3$ and $\phi = 0.95$. The posterior mean, 95% credible intervals and the number of effective posterior sample for the three inferred parameters are:

Posterior mean

$\phi = 0.3$

```
##        mean-chain:1 mean-chain:2 mean-chain:3 mean-chain:4 mean-all chains
## mu        9.9022538     9.904982    9.8996723    9.9008711       9.9019449
## sigma     1.5352551     1.539356    1.5359160    1.5290467       1.5348935
## phi       0.3590849     0.362436    0.3658191    0.3588313       0.3615428
## lp__   -183.3009852  -183.394613 -183.4623789 -183.2216045    -183.3448954
```

$\phi = 0.95$

```
##        mean-chain:1 mean-chain:2 mean-chain:3 mean-chain:4 mean-all chains
## mu       11.4837137   32.8910213   11.1718342   12.5074429      17.0135030
## sigma     1.4626159    1.4221244    1.4655517    1.4645298       1.4537055
## phi       0.9417743    0.9739345    0.9397024    0.9360139       0.9478563
## lp__   -174.0371338 -176.2110896 -174.0677048 -173.8565537    -174.5431205
```

95% Credible intervals

$\phi = 0.3$

```
## 95% CI of mu: 9.561357 10.2552
```

```
## 95% CI of sigma2: 1.389246 1.694134
```

```
## 95% CI of phi: 0.2237665 0.4966463
```

$\phi = 0.95$

```
## 95% CI of mu: 3.349978 48.01825
```

```
## 95% CI of sigma2: 1.325359 1.614094
```

```
## 95% CI of phi: 0.8786014 1.00419
```

Number of effective posterior sample

$\phi = 0.3$

```
##       mu
## 3402.324
```

```
##    sigma
## 3447.136
```

```
##      phi
## 3203.461
```

$\phi = 0.95$

```
##       mu
## 5.126568
```

```
##   sigma
## 58.8348
```

```
##       mu
## 5.126568
```

True values

$\phi = 0.3$

```
## Mu: 9.901945
```

```
## Sigma2: 1.534893
```

```
## Phi: 0.3615428
```

$\phi = 0.95$

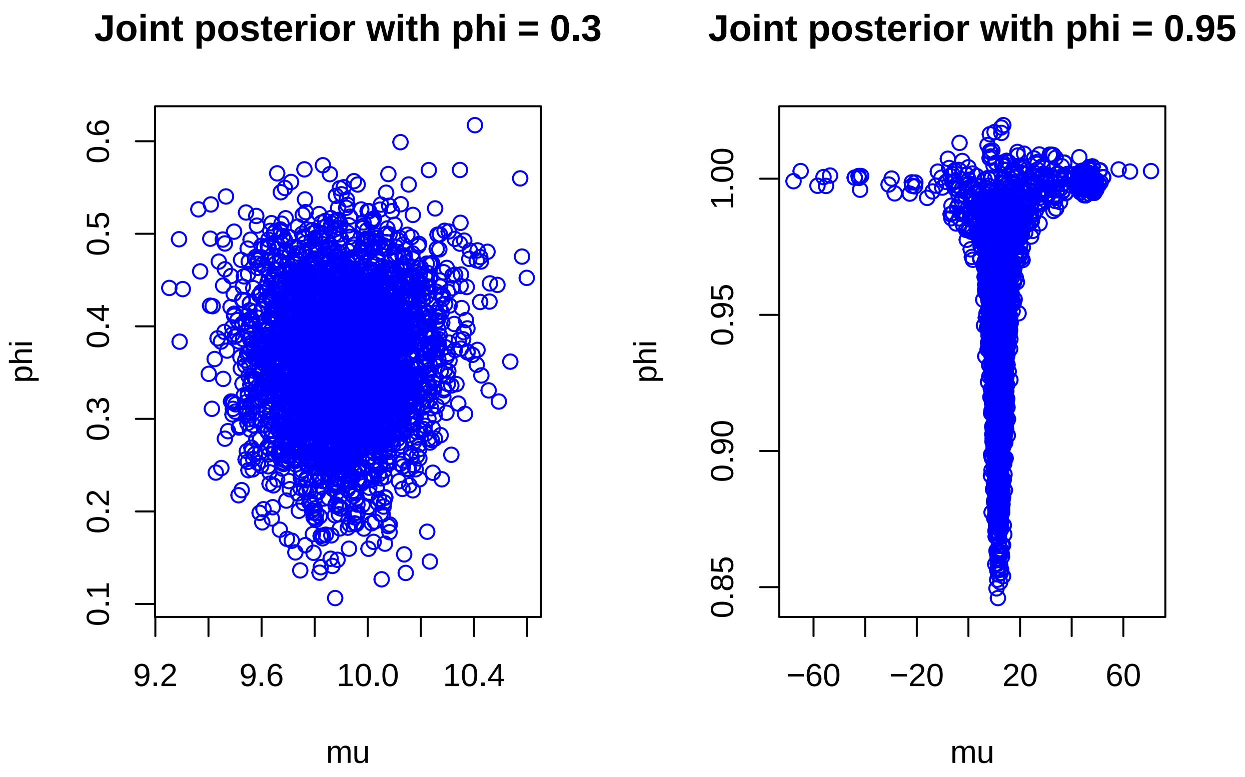
```
## Mu: 17.0135
```

```
## Sigma: 1.453705
```

```
## Phi: 0.9478563
```

(ii)

The convegence of μ and ϕ are evaluated as plots below:



In plot 2, μ and ϕ are more correlated than in plot 1. Also, it conveges to $\mu = 10$.

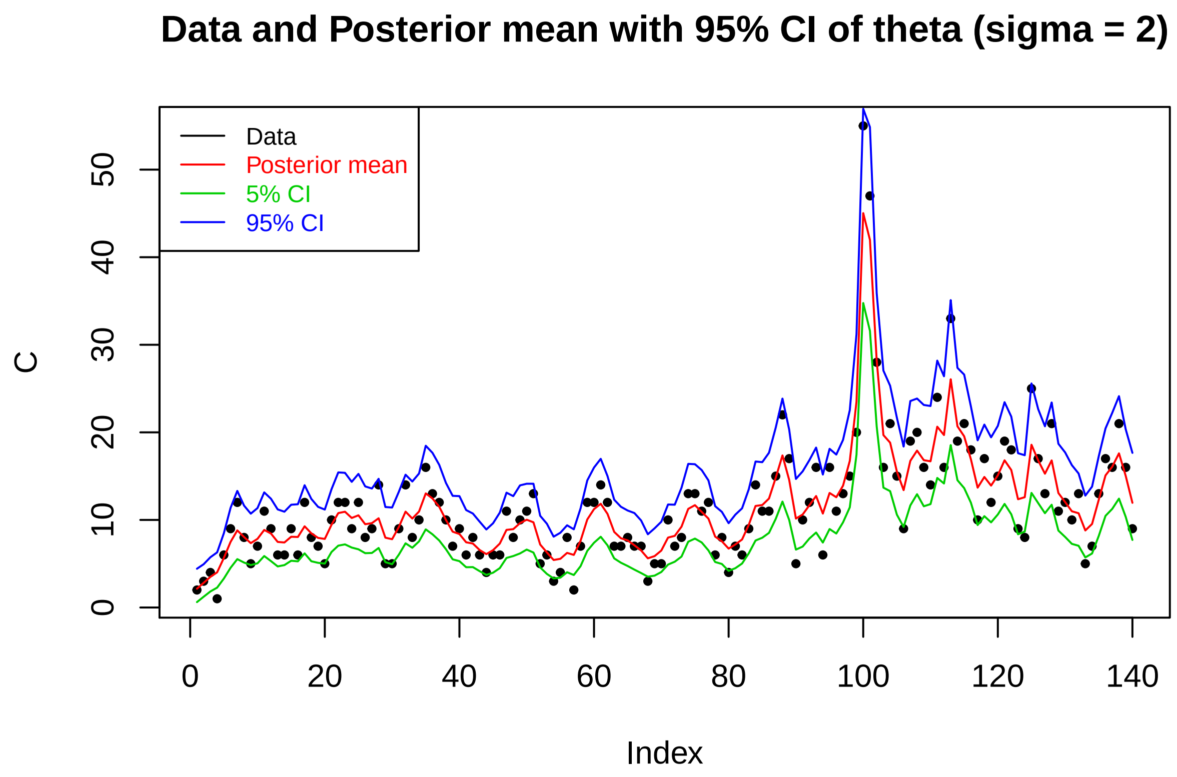
(c)

In this part, the data *campy.dat* is used. It contains the number of cases of campylobacter infections in the north of province Quebec (Canada). An extract of data are:

```
##   c
## 1 2
## 2 3
## 3 4
## 4 1
## 5 6
## 6 9
```

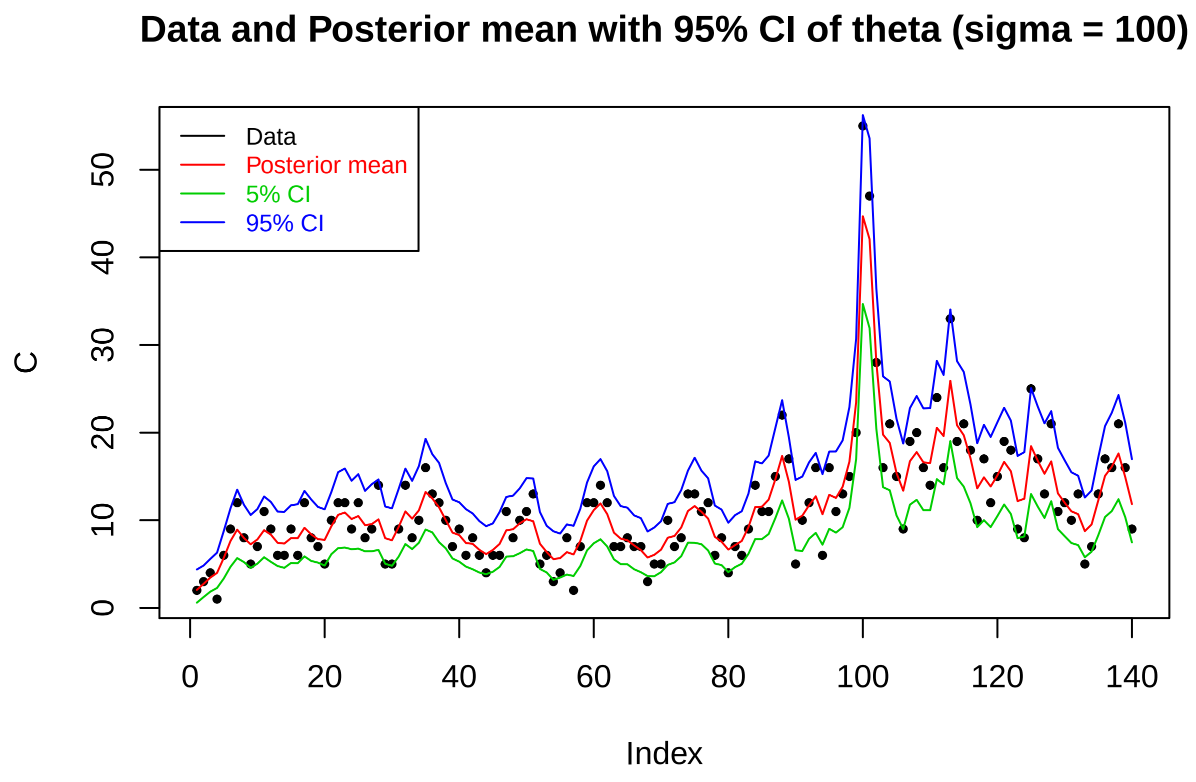
The implementation and estimation of the model in Stan are performed. The plot that contains both the data and the posterior mean and 95% credible intervals for the latent intensity $\theta_t = exp(x_t)$ given below.

The prior parameters set as: $\mu = 10, \sigma^2 = 1.4$ and $\phi = 0.2$



(d)

To see if θ_t effects smoothness of the curve (data). We change the prior for σ^2 from 1.4 to 100 and all other parameters are the same as part(d). Results as plot below.



It can be seen the curve slightly change, but there is not clear that the smoothness of a curve is affected by σ^2.

Contribution

Results and comments presented in this report have been developed and discussed together by the members of the group.

Appendix

Question 1

```
#####Question 1 #####
#####1 Mormal model
###(i)
#library needed
library(geoR) #require for rinvchsq()

#Reading the data
rainfallData <- read.table("rainfall.dat", header = FALSE)
head(rainfallData)
```

```r
data <- rainfallData$V1

#a.i
#Initial parameters
mu0 <- 1
sigmasq0 <- 1
v0 <- 1
tausq0 <- 10
nDraws = 1000

gibbSampler <- function(iter, data, mu0, tausq0, sigmasq0){
  results <- matrix(0, nrow = nDraws, ncol = 2)
  colnames(results) <- c("mu", "sigmasq")

  n = length(data)
  mu = mu0
  vn = v0 + n

  for(i in 1:nDraws){
    #compute sigmasq
    sigmasq <- rinvchisq(1, df = vn, scale = (v0*sigmasq0 + sum((data-mu)^2))/vn)
    results[i,2] <- sigmasq

    #compute mu
    tausqInv <- (n/sigmasq)+(1/tausq0)
    tausqn <- 1/tausqInv
    w <- (n/sigmasq)/tausqInv
    mun <- w*mean(data) + (1-w)*mu0

    mu <- rnorm(1, mean = mun, sd = sqrt(tausqn))
    results[i,1] <- mu
  }
  return(results)
}

gibbs <- gibbSampler(iter=nDraws, data=data, mu0=mu0, tausq0=tausq0, sigmasq0=sigmasq0)
gibbs <- as.data.frame(gibbs)

#check for posterior mean and sigma
meanGibbs<-colMeans(gibbs) #31.62246 1549.09369

###(ii)
par(mfrow=c(1,2))
plot(gibbs[,1], type = "l",col= "green",
     main = "Convergence of mu", ylab="mu")
plot(gibbs[,2], type = "l",col= "blue",
     main = "Convergence of sigma", ylab="sigma", ylim=c(1400, 2000))

par(mfrow=c(1,3))
hist(gibbs[,1], probability = TRUE, breaks = 50, main = "Gibbs draws", xlab = "mu")
lines(density(gibbs[,1]), col = "red", lwd = 2)
plot(cumsum(gibbs[,1])/seq(1, nDraws), type = "l",
     main = "Gibbs draws", xlab = "Iteration", ylab = "cummulative mean of mu")
```

```
acf(gibbs[,1], main='Gibbs draws', lag.max = 20)

#####(b) Mixture normal model

##########    BEGIN USER INPUT #################
#Data
x <- as.matrix(data)

#Modle options
nComp <- 2 #Number of mixture components

#Prior
alpha <- 10*rep(1,nComp) # Dirichlet(alpha)
muPrior <- rep(0,nComp) # Prior mean of mu
tau2Prior <- rep(10,nComp) # Prior std of mu
sigma2_0 <- rep(var(x),nComp) # s20 (best guess of sigma2)
nu0 <- rep(4,nComp) # degrees of freedom for prior on sigma2

# MCMC options
nIter <- 100 # Number of Gibbs sampling draws

# Plotting options
plotFit <- TRUE
lineColors <- c("blue", "green", "magenta", 'yellow')
sleepTime <- 0.1 # Adding sleep time between iterations for plotting
################   END USER INPUT ###############

###### Defining a function that simulates from the
rScaledInvChi2 <- function(n, df, scale){
  return((df*scale)/rchisq(n,df=df))
}

####### Defining a function that simulates from a Dirichlet distribution
rDirichlet <- function(param){
  nCat <- length(param)
  piDraws <- matrix(NA,nCat,1)
  for (j in 1:nCat){
    piDraws[j] <- rgamma(1,param[j],1)
  }
  piDraws = piDraws/sum(piDraws) # Diving every column of piDraws by the sum of the elements in that co
  return(piDraws)
}

# Simple function that converts between two different representations of the mixture allocation
S2alloc <- function(S){
  n <- dim(S)[1]
  alloc <- rep(0,n)
  for (i in 1:n){
    alloc[i] <- which(S[i,] == 1)
  }
  return(alloc)
}
```

```
# Initial value for the MCMC
nObs <- length(x)
S <- t(rmultinom(nObs, size = 1 , prob = rep(1/nComp,nComp))) # nObs-by-nComp matrix with component all
mu <- quantile(x, probs = seq(0,1,length = nComp))
sigma2 <- rep(var(x),nComp)
probObsInComp <- rep(NA, nComp)

# Setting up the plot
xGrid <- seq(min(x)-1*apply(x,2,sd),max(x)+1*apply(x,2,sd),length = 100)
xGridMin <- min(xGrid)
xGridMax <- max(xGrid)
mixDensMean <- rep(0,length(xGrid))
effIterCount <- 0
ylim <- c(0,2*max(hist(x)$density))

for (k in 1:nIter){
  message(paste('Iteration number:',k))
  alloc <- S2alloc(S) # Just a function that converts between different representations of the group al
  nAlloc <- colSums(S)
  print(nAlloc)
  # Update components probabilities
  pi <- rDirichlet(alpha + nAlloc)

  # Update mu's
  for (j in 1:nComp){
    precPrior <- 1/tau2Prior[j]
    precData <- nAlloc[j]/sigma2[j]
    precPost <- precPrior + precData
    wPrior <- precPrior/precPost
    muPost <- wPrior*muPrior + (1-wPrior)*mean(x[alloc == j])
    tau2Post <- 1/precPost
    mu[j] <- rnorm(1, mean = muPost, sd = sqrt(tau2Post))
  }

  # Update sigma2's
  for (j in 1:nComp){
    sigma2[j] <- rScaledInvChi2(1, df = nu0[j] + nAlloc[j],
                    scale = (nu0[j]*sigma2_0[j] +
                    sum((x[alloc == j] - mu[j])^2))/(nu0[j] + nAlloc[j]))
  }

  # Update allocation
  for (i in 1:nObs){
    for (j in 1:nComp){
      probObsInComp[j] <- pi[j]*dnorm(x[i], mean = mu[j], sd = sqrt(sigma2[j]))
    }
    S[i,] <- t(rmultinom(1, size = 1 , prob = probObsInComp/sum(probObsInComp)))
  }

  # Printing the fitted density against data histogram
  if (plotFit && (k%%1 ==0)){
    effIterCount <- effIterCount + 1
    hist(x, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax),
```

```r
         main = paste("Iteration number",k), ylim = ylim)
    mixDens <- rep(0,length(xGrid))
    components <- c()
    for (j in 1:nComp){
      compDens <- dnorm(xGrid,mu[j],sd = sqrt(sigma2[j]))
      mixDens <- mixDens + pi[j]*compDens
      lines(xGrid, compDens, type = "l", lwd = 2, col = lineColors[j])
      components[j] <- paste("Component ",j)
    }
    mixDensMean <- ((effIterCount-1)*mixDensMean + mixDens)/effIterCount

    lines(xGrid, mixDens, type = "l", lty = 2, lwd = 3, col = 'red')
    legend("topright", box.lty = 1, legend = c("Data histogram",components, 'Mixture'),
           col = c("black",lineColors[1:nComp], 'red'), lwd = 2)
    Sys.sleep(sleepTime)
  }
}

hist(x, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax), main = "Final fitted density")
lines(xGrid, mixDensMean, type = "l", lwd = 2, lty = 4, col = "red")
lines(xGrid, dnorm(xGrid, mean = mean(x), sd = apply(x,2,sd)), type = "l", lwd = 2, col = "blue")
legend("topright", box.lty = 1,
       legend = c("Data histogram","Mixture density","Normal density"),
       col=c("black","red","blue"), lwd = 2)

####(c)Graphical comparison
#Parameters from part (a) and (b)
muA = meanGibbs[1]
sigma2A = meanGibbs[2]
muB=mu
sigma2B=sigma2
piB=pi

normalDen<-rnorm(1000, mean = muA, sd = sqrt(sigma2A))
mixtureDen<- (pi[1])*rnorm(1000, mean = muB[1], sd = sqrt(sigma2B[1]))+
  (pi[2])*rnorm(1000, mean = muB[2], sd = sqrt(sigma2B[2]))

par(mfrow = c(1, 1))
hist(data, freq = FALSE, main='Histogram, Norma Density and Mixture Model of Posterior',  breaks = 30)
lines(density(normalDen), col = "red", lwd = 1)
lines(density(mixtureDen), col = "blue", lwd = 1)
legend("topright", box.lty = 1,
       legend = c("Data histogram","Normal density", "Mixture density"),
       col=c("black","red","yellow"), lwd = 1)

#######################
####(c) Graphical comparision
# par(mfrow = c(1, 1))
## Data histogram
# hist(data, freq=FALSE, breaks=20)
## Normal density
# muA = meanGibbs[1]
# sigma2A = meanGibbs[2]
# muB=mu
# sigmaB=sigma
# piB=pi
# lines(xGrid, dnorm(xGrid, mean = muA, sd=sqrt(sigma2A)), type = "l", lwd = 2, col = "blue")
## Mixture density
# lines(xGrid, mixDensMean, type = "l", lwd = 2, col = "red")
# legend("topright", box.lty = 1,
# legend = c("Data histogram", "Mixture density", "Normal density"),
# col = c("black", "red", "blue"), lwd = 2)
# ########################
```

Question 2

```
####Question2#####
###(a)

# Requirement
rm(list=ls())
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())

# Prior
T = 200
mu = 10
sigma = sqrt(2)
phi = 0.5

# Model
arStanModel = '
data{
int<lower=0> N;
vector[N] x;
}
parameters{
real mu;
real<lower=0> sigma;
real phi;
}
model{
for(t in 2:N)
x[t] ~ normal(mu + phi*(x[t-1]-mu), sigma);
}
'
#AR process
arProcess <- function(phi, init, sigma, T){
  y <- rep(0, T)
  y[1] <- init
  for(t in 2:T){
    y[t] <- mu + phi*(y[t-1]-mu) + rnorm(1, 0, sigma)
  }
  return(y)
```

```r
}

#Simulation of AR process
phis = seq(-1, 1, 0.05)
samples <- matrix(0, nrow = length(phis), ncol = T)
for(i in 1:length(phis)){
  samples[i,] <- arProcess(phis[i], mu, sigma, T)
}

#plot
par(mfrow = c(2,2))
plot(samples[2,], type = "l", main = "Phi = -0.95 ")
plot(samples[15,], type = "l",  main = "Phi = -0.30")
plot(samples[25,], type = "l",  main = "Phi = 0.20")
plot(samples[40,], type = "l",  main = "Phi = 0.955")

#What effect does the value of phi have on x[1:t]
#high phi values give smoothing effect???

###(b)
###(i)
# Simulate two AR process with:
phi1 = 0.3
phi2 = 0.95

x1 <- arProcess(phi1, mu, sigma, T)
x2 <- arProcess(phi2, mu, sigma, T)

# MCMC
fit1 <- stan(model_code = arStanModel, data = list(x = x1, N = T))
fit2 <- stan(model_code = arStanModel, data = list(x = x2, N = T))

# Posterior mean
posMean1 <- get_posterior_mean(fit1)
posMean2 <- get_posterior_mean(fit2)

#Estimate of true values
posMu1 <- posMean1 [1, 5]#9.880951
posSigma1 <- posMean1 [2, 5]#1.32709
posPhi1 <- posMean1[3, 5]#0.2886453

posMu2 <- posMean2[1, 5]#10.42592
posSigma2 <- posMean2[2, 5]#1.451759
posPhi2 <- posMean2[3, 5]#0.9188402

# 95% CI
para1 <- extract(fit1)
para2 <- extract(fit2)

CI_mu1 <- apply(as.matrix(para1$mu), 2, quantile, probs = c(0.025, 0.975))#(9.761905, 10.457614)
CI_sigma1 <- apply(as.matrix(para1$sigma), 2, quantile, probs = c(0.025, 0.975))#(1.346373, 1.637653)
CI_phi1 <- apply(as.matrix(para1$phi), 2, quantile, probs = c(0.025, 0.975))#(0.2445137, 0.5089993)
```

```
CI_mu2 <- apply(as.matrix(para2$mu), 2, quantile, probs = c(0.025, 0.975))#(6.513401, 14.874186)
CI_sigma2 <- apply(as.matrix(para2$sigma), 2, quantile, probs = c(0.025, 0.975))#(1.310611, 1.604546)
CI_phi2 <- apply(as.matrix(para2$phi), 2, quantile, probs = c(0.025, 0.975))#(0.8581515, 0.9873728)

# Number of errective posterior samples
summary1 <- summary(fit1)
posEff1 <- summary1$summary[, "n_eff"]
posEffMu1 <- posEff1["mu"] #2843.019
posEffSigma1 <- posEff1["sigma"] #3795.872
posEffPhi1 <- posEff1["phi"] #3240.529

summary2 <- summary(fit2)
posEff2 <- summary2$summary[, "n_eff"]
posEffMu2 <- posEff2["mu"] #1185.766
posEffSigma2 <- posEff2["sigma"] #2393.822
posEffPhi2 <- posEff2["phi"] #1245.068

###(ii)

# Plotting joint posterior
par(mfrow = c(1,2))
plot(x = para1$mu, y = para1$phi, col = "blue", xlab = "mu", ylab = "phi",
     main = "Joint posterior with phi = 0.3")
plot(x = para2$mu, y = para2$phi, col = "blue",xlab = "mu", ylab = "phi",
     main = "Joint posterior with phi = 0.95")

#comment:correlation??? convergence???

###(c)
# Requirement
rm(list=ls())
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())

# Loading data
campy <- read.table("campy.dat", header = TRUE)
campyData <- campy[,1]
head(campy)

#Prior
mu = 10
sigma = 1.4
phi = 0.2

# Model
poisonStanModel = '
data{
int<lower=0> N;
int c[N];
}
parameters{
real mu;
real<lower=0> sigma;
real phi;
vector[N] x;
}
model{
for(t in 2:N){
   x[t] ~ normal(mu + phi*(x[t-1]-mu), sigma);
}
for(n in 1:N){
      c[n] ~  poisson(exp(x[n]));
}
}
'
# Fit the model
fit3 <- stan(model_code = poisonStanModel,
             data = list(c = campyData, N = length(campyData)),
             chains = 1)

# Extracting relevant parameters
x <- extract(fit3, par = "x")
thetat <- exp(x$x)

# Mean
meanx <- apply(thetat, 2, mean)

# CI
quantilex <- apply(thetat, 2, quantile, probs = c(0.025, 0.975))

# Plot
plot(campyData, type = "p", pch= 19, cex = 0.5, col = 1,
     ylab = "C",
     main = "Data and Posterior mean with 95% CI of theta (sigma = 1.4)" )
lines(meanx, type = "l", col = 2)
lines(quantilex[1,], col = 3)
lines(quantilex[2,], col = 4)
legend("topleft",
       c("Data", "Posterior mean", "5% CI", "95% CI"),
       lty = 1,
       col = c(1,2,3,4), text.col = c(1,2,3,4),
       cex = 0.75)

###(d)
#Prior
mu = 10
sigma = 100
phi = 0.2

# Model
poisonStanModel2 = '
data{
int<lower=0> N;
int c[N];
}
```

```
parameters{
real mu;
real<lower=0> sigma;
real phi;
vector[N] x;
}
model{
for(t in 2:N){
x[t] ~ normal(mu + phi*(x[t-1]-mu), sigma);
}
for(n in 1:N){
c[n] ~  poisson(exp(x[n]));
}
}
'
# Fit the model
fit4 <- stan(model_code = poisonStanModel2,
             data = list(c = campyData, N = length(campyData)),
             chains = 1)

# Extracting relevant parameters
x2 <- extract(fit4, par = "x")
thetat2 <- exp(x2$x)

# Mean
meanx2 <- apply(thetat2, 2, mean)

# CI
quantilex2 <- apply(thetat2, 2, quantile, probs = c(0.025, 0.975))

# Plot
plot(campyData, type = "p", pch= 19, cex = 0.5, col = 1,
     ylab = "C",
     main = "Data and Posterior mean with 95% CI of theta (sigma = 100)" )
lines(meanx2, type = "l", col = 2)
lines(quantilex2[1,], col = 3)
lines(quantilex2[2,], col = 4)
legend("topleft",
       c("Data", "Posterior mean", "5% CI", "95% CI"),
       lty = 1,
       col = c(1,2,3,4), text.col = c(1,2,3,4),
       cex = 0.75)
# Has the posterior for thetat changed? varry slightly , capture more data points

###############END LAB3###############
```

Bayesian Learning Lab3

Syeda Farha Shazmeen(syesh076),Farhana Chowdhury Tondra(farch587),

6 May 2018

1. Normal model, mixture of normal model with semi-conjugate prior.

1.a) Normal Model

Given that daily precipitation is indepdent normally distributed:

$$y1...yn|\ mu, \sigma^2 \sim N(\theta, \sigma^2)$$

The mean and variance are given by:

$$\mu \sim N(\mu_o, \tau^2)$$

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

Conditional Posterior:

$$\mu|\sigma^2, x \sim N(\mu_n, \tau_n^2)$$

$$\sigma^2|\mu, x \sim Inv - \chi^2\left(v_n, \frac{v_0\sigma_0^2 + \sum_{i=1}^n (x_i - \mu^2)}{n + v_0}\right)$$

Where

$$\frac{1}{\tau_n^2} = \frac{n}{\sigma^2} + \frac{1}{\tau_0^2}$$

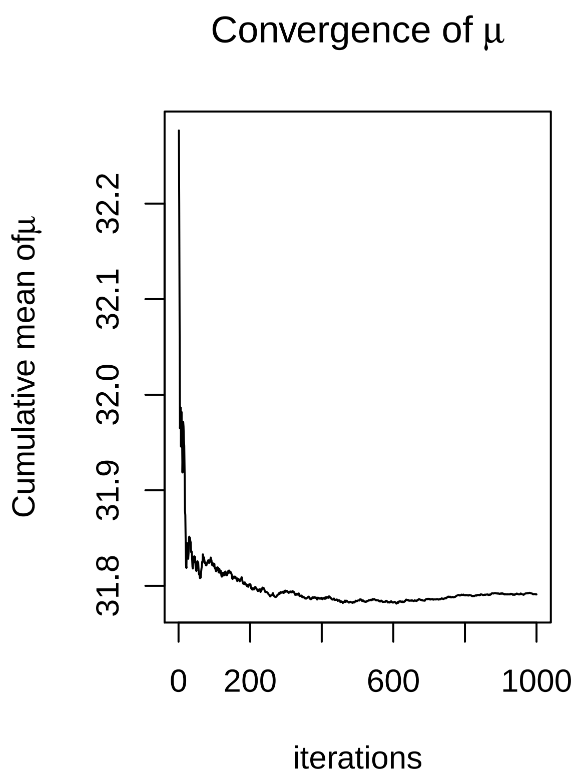
$$\mu_n = w\bar{x} + (1 - w)\mu_0$$

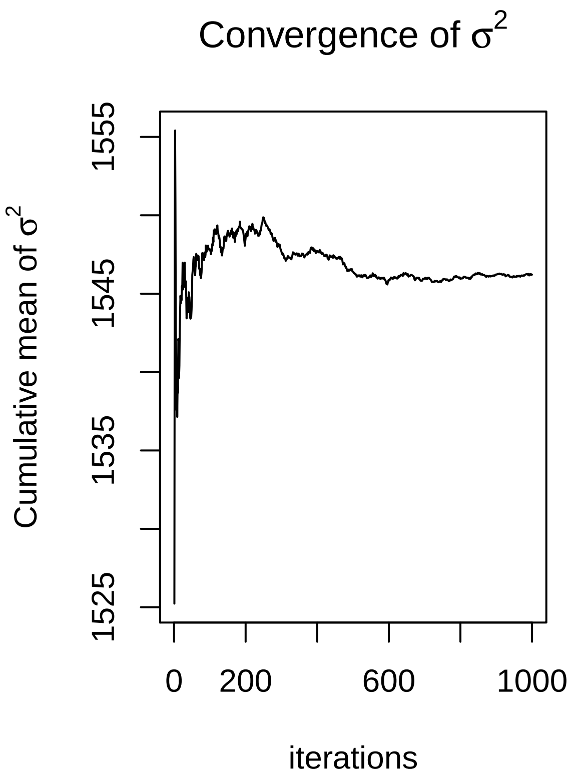
and

$$w = \frac{\frac{n}{\sigma^2}}{\frac{n}{\sigma^2} + \frac{1}{\tau_0^2}}$$

The posterior is given by joint probability :

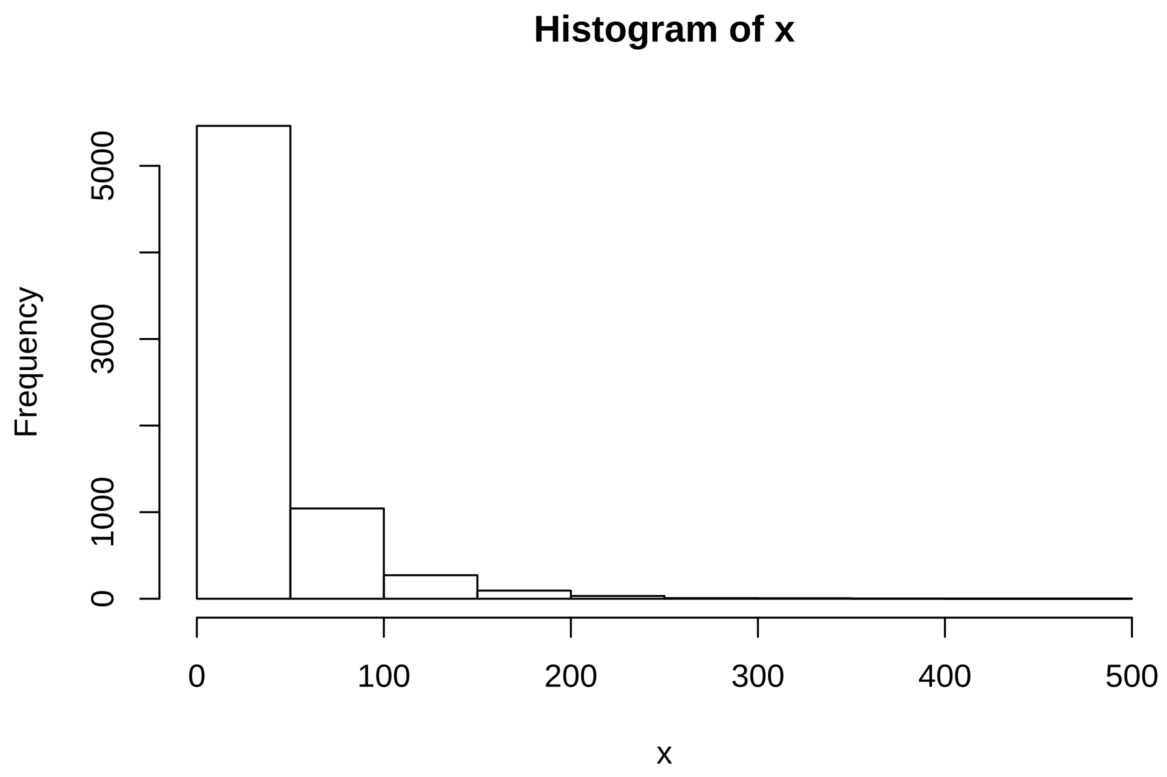
$$p(\mu, \sigma^2|y1....yn)$$





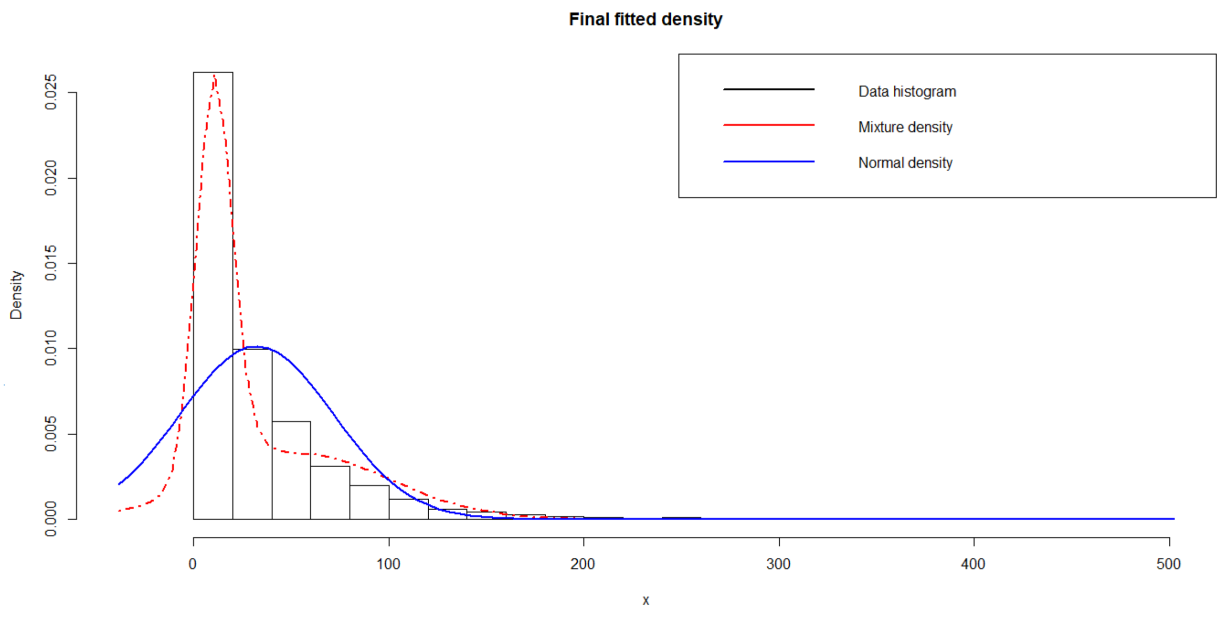
1.b) Mixture normal model.(USED THE SAMPLE CODE PROVIDED)

The daily precipitation y_1......y_n follow an iid two component mixture of normals models.



```
for (k in 1:nIter){
  message(paste('Iteration number:',k))
  alloc <- S2alloc(S) # Just a function that converts between
  #different representations of the group allocations
  nAlloc <- colSums(S)
  print(nAlloc)
  # Update components probabilities
  pi <- rDirichlet(alpha + nAlloc)

  # Update mu's
  for (j in 1:nComp){
    precPrior <- 1/tau2Prior[j]
    precData <- nAlloc[j]/sigma2[j]
    precPost <- precPrior + precData
    wPrior <- precPrior/precPost
    muPost <- wPrior*muPrior + (1-wPrior)*mean(x[alloc == j])
    tau2Post <- 1/precPost
    mu[j] <- rnorm(1, mean = muPost, sd = sqrt(tau2Post))
  }

  # Update sigma2's
  for (j in 1:nComp){
    sigma2[j] <- rScaledInvChi2(1, df = nu0[j] + nAlloc[j],
        scale = (nu0[j]*sigma2_0[j] +
      sum((x[alloc == j] - mu[j])^2))/(nu0[j] + nAlloc[j]))
  }
```

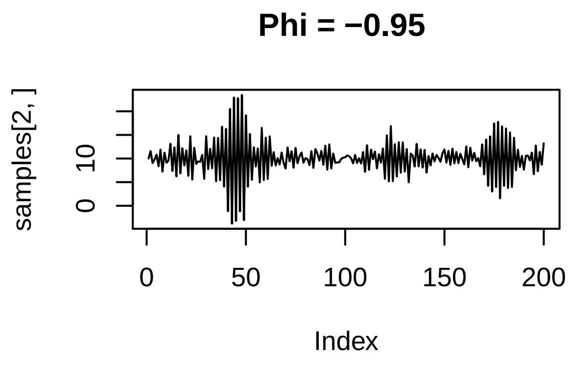
```r
  # Update allocation
  for (i in 1:nObs){
    for (j in 1:nComp){
      probObsInComp[j] <- pi[j]*dnorm(x[i], mean = mu[j], sd = sqrt(sigma2[j]))
    }
    S[i,] <- t(rmultinom(1, size = 1 , prob = probObsInComp/sum(probObsInComp)))
  }

  # Printing the fitted density against data histogram
  if (plotFit && (k%%1 ==0)){
    effIterCount <- effIterCount + 1
    hist(x, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax), main = paste("Iteration number",k),
    mixDens <- rep(0,length(xGrid))
    components <- c()
    for (j in 1:nComp){
      compDens <- dnorm(xGrid,mu[j],sd = sqrt(sigma2[j]))
      mixDens <- mixDens + pi[j]*compDens
      lines(xGrid, compDens, type = "l", lwd = 2, col = lineColors[j])
      components[j] <- paste("Component ",j)
    }
    mixDensMean <- ((effIterCount-1)*mixDensMean + mixDens)/effIterCount

    lines(xGrid, mixDens, type = "l", lty = 2, lwd = 3, col = 'red')
    legend("topleft", box.lty = 1, legend = c("Data histogram",components, 'Mixture'),
           col = c("black",lineColors[1:nComp], 'red'), lwd = 2)
    Sys.sleep(sleepTime)
  }

}
hist(x, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax),
     main = "Final fitted density")
lines(xGrid, mixDensMean, type = "l", lwd = 2, lty = 4, col = "red")
lines(xGrid, dnorm(xGrid, mean = mean(x), sd = apply(x,2,sd)),
      type = "l", lwd = 2, col = "blue")

legend("topright", box.lty = 1, legend = c("Data histogram","Mixture density","Normal density"),
       col=c("black","red","blue"), lwd = 2)

#########################    Helper functions    ##############################################
```

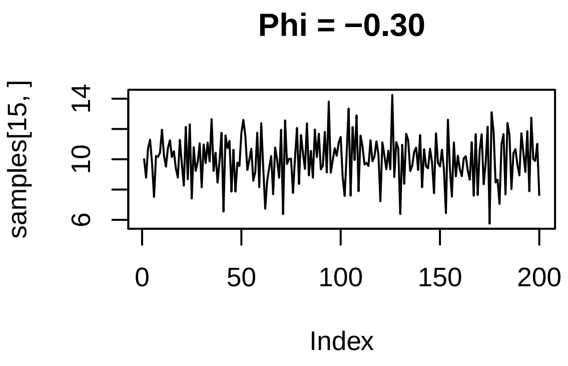


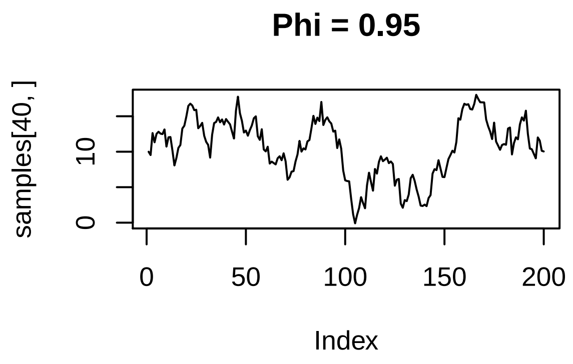
2. Time series models in Stan

2.a) Given AR(1)-process :

$$x_t = \mu + \phi(x_{t-1} - \mu) + \epsilon_t \ ; \ \epsilon_t \ N(0, \sigma^2)$$







So, what effect does the value of ϕ have on $x_{1:T}$? We can see that positive high phi value gives smoothing effect on AR process than negative values.

2.b) Simulate two AR(1)-processes, $x_{1:T}$ with $\phi = 0.3$ and $y_{1:T}$ with $\phi = 0.95$.

i)

```
##
##  Posterior mean is for phi=0.3 :
```

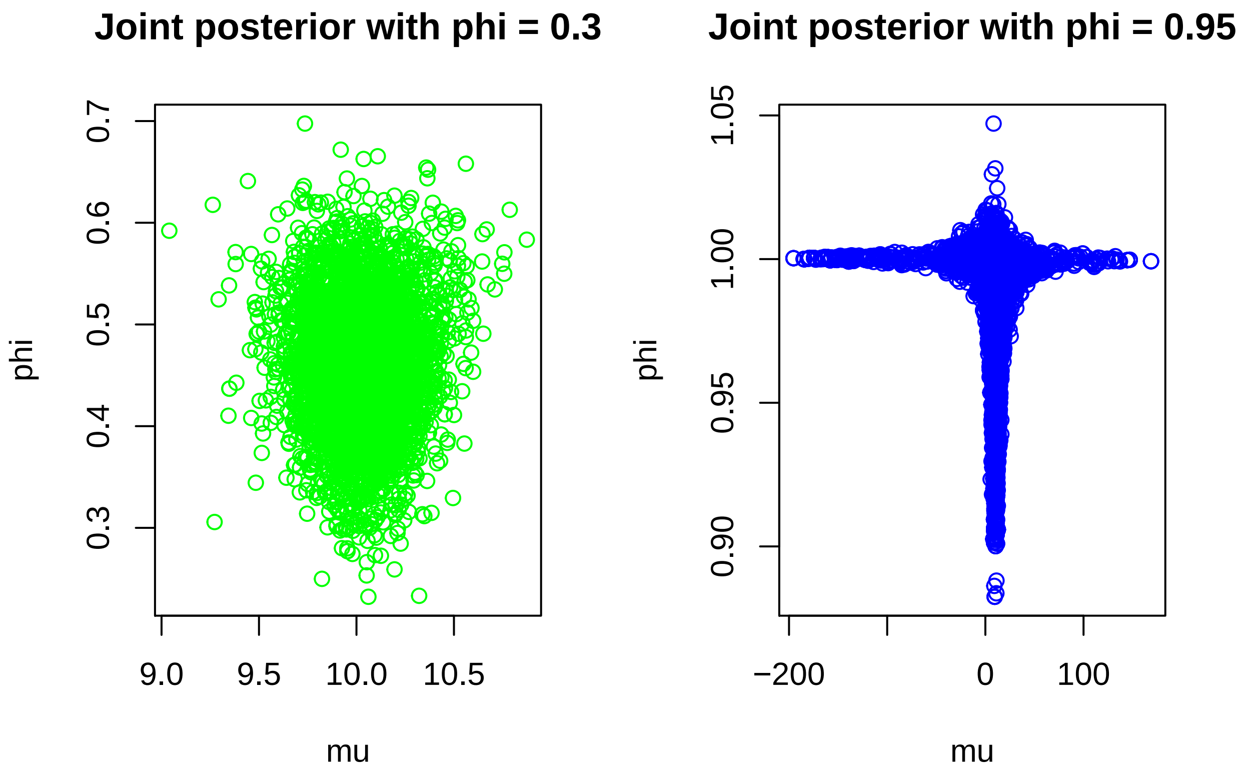
```
##           mean-chain:1 mean-chain:2 mean-chain:3 mean-chain:4
## mu          10.0227997   10.0277391   10.0225192   10.0308282
## sigma_sqr    2.0396011    2.0476789    2.0454711    2.0384366
## phi          0.4579904    0.4580291    0.4601641    0.4603947
## lp__      -168.2619224 -168.2607717 -168.3451279 -168.3292999
##           mean-all chains
## mu             10.0259715
## sigma_sqr       2.0427969
## phi             0.4591446
## lp__         -168.2992805
```

```
##
##  Posterior mean is for phi=0.95 :
```

```
##           mean-chain:1 mean-chain:2 mean-chain:3 mean-chain:4
## mu         -16.2551239   11.4029928   11.1943965   11.1512756
## sigma_sqr    2.1100176    2.0723599    2.0567896    2.0767347
## phi          0.9964772    0.9737209    0.9628447    0.9706576
## lp__      -170.9122250 -170.1824742 -170.0385062 -169.9806279
##           mean-all chains
## mu              4.3733852
## sigma_sqr       2.0789754
## phi             0.9759251
## lp__         -170.2784583

##
##  95% credible intervals of mu for phi = 0.3 is :

##            [,1]
## 2.5%   9.659471
## 97.5% 10.411471

##
##  95% credible intervals of sigma_square for phi = 0.3 is :

##           [,1]
## 2.5%  1.670938
## 97.5% 2.477508

##
##  95% credible intervals of phi for phi = 0.3 is :

##            [,1]
## 2.5%  0.3316620
## 97.5% 0.5837348

##
##  95% credible intervals of mu for phi = 0.95 is :

##             [,1]
## 2.5%  -140.87049
## 97.5%   55.42477

##
##  95% credible intervals of sigma_square for phi = 0.95 is :

##           [,1]
## 2.5%  1.707495
## 97.5% 2.519149

##
##  95% credible intervals of phi for phi = 0.95 is :

##            [,1]
## 2.5%  0.9116376
## 97.5% 1.0069035

##
##  The number of effective posterior samples for the three inferred parameter when phi=0.3 are:

##        mu sigma_sqr       phi
##  3501.651  4000.000  4000.000

##
##  The number of effective posterior samples for the three inferred parameter when phi=0.9 are :

##         mu  sigma_sqr        phi
## 118.104637 688.395866   7.728055
```

```
##
##  Estimated true values are :
##

##                  mu sigma_square       phi
## phi=0.3  10.0259715    2.0427969 0.4591446
## phi=0.95  4.3733852    2.0789754 0.9759251
```

ii) Plotting joint posterior and obvsering convergence



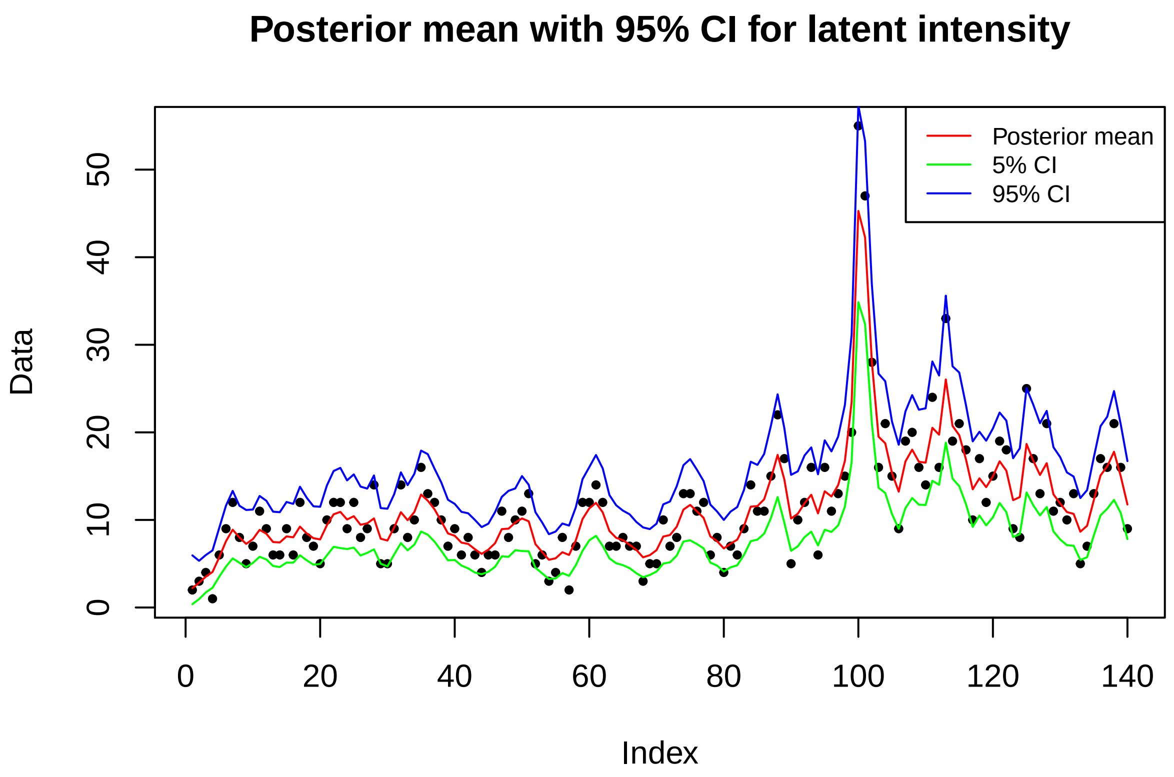
We can conclude by observing the joint posterior plots that, convergence reaches when phi is positive and close to 1.

2.c) Assume that the number of infections c_t at each

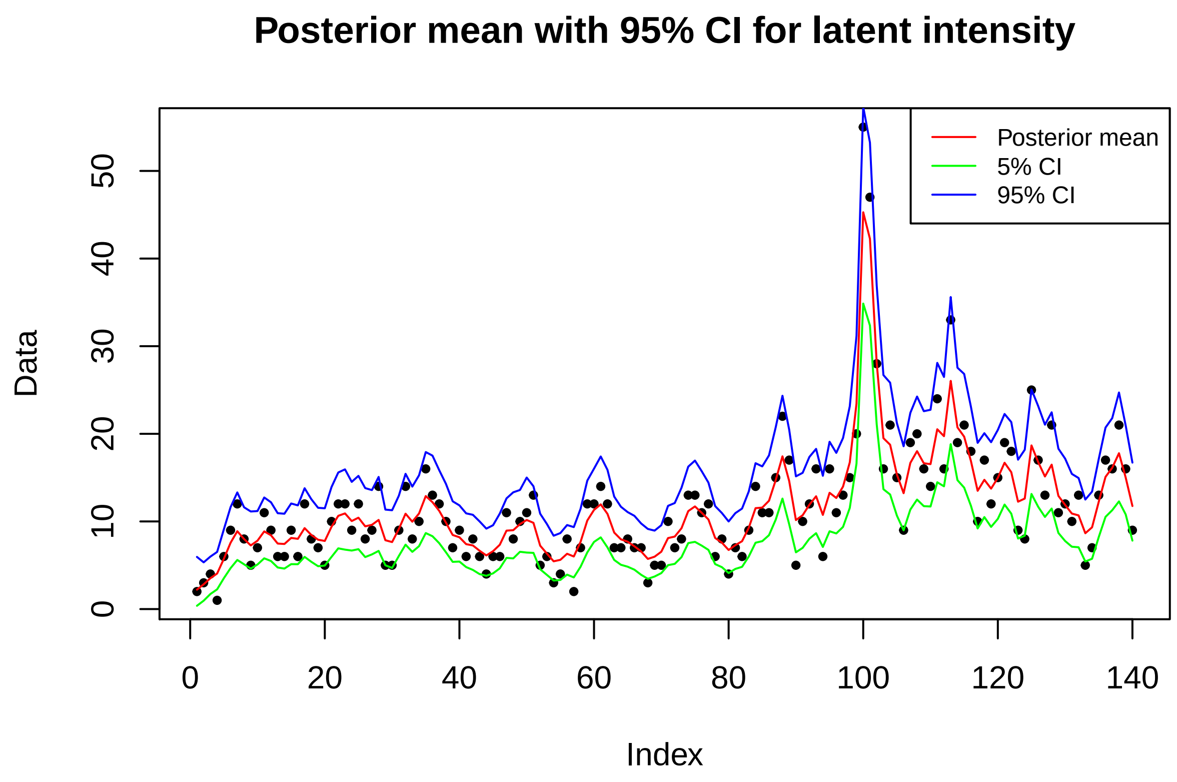
time point follows an independent Poisson distribution when conditioned on a latent AR(1)-process x_t, that is

$$c_t|x_t \; Poisson(exp(x_t))$$

Where x_t is an AR(1)-process and considered here as parameters.


Posterior mean with 95% CI for latent intensity
Data
Index
Posterior mean
5% CI
95% CI
0
10
20
30
40
50
0
20
40
60
80
100
120
140

2.d) Change the prior for σ^2 so that it becomes informative about that the AR(1)-process increments ϵ_t should be small.



No change at all in posterior after changing the prior $\sigma^2 = 4$, so the prior belief that the true underlying intensity varies more smoothly than the data suggests is wrong.

Appendix

```
###1.a)

library(geoR)
library(dplyr)
rainfall<-read.table("rainfall.dat")
names(rainfall)<-c("precipitation")
rainfall_precep<-rainfall$precipitation

#the first mu parameter for conditional posterior

param_mean_draw<-function(data,mu_0,tou_0_sq,sigma_square)
{
  n<-length(data)
  tau_n_sq<-(n/sigma_square)+(1/tou_0_sq)
  inv_tau_sq<-1/tau_n_sq
  w=(n/sigma_square)/(tau_n_sq)
  mu_n=w*mean(data)+(1-w)*mu_0
```

```r
  draw1<-rnorm(1,mu_n,inv_tau_sq)
  return(draw1)
}


param_var_draw<-function(data,v_0,mu,sigma_sq_zero)
{
  n=length(data)
  vn=v_0+n
  scale<-(v_0*sigma_sq_zero+sum((data-mu)^2))/(vn)
  draw2<-rinvchisq(1,df=vn,scale = scale)
}

#define the prior values

sigma_square <- 10
sigma_sq_zero <- 10
tou_0_sq <- 10
v_0 <- 10
mu_0 <- 10
max_iter<-1000
draws<-data.frame(iter=vector("numeric",length=max_iter),
      mean=vector("numeric",length = max_iter),
      sigma=vector("numeric",length=max_iter))
for(i in 1:max_iter)
{

  mu<-param_mean_draw(rainfall_precep,mu_0,tou_0_sq,sigma_square)

  sigma_square<-param_var_draw(data=rainfall_precep,v_0=v_0,mu=mu,
            sigma_sq_zero=sigma_sq_zero)
  draws[i,]<-c(i,mu,sigma_square)
}
draws$cum_mean<-cummean(draws$mean)
draws$cum_var<-cummean(draws$sigma)
layout(matrix(c(1, 2), nrow = 1))
plot(draws$iter, draws$cum_mean, type = "l",
     main=expression(paste("Convergence of ",  mu)),
     xlab="iterations",ylab=expression(paste("Cumulative mean of\t",mu)))
plot(draws$iter, draws$cum_var, type = "l",
     main=expression(paste("Convergence of ",  sigma^2)),
     xlab="iterations",ylab=expression(paste("Cumulative mean of\t ", sigma^2)))

#1.b (used the sample code provided)
# Estimating a simple mixture of normals

##########    BEGIN USER INPUT #################\
rainfall<-read.table("rainfall.dat")
names(rainfall)<-c("precipitation")
rainfall_precep<-rainfall$precipitation
# Data options
data(faithful)
rawData <- rainfall
```

```r
x <- as.matrix(rawData['precipitation'])

# Model options
nComp <- 2    # Number of mixture components

# Prior options
alpha <- 10*rep(1,nComp) # Dirichlet(alpha)
muPrior <- rep(0,nComp) # Prior mean of mu
tau2Prior <- rep(10,nComp) # Prior std of mu
sigma2_0 <- rep(var(x),nComp) # s20 (best guess of sigma2)
nu0 <- rep(4,nComp) # degrees of freedom for prior on sigma2

# MCMC options
nIter <- 100 # Number of Gibbs sampling draws

# Plotting options
plotFit <- TRUE
lineColors <- c("blue", "green", "magenta", 'yellow')
sleepTime <- 0.1 # Adding sleep time between iterations for plotting
################   END USER INPUT ###############

###### Defining a function that simulates from the
rScaledInvChi2 <- function(n, df, scale){
  return((df*scale)/rchisq(n,df=df))
}

####### Defining a function that simulates from a Dirichlet distribution
rDirichlet <- function(param){
  nCat <- length(param)
  piDraws <- matrix(NA,nCat,1)
  for (j in 1:nCat){
    piDraws[j] <- rgamma(1,param[j],1)
  }
  piDraws = piDraws/sum(piDraws) # Diving every column of piDraws
  #by the sum of the elements in that column.
  return(piDraws)
}

# Simple function that converts between two different
# representations of the mixture allocation
S2alloc <- function(S){
  n <- dim(S)[1]
  alloc <- rep(0,n)
  for (i in 1:n){
    alloc[i] <- which(S[i,] == 1)
  }
  return(alloc)
}

# Initial value for the MCMC
nObs <- length(x)
S <- t(rmultinom(nObs, size = 1 , prob = rep(1/nComp,nComp)))
# nObs-by-nComp matrix with component allocations.
```

```
mu <- quantile(x, probs = seq(0,1,length = nComp))
sigma2 <- rep(var(x),nComp)
probObsInComp <- rep(NA, nComp)

# Setting up the plot
xGrid <- seq(min(x)-1*apply(x,2,sd),max(x)+1*apply(x,2,sd),length = 100)
xGridMin <- min(xGrid)
xGridMax <- max(xGrid)
mixDensMean <- rep(0,length(xGrid))
effIterCount <- 0
ylim <- c(0,2*max(hist(x)$density))

for (k in 1:nIter){
  message(paste('Iteration number:',k))
  alloc <- S2alloc(S) # Just a function that converts between
  #different representations of the group allocations
  nAlloc <- colSums(S)
  print(nAlloc)
  # Update components probabilities
  pi <- rDirichlet(alpha + nAlloc)

  # Update mu's
  for (j in 1:nComp){
    precPrior <- 1/tau2Prior[j]
    precData <- nAlloc[j]/sigma2[j]
    precPost <- precPrior + precData
    wPrior <- precPrior/precPost
    muPost <- wPrior*muPrior + (1-wPrior)*mean(x[alloc == j])
    tau2Post <- 1/precPost
    mu[j] <- rnorm(1, mean = muPost, sd = sqrt(tau2Post))
  }

  # Update sigma2's
  for (j in 1:nComp){
    sigma2[j] <- rScaledInvChi2(1, df = nu0[j] + nAlloc[j],
           scale = (nu0[j]*sigma2_0[j] +
    sum((x[alloc == j] - mu[j])^2))/(nu0[j] + nAlloc[j]))
  }

  # Update allocation
  for (i in 1:nObs){
    for (j in 1:nComp){
      probObsInComp[j] <- pi[j]*dnorm(x[i], mean = mu[j], sd = sqrt(sigma2[j]))
    }
    S[i,] <- t(rmultinom(1, size = 1 , prob = probObsInComp/sum(probObsInComp)))
  }

  # Printing the fitted density against data histogram
  if (plotFit && (k%%1 ==0)){
    effIterCount <- effIterCount + 1
    hist(x, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax),
         main = paste("Iteration number",k), ylim = ylim)
    mixDens <- rep(0,length(xGrid))
    components <- c()
    for (j in 1:nComp){
      compDens <- dnorm(xGrid,mu[j],sd = sqrt(sigma2[j]))
      mixDens <- mixDens + pi[j]*compDens
      lines(xGrid, compDens, type = "l", lwd = 2, col = lineColors[j])
      components[j] <- paste("Component ",j)
    }
    mixDensMean <- ((effIterCount-1)*mixDensMean + mixDens)/effIterCount

    lines(xGrid, mixDens, type = "l", lty = 2, lwd = 3, col = 'red')
    legend("topleft", box.lty = 1, legend = c("Data histogram",components, 'Mixture'),
    col = c("black",lineColors[1:nComp], 'red'), lwd = 2)
    Sys.sleep(sleepTime)
  }

}
hist(x, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax), main = "Final fitted density")
lines(xGrid, mixDensMean, type = "l", lwd = 2, lty = 4, col = "red")
lines(xGrid, dnorm(xGrid, mean = mean(x), sd = apply(x,2,sd)),
      type = "l", lwd = 2, col = "blue")

legend("topright", box.lty = 1,
       legend = c("Data histogram","Mixture density","Normal density"),
       col=c("black","red","blue"), lwd = 2)



###2.a)
rm(list=ls())
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())

# Prior
T = 200
mu = 10
sigma_sqr = 2
phi = 0.5

# Model
arModel = '
data{
int<lower=0> N;
vector[N] x;
}
parameters{
real mu;
real<lower=0> sigma_sqr;
real phi;
}
model{
for(t in 2:N)
  x[t] ~ normal(mu + phi*(x[t-1]-mu), sqrt(sigma_sqr));
```

```r
}
'
arProcess <- function(phi, init, sigma_sqr, T){
  y <- c()
  y[1] <- init
  for(t in 2:T){
    y[t] <- mu + phi*(y[t-1]-mu) + rnorm(1, 0, sqrt(sigma_sqr))
  }
  return(y)
}

phi_range = seq(-1, 1, 0.05)
samples <- matrix(0, nrow = length(phi_range), ncol = T)
for(i in 1:length(phi_range)){
  samples[i,] <- arProcess(phi_range[i], mu, sigma_sqr, T)
}

par(mfrow = c(2,2))
plot(samples[2,], type = "l", main = "Phi = -0.95 ")
plot(samples[15,], type = "l",  main = "Phi = -0.30")
plot(samples[40,], type = "l",  main = "Phi = 0.95")

###2.b)
phi_1 = 0.3
phi_2 = 0.95

x <- arProcess(phi_1, mu, sigma_sqr, T)
y <- arProcess(phi_2, mu, sigma_sqr, T)

# MCMC
fit1 <- stan(model_code = arModel, data = list(x = x, N = T))
fit2 <- stan(model_code = arModel, data = list(x = y, N = T))

# Posterior mean
Mean1 <- get_posterior_mean(fit1)
Mean2 <- get_posterior_mean(fit2)

cat("Posterior mean is for phi=0.3 : ", Mean1)
cat("Posterior mean is for phi=0.95 : ", Mean2)

# 95% credible intervals
postfit1 <- extract(fit1)
postfit2 <- extract(fit2)


CI_mu1 <- apply(as.matrix(postfit1$mu), 2, quantile, probs = c(0.025, 0.975))
CI_sigma1 <- apply(as.matrix(postfit1$sigma_sqr), 2, quantile, probs = c(0.025, 0.975))
CI_phi1 <- apply(as.matrix(postfit1$phi), 2, quantile, probs = c(0.025, 0.975))
cat("95% credible intervals of mu for phi = 0.3 is : ")
CI_mu1
cat("95% credible intervals of sigma_square for phi = 0.3 is : ")
CI_sigma1
cat("95% credible intervals of phi for phi = 0.3 is : ")
```

```
CI_phi1

CI_mu2 <- apply(as.matrix(postfit2$mu), 2, quantile, probs = c(0.025, 0.975))
CI_sigma2 <- apply(as.matrix(postfit2$sigma_sqr), 2, quantile, probs = c(0.025, 0.975))
CI_phi2 <- apply(as.matrix(postfit2$phi), 2, quantile, probs = c(0.025, 0.975))
cat("95% credible intervals of mu for phi = 0.95 is : ")
CI_mu2
cat("95% credible intervals of sigma_square for phi = 0.95 is : ")
CI_sigma2
cat("95% credible intervals of phi for phi = 0.95 is : ")
CI_phi2

# The number of effective posterior samples for the three inferred parameters
result1 <- summary(fit1)
posEff1 <- result1$summary[, "n_eff"]
cat("The number of effective posterior samples for the
    three inferred parameter when phi=0.3 are: ")
posEff1[1:3]
result2 <- summary(fit2)
posEff2 <- result2$summary[, "n_eff"]
cat("The number of effective posterior samples for the
    three inferred parameter when phi=0.9 are : " )
posEff2[1:3]

#True values
mat<- matrix(c(posMean1 [1, 5],posMean1 [2, 5],posMean1 [3, 5],
      posMean2 [1, 5],posMean2 [2, 5],posMean2 [3,5]),ncol=3,byrow=TRUE)
colnames(mat) <- c("mu","sigma_square","phi")
rownames(mat) <- c("phi=0.3","phi=0.95")
cat("Estimated true values are : ")
as.table(mat)

#### ii)
par(mfrow = c(1,2))
plot(x = postfit1$mu, y = postfit1$phi, col = "green", xlab = "mu",
     ylab = "phi",main = "Joint posterior with phi = 0.3")
plot(x = postfit2$mu, y = postfit2$phi, col = "blue",xlab = "mu",
     ylab = "phi",main = "Joint posterior with phi = 0.95")
###2.c)
rm(list=ls())
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())

#prior
mu = 10
sigma_sqr = 2
phi = 0.5
campy = read.delim("campy.dat")

PoissonModel = '
data {
int<lower=0> N; //length of data
```

```
int y[N] ;  // data
}


parameters {
real mu;
real<lower=0> sigma_sqr;
real<lower=-1, upper=1> phi;
vector[N] xt;
}


model {

for(t in 2:N){
xt[t] ~ normal(mu + phi*( xt[t-1] - mu ), sqrt(sigma_sqr));
y[t] ~  poisson(exp(xt[t]));
}

}'
data = list(N=nrow(campy), y=campy$c)
burnin = 1000
niter = 2000
fit2<-stan(model_code=PoissonModel,
           data=data,
           warmup=burnin,
           iter=niter,
           chains=1,seed = 12345)

Post_fit2 <- extract(fit2)
theta_t <-  exp(Post_fit2$xt)

#Posterior mean
mean_theta_t <- c()
for (i in 1:ncol(theta_t) ) {
  mean_theta_t[i] <- mean(theta_t[,i])
}

#Posterior mean with 95% CI for latent intensity
lower <- c()
upper <- c()
for (i in 1:ncol(theta_t) ) {
  lower[i] <- quantile((theta_t[,i]), probs = c(0.025, 0.975))[1]
  upper[i] <- quantile((theta_t[,i]), probs = c(0.025, 0.975))[2]

}
#plot
plot(campy$c, type = "p", pch= 19, cex = 0.5, col = 1,ylab = "Data",
     main = "Posterior mean with 95% CI for latent intensity" )
lines(mean_theta_t, type = "l", col = "red")
lines(lower, col = "green")
lines(upper, col = "blue")
legend("topright",c("Posterior mean", "5% CI", "95% CI"),lty = c(1,1,1),
       col = c("red","green","blue"),cex = 0.75)

###2.d)
rm(list=ls())
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())

#prior
mu = 10
sigma_sqr = 4
phi = 0.5
campy = read.delim("campy.dat")

PoissonModel = '
data {
int<lower=0> N; //length of data
int y[N] ;  // data
}


parameters {
real mu;
real<lower=0> sigma_sqr;
real<lower=-1, upper=1> phi;
vector[N] xt;
}


model {

for(t in 2:N){
xt[t] ~ normal(mu + phi*( xt[t-1] - mu ), sqrt(sigma_sqr));
y[t] ~  poisson(exp(xt[t]));
}

}'
data = list(N=nrow(campy), y=campy$c)
burnin = 1000
niter = 2000
fit2<-stan(model_code=PoissonModel,
           data=data,
           warmup=burnin,
           iter=niter,
           chains=1,seed = 12345)

Post_fit2 <- extract(fit2)
theta_t <-  exp(Post_fit2$xt)

#Posterior mean
mean_theta_t <- c()
for (i in 1:ncol(theta_t) ) {
  mean_theta_t[i] <- mean(theta_t[,i])
```

```
}

#Posterior mean with 95% CI for latent intensity
lower <- c()
upper <- c()
for (i in 1:ncol(theta_t) ) {
  lower[i] <- quantile((theta_t[,i]), probs = c(0.025, 0.975))[1]
  upper[i] <- quantile((theta_t[,i]), probs = c(0.025, 0.975))[2]

}
#plot
plot(campy$c, type = "p", pch= 19, cex = 0.5, col = 1,ylab = "Data",
     main = "Posterior mean with 95% CI for latent intensity" )
lines(mean_theta_t, type = "l", col = "red")
lines(lower, col = "green")
lines(upper, col = "blue")
legend("topright",c("Posterior mean", "5% CI", "95% CI"),lty = c(1,1,1),
       col = c("red","green","blue"),cex = 0.75)
```