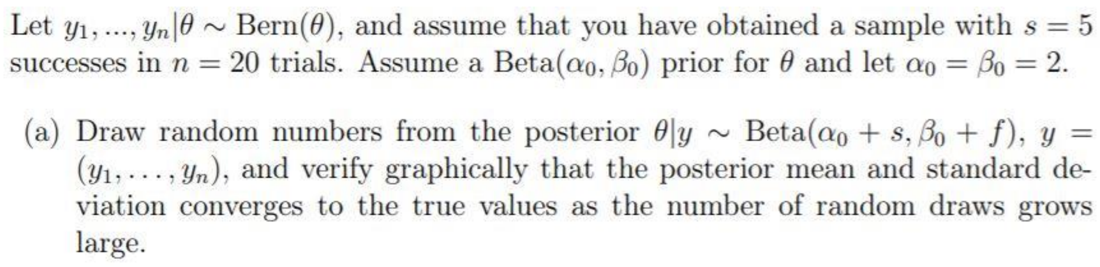
Lab1

Computer Lab 1

Assignment 1

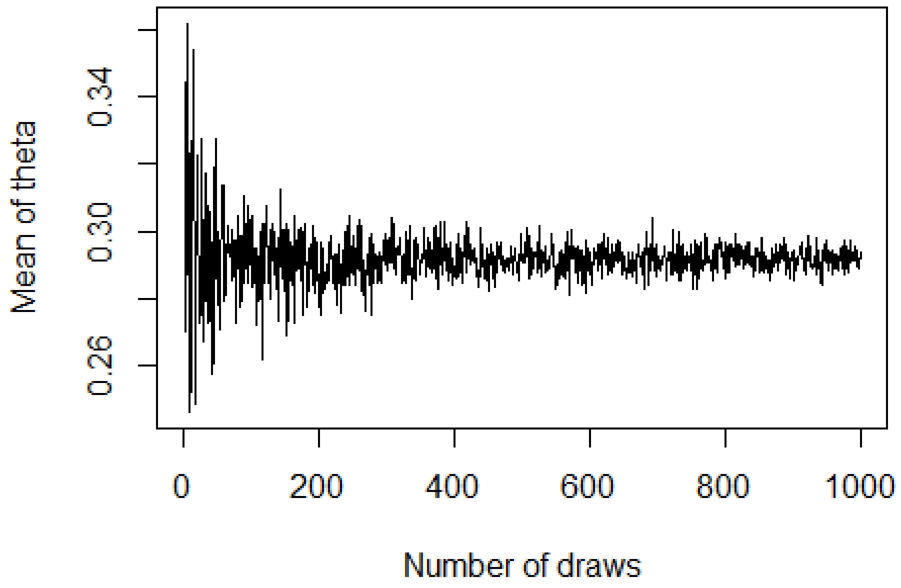
*(1a)

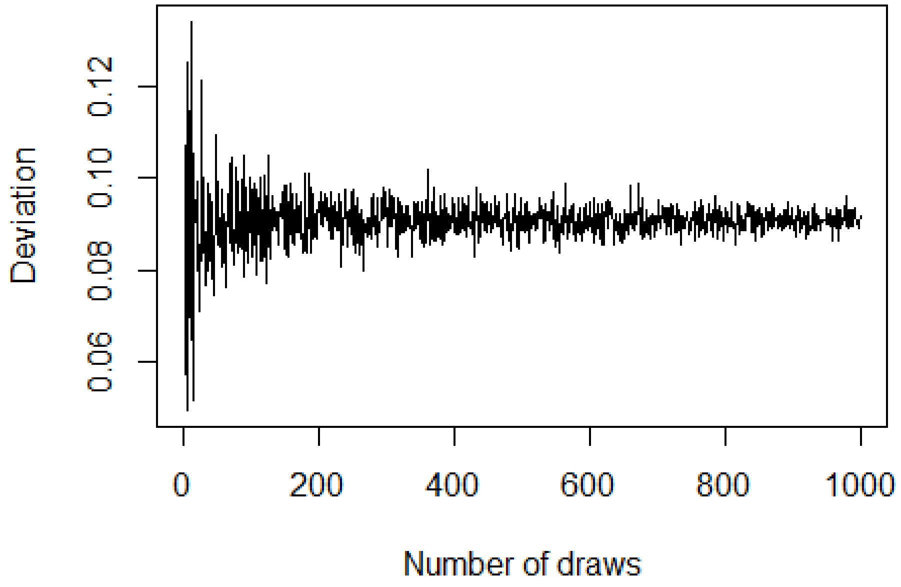


Let $y_1, ..., y_n|\theta \sim \text{Bern}(\theta)$, and assume that you have obtained a sample with $s = 5$ successes in $n = 20$ trials. Assume a $\text{Beta}(\alpha_0, \beta_0)$ prior for θ and let $\alpha_0 = \beta_0 = 2$.

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

As n gets bigger the mean and standard deviation converges towards its true value. Bigger amount of data (n) results in that the prior has less influence on the posterior. Mean -> 0,29 st -> 0,09





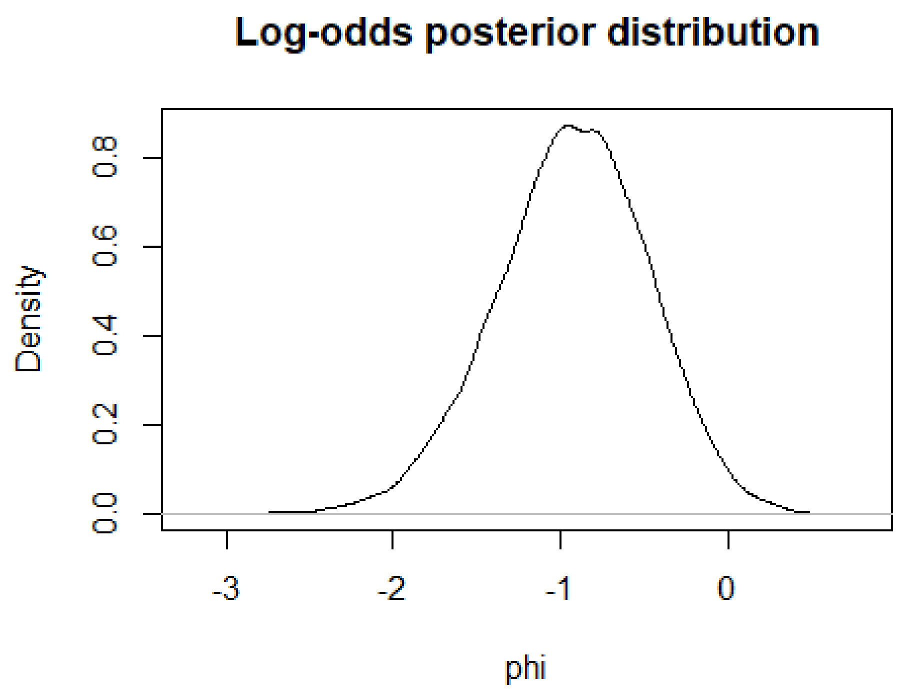
(b) Use simulation (nDraws = 10000) to compute the posterior probability Pr(phi > 0.3|y) and compare with the exact value [Hint: pbeta()].

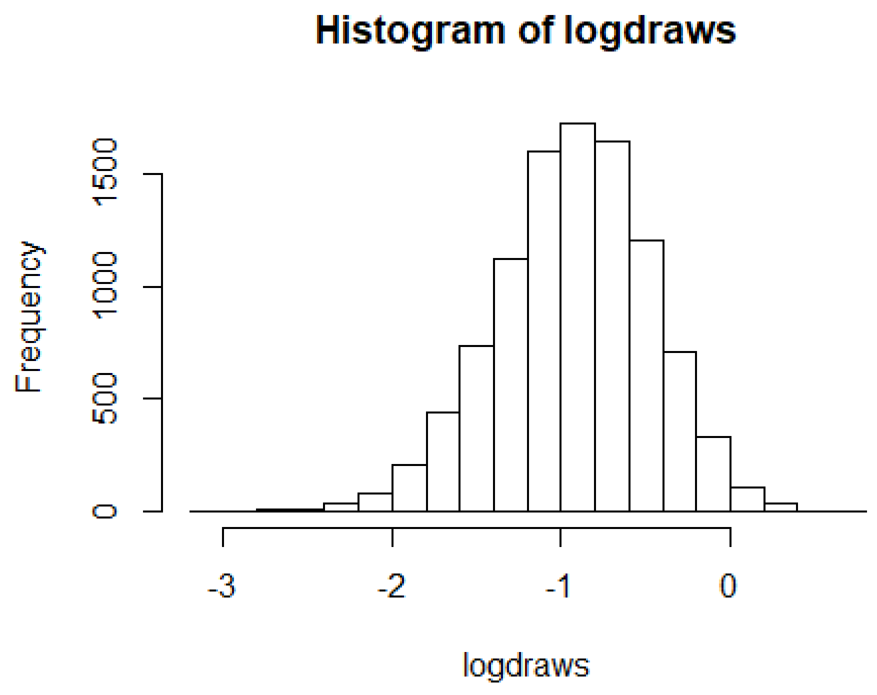
When similating nDraws = 10000 and counting the cases where the estimated probablity is bigger than 0,3 we see that the value is fairly close to the true value given the data y. Depening on the amount of draws we see that the posterior probaliblity for theta>0,3 given the data will be closer and closer to the true value.

```
## [1] 0.4433

## [1] 0.4399472
```

c) Compute the posterior distribution of the log-odds phi = log(phu / (1-phi) by simulation (nDraws = 10000). [Hint: hist() and density() might come in handy]

The log-odds posterior distribution can be seen in the plot. It looks like the same distribution as the data of shown in histogram.


Log-odds posterior distribution
Density
0.0
0.2
0.4
0.6
0.8
-3
-2
-1
0
phi


Histogram of logdraws
Frequency
0
500
1000
1500
-3
-2
-1
0
logdraws

Assignment 2

Assume that you have asked 10 randomly selected persons about their monthly income (in thousands Swedish Krona) and obtained the following ten observations: 44, 25, 45, 52, 30, 63, 19, 50, 34 and 67. A common model for non-negative continuous variables is the log-normal distribution. The log-normal distribution $\log\mathcal{N}(\mu, \sigma^2)$ has density function

$$p(y|\mu, \sigma^2) = \frac{1}{y \cdot \sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(\log y - \mu)^2\right],$$

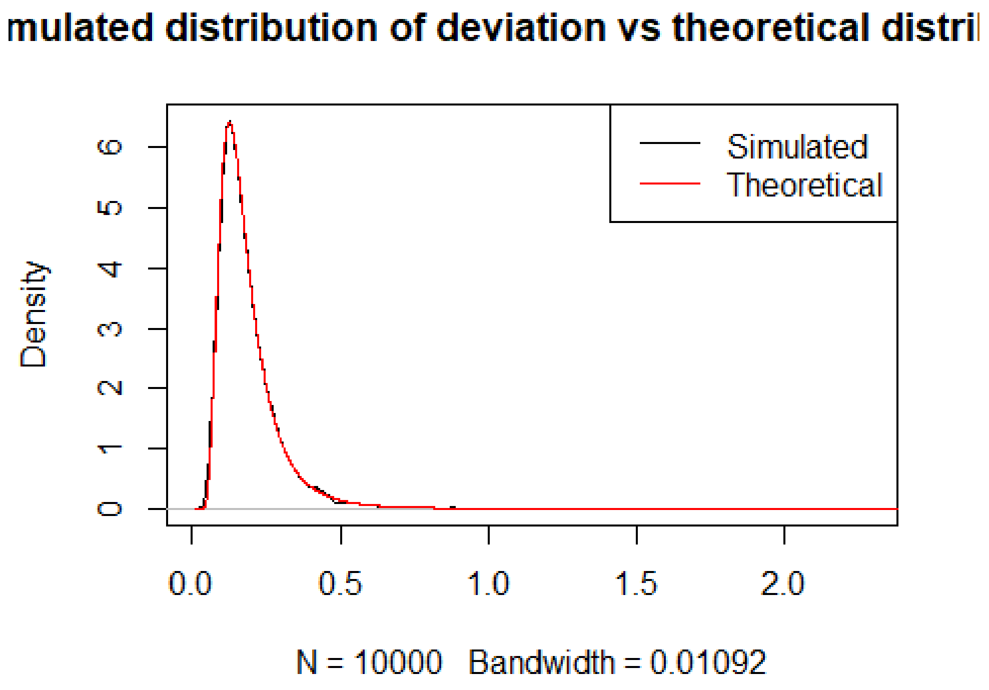
for $y > 0$, $\mu > 0$ and $\sigma^2 > 0$. The log-normal distribution is related to the normal distribution as follows: if $y \sim \log\mathcal{N}(\mu, \sigma^2)$ then $\log y \sim \mathcal{N}(\mu, \sigma^2)$. Let $y_1, ..., y_n|\mu, \sigma^2 \overset{iid}{\sim} \log\mathcal{N}(\mu, \sigma^2)$, where $\mu = 3.7$ is assumed to be known but σ^2 is unknown with non-informative prior $p(\sigma^2) \propto 1/\sigma^2$. The posterior for σ^2 is the $Inv - \chi^2(n, \tau^2)$ distribution, where

$$\tau^2 = \frac{\sum_{i=1}^{n}(\log y_i - \mu)^2}{n}.$$

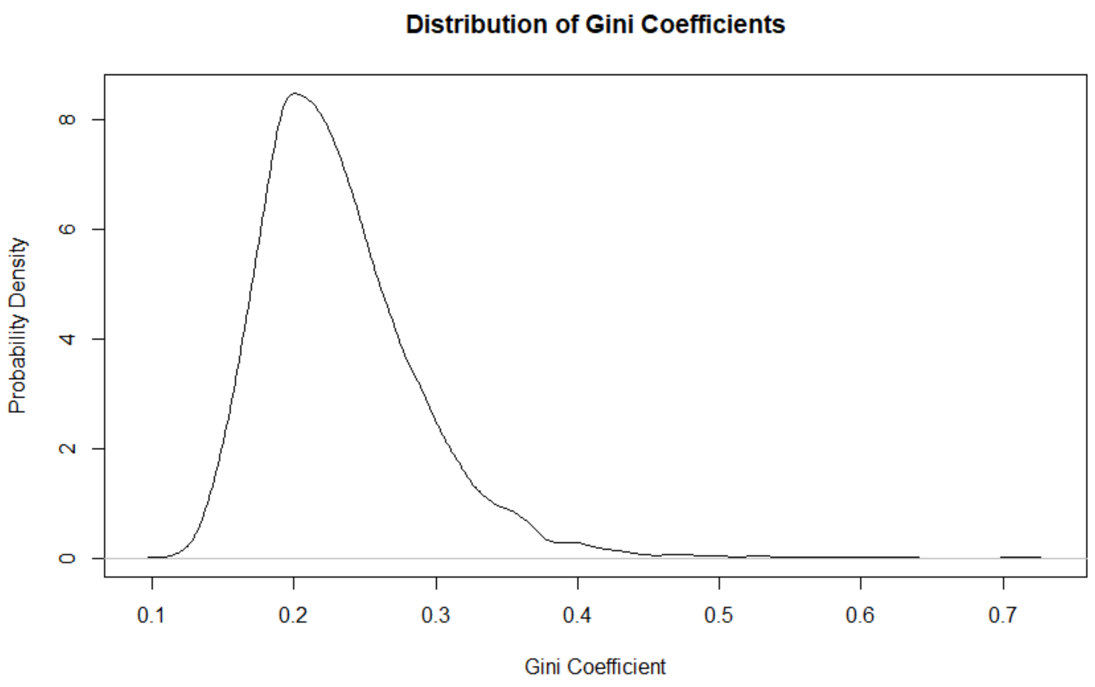
Simulate 10,000 draws from the posterior of σ^2 (assuming $\mu = 3.7$) and compare with the theoretical $Inv - \chi^2(n, \tau^2)$ posterior distribution.

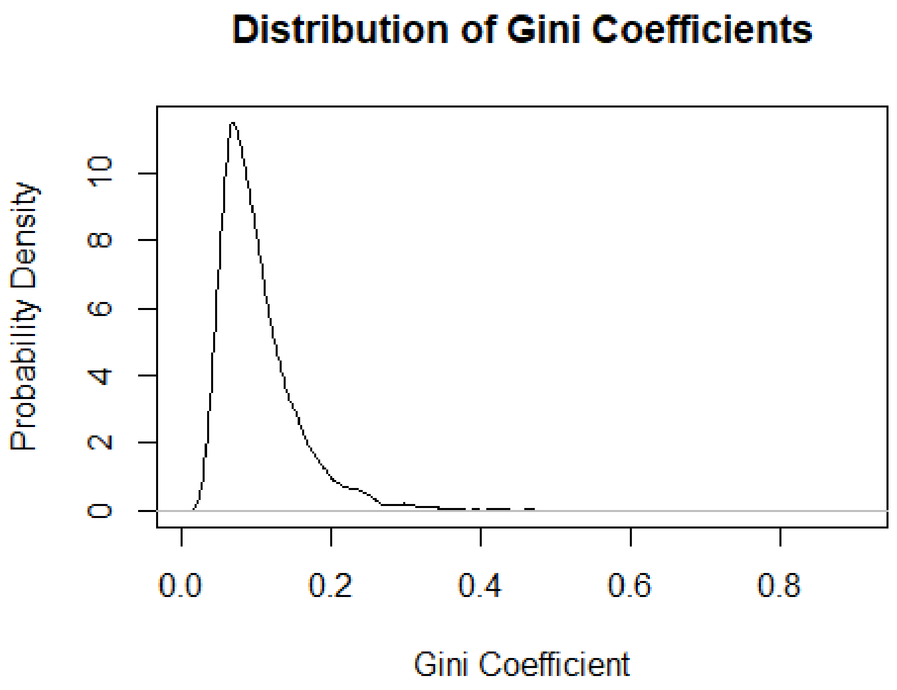
(2a)

We simulated a distribution from the posterior by drawing 10000 values and using density() to get the distribution. We are aware that density() only fits the data and does not yield the true distribution, but since n is so big (10000) we believe that density() will fit the model realisticly. Using density makes it easier to compare distribution than if we would use a histogram. We compared this distribution with the theoretical distribution and found that they were almost identical.



2b) Use the posterior draws in a) to compute the posterior distribution of the Gini coefficient G for the current data set.





When plotting the posterior distribution of the Gini Coefficients we can observe that it is very similar to the plot of the posterior distribution sigma^2. It is reasonable that the distribution is similar since the Gini Coeffiecients is just a transfromaion of the sigma^2, since the income has a conjugate prior which is scaled inverse chi-squared.

2c)

Use the posterior draws from b) to compute a 90% equal tail credible interval for G. A 90% equal tail interval (a, b) cuts off 5% percent of the posterior probability mass to the left of a, and 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 90% Highest Posterior Density interval for G. Compare the two intervals

We obtained a credible interval of (0.046,0.21) and a highest posterior density interval of (0.33,0.176). We used the default kernel (Gaussian) with the default bandwidth for the HDI. In symmetric distributions, credible interval and HDI will return same results. With skewed distributions such as this one the HDI will move to the more probable values while credible interval is limited to the 5 and 95 percentile. Since the HDI covers more probable values we believe that this interval seems to be a more intuitive and meaningful summary of the posterior.

```
## Warning: package 'HDInterval' was built under R version 3.6.3
```

Assignment 3

(3a) Plot the posterior distribution of κ for the wind direction data over a fine grid of κ value

The posterior distribution of k is obtained through the proportinality on likelihood vector of y|k multiplied by probability distribution p(k). Since the prior distribution of k is given the probability vector of k can be calculated. The likelihood vector of y|k is obtained by feeding the given likelihood function with the k values used to create the probability vector of k. By the final mltiplikation likelilhood(y|k)*p(k) we the get the posterior density for all values of k.

p(k|y) proportial to p(y|k)*p(k)

Bayesian inference for the concentration parameter in the von Mises distribution. This exercise is concerned with directional data. The point is to show you that the posterior distribution for somewhat weird models can be obtained by plotting it over a grid of values. The data points are observed wind directions at a given location on ten different days. The data are recorded in degrees:

$$(40, 303, 326, 285, 296, 314, 20, 308, 299, 296),$$

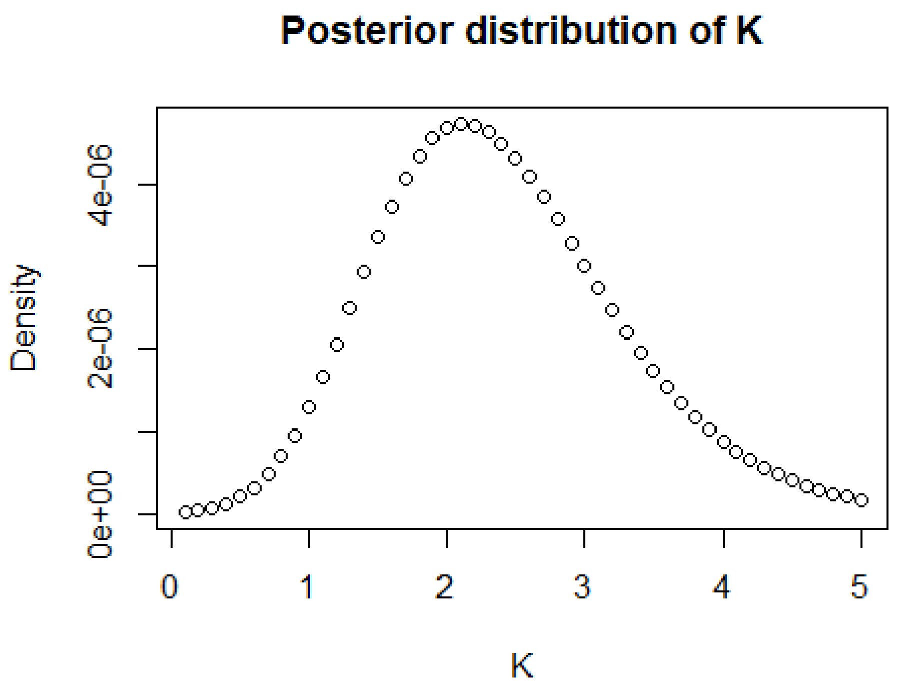
where North is located at zero degrees (see Figure 1 on the next page, where the angles are measured clockwise). To fit with Wikipedias description of probability distributions for circular data we convert the data into radians $-\pi \leq y \leq \pi$. The 10 observations in radians are

$$(-2.44, 2.14, 2.54, 1.83, 2.02, 2.33, -2.79, 2.23, 2.07, 2.02).$$

Assume that these data points are independent observations following the von Mises distribution

$$p(y|\mu, \kappa) = \frac{\exp\left[\kappa \cdot \cos(y - \mu)\right]}{2\pi I_0(\kappa)}, \quad -\pi \leq y \leq \pi,$$

where $I_0(\kappa)$ is the modified Bessel function of the first kind of order zero [see `?besselI` in R]. The parameter μ ($-\pi \leq \mu \leq \pi$) is the mean direction and $\kappa > 0$ is called the concentration parameter. Large κ gives a small variance around μ, and vice versa. Assume that μ is known to be 2.39. Let $\kappa \sim \text{Exponential}(\lambda = 1)$ a priori, where λ is the rate parameter of the exponential distribution (so that the mean is $1/\lambda$).



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

By finding the maximum value of the density and extracting its index we can then get the k value which results in the highest probability

```
# Bernoulli ... again.
# Let y1, ..., yn|?? ??? Bern(??), and assume that you have obtained a sample
with s = 5
# successes in n = 20 trials. Assume a Beta(??0, ??0) prior for ?? and let
??0 = ??0 = 2.

# a)
# Draw random numbers from the posterior ??|y ??? Beta(??0 + s, ??0 + f), y =
(y1, . . . , yn),
# and verify graphically that the posterior mean and standard deviation
# converges to the true values as the number of random draws grows large.

vec = c()
deviation = c()
number = c()

# Prior
a = b = 2
s = 5
n = 20
f = n-s

# Posterior
posterior_alpha = s + a
posterior_beta = b + f

for (i in 1:1000){
  post = rbeta (i, posterior_alpha, posterior_beta)
  mean =mean(post)
  st = sd(post)
  vec = append(vec,mean)
  deviation = append(deviation,st)
  number = append(number, i)
}

# Plot theta means
plot(number, vec, type='l', xlab='Number of draws', ylab='Mean of theta')

# Plot theta deviations
plot(deviation)
plot(number, deviation, type='l', xlab='Number of draws', ylab='deviation')


#USing the alpha and B from teh psoterior distribution
```

```r
variance_posterior= ((a+s)*(b+f)/((a+s+b+f)^2*(a + s + b +f +1)))^(1/2)
expected_mean = (a +s)/(a + s + b +f)


# The posterior a beta(alpha + s, beta + f). According to the table of
distributions we see formulas for
#E[posterior] and v[posterior] => E[posterior] = (a +s)/(a + s + b +f).
#V[posterior] = (a+s)(b+f)/(a+s+b+f)^2*(a + s + b +f +1)
# Plot these and see that as n gets large the values converge to this


# b)
# Use simulation (nDraws = 10000) to compute the posterior probability
# Pr(?? > 0.3|y) and compare with the exact value [Hint: pbeta()].

draws = rbeta (10000, posterior_alpha, posterior_beta)
bigger_than = ifelse(draws>0.3,1,0)
prob_bigger_than = sum(bigger_than)/length(draws)
prob_bigger_than


theta_bigger_than = pbeta(0.3, posterior_alpha, posterior_beta, ncp = 0,
lower.tail = FALSE, log.p = FALSE)
print(theta_bigger_than)

# c)
# Compute the posterior distribution of the log-odds ?? = log(?? / 1?????)
# by simulation (nDraws = 10000). [Hint: hist() and density() might come in
handy]

logdraws = log(draws/(1-draws))
densityLogDraws = density(logdraws)
plot(densityLogDraws,
     main = "Log-odds posterior distribution",
     ylab = "Density",
     xlab = "phi")

hist(logdraws)

## ASSIGNEMNT 2

# Log-normal distribution and the Gini coefficient.
# Assume that you have asked 10 randomly selected persons about their monthly
income
# (in thousands Swedish Krona) and obtained the following ten observations:
44,
# 25, 45, 25, 30, 33, 19, 50, 34 and 67. A common model for non-negative
continuous
# variables is the log-normal distribution
# a)
# Simulate 10, 000 draws from the posterior of ??2
# (assuming 弈牯 = 3.7) and compare with the theoretical
# Inv ??? ??2(n, ?? 2) posterior distribution.

my = 3.7
n = 10

Y = c(44,25,45,52,30,63,19,50,34,67)
#compute sample variance s^2
T2 =  sum((log(Y)-my)^2)/n

#Draw X from chi2(n) (This is a draw from Inv- X^2(n,s^2))
set.seed(12345)
XposteriorDraw = rchisq(10000,10)

# get deviation^2 from X ( deviation^2 = df*s^2 /X)
deviationPostDraw = 10*T2/XposteriorDraw
distDeviationPostDraw = density(deviationPostDraw)

library(invgamma)

# function for scaled inverse chi-squared pdf
invscaledchi2 <- function(x, df, tao2) {
  a <- df / 2
  ((n*tao2 / 2)^a)/gamma(a) * x^(-a-1) * exp(-(n*tao2 / 2)/x)
}


# sequence of x-values to illustrate the distribution
xrange = seq(0.01,3.0,0.001)

# values from inverse chisquared mapped on x-range
deviations = invscaledchi2(xrange,10,T2)

# plot simulated distribution with theoretical distribution
plot(distDeviationPostDraw)
lines(xrange, deviations, type="l",col="red")


# b)
# Use
#the posterior draws in a) to compute the posterior distribution of the Gini
#coefficient G for the current data se

#Use draws from posterior to form GiniCoeffient
Gini_coefficients = 2*pnorm(sqrt(deviationPostDraw)/sqrt(2),0,1)-1
Gini_density = density(Gini_coefficients)
```

```r
plot(Gini_density, main = "Distribution of Gini Coefficients", xlab = "Gini
Coefficient", ylab = "Probability Density")

#c
#Use the posterior draws from b) to compute a 90% equal tail credible
interval
#for G. A 90% equal tail interval (a, b) cuts off 5% percent of the posterior
#probability mass to the left of a, and 5% to the right of b.

GiniQuantiles = quantile(Gini_coefficients, probs=seq(0,1,0.05))
Crediblerange = c(GiniQuantiles[2],GiniQuantiles[20])
# gets range (0.1385257,0.3431806)

#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 90%
Highest
#Posterior Density interval for G. Compare the two intervals

# using library from CRAN to get highest (posterior) density interval
library(HDInterval)
hdiRange = hdi(Gini_density, credMass=0.9)
# gets range (0.1371345,0.3431806)

hist(Gini_coefficients)
plot(Gini_density)

vec = c()
#Alternatively using sort of the densities
sorted_normalized_y = sort(Gini_density$y, decreasing =
TRUE)/sum(Gini_density$y)
sorted_x = Gini_density$x[order(-Gini_density$y)]

count = 0
summa = 0
while(summa <= 0.95){
  count = count + 1
  summa = sorted_normalized[count] + summa

}
a = min(sorted_x[1:count-1])
b = max(sorted_x[1:count-1])

# Assignment 3
# a)
# Plot the posterior distribution of ?? for the wind direction data over a
fine grid
# of ?? values
```

```r
y = c(-2.44, 2.14, 2.54, 1.83, 2.02, 2.33, -2.79, 2.23, 2.07, 2.02)
my = 2.39
lambda = 1

k_prior = function(lambda, k){
  lambda*exp(-lambda*k)
}

likelihood_function = function(k, y, my){
  prod(exp(k*cos(y-my))/(2*pi*besselI(k, 0)))
}

sequence = seq(0.1, 5, 0.1)

k_prior_vector = k_prior(lambda, sequence)

likelihood_vector = c()
test = c()
for (k in sequence){
  likelihood_vector = append(likelihood_vector, likelihood_function(k, y,
my))
}

k_posterior = likelihood_vector*k_prior_vector
plot(sequence, k_posterior, main = "Posterior distribution of K", xlab = "K",
ylab = "Density" )


# b)
# Find the approximate  posterior mode of k  from the information in a

k_index = which(k_posterior == max(k_posterior))
k_mode = sequence[k_index]
k_mode
print(k_mode)
```

Lab2

TDDE07

Computer Lab 2

Authors:

Philip Kantola phika529

Arun Uppugunduri aruup817

Assignment 1

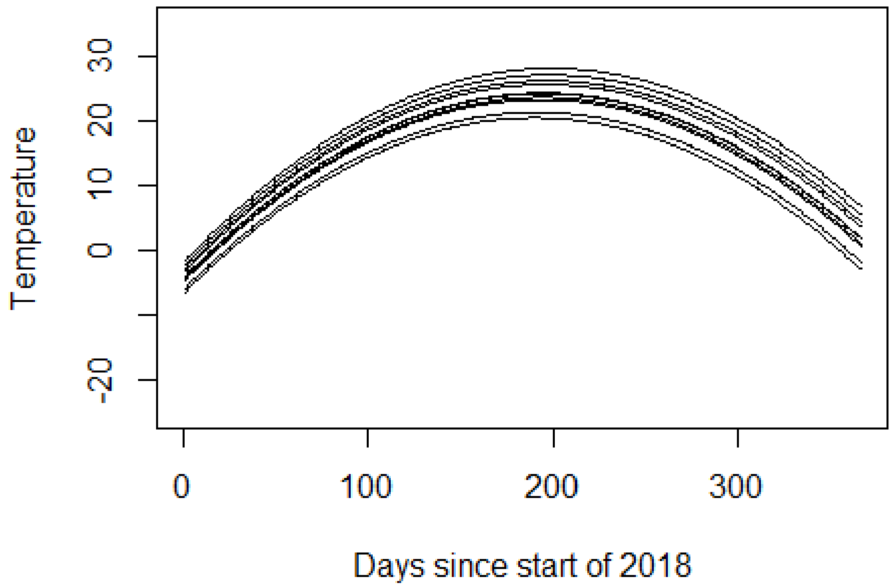
1a

We started by drawing 10 draws from our given scaled chi inverse function. For each o

check(These are draws from our marginal distribution of sigma.)

We simulated 10 different beta-vectors with our prior hyperparameters and plotted them (see plot below). The low value of the values in precision matrix means that we are unsure on our prior, this is the reason that our plotted curves gives very different results. Altering v0, sigma0 and our precision matrix will render the same result in the end, the variance of the beta parameters. The hyperparameters which we can change to make the curves fit our prior believes are essentially the rho vector. In the plot below we used the default values of [-10,100,-100]. Based on historical data we saw that these given values gave us what we belived to be the best distribution.

```
## Warning: package 'mvtnorm' was built under R version 3.6.2
```



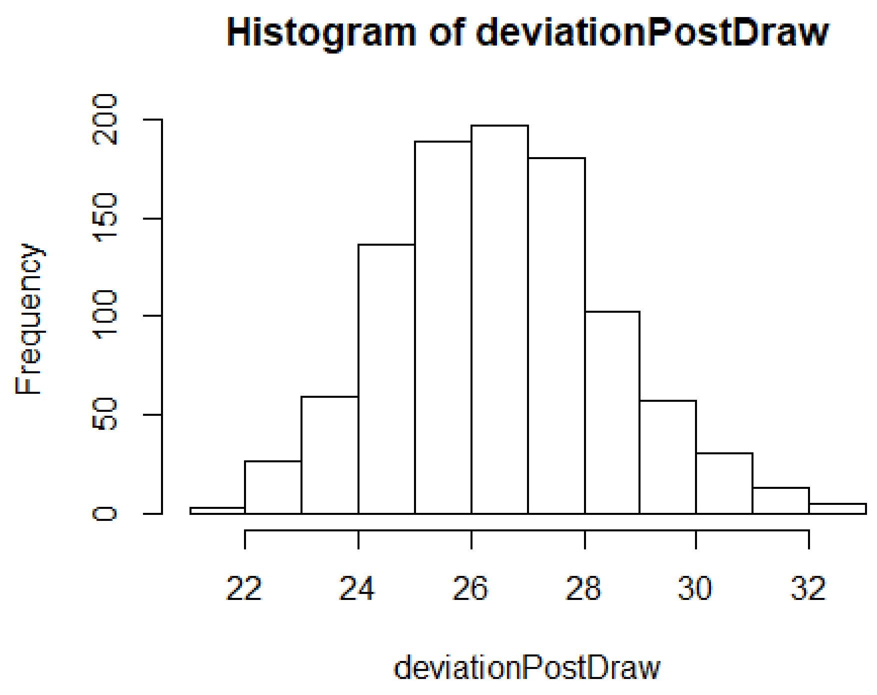
b

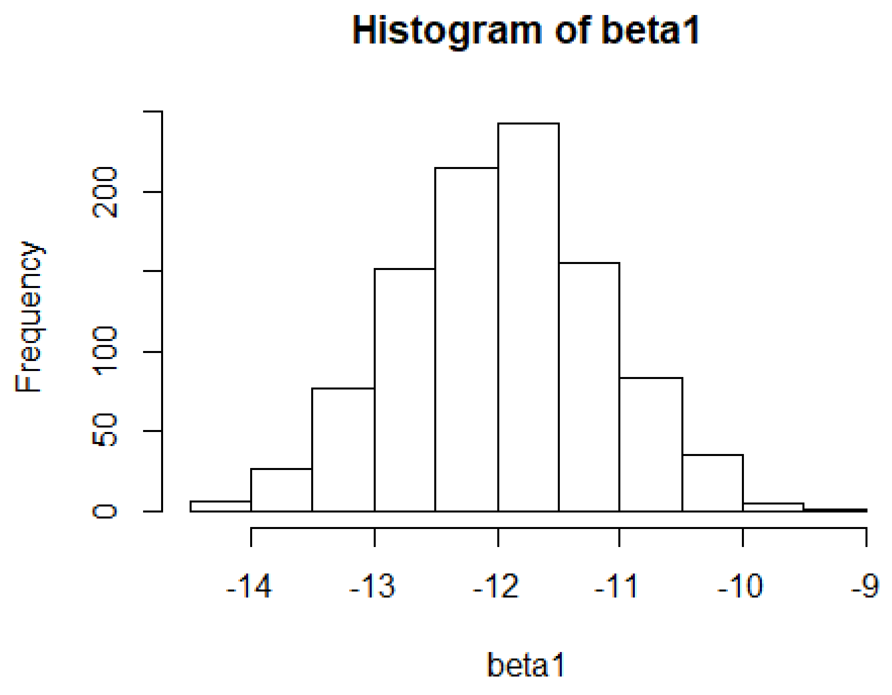
Write a program that simulates from the joint posterior distribution of beta0, beta1, beta2 and sigma^2 Plot the marginal posteriors for each parameter as a histogram.

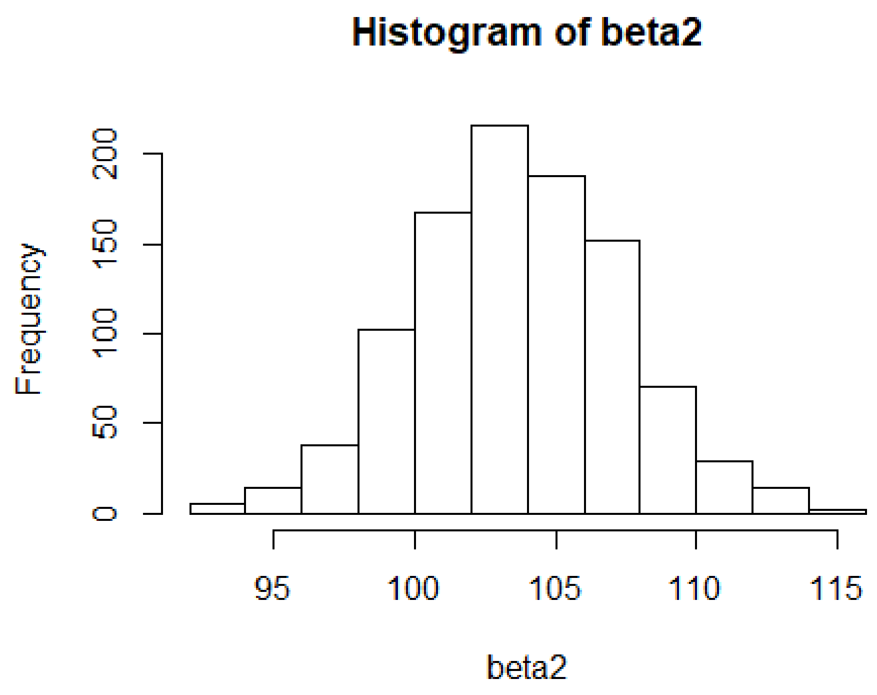
Also produce a figure with the scatterplot of the temperature data and overlay it with the curve for the posterior median of the regression function

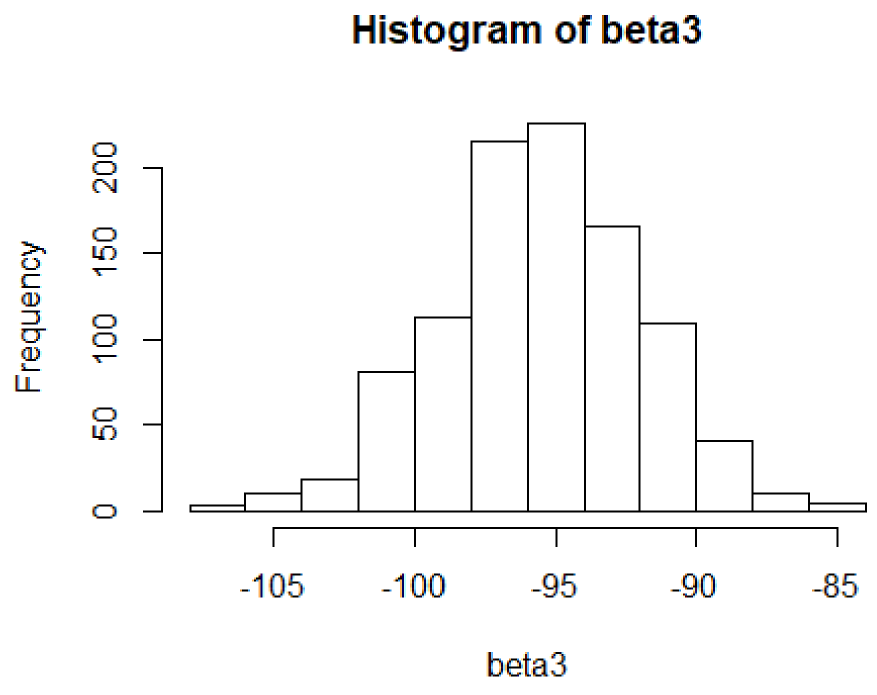
Also overlay curves for the lower 2.5% and upper 97,5% credible interval for f(time)

From the histograms we have obtained all parameters was shown to have distributions similar to the normal distribution. This was expected since the betas was drawn from a normal distribution and since sigma^2 comes from a scaled inverse chi^2 distribution which can be approximated to a normal distribution if n is big. (we chose n as 1000 in this case)

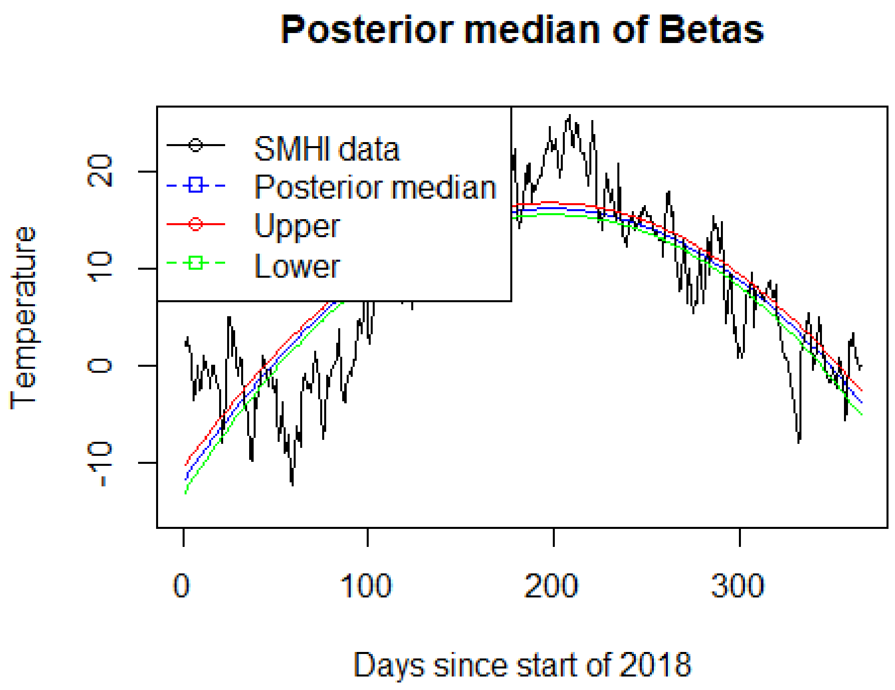

Histogram of deviationPostDraw
Frequency
0
50
100
150
200
22
24
26
28
30
32
deviationPostDraw


Histogram of beta1
Frequency
0
50
100
200
-14
-13
-12
-11
-10
-9
beta1





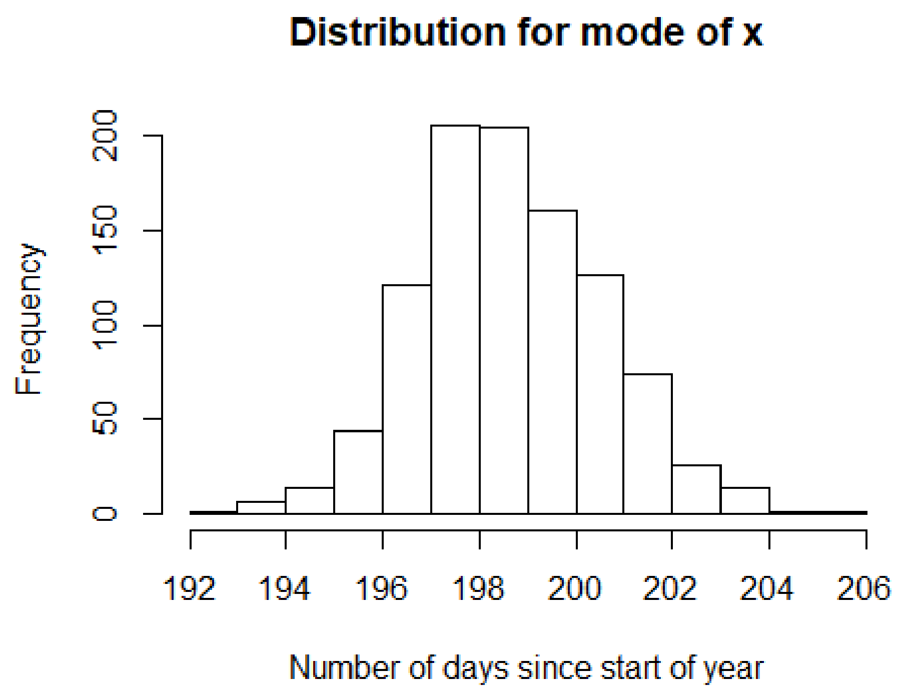
We took the median of f(x) computed for all days and plotted the resulting curve together with the actual data. In the same plot we also plotted the 2.5 and 97.5 percentile of f(x), the credible interval. See plot below. The credible interval does not cover 95% of the data which it is not meant to do, since the interval will cover the 95% probability interval of the curve which describes the weather.



#1c

It is of interest to locate the time with the highest expected temperature (that is, the time where f(time) is maximal). Let's call this value xmax. Use the simulations in b) to simulate from the posterior distribution of xmax.

Seen below is the histogram for the posterior distribution of the posterior mode. Based on the result it appears to be normally distributed. Looking at the plotted curves for f(time) the historgram looks reasonable showing a max temperature around time = 199 days.



1d

Say that you want to estimate a polynomial model of order 7 but suspect that higer order terms might not be needed and worry about overfitting. Suggest a suitable prior that mitgates this potential problem.

This is solved by adding a penalizing factor lambda adding more shrinkage and gaining more precision in the model. Since lambda is unknown we simply assign a prior to it which will then be have the distibuted acording to the scaled inverse chi square with parameters n_ and lambda_0. Slide 91. Omega_0 is specified as lamda*Identity matrix and using hyperparameer my_0 = (0,0,0) (Slide 91).

the prior is as follows: beta|sigma^2 ~ N(0,sigma^2/lambda). We We then say that Omega_0 = lambda*identity vector and we get a joint prior of Beta, Sigma^2, lambda => p(B, sigma^2, lambda| data).

Assignment 2

2a

Since the prior for beta is normal and the likelihood a product of the logistic regression the posterior is difficult to compute, instead we obtained a large sample approximate posterior for the beta vector. We first optimized over the log-logistic regression expression and obtained the following numerical values for the beta mode and inverted Hessian.(first row is mode for beta vector and the following rows is the Hessian)

```
## [1]  0.62672884 -0.01979113  0.18021897  0.16756670 -0.14459669 -0.0820656
1
## [7] -1.35913317 -0.02468351
```

```
##              [,1]       [,2]       [,3]        [,4]        [,5]       [,6]
## [1,]   37.697939   793.0147   469.6456   367.20017   53.450998  1582.5424
## [2,]  793.014693 21726.2184 10340.0749  7713.59763 1095.613234 33614.3881
## [3,]  469.645561 10340.0749  6070.3314  4544.61312  648.975117 19684.1213
## [4,]  367.200173  7713.5976  4544.6131  5345.13635  981.781722 16227.6889
## [5,]   53.450998  1095.6132   648.9751   981.78172  214.577828  2472.3781
## [6,] 1582.542441 33614.3881 19684.1213 16227.68889 2472.378069 68845.6824
## [7,]    9.469258   191.1836   127.1882    74.89198    7.675926   321.3491
## [8,]   48.177234   995.4926   584.1196   373.17114   42.890848  1893.1488
##             [,7]       [,8]
## [1,]   9.469258   48.17723
## [2,] 191.183648  995.49260
## [3,] 127.188152  584.11957
## [4,]  74.891981  373.17114
## [5,]   7.675926   42.89085
## [6,] 321.349083 1893.14881
## [7,]  11.923173   12.79632
## [8,]  12.796324  123.04393
```

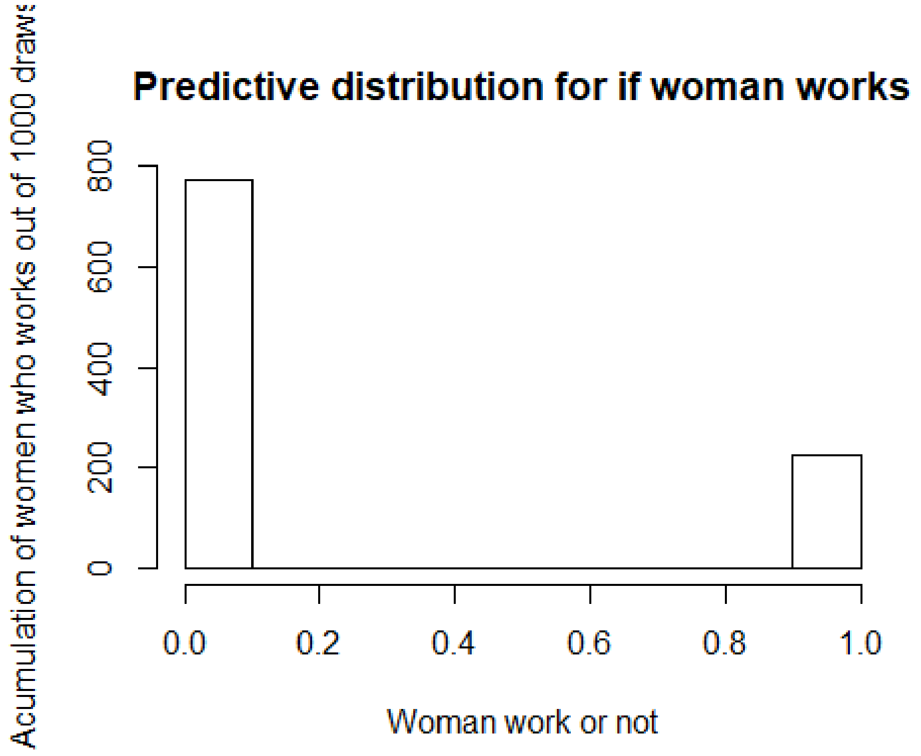
We approximated an 95% credible interval for the variable NSmallChild (or rather, for the coefficient) and obtained the range below. We compared with GLM-model and concluded that this is a reasonable result. the effect of the number of small childs prove to be very large, if a woman has two small children the probability of her working becomes very small, this is obvious when comparing to the weight of the other variables from GLM. (About the values below, the first row represents the credible interval, the second row are the estimated beta-values obtained from GML)

```
##       2.5%      97.5%
## -2.1296887 -0.5759195
```

```
## (Intercept)    Constant  HusbandInc   EducYears    ExpYears   ExpYears2
##  0.64430363          NA -0.01977457  0.17988062  0.16751274 -0.14435946
##         Age NSmallChild   NBigChild
## -0.08234033 -1.36250239 -0.02542986
```

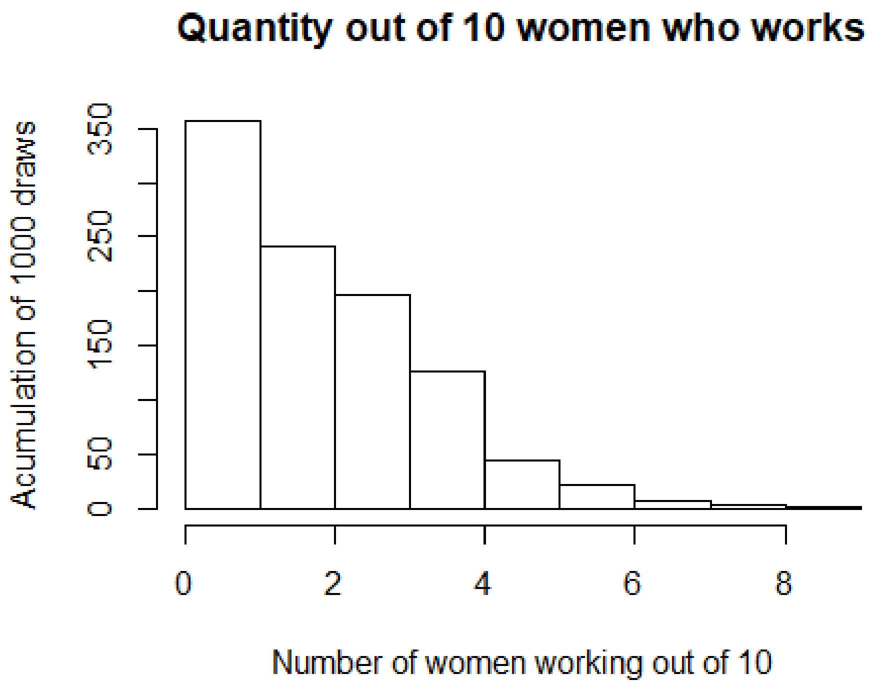
2 b) Write a function that simulates from the predictive distribution of the response variable in a logistic regression. Use your normal approximation from 2(a). Use that function to simulate and plot the predictive distribution for the Work variable for a 40 year old woman, with two children (3 and 9 years old), 8 years of education, 10 years of experience. and a husband with an income of 10.

We simulated values for our response variable y from the logistic regression with the help of draws from the posterior of beta vector. As seen in the histogram below the probability that this type of woman works is low.



2 c Now, consider 10 women which all have the same features as the woman in 2(b). Rewrite your function and plot the predictive distribution for the number of women, out of these 10, that are working. [Hint: Which distribution can be described as a sum of Bernoulli random variables?]

We simulated the quantity out of 10 woman with same data whom works with a binomial distribution where p was drawn from the posterior distribution of work probability. Results can be seen in the following histogram.



Assignment 1

```
# for multivariate normal functions
library(mvtnorm)

V_0 = 4
sigma_0 = 1
my_0 = c(-10,100,-100)
set.seed(12345)
omega_0 = 0.01*diag(3)
omega_0_Inverse = solve(omega_0)

set.seed(12345)
#Draw 10 draws from out given chi sqared distribution
XDraw = rchisq(10,V_0)
#Transform our 10 draws to the scaled inverse chi square
deviationDraw = V_0*sigma_0/XDraw

# For each of the draws above we create the joint prior obtaining the distrib
ution for Beta given sigma
# each draw gives us a set of values for the betas
#rmvnorm because of that my_0 is a vector => multivariate
# sigma is covarance matrix
joint_prior = list()
for(i in 1:10){
  set.seed(12345)
  joint_prior[[i]] =  rmvnorm(1, mean = my_0, sigma = deviationDraw[i]*omega_
0_Inverse)
}

# Function for computing the regression function for different values of x (t
ime)
function_temperature = function(betas,x){
  betas[1] + betas[2]*x + betas[3]*x^2
}

#Plot the temperature regression curve for every set of values for betas
time = seq(0,1,1/365)
plot(function_temperature(joint_prior[[1]][1,],time), type = 'l', ylim = c(-2
5,35), xlab = "Days since start of 2018", ylab = "Temperature")
for(i in 2:10){
  lines(function_temperature(joint_prior[[i]][1,],time), type = 'l')

}


# (b)

data = read.table("TempLinkoping.txt", header = TRUE)
X = data$time
data$xsquare = X^2
data$intercept = 1
Y = data$temp
X_matrix = matrix(0, nrow = 365, ncol = 3)
X_matrix[,1] = data$intercept
X_matrix[,2] = data$time
X_matrix[,3] = data$xsquare

beta_hat = solve((t(X_matrix)%*%X_matrix))%*%t(X_matrix)%*%Y

# get value for my_n
my_n = solve(((t(X_matrix)%*%X_matrix)) + omega_0)%*%(((t(X_matrix)%*%X_matri
x%*%beta_hat))+omega_0%*%my_0)

omega_n = (t(X_matrix)%*%X_matrix) + omega_0
omega_n_inverse = solve(omega_n)

# degrees of freedom
Vn = V_0 + length(X)


sigma_n = ((V_0*sigma_0 + (t(Y)%*%Y + t(my_0)%*%omega_0%*%my_0)[1] - (t(my_n)
%*%omega_n%*%my_n)[1]))/Vn

# number of draws
```

```r
n = 1000

# draw values from the marginal posterior distribution of sigma
set.seed(12345)
#Draw n draws from our given chi sqared distribution
XPostDraw = rchisq(n,Vn)
#Transform our n draws to the scaled inverse chi square
deviationPostDraw = Vn*sigma_n/XPostDraw

# For each of the draws above we create the joint marginal posterior obtainin
g the distribution for Beta given sigma
# each draw gives us a set of values for the betas
joint_posterior = matrix(0, nrow = n, ncol = 3)
beta1 = c()
beta2 = c()
beta3 = c()


set.seed(12345)
for(i in 1:n){
  joint_posterior[i,] =  rmvnorm(1, mean = my_n, sigma = omega_n_inverse*devi
ationPostDraw[i])
  beta1 = append(beta1,joint_posterior[i,1])
  beta2 = append(beta2,joint_posterior[i,2])
  beta3 = append(beta3,joint_posterior[i,3])
  temp = temperature_posterior(joint_posterior[i,], X_matrix)
}


# visualize the simulated parameters in histograms
hist(deviationPostDraw)
hist(beta1)
hist(beta2)
hist(beta3)

# Function f(time) for certain beta values on every values of x (time)
temperature_posterior = function(betas,x){
  x%*%betas
}

temperature_matrix = matrix(0, nrow = 365, ncol = 1000)
for(i in 1:n){
  temperature_matrix[,i] = temperature_posterior(joint_posterior[i,], X_matri
x)
}

#Get median value of f(x) computed for each day
median_temperature_vec = apply(temperature_matrix,1, median)
```

```
CI_temperature_vec = apply(X = temperature_matrix, MARGIN=1, FUN=function(x)
quantile(x,c(0.05,0.95)))


#Plot posterior median, upper/lower credible interval for the Beta values and
SMHI data
plot(median_temperature_vec, type = 'l', ylim = c(-15,25), main = "Posterior
median of Betas", xlab = "Days since start of 2018", ylab = "Temperature", co
l = "blue")
lines(Y, col = "Black")
lines(CI_temperature_vec[1,], col = "green")
lines(CI_temperature_vec[2,], col = "red")
legend("topleft", c("SMHI data", "Posterior median", "Upper", "Lower"), col =
c("black", "blue", "red", "green"), pch = 21:22, lty = 1:2)


# 1 c)
# function which takes in beta values and returns which x (which day) gives t
he maximum value
maximal_time = function(beta2,beta3){

   x =  -beta2/(2*beta3)
}

# collect all the max x-values for every set of betas
max_vector = c()
for(i in 1:n){
  # multiply with 366 to get nr of days instead of range 0,1
  max_vector = append(max_vector,maximal_time((joint_posterior[i,2]),(joint_p
osterior[i,3]))*365)
}

hist(max_vector, main="Distribution for mode of x", xlab="Number of days sinc
e start of year")
```

Assignment 2

```
#Assignment 2

women_data = read.table('WomenWork.dat', header = TRUE)
women_data
n = nrow(women_data)

tao = 10
#covariance_matrix = c(1:8)
Y = as.vector(women_data[,1])
X = as.matrix(women_data[,-1])
col = ncol(X)
```

```r
# parameters for the prior distribution of beta
my = rep(0, col)
sigma_prior = diag(tao^2, nrow = col)

# function which returns an expression proportional to log beta posterior, th
is can be optimized for
# values of beta mode and hessian J (observed hessian evaluated at posterior
mode)
log.posterior = function(betas, x,Y,my,sigma_prior){

  # is simply log of the product of density function
  log_likelihood = sum((X%*%betas)*Y-log(1+exp(X%*%betas)))

  #log of deensity for nultivariate normal => dmvnorm(density multivariate)
  log_prior = dmvnorm(x=betas,mean=my,sigma=sigma_prior, log=TRUE)

  return(log_likelihood + log_prior)

}

# Different starting values. Ideally, any random starting value gives you the
same optimum (i.e. optimum is unique)
initVal <- as.vector(rep(0,col));

# function which optmizes over expression log.posterior with respect to its f
irst argument (betas).
# returns optimal values for beta (mode), and hessian in the mode
OptParams<-optim(initVal,log.posterior,gr=NULL,X,Y,my,sigma_prior,method=c("B
FGS"),control=list(fnscale=-1),hessian=TRUE)

betas_posterior = OptParams$par
# takes negative so that the posterior can be approx. as normal
# J = -second derivate evaluated in theta_hat
hessian_posterior = -OptParams$hessian

# take inverse for using it in the formula
hessian_posterior = solve(hessian_posterior)

#Draw samples from Betas posterior distribution.
set.seed(12345)
posterior_distribution = rmvnorm(n = 1000, mean = betas_posterior, sigma = (h
essian_posterior))

#Distribution gives for each draw a beta vector of length 8
#Take out vecotr of interest (no of small children)
n_child = posterior_distribution[,7]

#95% credible interval
quantiles = quantile(n_child, probs = seq(0,1,0.025))
```

```r
quantiles
interval = c(quantiles[2], quantiles[40])
interval

#Comparision with glm model
glmModel <- glm(Work ~ ., data = women_data, family = binomial)
glmModel$coefficients

#2b
#Write a function that simulates from the predictive distribution of the resp
onse
#variable in a logistic regression. Use your normal approximation from 2(a).
#Use that function to simulate and plot the predictive distribution for the W
ork
#variable for a 40 year old woman, with two children (3 and 9 years old), 8 y
ears
#of education, 10 years of experience. and a husband with an income of 10.

# expression for the response variable y (logistic regression)
work_distribution = function(x_data, betas){
  exp(t(x_data)%*%betas) / (1+exp(t(x_data)%*%betas))
}

# the properties of the type of woman we want the work distribution for
x_data = c(1,10.000,8,10,1,40,1,1)

outcome = c()

# Simulating from teh predictive distribution (hence giving teh actual outcom
es) for y with respect to our x values by simulating beta values from the bet
a posterior.
set.seed(12345)
for (i in 1:1000){
  betas = rmvnorm(n = 1, mean = betas_posterior, sigma = (hessian_posterior))
  betas = as.vector(betas)
  prob = work_distribution(x_data, betas)
  # y is binary, therefore bernoulli distributed. Same thing as binomial with
one draw.
  outcome = append(outcome, rbinom(1,1, prob))


}
# histogram representing prob of woman work
hist(outcome, main="Predictive distribution for if woman works", xlab="Woman
work or not", ylab="Acumulation of women who works out of 1000 draws")



# 2c
```

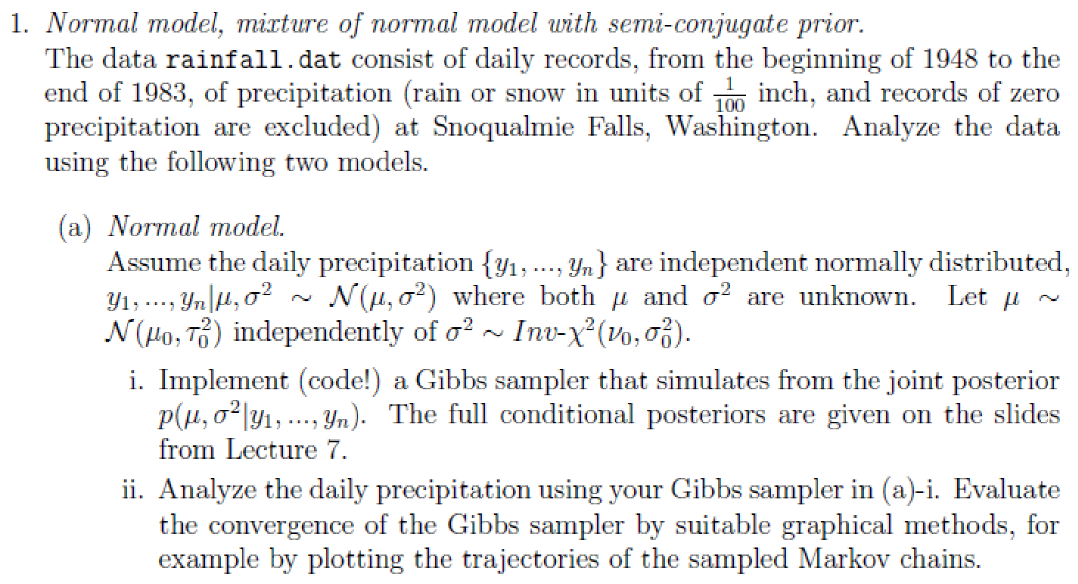
```
#Now, consider 10 women which all have the same features as the woman in 2(b)
.
#Rewrite your function and plot the predictive distribution for the number of
#women, out of these 10, that are working. [Hint: Which distribution can be
#described as a sum of Bernoulli random variables?]

# the distribution will be a binomial

# expression for the binomial dist of the quantity out of 10 woman of same ty
pe that works. Depends on our
# posterior dist for the probability that one woman of this type works
work_binomial = function(x_data, betas){
  p = work_distribution(x_data, betas)
  rbinom(1,10,p)
}

work_binomial_result = c()

# Simulating the distribution for quantity out of 10 woman works (binomial)
# using posterior dist for beta vectors and for work probability.(work distri
bution)
set.seed(12345)
for (i in 1:1000){
  betas = rmvnorm(n = 1, mean = betas_posterior, sigma = (hessian_posterior))
  betas = as.vector(betas)
  work_binomial_result = append(work_binomial_result,work_binomial(x_data,bet
as))
}

hist(work_binomial_result, main="Quantity out of 10 women who works", xlab="N
umber of women working out of 10", ylab="Acumulation of 1000 draws")

# one can check these quantiles to compare with prior histogram mode
#quantile(work_binomial_result, probs = seq(0,1,0.025))
```

aruup817
phika529

Lab3

Lab3

Assignment 1



1. *Normal model, mixture of normal model with semi-conjugate prior.*
   The data `rainfall.dat` consist of daily records, from the beginning of 1948 to the end of 1983, of precipitation (rain or snow in units of $\frac{1}{100}$ inch, and records of zero precipitation are excluded) at Snoqualmie Falls, Washington. Analyze the data using the following two models.

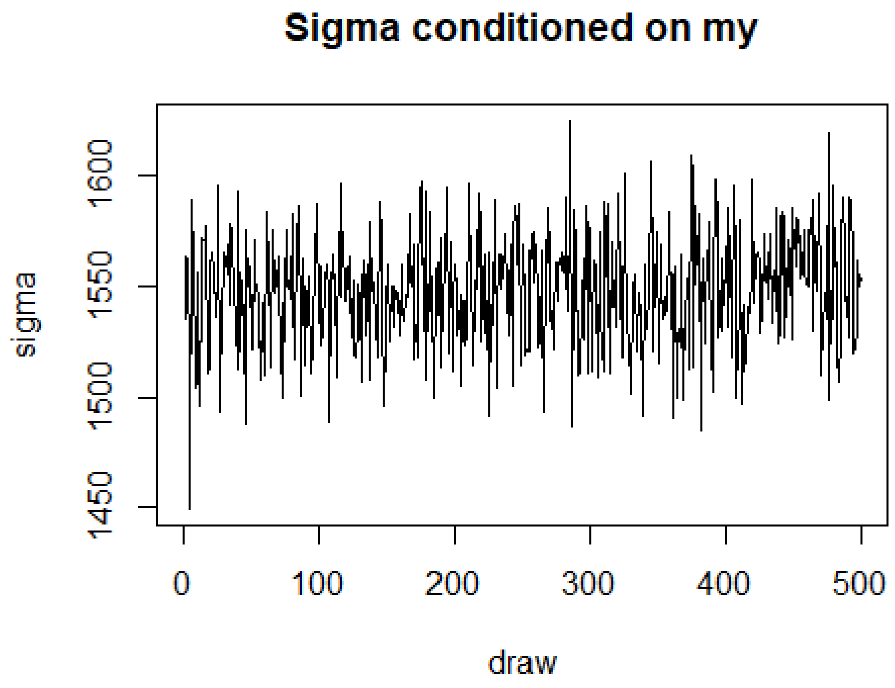
   (a) *Normal model.*
   Assume the daily precipitation $\{y_1, ..., y_n\}$ are independent normally distributed, $y_1, ..., y_n|\mu, \sigma^2 \sim \mathcal{N}(\mu, \sigma^2)$ where both μ and σ^2 are unknown. Let $\mu \sim \mathcal{N}(\mu_0, \tau_0^2)$ independently of $\sigma^2 \sim Inv\text{-}\chi^2(\nu_0, \sigma_0^2)$.

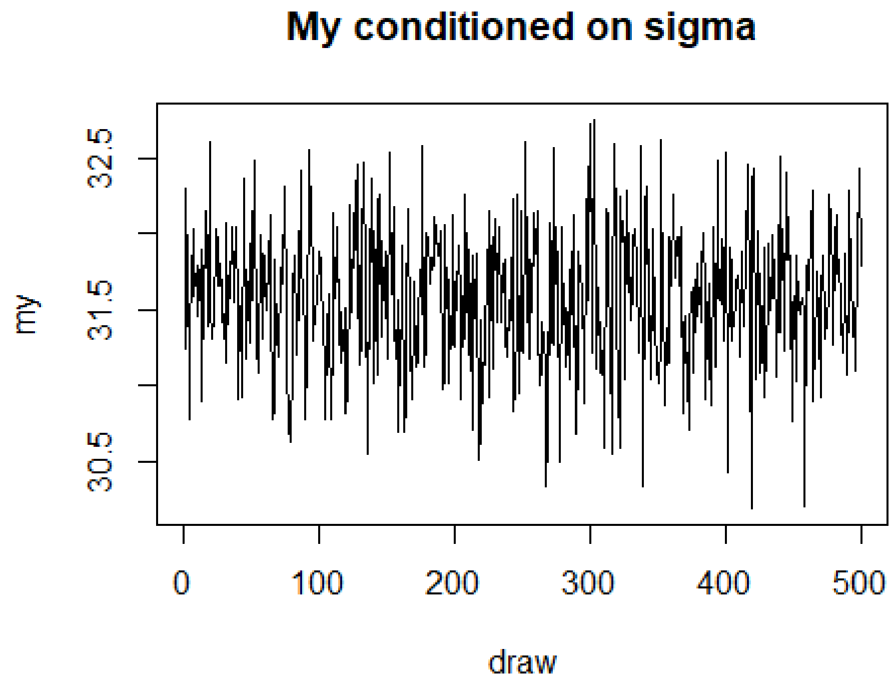
   i. Implement (code!) a Gibbs sampler that simulates from the joint posterior $p(\mu, \sigma^2|y_1, ..., y_n)$. The full conditional posteriors are given on the slides from Lecture 7.

   ii. Analyze the daily precipitation using your Gibbs sampler in (a)-i. Evaluate the convergence of the Gibbs sampler by suitable graphical methods, for example by plotting the trajectories of the sampled Markov chains.

We used low random initial values to calculate the priors, we then implemented a gibbs sampler which gave us our conditional posteriors for my and sigma. We plotted the results from 500 draws from the conditional posteriors as can be seen in the following graphs. The results seem to fall around a constant value with little to no autocorrelation. Compared with the properties of the given data our my and sigma seem to be somewhat correct.

aruup817
phika529





aruup817
phika529

(b) *Mixture normal model*
Let us now instead assume that the daily precipitation $\{y_1, ..., y_n\}$ follow an iid two-component **mixture of normals** model:

$$p(y_i|\mu, \sigma^2, \pi) = \pi\mathcal{N}(y_i|\mu_1, \sigma_1^2) + (1-\pi)\mathcal{N}(y_i|\mu_2, \sigma_2^2),$$

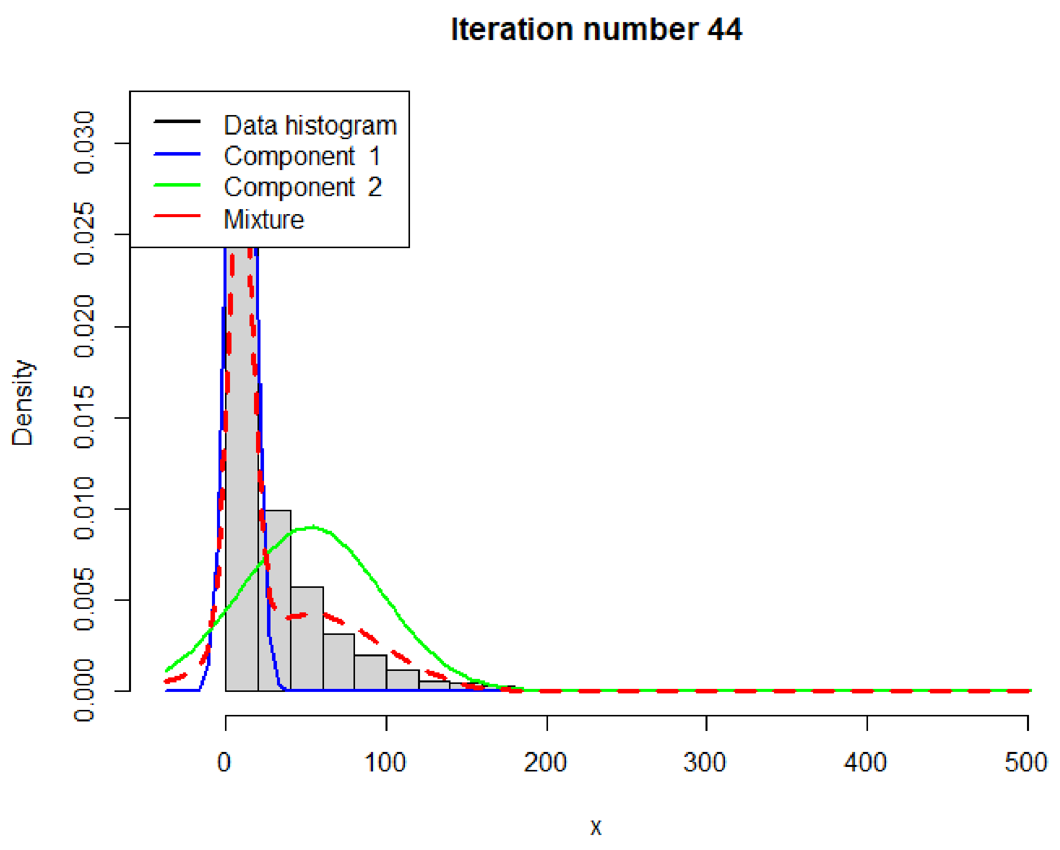
where

$$\mu = (\mu_1, \mu_2) \quad \text{and} \quad \sigma^2 = (\sigma_1^2, \sigma_2^2).$$

Use the Gibbs sampling data augmentation algorithm in `NormalMixtureGibbs.R` (available under Lecture 7 on the course page) to analyze the daily precipitation data. Set the prior hyperparameters suitably. Evaluate the convergence of the sampler.

We modified the hyper parameters unti the mixed graph displayed followed the data in a good way. As can be seen in the plot below we believe that the final model catches the true data in a good way.

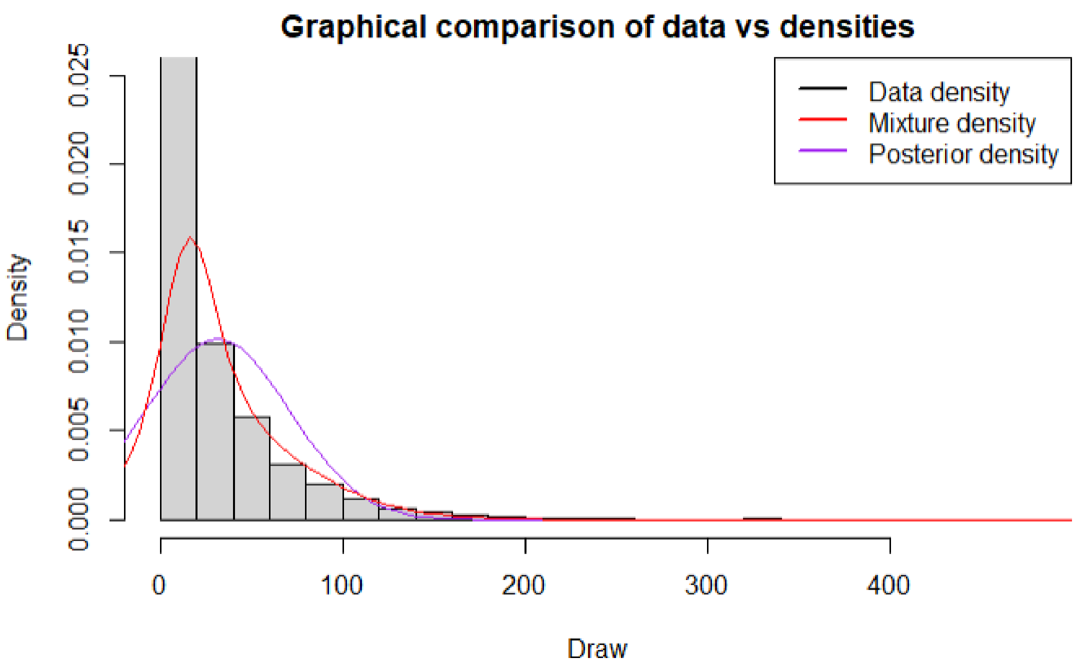
comment from teacher Evaluate the convergence of the samplet by saving the sampled parameters (theta. sigma2 and w) in Mattias code at each iteration and then plot the parameter trajectories over the iterations.



Plot the following densities in one figure: 1) a histogram or kernel density estimate of the data. 2) Normal density in (a). 3) Mixture of normals density in (b). Base your plots on the mean over all posterior draws.

aruup817
phika529

From the following graph we draw the conclusion that the mixture model from b) fits the data best. A normal distribution alone is not suitable to fit this data. Mixed normal model is doing a good job but a more convenient solution might be to fit an exponential distribution.



#Assignment 2

aruup817
phika529

2. *Metropolis Random Walk for Poisson regression.*
Consider the following Poisson regression model

$$y_i|\beta \sim \text{Poisson}\left[\exp\left(\mathbf{x}_i^T\beta\right)\right],\ i = 1, ..., n,$$

where y_i is the count for the ith observation in the sample and x_i is the p-dimensional vector with covariate observations for the ith observation. Use the data set `eBayNumberOfBidderData.dat`. This dataset contains observations from 1000 eBay auctions of coins. The response variable is **nBids** and records the number of bids in each auction. The remaining variables are features/covariates ($\mathbf{x}$):

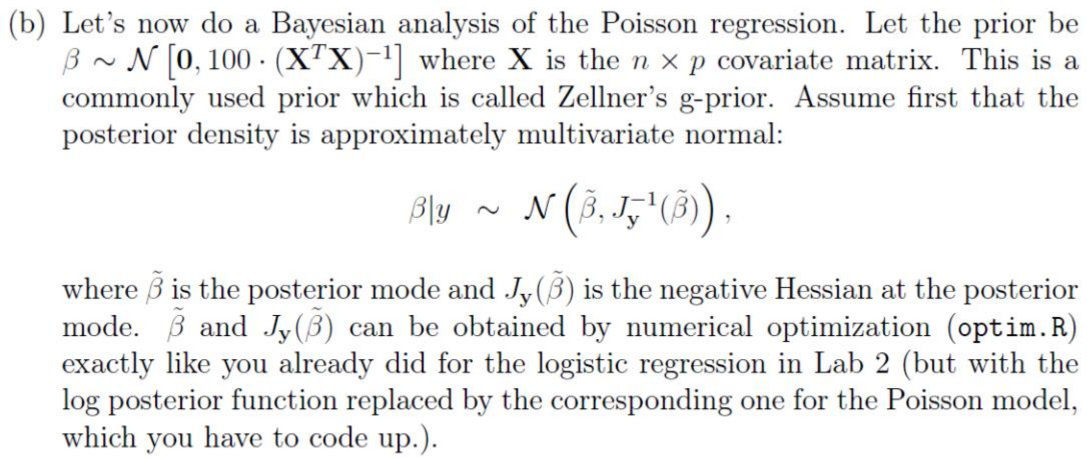
- **Const** (for the intercept)
- **PowerSeller** (is the seller selling large volumes on eBay?)
- **VerifyID** (is the seller verified by eBay?)
- **Sealed** (was the coin sold sealed in never opened envelope?)
- **MinBlem** (did the coin have a minor defect?)
- **MajBlem** (a major defect?)
- **LargNeg** (did the seller get a lot of negative feedback from customers?)
- **LogBook** (logarithm of the coins book value according to expert sellers. Standardized)
- **MinBidShare** (a variable that measures ratio of the minimum selling price (starting price) to the book value. Standardized).

(a) Obtain the maximum likelihood estimator of β in the Poisson regression model for the eBay data [Hint: `glm.R`, don't forget that `glm()` adds its own intercept so don't input the covariate Const]. Which covariates are significant?

2a The maximum likelihood estimators for beta are given by the coefficients from the glm model. These values are the ones that maximizes the likelihood for the given data. Column 2 was taken away since this was meerly a constant and glm model adds its own intercept. The values that have the biggest impact on the model are variables Sealed, MajBlem and MiniBidShare.

```
## (Intercept) PowerSeller    VerifyID      Sealed     Minblem     MajBlem
##  3.61513725 -0.08454439 -0.58738252  1.49036013 -0.35181938 -1.06208528
##     LargNeg     LogBook MinBidShare
##  0.31551113 -0.10342827 -5.13588752
```

aruup817
phika529



(b) Let's now do a Bayesian analysis of the Poisson regression. Let the prior be $\beta \sim \mathcal{N}\left[\mathbf{0}, 100 \cdot (\mathbf{X}^T\mathbf{X})^{-1}\right]$ where $\mathbf{X}$ is the $n \times p$ covariate matrix. This is a commonly used prior which is called Zellner's g-prior. Assume first that the posterior density is approximately multivariate normal:

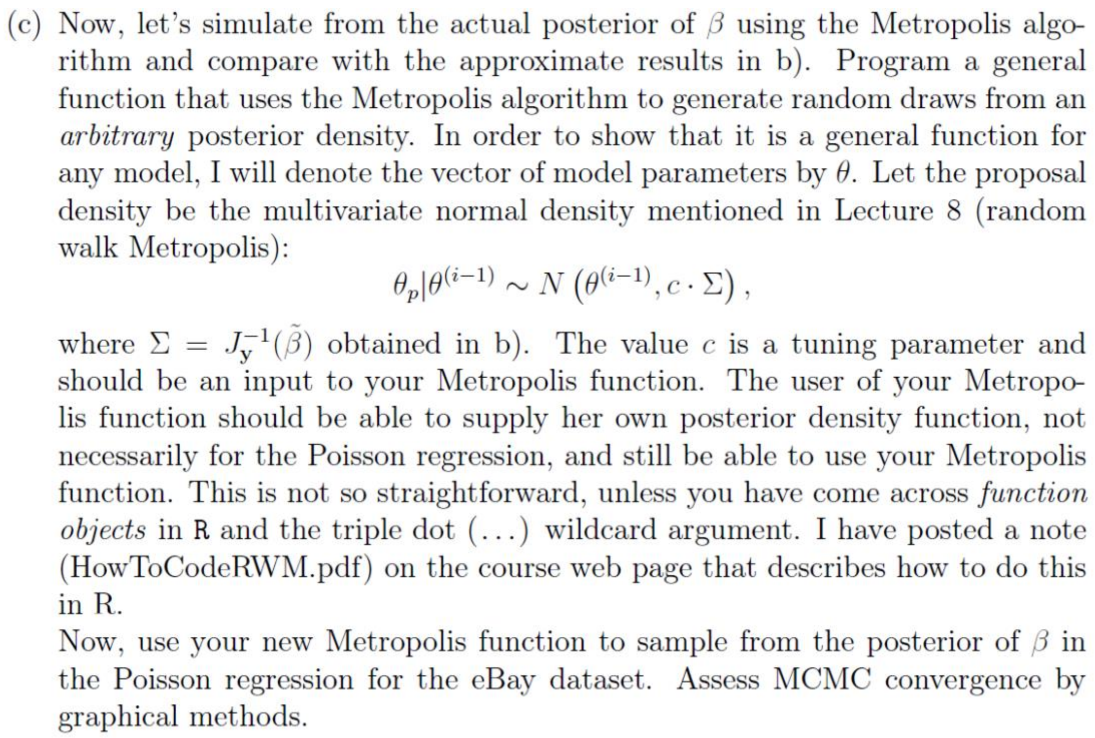
$$\beta|y \;\sim\; \mathcal{N}\left(\tilde{\beta}, J_{\mathbf{y}}^{-1}(\tilde{\beta})\right),$$

where $\tilde{\beta}$ is the posterior mode and $J_{\mathbf{y}}(\tilde{\beta})$ is the negative Hessian at the posterior mode. $\tilde{\beta}$ and $J_{\mathbf{y}}(\tilde{\beta})$ can be obtained by numerical optimization (`optim.R`) exactly like you already did for the logistic regression in Lab 2 (but with the log posterior function replaced by the corresponding one for the Poisson model, which you have to code up.).

2b

Using the function optim we obtain the following beta and hessian values (first row is the beta values);

```
## [1]  1.06984118 -0.02051246 -0.39300599  0.44355549 -0.05246627 -
0.22123840
## [7]  0.07069683 -0.12021767 -1.89198501

##                [,1]          [,2]          [,3]          [,4]
[,5]
##  [1,]  9.454625e-04 -7.138972e-04 -2.741517e-04 -2.709016e-04 -4.454554e-
04
##  [2,] -7.138972e-04  1.353076e-03  4.024623e-05 -2.948968e-04  1.142960e-
04
##  [3,] -2.741517e-04  4.024623e-05  8.515360e-03 -7.824886e-04 -1.013613e-
04
##  [4,] -2.709016e-04 -2.948968e-04 -7.824886e-04  2.557778e-03  3.577158e-
04
##  [5,] -4.454554e-04  1.142960e-04 -1.013613e-04  3.577158e-04  3.624606e-
03
##  [6,] -2.772239e-04 -2.082668e-04  2.282539e-04  4.532308e-04  3.492353e-
04
##  [7,] -5.128351e-04  2.801777e-04  3.313568e-04  3.376467e-04  5.844006e-
05
##  [8,]  6.436765e-05  1.181852e-04 -3.191869e-04 -1.311025e-04  5.854011e-
05
##  [9,]  1.109935e-03 -5.685706e-04 -4.292828e-04 -5.759169e-05 -6.437066e-
05
##                [,6]          [,7]          [,8]          [,9]
##  [1,] -2.772239e-04 -5.128351e-04  6.436765e-05  1.109935e-03
##  [2,] -2.082668e-04  2.801777e-04  1.181852e-04 -5.685706e-04
##  [3,]  2.282539e-04  3.313568e-04 -3.191869e-04 -4.292828e-04
##  [4,]  4.532308e-04  3.376467e-04 -1.311025e-04 -5.759169e-05
##  [5,]  3.492353e-04  5.844006e-05  5.854011e-05 -6.437066e-05
##  [6,]  8.365059e-03  4.048644e-04 -8.975843e-05  2.622264e-04
##  [7,]  4.048644e-04  3.175060e-03 -2.541751e-04 -1.063169e-04
##  [8,] -8.975843e-05 -2.541751e-04  8.384703e-04  1.037428e-03
##  [9,]  2.622264e-04 -1.063169e-04  1.037428e-03  5.054757e-03
```



(c) Now, let's simulate from the actual posterior of β using the Metropolis algorithm and compare with the approximate results in b). Program a general function that uses the Metropolis algorithm to generate random draws from an *arbitrary* posterior density. In order to show that it is a general function for any model, I will denote the vector of model parameters by θ. Let the proposal density be the multivariate normal density mentioned in Lecture 8 (random walk Metropolis):

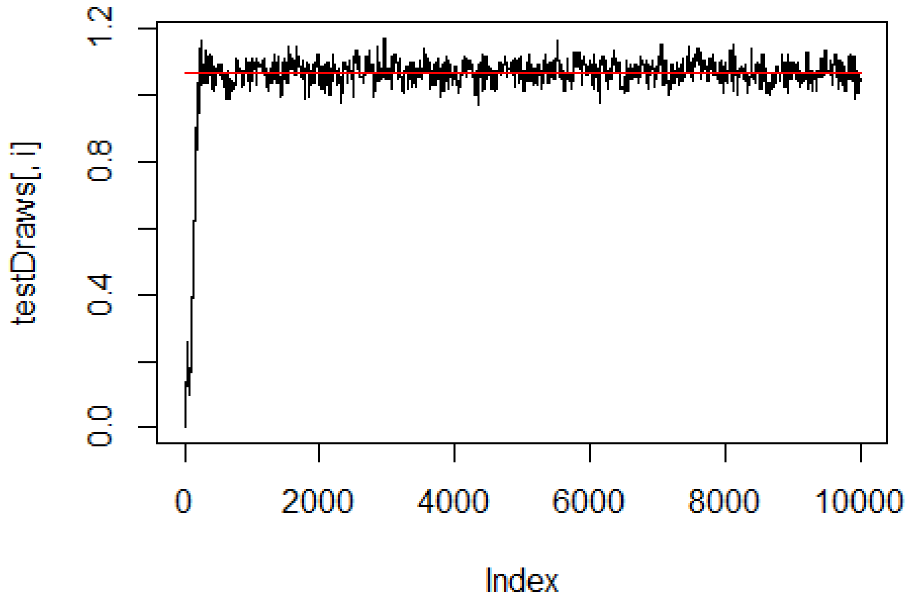
$$\theta_p|\theta^{(i-1)} \sim N\left(\theta^{(i-1)}, c \cdot \Sigma\right),$$

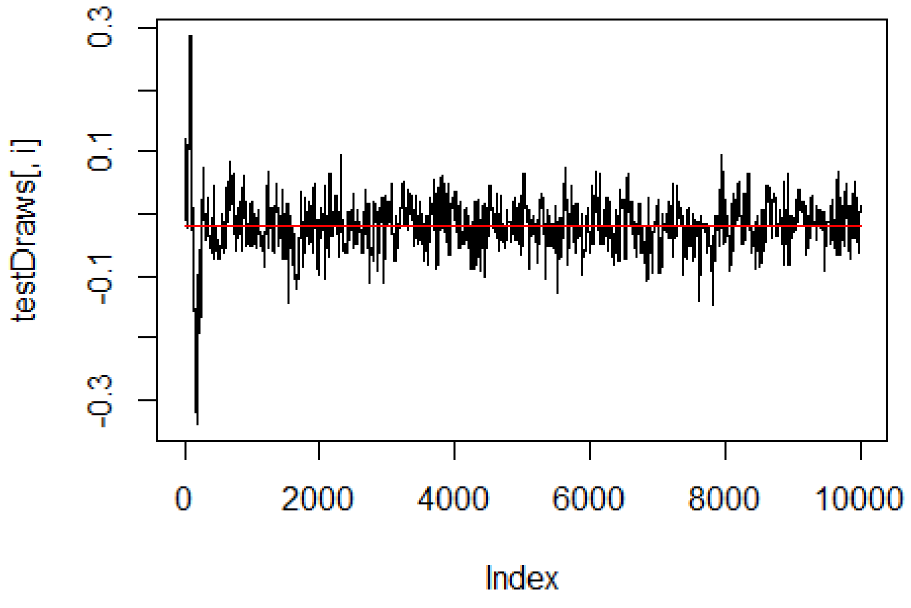
where $\Sigma = J_{\mathbf{y}}^{-1}(\tilde{\beta})$ obtained in b). The value c is a tuning parameter and should be an input to your Metropolis function. The user of your Metropolis function should be able to supply her own posterior density function, not necessarily for the Poisson regression, and still be able to use your Metropolis function. This is not so straightforward, unless you have come across *function objects* in R and the triple dot (...) wildcard argument. I have posted a note (HowToCodeRWM.pdf) on the course web page that describes how to do this in R.
Now, use your new Metropolis function to sample from the posterior of β in the Poisson regression for the eBay dataset. Assess MCMC convergence by graphical methods.

2c

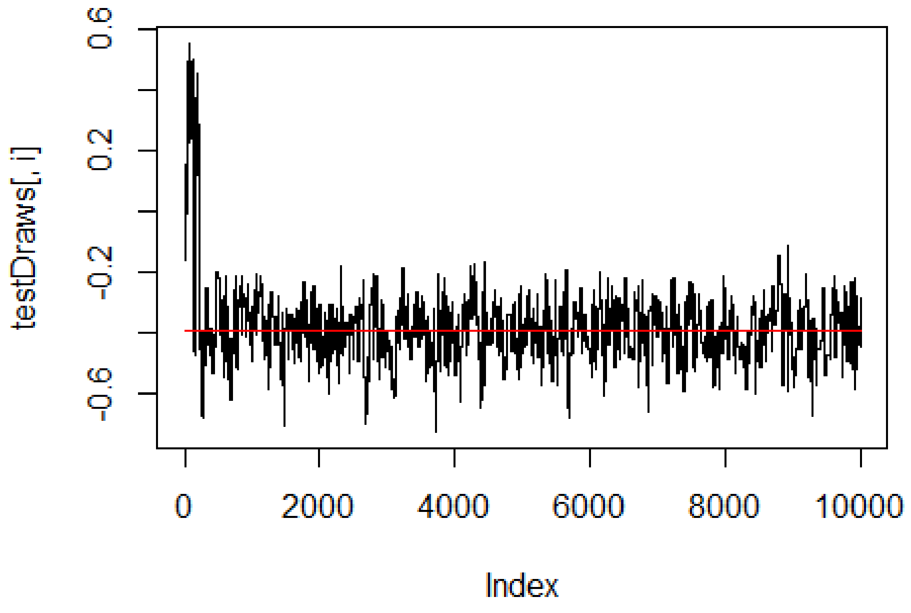
After running the metropolis algorithm for n = 10000 iterations we can see from the plots below that the values for beta converges to a set of beta values.

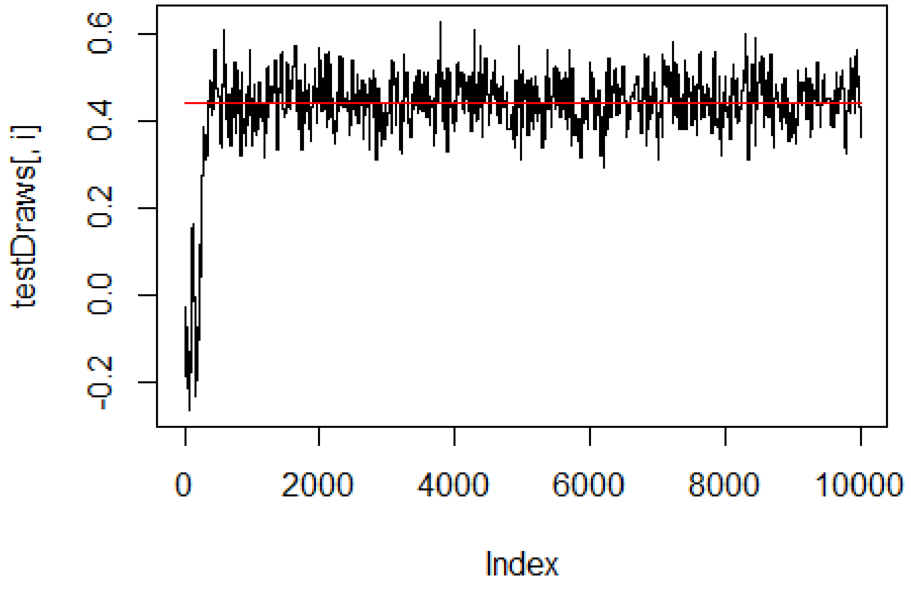
aruup817
phika529



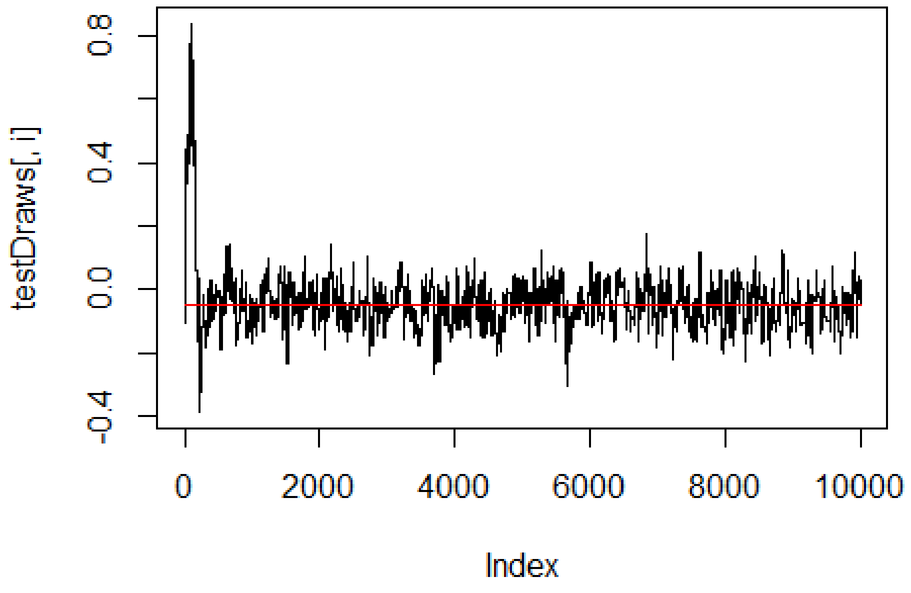


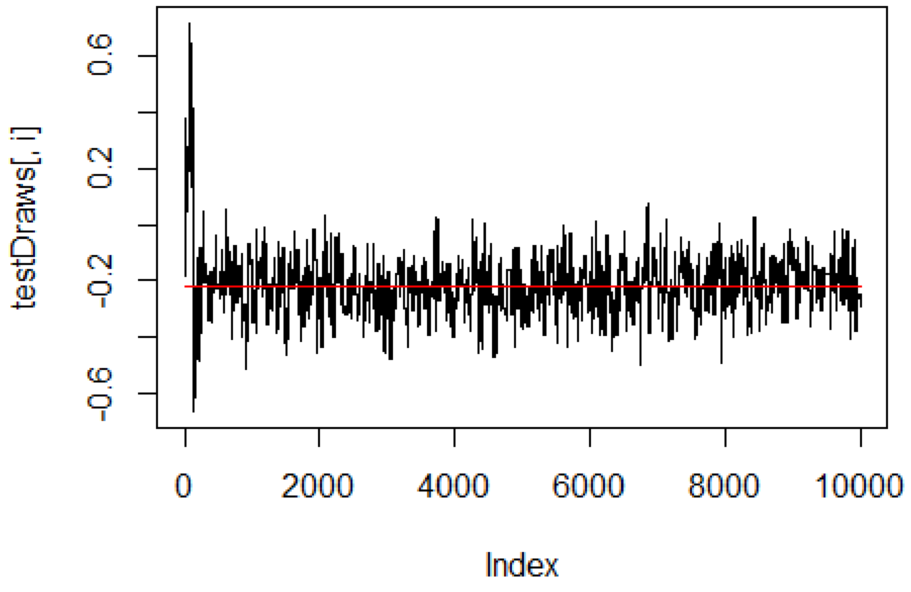
aruup817
phika529



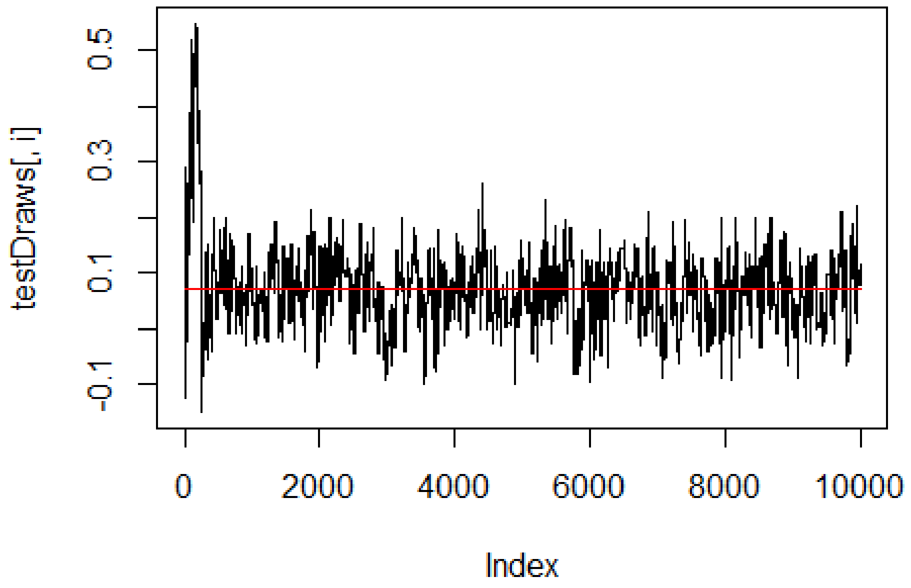


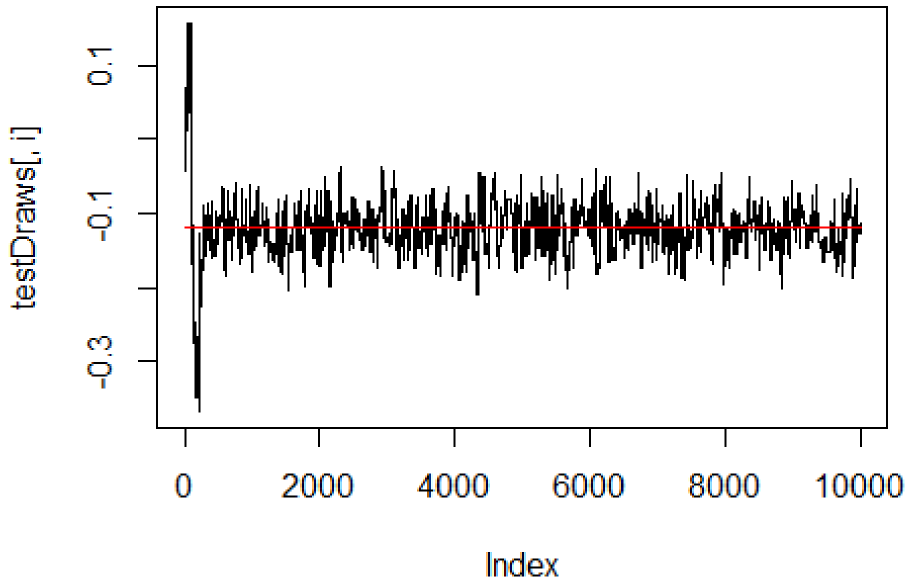
aruup817
phika529



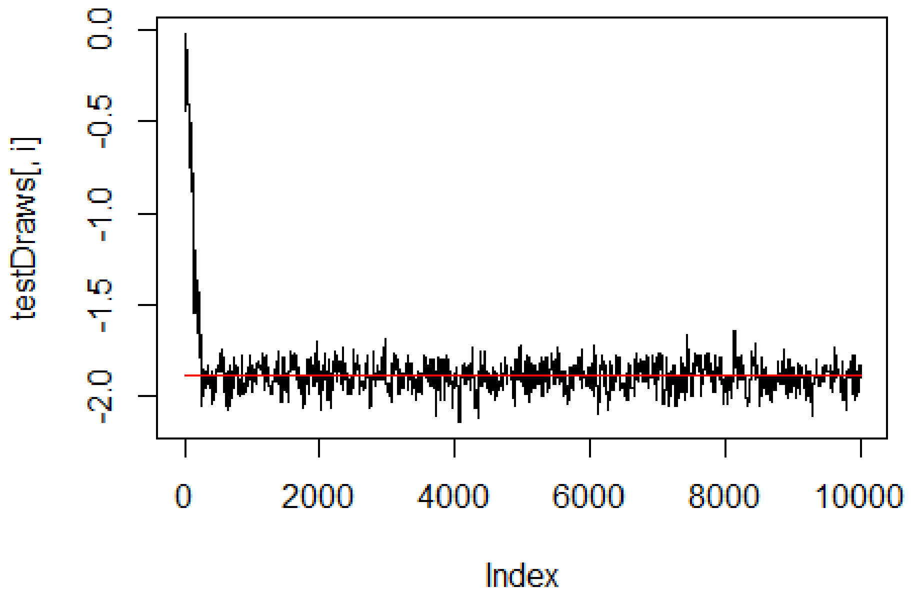


aruup817
phika529





aruup817
phika529

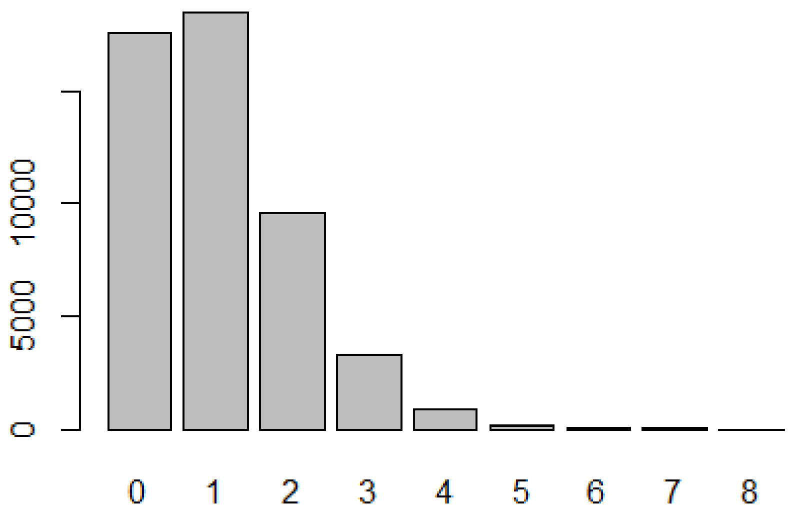


(d) Use the MCMC draws from c) to simulate from the predictive distribution of the number of bidders in a new auction with the characteristics below. Plot the predictive distribution. What is the probability of no bidders in this new auction?

- **PowerSeller** = 1
- **VerifyID** = 1
- **Sealed** = 1
- **MinBlem** = 0
- **MajBlem** = 0
- **LargNeg** = 0
- **LogBook** = 1
- **MinBidShare** = 0.5

We calculated the mean of our beta values from 1000 iterations (skipped the burn-in phase), and used these for the poisson regression.When given a Poisson regression model for betas and an input vector x,the predicted mean of the associated Poisson distribution is given by e^beta*x. Beta values can normally be estimated by MLE, our as in our case, by metropolis. We used the lambda obtained from the poisson regression to simulate draws from a poisson distribution with our chosen x-values. The distribution than gave us n=50000 values for y, and approximated 35% of these said 0, so to say no bids. From this we draw the conclusion that the probability that an auction with these qualities gets 0 bids is 35%.

aruup817
phika529



Asignment 1

```
data = read.table("rainfall.dat")

# Setup
my0 <- 1
#Increased value of tao0 results in bigger value of w hence we myn will go
more towards the data mean
tao0squared <- 10
v0 <- 1
sigmaSquared <- 1  #init value for sigma, gets updated in Gibbs sampling
sigma0Squared <- 1
nDraws <- 500 # Number of draws
dataMean = mean(data[,1])
n = length(data[,1])
vn = v0 + n

# Gibbs sampling
gibbsDraws <- matrix(0,nDraws,2)
theta2 <- 0
for (i in 1:nDraws){

  ## from lecture 2
  taonSquared = 1/(n/sigmaSquared + 1/tao0squared)
  w = (n/sigmaSquared)/(n/sigmaSquared + 1/tao0squared)
  myn = w*dataMean + (1-w)*my0
  ##
```

aruup817
phika529

```
  ## from lecture 7

  #Compute Full conditional posterior (Normal model with conditionally
conjugate prior)
  my <- rnorm(1, mean = myn, sd = sqrt(taonSquared))
  gibbsDraws[i,1] <- my

  sigmaSquaredVariance = ((v0*sigma0Squared + sum((data[,1]-my)^2))/n+v0)
  #Make it inverse chai squared
  sigmaSquared <- vn*sigmaSquaredVariance/(rchisq(1, vn))
  gibbsDraws[i,2] <- sigmaSquared
  ##
  }

# Plot the values from the conditional posteriors of my|sigma and sigma|my
plot(gibbsDraws[,2], type="lines", xlab = "draw", ylab = "sigma", main =
"Sigma conditioned on my")

plot(gibbsDraws[,1], type = "lines", xlab = "draw", ylab = "my", main = "My
conditioned on sigma")
lines(dataMean, col = "red")


# (1b)


##Mattias Code
##########    BEGIN USER INPUT ################
# Data options
data(data)
rawData <- faithful
x <- as.matrix(data[,1])

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

# Prior options
alpha <- 10*rep(1,nComp) # Dirichlet(alpha)
muPrior <- rep(10,nComp) # Prior mean of mu
tau2Prior <- rep(5,nComp) # Prior std of mu
sigma2_0 <- rep(var(x),nComp) # s20 (best guess of sigma2)
nu0 <- rep(2,nComp) # degrees of freedom for prior on sigma2

# MCMC options
nIter <- 1000 # Number of Gibbs sampling draws

# Plotting options
plotFit <- TRUE
```

aruup817
phika529

```
lineColors <- c("blue", "green")
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))
```

aruup817
phika529

```
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
```

aruup817
phika529

```
    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)
  }

}


#########################   Enf of mattias code
###############################################

#Extract what we are interested in
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)



# (1c)

my_posterior_mean = mean(gibbsDraws[,1])
sigma_posterior_mean = mean(gibbsDraws[,2])

# plot density of data
hist(x, breaks=20, freq=FALSE, ylim=c(0,0.025) ,main="Graphical comparison of
data vs densities", xlab="Draw")
# plot posterior density from a)
lines(xGrid, dnorm(xGrid, mean=my_posterior_mean, sd =
sqrt(sigma_posterior_mean)), type="lines", col="purple")
# plot mixture model density from b)
lines(xGrid, mixDensMean, type="lines", col = "red")
legend("topright", box.lty = 1, legend = c("Data density","Mixture
density","Posterior density"), col=c("black","red","purple"), lwd = 2)
```

##Assignment 2

```
#2a)


bids_DATA = read.table('eBayNumberOfBidderData.dat', header = TRUE)
data = bids_DATA[,-2]

#Create the glm model. Gives us valus on the parameters which maximizes the
```

```
likelihood for the given data.
#Column 2 is taken away since glm model adds its own intercept

glm_model = glm(nBids ~ ., data = bids_DATA[,-2])
Beta_covariates = glm_model$coefficients
Beta_covariates



#2b
library(mvtnorm)



#covariance_matrix = c(1:8)
Y = as.vector(bids_DATA[,1])
X = as.matrix(bids_DATA[,-1])
col = ncol(X)

# parameters for the prior distribution of beta
my = as.vector(rep(0, col))
sigma_prior = 100*solve(t(X)%*%X)

# function which returns an expression proportional to log beta posterior,
this can be optimized for
# values of beta mode and hessian J (observed hessian evaluated at posterior
mode)
#Log likelikihood is given by the log produkt of the poisson densituy
distribution
log.posterior = function(betas, x,Y,my,sigma_prior){

  # logproduct of poission pdf
  log_likelihood = sum(-log(factorial(Y))+(betas%*%t(X))*Y -
exp(betas%*%t(X)))

  # log of multivariate pdf => dmvnorm
  log_prior = dmvnorm(x=betas,mean=my,sigma=sigma_prior, log=TRUE)

  if(abs(log_likelihood == Inf)){
    log_likelihood = -5000
  }

  return(log_likelihood + log_prior)
}

# Different starting values. Ideally, any random starting value gives you the
same optimum (i.e. optimum is unique)
initVal <- as.vector(rep(0,col));

# function which optmizes over expression log.posterior with respect to its
first argument (betas).
```

aruup817
phika529

```
# returns optimal values for beta (mode), and hessian in the mode

#Find beta and hessian values in mode
OptParams<-
optim(initVal,log.posterior,gr=NULL,X,Y,my,sigma_prior,method=c("BFGS"),contr
ol=list(fnscale=-1),hessian=TRUE)

#Beta values
betas_posterior = OptParams$par
betas_posterior

# Takes negative so that the posterior can be approx. as normal
hessian_posterior = -solve(OptParams$hessian)
hessian_posterior

#2c
#Metropolis algorithm Le 8



#Draw samples from Betas posterior distribution.
set.seed(12345)

#metopolis algorithm, simulates from posterior of beta.
# args:  c = arbitrary tuning parameter, ... is a wildcard argument and
represents the arguments of the posterior_density
# higher value of c gives bigger variance / bigger steps
# n = number of iterations (return n values for betas)
metropolis = function(n, c, hessian, betas,  posterior_density, ...){

  # this step depends on previous position. Previous position becomes this
turns mean.
  proposal_draws_previous = betas;
  acceptedDraws = matrix(0, ncol=9,nrow=n)

  set.seed(12345)
  for(i in 1:n){
      # draws from the proposal distribution.
      proposal_draws = rmvnorm(1, proposal_draws_previous, c*hessian)
      # create a ratio depending on if this draw is better than previous,
take exp to remove logarithm (logposterior)
      acceptance_ratio = min(1,exp(posterior_density(proposal_draws, ...)-
posterior_density(proposal_draws_previous, ...)))
      # draw a random uniformed variable to compare wiht acceptance ratio
      random_acceptance = runif(1,0,1)
      # if acceptance ratio is bigger than random variable than we move to
the new position, otherwise we stay
      if(acceptance_ratio >= random_acceptance){
        proposal_draws_previous = proposal_draws
        betas = proposal_draws
```

aruup817
phika529

```
      }
      acceptedDraws[i,] = betas

  }
   acceptedDraws;
}

testDraws = metropolis(10000, c=1, hessian= hessian_posterior, betas = my,
posterior_density = log.posterior, X, Y, my, sigma_prior)

# plot the result for each dimension of beta.
for(i in 1:9){
  plot(testDraws[,i], type='s')
  a = c(rep(betas_posterior[i],nrow(testDraws)))
  lines(a, col='red')
}

#2d

# our x-values that we want to predict number of bids for
x_values <- c(1,1,1,1,0,0,0,1,0.5)

# mean of our beta values from the posterior
mean_betas = c()

# dont count in the first 1000 iterations in mean calculation (burn-in phase)
for(i in 1:9){
  mean_betas = append(mean_betas, mean(testDraws[1000:10000,i]))
}

#when given a Poisson regression model beta and an input vector x,
#the predicted mean of the associated Poisson distribution is given by
e^beta*x
#theta can be estimated by MLE, our as in our case, by metropolis.
poisson_regression <- exp(mean_betas%*% x_values)[1]

# simulate value for y considering our x-values
simulation <- rpois(50000,poisson_regression)
hist(simulation, breaks=8)

# percentage of draws where y = 0
probZero = sum(simulation==0)/length(simulation)
```

Lab4

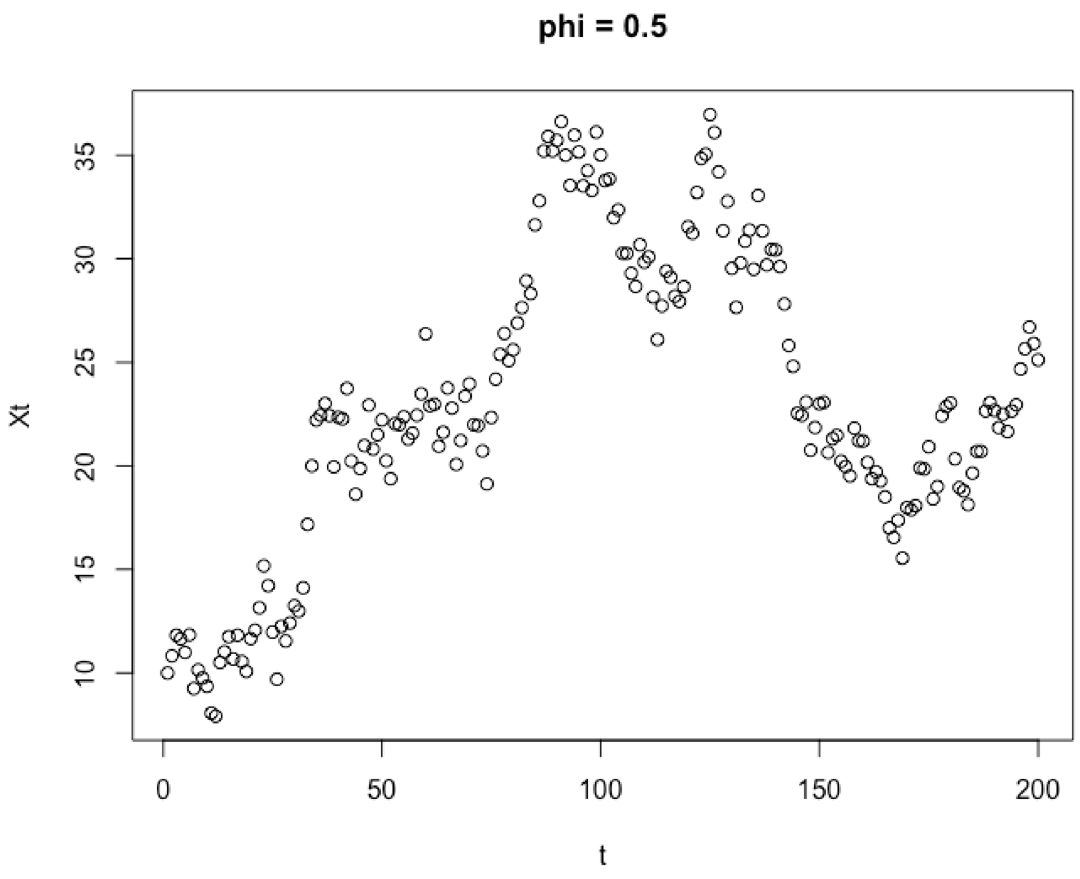
#Assignment 1

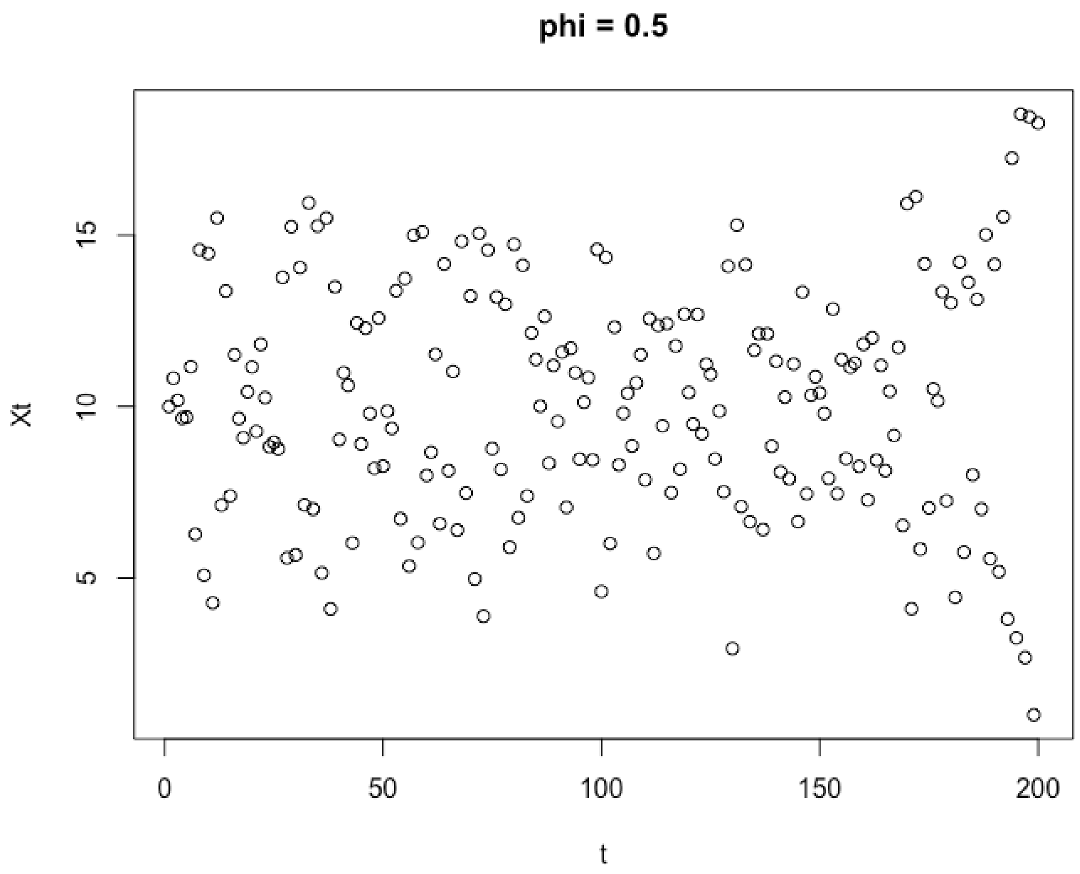
(a) Write a function in R that simulates data from the AR(1)-process

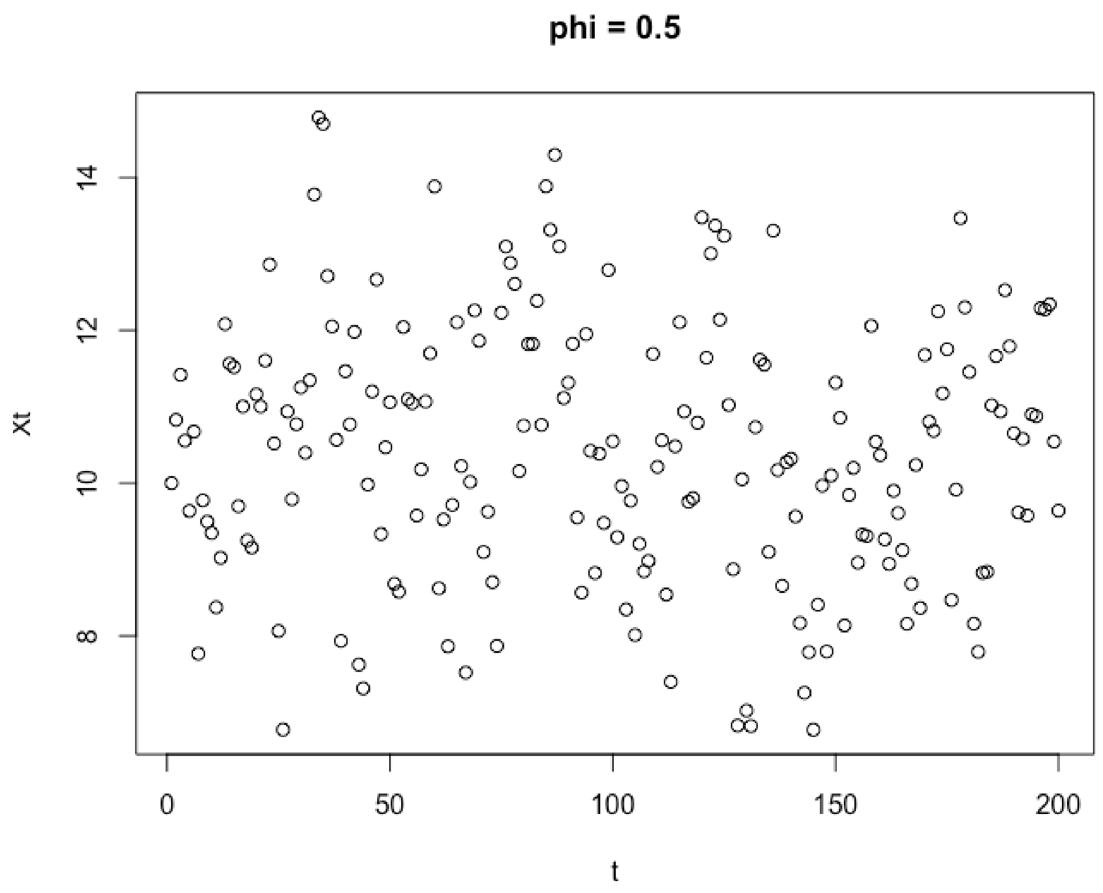
$$x_t = \mu + \phi\,(x_{t-1} - \mu) + \varepsilon_t, \;\; \varepsilon_t \overset{iid}{\sim} N\left(0, \sigma^2\right),$$

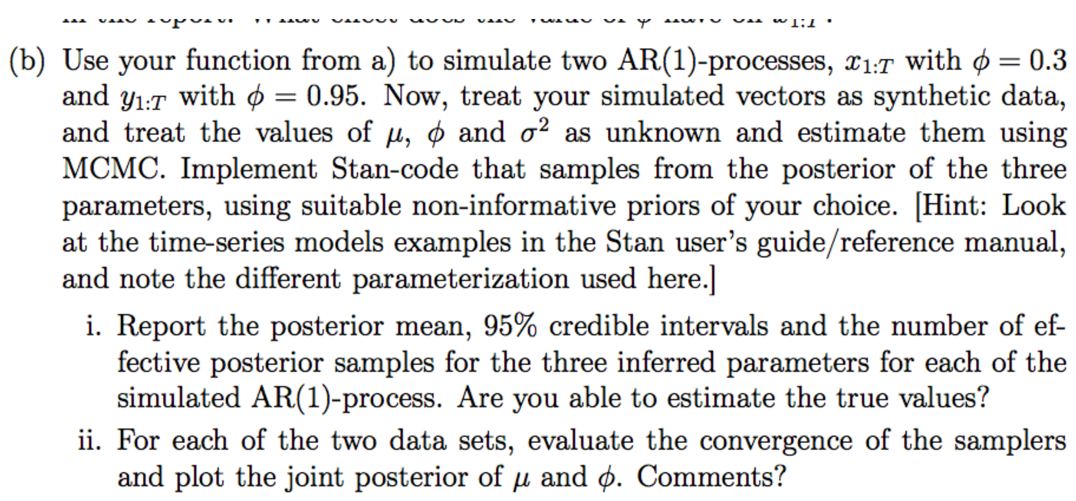
for given values of μ, ϕ and σ^2. Start the process at $x_1 = \mu$ and then simulate values for x_t for $t = 2, 3 \ldots, T$ and return the vector $x_{1:T}$ containing all time points. Use $\mu = 10, \sigma^2 = 2$ and $T = 200$ and look at some different realizations (simulations) of $x_{1:T}$ for values of ϕ between -1 and 1 (this is the interval of ϕ where the AR(1)-process is stable). Include a plot of at least one realization in the report. What effect does the value of ϕ have on $x_{1:T}$?

1a We plottet the AR process for phi = 0.99, phi = -0.99 and phi = 0.5. The value of phi determines the dependence on the previous term where a higher positive value will result in a higher dependence. When the parameter is negative the value at t is, in general, the oppostite sign of that at t-1. Resultingly, the graph will show no sign of dependence.




phi = 0.5
Xt
t
0
50
100
150
200
5
10
15




(b) Use your function from a) to simulate two AR(1)-processes, $x_{1:T}$ with $\phi = 0.3$ and $y_{1:T}$ with $\phi = 0.95$. Now, treat your simulated vectors as synthetic data, and treat the values of μ, ϕ and σ^2 as unknown and estimate them using MCMC. Implement Stan-code that samples from the posterior of the three parameters, using suitable non-informative priors of your choice. [Hint: Look at the time-series models examples in the Stan user's guide/reference manual, and note the different parameterization used here.]

i. Report the posterior mean, 95% credible intervals and the number of effective posterior samples for the three inferred parameters for each of the simulated AR(1)-process. Are you able to estimate the true values?

ii. For each of the two data sets, evaluate the convergence of the samplers and plot the joint posterior of μ and ϕ. Comments?

1b

If no priors are specified the models assumes flat priors. Our result from the stan gave following data when setting phi to = 0.95:

For my: Credible 0.95 interval -> (0.180,1.367) Mean -> 0.765 Nr of effective posterior samples -> 2416

For phi: Credible 0.95 interval -> (0.891,0.986) Mean -> 0.939 Nr of effective posterior samples -> 2489

For sigma: Credible 0.95 interval -> (1.234,1.502) Mean -> 1.361 Nr of effective posterior samples -> 3740

Strangely enough the number of effective samples for sigma exceeded the number of independent draws for said parameter. Overall the stan model fails to estimate the value for my, but estimates sigma very well. The value for phi was pretty accurate as well, almost 0.95. Since phi is close to 1, my gives very little impact to each iteration which might make it hard to estimate.

Our result from the stan gave following data when setting phi to = 0.3:

For my: Credible 0.95 interval -> (4.71,7.29) Mean -> 6 Nr of effective posterior samples -> 2413

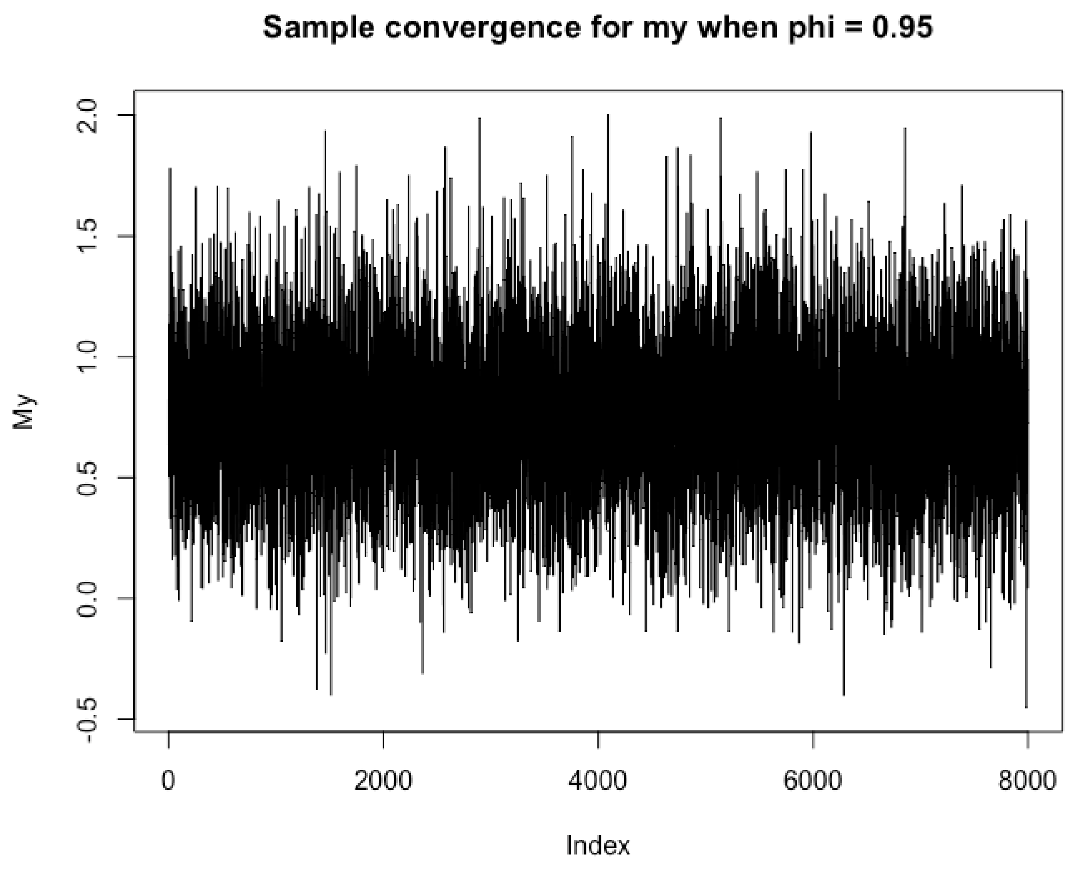
For phi: Credible 0.95 interval -> (0.274,0.443) Mean -> 0.399 Nr of effective posterior samples -> 2489

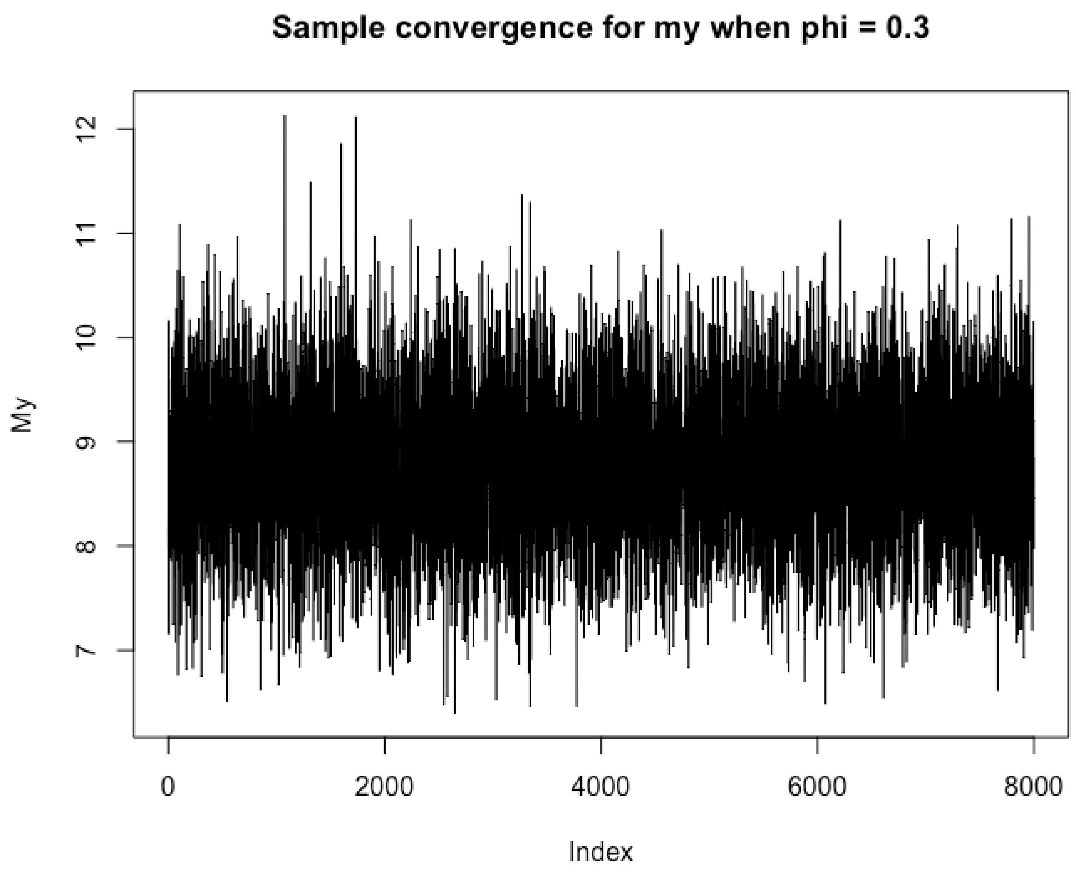
For sigma: Credible 0.95 interval -> (1.283,1.565) Mean -> 1.413 Nr of effective posterior samples -> 3546

The stan model again fails to estimate the value for my, but is much closer this time. The model again estimates sigma very well. The value for phi was a little less accurate but the real value is still contained in the credible interval. The model did better job predicting my when less weight was given to the previous term.

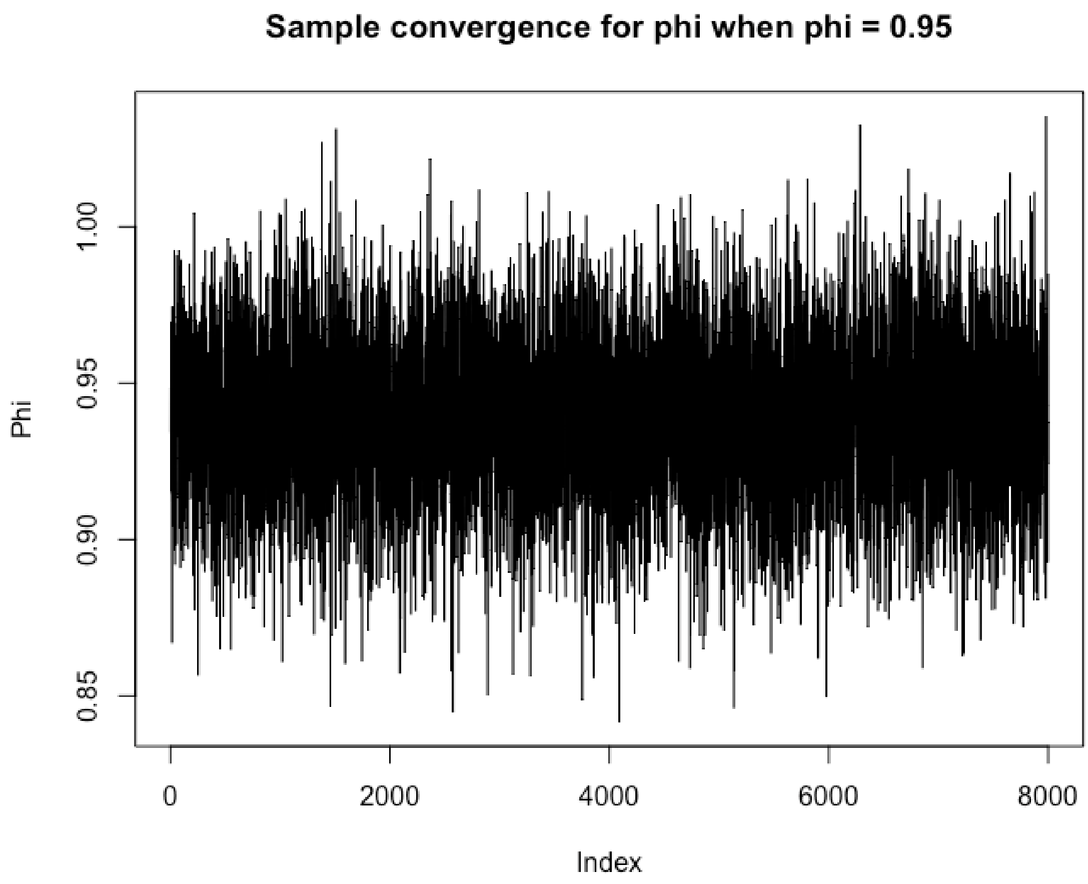
For each of the two data sets, evaluate the convergence of the samplers and plot the joint posterior of my and phi.

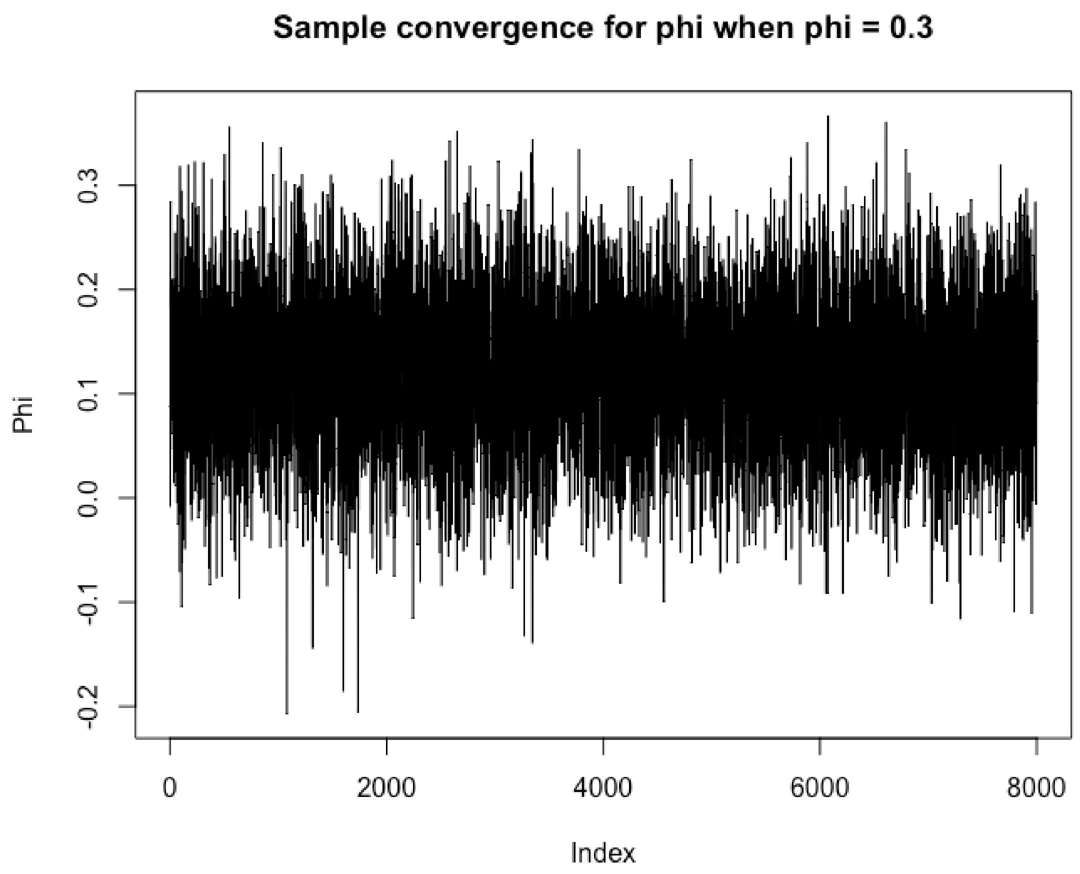
My converges to 0.765 when the real phi is 0.95, and to 6 when the real phi is 0.3, which can be seen in the graphs below. The estimations are pretty bad considering that the real my is 10, especially for when the real phi is 0.95. As mentioned earlier this might be due to that my matters less when phi is higher, since then more weight is given to the previous sample point instead.


Sample convergence for my when phi = 0.95
My
Index
-0.5
0.0
0.5
1.0
1.5
2.0
0
2000
4000
6000
8000

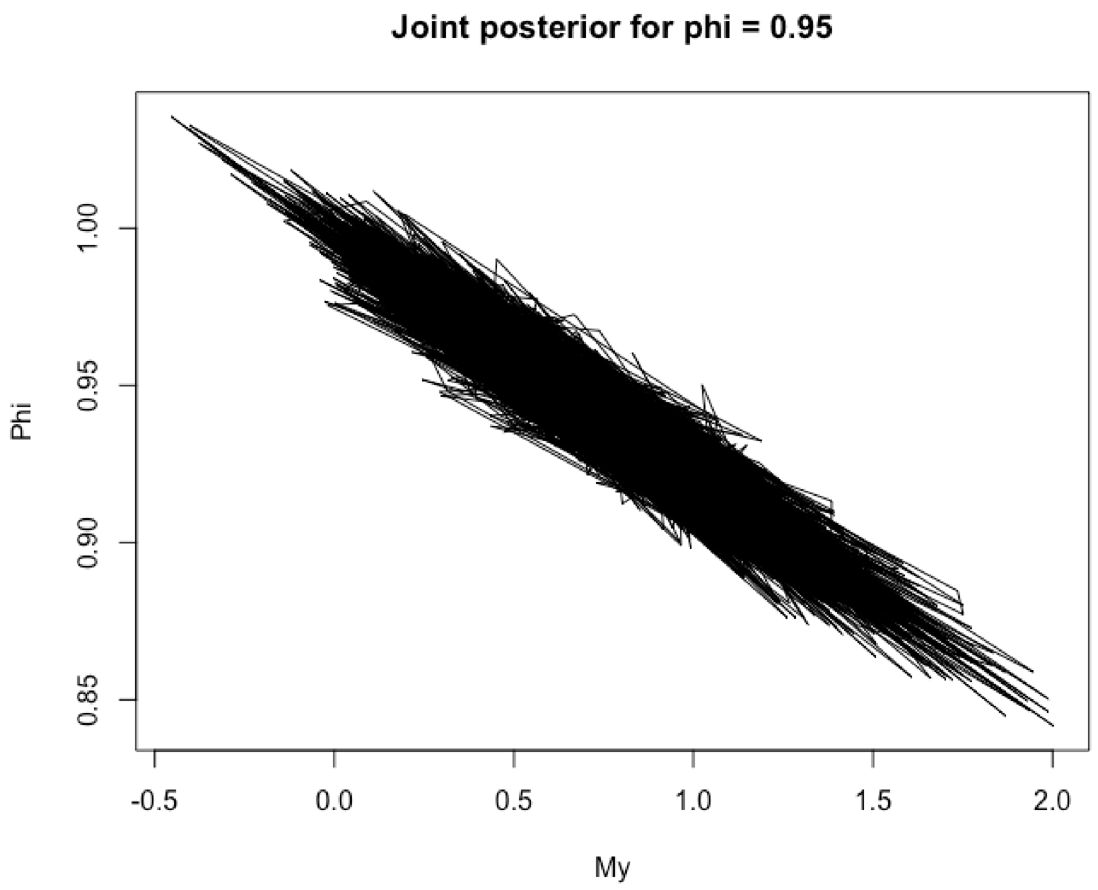


Phi converges to 0.939 when the real phi is 0.95, and to 0.399 when the real phi is 0.3, which can be seen in the following graphs; (as mentioned earlier, when more weight is given to the previous sample point phi is estimated better)

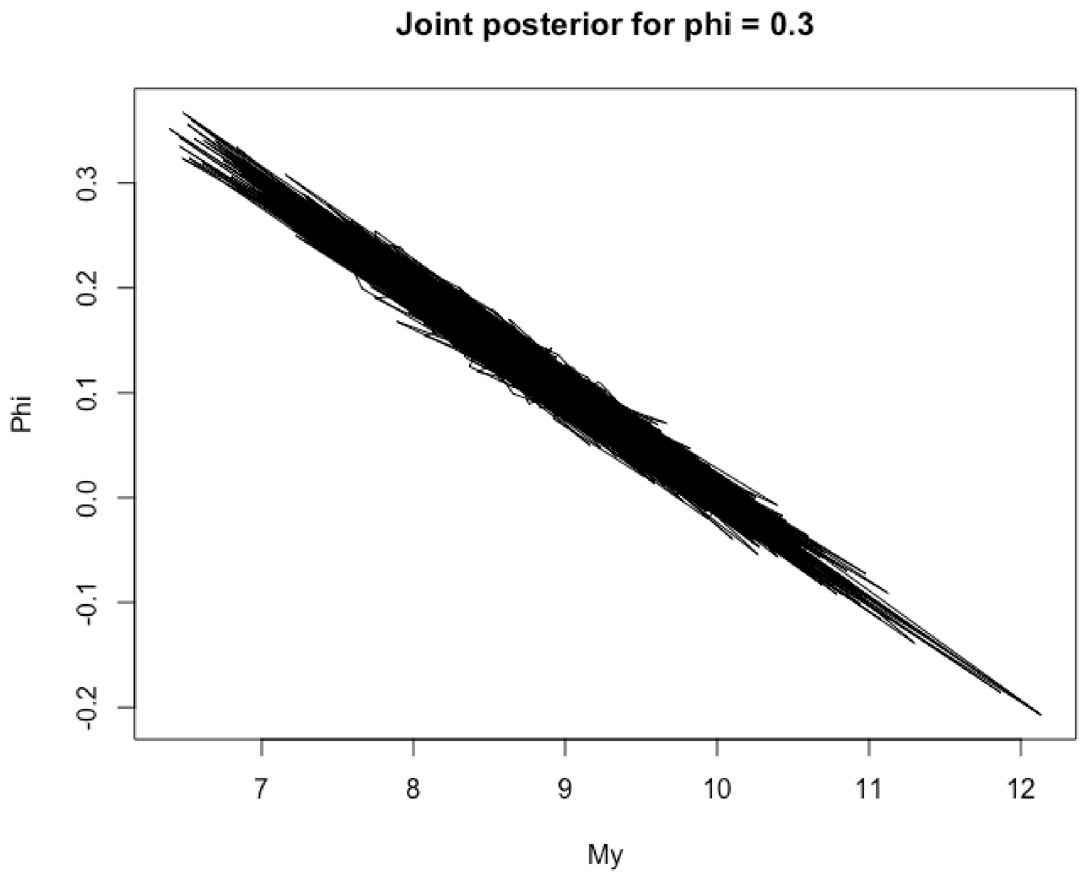

Sample convergence for phi when phi = 0.95
Phi
1.00
0.95
0.90
0.85
0
2000
4000
6000
8000
Index



We plotted the joint posterior of phi and my and got the following plots. As seen below it seems not to be possible to obtain correct values for phi and my at the same time. If bigger weight is given to one of them, the estimation becomes better.



Joint posterior for phi = 0.95
Phi
My
1.00
0.95
0.90
0.85
-0.5
0.0
0.5
1.0
1.5
2.0



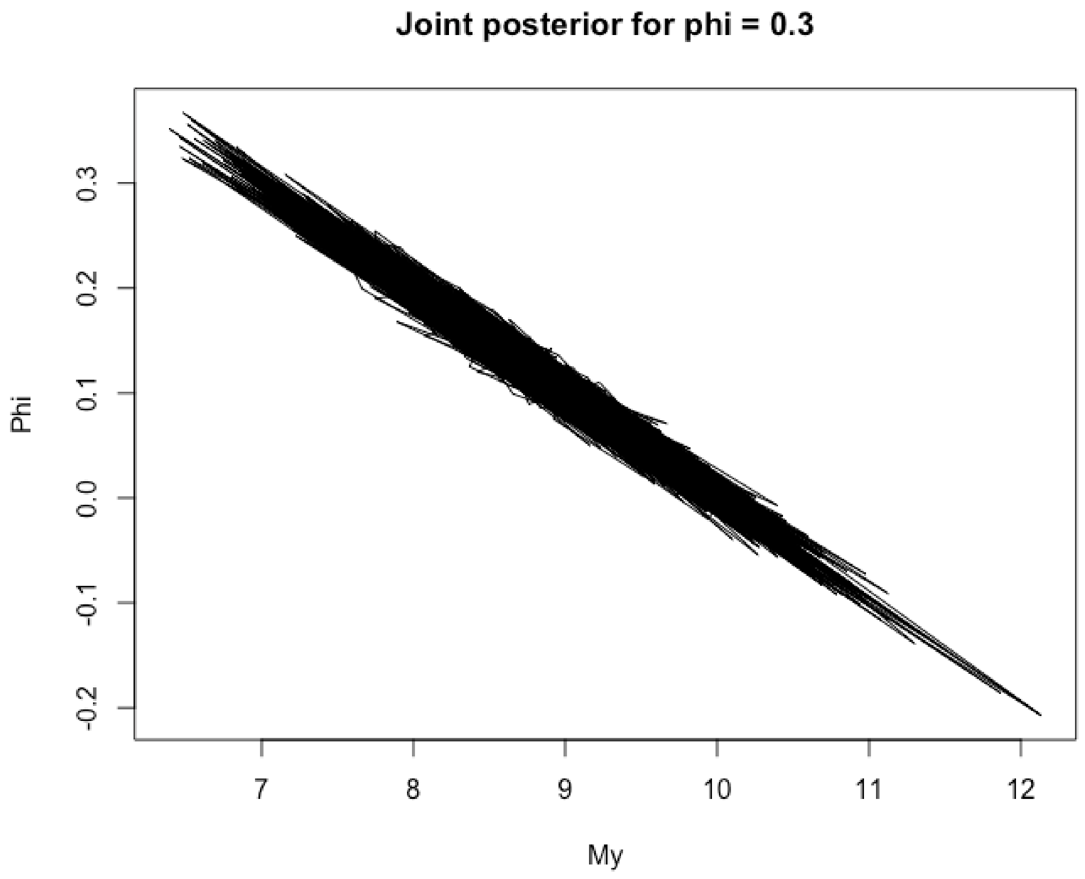
(c) The data **campy.dat** contain the number of cases of campylobacter infections in the north of the province Quebec (Canada) in four week intervals from January 1990 to the end of October 2000. It has 13 observations per year and 140 observations in total. Assume that the number of infections c_t at each time point follows an independent Poisson distribution when conditioned on a latent AR(1)-process x_t, that is

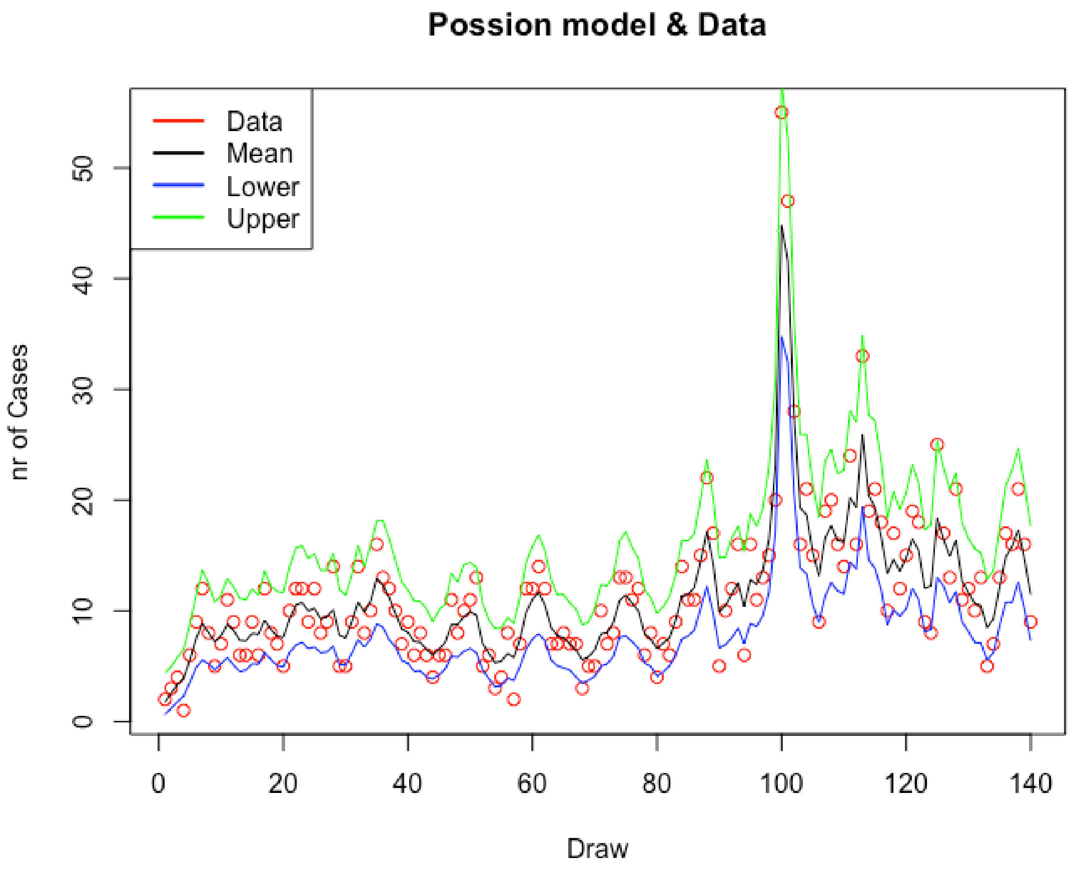
$$c_t|x_t \sim Poisson\left(\exp\left(x_t\right)\right),$$

where x_t is an AR(1)-process as in a). Implement and estimate the model in Stan, using suitable priors of your choice. Produce a plot that contains both the data and the posterior mean and 95% credible intervals for the latent intensity $\theta_t = \exp\left(x_t\right)$ over time. [Hint: Should x_t be seen as data or parameters?]

1c

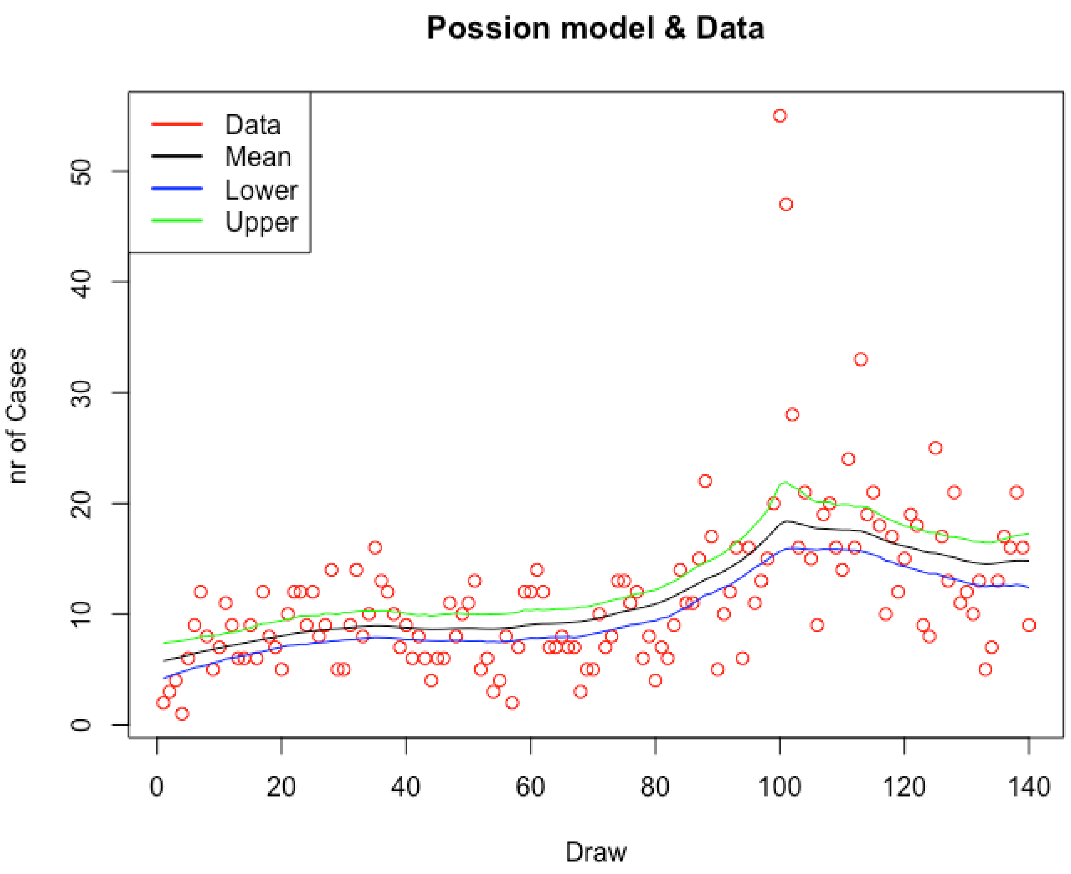
Below one can see a plot of the data, posterior mean and the credible interval for the posterior. The mean follows the mean of the data as expected. Since our prior for sigma is uninformative the posterior will only be based of the data.





1d

Below one can see a plot of the data, posterior mean and the credible interval for the posterior. We have created an informative prior for sigma which says that the underlying data varies more smoothly than the data shows. As can see in the following plot the posterior is not as sensitive to changes in the data. The credible interval has decreased as well, since we are more sure on the value fo our posterior parameter.



```r
library(rstan)


t = seq(1,200, 1)
my = 10
sigma_square = 2
phi = 0.5

time_points = c()
set.seed(12345)
for (i in 1:length(t)){
  if(i == 1){
    time_points[i] = my
    next
  }

  error_term = rnorm(1, mean = 0, sd = sqrt(sigma_square))

  time_points[i] = my + phi*(time_points[i-1] - my) + error_term
}

plot(t, time_points, main = "phi = 0.5", ylab = "Xt")
```

```
#1b)


time_serie = function(phi){
array = c()

for (i in 1:length(t)){
  if(i == 1){
    array[i] = my
    next
  }

  error_term = rnorm(1, mean = 0, sd = sqrt(sigma_square))
  array[i] = my + phi*(array[i-1] - my) + error_term
}
  return(array)
}


time_serie_0.95 = time_serie(0.95)
time_serie_0.3 = time_serie(0.01)

# aplha = my
# beta = phi
# sigma = sigma

alpha = 1
beta = 0.5
sigma = 2

stanModel = ' data {
  int<lower=0> N;
  vector[N] y;
}
parameters {
  real alpha;
  real beta;
  real<lower=0> sigma;
}

model {
  for (n in 2:N)
    y[n] ~ normal(alpha + beta * y[n-1], sigma);
}'

burnin = 1000
```

```r
niter = 3000

# for phi = 0.95
data = list(N=length(t), y=time_serie_0.95)
fit_0.95 = stan(model_code=stanModel,data=data,
           warmup=burnin,iter=niter,chains=4)
# Print the fitted model
print(fit_0.95,digits_summary=3)

# for phi = 0.3
data = list(N=length(t), y=time_serie_0.3)
fit_0.3 = stan(model_code=stanModel,data=data,
           warmup=burnin,iter=niter,chains=4)
# Print the fitted model
print(fit_0.3,digits_summary=3)


# Extract posterior samples
postDraws_0.95 <- extract(fit_0.95)
postDraws_0.3 <- extract(fit_0.3)

par(mfrow = c(1,1))
plot(postDraws_0.95$beta,type="l",ylab="Phi", main="Sample convergence for
phi when phi = 0.95")
plot(postDraws_0.3$beta,type="l",ylab="Phi", main="Sample convergence for phi
when phi = 0.3")

plot(postDraws_0.95$alpha,type="l",ylab="My", main="Sample convergence for my
when phi = 0.95")
plot(postDraws_0.3$alpha,type="l",ylab="My", main="Sample convergence for my
when phi = 0.3")


plot(postDraws_0.95$alpha, postDraws_0.95$beta,type="l",ylab="Phi",xlab =
"My", main="Joint posterior for phi = 0.95")
plot(postDraws_0.3$alpha, postDraws_0.3$beta,type="l",ylab="Phi",xlab = "My",
main="Joint posterior for phi = 0.3")


# 1c

c_data = read.table('campy.dat', header = TRUE)



C_DataRStan<-
  list(N = nrow(c_data),
       c = c_data$c)
```

```r
# The poissons model
C_Model = '
data {
  int<lower=0> N;
  int c[N];
}



parameters {
  real alpha;
  real <lower=-1, upper= 1> beta;
  real<lower=0> sigma;
  real Xt[N];
}

model {

  // Prior
  for (n in 2:N){
    Xt[n] ~ normal(alpha + beta * Xt[n-1], sigma);
  }
  // Model/likelihood
  for(n in 1:N){

  c[n] ~ poisson(exp(Xt[n]));

  }
}'

fit_C_Model<-stan(model_code=C_Model,
               data=C_DataRStan,
               warmup=1000,
               iter=2000,
               chains=4)

print(fit_C_Model,digits=4)

# Plot some results


poster_means_vector = fit_C_Model@.MISC[["summary"]][["msd"]][-c(1,2,3)]

cred_lower = fit_C_Model@.MISC[["summary"]][["c_quan"]][,1,1][-c(1,2,3,144)]
cred_upper = fit_C_Model@.MISC[["summary"]][["c_quan"]][,9,1][-c(1,2,3,144)]

N = nrow(c_data)
xGrid = seq(1,140,1)
plot(xGrid, c_data$c, col = "red", main = "Possion model & Data", ylab = "nr
of Cases", xlab = "Draw")
lines(exp(poster_means_vector[1:N]), ylim = c(0,60), col = "black", type =
'l')
lines(exp(cred_lower), col = "blue")
lines(exp(cred_upper), col = "green")

legend('topleft',legend = c('Data', 'Mean','Lower','Upper'), col = c('red',
'black', 'blue', 'green'), lwd=2)


##1d)

# The poissons model
Sigma_with_a_kind_of_specific_prior_Model = '
data {
  int<lower=0> N;
  int c[N];
}

parameters {
  real alpha;
  real <lower=-1, upper= 1> beta;
  real<lower=0> sigma;
  vector[N] Xt;
}

model {

  // Prior for sigma
  sigma ~ scaled_inv_chi_square(100,0.1);
  beta ~ uniform(-1,1);


  // Prior
  for (n in 2:N){
    Xt[n] ~ normal(alpha + beta * (Xt[n-1]), sigma);
  }
  // Model/likelihood
  for(n in 1:N){

  c[n] ~ poisson(exp(Xt[n]));

  }
}'

fit_Model<-stan(model_code=Sigma_with_a_kind_of_specific_prior_Model,
                  data=C_DataRStan,
                  warmup=1000,
                  iter=2000,
                   chains=4)

print(fit_Model,digits=4)

# Plot some results
res<-extract(fit_Model)


poster_means_vector = fit_Model@.MISC[["summary"]][["msd"]][-c(1,2,3)]

cred_lower = fit_Model@.MISC[["summary"]][["c_quan"]][,1,1][-c(1,2,3,144)]
cred_upper = fit_Model@.MISC[["summary"]][["c_quan"]][,9,1][-c(1,2,3,144)]

N = nrow(c_data)
xGrid = seq(1,140,1)
plot(xGrid, c_data$c, col = "red", main = "Possion model & Data", ylab = "nr
of Cases", xlab = "Draw")
lines(exp(poster_means_vector[1:N]), ylim = c(0,60), col = "black", type =
'l')
lines(exp(cred_lower), col = "blue")
lines(exp(cred_upper), col = "green")

legend('topleft',legend = c('Data', 'Mean','Lower','Upper'), col = c('red',
'black', 'blue', 'green'), lwd=2)
```

Mattias Villani
Department of Statistics
Stockholm University
Dept of Computer and Information Science
Linköping University

Bayesian Statistics I
Bayesian Learning

Computer Lab 1

You are recommended to use R for solving the labs.
You work and submit your labs in pairs, but both of you should contribute equally and understand all parts of your solutions.
`It is not allowed to share exact solutions` with other student pairs.
The submitted lab reports will be verified through URKUND and indications of plagiarism will be investigated by the Disciplinary Board.
Submit your solutions via LISAM, no later than April 18 at 23:30.

1. *Bernoulli ... again.*
   Let $y_1, ..., y_n|\theta \sim \text{Bern}(\theta)$, and assume that you have obtained a sample with $s = 5$ successes in $n = 20$ trials. Assume a $\text{Beta}(\alpha_0, \beta_0)$ prior for θ and let $\alpha_0 = \beta_0 = 2$.

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

   (b) Use simulation (`nDraws = 10000`) to compute the posterior probability $\Pr(\theta > 0.3|y)$ and compare with the exact value [Hint: `pbeta()`].

   (c) Compute the posterior distribution of the log-odds $\phi = \log \frac{\theta}{1-\theta}$ by simulation (`nDraws = 10000`). [Hint: `hist()` and `density()` might come in handy]

2. *Log-normal distribution and the Gini coefficient.*
   Assume that you have asked 10 randomly selected persons about their monthly income (in thousands Swedish Krona) and obtained the following ten observations: 44, 25, 45, 52, 30, 63, 19, 50, 34 and 67. A common model for non-negative continuous variables is the log-normal distribution. The log-normal distribution $\log \mathcal{N}(\mu, \sigma^2)$ has density function

$$p(y|\mu, \sigma^2) = \frac{1}{y \cdot \sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(\log y - \mu)^2\right],$$

   for $y > 0$, $\mu > 0$ and $\sigma^2 > 0$. The log-normal distribution is related to the normal distribution as follows: if $y \sim \log \mathcal{N}(\mu, \sigma^2)$ then $\log y \sim \mathcal{N}(\mu, \sigma^2)$. Let $y_1, ..., y_n|\mu, \sigma^2 \overset{iid}{\sim} \log \mathcal{N}(\mu, \sigma^2)$, where $\mu = 3.7$ is assumed to be known but σ^2 is unknown with non-informative prior $p(\sigma^2) \propto 1/\sigma^2$. The posterior for σ^2 is the $Inv - \chi^2(n, \tau^2)$ distribution, where

$$\tau^2 = \frac{\sum_{i=1}^{n}(\log y_i - \mu)^2}{n}.$$

(a) Simulate 10,000 draws from the posterior of σ^2 (assuming $\mu = 3.7$) and compare with the theoretical $Inv - \chi^2(n, \tau^2)$ posterior 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. See Wikipedia for more information. It can be shown that $G = 2\Phi\left(\sigma/\sqrt{2}\right) - 1$ when incomes follow a $\log\mathcal{N}(\mu, \sigma^2)$ 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.

(c) Use the posterior draws from b) to compute a 90% equal tail credible interval for G. A 90% equal tail interval (a, b) cuts off 5% percent of the posterior probability mass to the left of a, and 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 90% Highest Posterior Density interval for G. Compare the two intervals.

3. *Bayesian inference for the concentration parameter in the von Mises distribution.* This exercise is concerned with directional data. The point is to show you that the posterior distribution for somewhat weird models can be obtained by plotting it over a grid of values. The data points are observed wind directions at a given location on ten different days. The data are recorded in degrees:

$$(40, 303, 326, 285, 296, 314, 20, 308, 299, 296),$$

where North is located at zero degrees (see Figure 1 on the next page, where the angles are measured clockwise). To fit with Wikipedias description of probability distributions for circular data we convert the data into radians $-\pi \leq y \leq \pi$. The 10 observations in radians are

$$(-2.44, 2.14, 2.54, 1.83, 2.02, 2.33, -2.79, 2.23, 2.07, 2.02).$$

Assume that these data points are independent observations following the von Mises distribution

$$p(y|\mu, \kappa) = \frac{\exp\left[\kappa \cdot \cos(y - \mu)\right]}{2\pi I_0(\kappa)}, \quad -\pi \leq y \leq \pi,$$

where $I_0(\kappa)$ is the modified Bessel function of the first kind of order zero [see `?besselI` in R]. The parameter μ ($-\pi \leq \mu \leq \pi$) is the mean direction and $\kappa > 0$ is called the concentration parameter. Large κ gives a small variance around μ, and vice versa. Assume that μ is known to be 2.39. Let $\kappa \sim$ Exponential($\lambda = 1$) a priori, where λ is the rate parameter of the exponential distribution (so that the mean is $1/\lambda$).

(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).

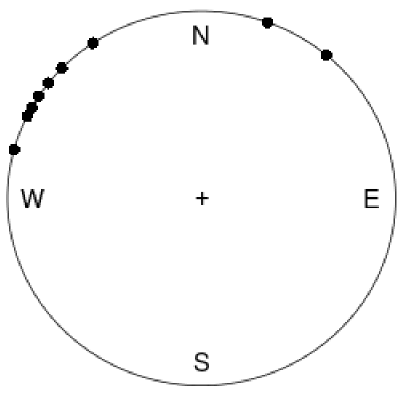


Figure 1: The wind direction data. Angles are measured clock-wise starting from North.

May Bayes be with you!

Mattias Villani
Department of Statistics
Stockholm University
Dept of Computer and Information Science
Linköping University

Bayesian Statistics I
Bayesian Learning

Computer Lab 2

You are recommended to use R for solving the labs.
You work and submit your labs in pairs, but both of you should contribute equally and understand all parts of your solutions.
`It is not allowed to share exact solutions` with other student pairs.
The submitted lab reports will be verified through URKUND and indications of plagiarism will be investigated by the Disciplinary Board.
Submit your solutions via LISAM, no later than April 29 at 23:30.

1. **Linear and polynomial regression**
   The dataset `TempLinkoping.txt` contains daily average temperatures (in Celcius degrees) at Malmslätt, Linköping over the course of the year 2018. The response variable is *temp* and the covariate is

$$time = \frac{\text{the number of days since beginning of year}}{365}.$$

   The task is to perform a Bayesian analysis of a quadratic regression

$$temp = \beta_0 + \beta_1 \cdot time + \beta_2 \cdot time^2 + \varepsilon,\ \varepsilon \overset{iid}{\sim} N(0, \sigma^2).$$

   (a) *Determining the prior distribution of the model parameters.* Use the conjugate prior for the linear regression model. Your task is to set the prior hyperparameters μ_0, Ω_0, ν_0 and σ_0^2 to sensible values. Start with $\mu_0 = (-10, 100, -100)^T$, $\Omega_0 = 0.01 \cdot I_3$, $\nu_0 = 4$ and $\sigma_0^2 = 1$. Check if this prior agrees with your prior opinions by simulating draws from the joint prior of all parameters and for every draw compute the regression curve. This gives a collection of regression curves, one for each draw from the prior. Do the collection of curves look reasonable? If not, change the prior hyperparameters until the collection of prior regression curves agrees with your prior beliefs about the regression curve.
   [Hint: the R package `mvtnorm` will be handy. And use your *Inv*-χ^2 simulator from Lab 1.]

   (b) Write a program that *simulates from the joint posterior distribution* of β_0, β_1,β_2 and σ^2. Plot the marginal posteriors for each parameter as a histogram. Also produce another figure with a scatter plot of the temperature data and overlay a curve for the posterior median of the regression function $f(time) = \beta_0 + \beta_1 \cdot time + \beta_2 \cdot time^2$, computed for every value of *time*. Also overlay curves for the lower 2.5% and upper 97.5% posterior credible interval for $f(time)$. That is, compute the 95% equal tail posterior probability intervals for every value of *time* and then connect the lower and upper limits of the interval by curves. Does the interval bands contain most of the data points? Should they?

(c) It is of interest to locate the *time* with the highest expected temperature (that is, the *time* where $f(time)$ is maximal). Let's call this value $\tilde{x}$. Use the simulations in b) to simulate from the *posterior distribution of* $\tilde{x}$. [Hint: the regression curve is a quadratic. You can find a simple formula for $\tilde{x}$ given β_0, β_1 and β_2.]

(d) Say now that you want to *estimate a polynomial model of order* 7, but you suspect that higher order terms may not be needed, and you worry about overfitting. Suggest a suitable prior that mitigates this potential problem. You do not need to compute the posterior, just write down your prior. [Hint: the task is to specify μ_0 and Ω_0 in a smart way.]

2. **Posterior approximation for classification with logistic regression**
The dataset `WomenWork.dat` contains $n = 200$ observations (i.e. women) on the following nine variables:

Variable	Data type	Meaning	Role
Work	Binary	Whether or not the woman works	Response
Constant	1	Constant to the intercept	Feature
HusbandInc	Numeric	Husband's income	Feature
EducYears	Counts	Years of education	Feature
ExpYears	Counts	Years of experience	Feature
ExpYears2	Numeric	(Years of experience/10)^2	Feature
Age	Counts	Age	Feature
NSmallChild	Counts	Number of child ≤ 6 years in household	Feature
NBigChild	Counts	Number of child > 6 years in household	Feature

(a) Consider the logistic regression

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

where y is the binary variable with $y = 1$ if the woman works and $y = 0$ if she does not. $\mathbf{x}$ is a 8-dimensional vector containing the eight features (including a one for the constant term that models the intercept).
The goal is to approximate the posterior distribution of the 8-dim parameter vector β with a multivariate normal distribution

$$\beta|\mathbf{y}, \mathbf{X} \sim N\left(\tilde{\beta}, J_{\mathbf{y}}^{-1}(\tilde{\beta})\right),$$

where $\tilde{\beta}$ is the posterior mode and $J(\tilde{\beta}) = -\frac{\partial^2 \ln p(\beta|\mathbf{y})}{\partial\beta\partial\beta^T}|_{\beta=\tilde{\beta}}$ is the observed Hessian evaluated at the posterior mode. Note that $\frac{\partial^2 \ln p(\beta|\mathbf{y})}{\partial\beta\partial\beta^T}$ is an 8×8 matrix with second derivatives on the diagonal and cross-derivatives $\frac{\partial^2 \ln p(\beta|\mathbf{y})}{\partial\beta_i\partial\beta_j}$ on the off-diagonal. It is actually not hard to compute this derivative by hand, but don't worry, we will let the computer do it numerically for you. Now, both $\tilde{\beta}$ and $J(\tilde{\beta})$ are computed by the `optim` function in R. See my code `https://github.com/mattiasvillani/BayesLearnCourse/raw/master/Code/MainOptimizeSpam.zip` where I have coded everything up for the spam prediction example (it also does probit regression, but that is not needed here). I want you to implement you own version of this. You can use my code as a template, but I want you

to write your own file so that you understand every line of your code. Don't just copy my code. Use the prior $\beta \sim \mathcal{N}(0, \tau^2 I)$, with $\tau = 10$.
Your report should include your code as well as numerical values for $\tilde{\beta}$ and $J_{\mathbf{y}}^{-1}(\tilde{\beta})$ for the `WomenWork` data. Compute an approximate 95% credible interval for the variable NSmallChild. Would you say that this feature is an important determinant of the probability that a women works?
[Hint: To verify that your results are reasonable, you can compare to you get by estimating the parameters using maximum likelihood: `glmModel <- glm(Work ~ 0 + ., data = WomenWork, family = binomial)`.]

(b) Write a function that simulates from the predictive distribution of the response variable in a logistic regression. Use your normal approximation from 2(a). Use that function to simulate and plot 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.
[Hints: The R package `mvtnorm` will again be handy. Remember my discussion on how Bayesian prediction can be done by simulation.]

(c) Now, consider 10 women which all have the same features as the woman in 2(b). Rewrite your function and plot the predictive distribution for the number of women, out of these 10, that are working. [Hint: Which distribution can be described as a sum of Bernoulli random variables?]

HAVE FUN!

Mattias Villani
Department of Statistics
Stockholm University
Dept of Computer and Information Science
Linköping University

Bayesian Statistics I
Bayesian Learning

Computer Lab 4

You are recommended to use R for solving the labs.
You work and submit your labs in pairs, but both of you should contribute equally. and understand all parts of your solutions.
`It is not allowed to share exact solutions` with other student pairs.
The submitted lab reports will be verified through URKUND and indications of plagiarism will be investigated by the Disciplinary Board.
Submit your solutions via LISAM, no later than May 26 at 23:30.

1. *Time series models in Stan*

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

   $$x_t = \mu + \phi\left(x_{t-1} - \mu\right) + \varepsilon_t, \;\; \varepsilon_t \overset{iid}{\sim} N\left(0, \sigma^2\right),$$

   for given values of μ, ϕ and σ^2. Start the process at $x_1 = \mu$ and then simulate values for x_t for $t = 2, 3 \ldots, T$ and return the vector $x_{1:T}$ containing all time points. Use $\mu = 10, \sigma^2 = 2$ and $T = 200$ and look at some different realizations (simulations) of $x_{1:T}$ for values of ϕ between -1 and 1 (this is the interval of ϕ where the AR(1)-process is stable). Include a plot of at least one realization in the report. What effect does the value of ϕ have on $x_{1:T}$?

   (b) Use your function from a) to simulate two AR(1)-processes, $x_{1:T}$ with $\phi = 0.3$ and $y_{1:T}$ with $\phi = 0.95$. Now, treat your simulated vectors as synthetic data, and treat the values of μ, ϕ and σ^2 as unknown and estimate them using MCMC. Implement Stan-code that samples from the posterior of the three parameters, using suitable non-informative priors of your choice. [Hint: Look at the time-series models examples in the Stan user's guide/reference manual, and note the different parameterization used here.]

      i. Report the posterior mean, 95% credible intervals and the number of effective posterior samples for the three inferred parameters for each of the simulated AR(1)-process. Are you able to estimate the true values?

      ii. For each of the two data sets, evaluate the convergence of the samplers and plot the joint posterior of μ and ϕ. Comments?

   (c) The data `campy.dat` contain the number of cases of campylobacter infections in the north of the province Quebec (Canada) in four week intervals from January 1990 to the end of October 2000. It has 13 observations per year and 140 observations in total. 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

   $$c_t | x_t \sim Poisson\left(\exp\left(x_t\right)\right),$$

where x_t is an AR(1)-process as in a). Implement and estimate the model in Stan, using suitable priors of your choice. Produce a plot that contains both the data and the posterior mean and 95% credible intervals for the latent intensity $\theta_t = \exp(x_t)$ over time. [Hint: Should x_t be seen as data or parameters?]

(d) Now, assume that we have a prior belief that the true underlying intensity θ_t varies more smoothly than the data suggests. Change the prior for σ^2 so that it becomes informative about that the AR(1)-process increments ε_t should be small. Re-estimate the model using Stan with the new prior and produce the same plot as in c). Has the posterior for θ_t changed?

HAVE FUN!

Mattias Villani
Department of Statistics
Stockholm University
Dept of Computer and Information Science
Linköping University

Bayesian Statistics I
Bayesian Learning

Computer Lab 3

You are recommended to use R for solving the labs.
You work and submit your labs in pairs, but both of you should contribute equally and understand all parts of your solutions.
`It is not allowed to share exact solutions` with other student pairs.
The submitted lab reports will be verified through URKUND and indications of plagiarism will be investigated by the Disciplinary Board.
Submit your solutions via LISAM, no later than May 17 at 23:30.

1. *Normal model, mixture of normal model with semi-conjugate prior.*
   The data `rainfall.dat` consist of daily records, from the beginning of 1948 to the end of 1983, of precipitation (rain or snow in units of $\frac{1}{100}$ inch, and records of zero precipitation are excluded) at Snoqualmie Falls, Washington. Analyze the data using the following two models.

   (a) *Normal model.*
   Assume the daily precipitation $\{y_1, ..., y_n\}$ are independent normally distributed, $y_1, ..., y_n|\mu, \sigma^2 \sim \mathcal{N}(\mu, \sigma^2)$ where both μ and σ^2 are unknown. Let $\mu \sim \mathcal{N}(\mu_0, \tau_0^2)$ independently of $\sigma^2 \sim Inv\text{-}\chi^2(\nu_0, \sigma_0^2)$.

   i. Implement (code!) a Gibbs sampler that simulates from the joint posterior $p(\mu, \sigma^2|y_1, ..., y_n)$. The full conditional posteriors are given on the slides from Lecture 7.
   ii. Analyze the daily precipitation using your Gibbs sampler in (a)-i. Evaluate the convergence of the Gibbs sampler by suitable graphical methods, for example by plotting the trajectories of the sampled Markov chains.

   (b) *Mixture normal model.*
   Let us now instead assume that the daily precipitation $\{y_1, ..., y_n\}$ follow an iid two-component **mixture of normals** model:

$$p(y_i|\mu, \sigma^2, \pi) = \pi\mathcal{N}(y_i|\mu_1, \sigma_1^2) + (1-\pi)\mathcal{N}(y_i|\mu_2, \sigma_2^2),$$

   where

$$\mu = (\mu_1, \mu_2) \quad \text{and} \quad \sigma^2 = (\sigma_1^2, \sigma_2^2).$$

   Use the Gibbs sampling data augmentation algorithm in `NormalMixtureGibbs.R` (available under Lecture 7 on the course page) to analyze the daily precipitation data. Set the prior hyperparameters suitably. Evaluate the convergence of the sampler.

(c) *Graphical comparison.*
Plot the following densities in one figure: 1) a histogram or kernel density estimate of the data. 2) Normal density $\mathcal{N}(y_i|\mu,\sigma^2)$ in (a); 3) Mixture of normals density $p(y_i|\mu,\sigma^2,\pi)$ in (b). Base your plots on the mean over all posterior draws.

2. *Metropolis Random Walk for Poisson regression.*
Consider the following Poisson regression model

$$y_i|\beta \sim \text{Poisson}\left[\exp\left(\mathbf{x}_i^T\beta\right)\right],\ i = 1, ..., n,$$

where y_i is the count for the ith observation in the sample and x_i is the p-dimensional vector with covariate observations for the ith observation. Use the data set `eBayNumberOfBidderData.dat`. This dataset contains observations from 1000 eBay auctions of coins. The response variable is **nBids** and records the number of bids in each auction. The remaining variables are features/covariates ($\mathbf{x}$):

- **Const** (for the intercept)
- **PowerSeller** (is the seller selling large volumes on eBay?)
- **VerifyID** (is the seller verified by eBay?)
- **Sealed** (was the coin sold sealed in never opened envelope?)
- **MinBlem** (did the coin have a minor defect?)
- **MajBlem** (a major defect?)
- **LargNeg** (did the seller get a lot of negative feedback from customers?)
- **LogBook** (logarithm of the coins book value according to expert sellers. Standardized)
- **MinBidShare** (a variable that measures ratio of the minimum selling price (starting price) to the book value. Standardized).

(a) Obtain the maximum likelihood estimator of β in the Poisson regression model for the eBay data [Hint: `glm.R`, don't forget that `glm()` adds its own intercept so don't input the covariate Const]. Which covariates are significant?

(b) Let's now do a Bayesian analysis of the Poisson regression. Let the prior be $\beta \sim \mathcal{N}\left[\mathbf{0}, 100 \cdot (\mathbf{X}^T\mathbf{X})^{-1}\right]$ where $\mathbf{X}$ is the $n \times p$ covariate matrix. This is a commonly used prior which is called Zellner's g-prior. Assume first that the posterior density is approximately multivariate normal:

$$\beta|y \quad \sim \quad \mathcal{N}\left(\tilde{\beta}, J_{\mathbf{y}}^{-1}(\tilde{\beta})\right),$$

where $\tilde{\beta}$ is the posterior mode and $J_{\mathbf{y}}(\tilde{\beta})$ is the negative Hessian at the posterior mode. $\tilde{\beta}$ and $J_{\mathbf{y}}(\tilde{\beta})$ can be obtained by numerical optimization (`optim.R`) exactly like you already did for the logistic regression in Lab 2 (but with the log posterior function replaced by the corresponding one for the Poisson model, which you have to code up.).

(c) Now, let's simulate from the actual posterior of β using the Metropolis algorithm and compare with the approximate results in b). Program a general function that uses the Metropolis algorithm to generate random draws from an *arbitrary* posterior density. In order to show that it is a general function for any model, I will denote the vector of model parameters by θ. Let the proposal density be the multivariate normal density mentioned in Lecture 8 (random walk Metropolis):

$$\theta_p|\theta^{(i-1)} \sim N\left(\theta^{(i-1)}, c \cdot \Sigma\right),$$

where $\Sigma = J_{\mathbf{y}}^{-1}(\tilde{\beta})$ obtained in b). The value c is a tuning parameter and should be an input to your Metropolis function. The user of your Metropolis function should be able to supply her own posterior density function, not necessarily for the Poisson regression, and still be able to use your Metropolis function. This is not so straightforward, unless you have come across *function objects* in `R` and the triple dot (...) wildcard argument. I have posted a note (HowToCodeRWM.pdf) on the course web page that describes how to do this in R.
Now, use your new Metropolis function to sample from the posterior of β in the Poisson regression for the eBay dataset. Assess MCMC convergence by graphical methods.

(d) Use the MCMC draws from c) to simulate from the predictive distribution of the number of bidders in a new auction with the characteristics below. Plot the predictive distribution. What is the probability of no bidders in this new auction?

- **PowerSeller** = 1
- **VerifyID** = 1
- **Sealed** = 1
- **MinBlem** = 0
- **MajBlem** = 0
- **LargNeg** = 0
- **LogBook** = 1
- **MinBidShare** = 0.5

HAVE FUN!