Labs

Christian von Koch

2020-05-31

#Lab 1

##Assignment 1

## Assignment 1: Let y1,...,yn be Bernoulli distributed with parameter theta. Assume that you have obtained a sample  
## with s=5 successes in n=20 trials. Assume a Beta(alpha0, beta0) prior for theta and let alpha0=beta0=2.  
  
## a) Draw random numbers from the posterior theta given y ~ Beta(alpha0+s, beta0+f) and verify graphically that the  
## posterior mean and standard deviation converges to the true values as the number of random draws grows large.   
  
set.seed(12345)  
alpha0=2  
beta0=2  
s=5  
f=15  
n=20  
  
# Function for calculating the mean of a beta-distribution  
calcMeanBeta = function(alpha, beta) {  
 return(alpha/(alpha+beta))  
}  
  
# Function for calculating the standard deviation of a beta-distribution  
calcStdDevBeta = function(alpha, beta) {  
 return(sqrt(alpha\*beta/((alpha+beta)^2\*(alpha+beta+1))))  
}  
  
# Function for calculating the mean squared error of drawed data  
calcMSE = function(n, mean, data){  
 return(sqrt(1/(n-1)\*sum((data-mean)^2)))  
}  
  
# Function for drawing random values from the betadistribution  
drawBetaValues = function(n, alpha, beta) {  
 return(rbeta(n, alpha, beta))  
}  
  
MeanOfPosterior = calcMeanBeta(alpha0+s, beta0+f)  
StdOfPosterior = calcStdDevBeta(alpha0+s, beta0+f)  
nVector = seq(1, 5000, 1)  
meanVector=c()  
stdVector=c()  
for (i in nVector) {  
 set.seed(12345)  
 betaValues= drawBetaValues(i, alpha0+s, beta0+f)  
 meanVector=c(meanVector, mean(betaValues))  
 stdVector=c(stdVector, calcMSE(i, mean(betaValues), betaValues))  
}  
plot(nVector, meanVector, main="Plot of how the mean converges with respect to number of draws",  
 xlab="Number of draws", ylab="Mean", type="l")  
abline(h=MeanOfPosterior, col="red")  
plot(nVector, stdVector, main="Plot of how the standard deviation converges with respect to the number of draws",  
 xlab="Number of draws", ylab="Standard deviation", type="l")  
abline(h=StdOfPosterior, col="red")  
## As seen in the plot the posterior mean as well as the posterior standard deviation converges towards its true  
## value of approx 0.29 and 0.09 respectively as the number of randow draws grows large.  
  
## b) Use simulation (nDraws=10000) to compute the posterior probability Pr(theta>0.3 given y) and compare with  
## with the exact value  
  
trueProb=1-pbeta(0.3, alpha0+s, beta0+f)  
set.seed(12345)  
draw10000=rbeta(10000, alpha0+s, beta0+f)  
probHat=sum(draw10000>0.3)/10000  
print(trueProb)  
print(probHat)  
  
## As seen in the results from both calculations the probHat is very close to the true probability from the beta  
## distribution. As the number of draws increases the approximated probability will converge towards the true  
## value.  
  
## c) Compute the posterior distribution of the log-odds phi= log(theta/(1-theta)) by simulation (nDraws=10000)  
  
phi=log(draw10000/(1-draw10000))  
hist(phi, breaks=20, main="Distribution of the log-odds")  
plot(density(phi), main="Density function of phi")

##Assignment 2

## Assignment 2: Assume that you have asked 10 randomly selected persons about their monthly   
## income(inthousandsSwedishKrona)andobtainedthefollowingtenobservations: 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(N(my, sigma^2)) has density function ... for y > 0, my > 0 and sigma > 0. The log-normal  
## distribution is related to the normal distribution as follows: if y ~ log N(my, sigma^2) then   
## log y ~ N(my, sigma^2). Let y1,...,yn given my and simga^2 ~ log N(my, sigma^2), where my=3.7 is assumed to be  
## known but sigma^2 is unknown with noninformative prior p(sigma^2) is proportional to 1/sigma^2. The posterior  
## for sigma^2 is the Inv - chitwo distribution with X(n, thao^2) distribution, where thao^2=sum((log(yi)-my)^2)/n  
  
## a) Simulate 10 000 draws from the posterior of sigma^2 (assuming my=3.7) and compare with the theoretical   
## with the theoretical Inv - chitwo distribution with X(n, thao^2) posterior distribution.  
  
library(geoR)  
x=c(44, 25, 45, 52, 30, 63, 19, 50, 34, 67)  
n=length(x)  
my=3.7  
  
#Function for calculating thao^2  
calcThao = function(data, my, n) {  
 return(sum((log(data)-my)^2)/n)  
}  
  
invchisquare <- function(x, df, taosq){  
 first = ((taosq\*df/2)^(df/2))/gamma(df/2)  
 second = (exp((-df\*taosq)/(2\*x)))/(x^(1+df/2))  
 return(first\*second)  
}  
  
thaosq=calcThao(x, my, n)  
set.seed(12345)  
drawX=rchisq(10000, n)  
sigmasq=(n)\*thaosq/drawX  
xvals=seq(0.001, 3, 0.001)  
plot(density(sigmasq), main="Density of simulated sigma^2, black = simulated distrib., red = actual distrib.")  
lines(xvals,invchisquare(xvals, n, thaosq), col="red")  
  
## As seen in the plot the theoretical distribution (red line) follows the simulated one with good precision. This  
## indicates that the simulation has been made correctly.  
  
## b) The most common measure of income inequality is the Gini coefficient, G, where 0<=G<=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\*CDF-normal(sigma/sqrt(2))-1 when income follow a log N(my, sigma^2)  
## distribution. Use the posterior draws in a) to compute the posterior distribution of the Gini coefficient G  
## for the current data set.  
  
G=2\*pnorm(sqrt(sigmasq/2), mean=0, sd=1)-1  
hist(G, breaks=100)  
plot(density(G), main="Density function of simulated values of the Gini coefficient")  
  
## As seen in the plot the gini coefficient is centered at around 0.2 which means a rather inequal distribution.  
  
## 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 defaultsettings,   
## and use that kernel density estimate to compute a 90% Highest Posterior Density interval for G. Compare the   
## two intervals.  
  
GSorted=sort(G)[(0.05\*length(G)+1):(0.95\*length(G))]  
# 90 % credible interval for G through the simulated draws  
G\_CredInterval=c(min(GSorted),max(GSorted))  
print(G\_CredInterval)  
plot(density(G), main="Density function of simulated values of the Gini coefficient with credible intervals")  
abline(v = G\_CredInterval[1], col="blue")  
abline(v = G\_CredInterval[2], col="blue")  
  
GDensity=density(G)  
GDensity.df=data.frame(x=GDensity$x, y=GDensity$y)  
GDensity.df=GDensity.df[order(-GDensity.df[,2]),]  
index=dim(GDensity.df)[1]  
GDensity.df$y=cumsum(GDensity.df$y)/sum(GDensity.df$y)  
GDensity\_CredInterval\_Vals=GDensity.df[GDensity.df$y<0.90,]  
GDensity\_CredInterval=c(min(GDensity\_CredInterval\_Vals$x), max(GDensity\_CredInterval\_Vals$x))  
print(GDensity\_CredInterval)  
abline(v = GDensity\_CredInterval[1], col="red")  
abline(v = GDensity\_CredInterval[2], col="red")  
title(sub="Blue = Simulated credible interval, Red = Kernel estimated credible interval")  
  
## As seen in the plot the credible intervals are quite similar with small deviations.

##Assignment 3

## Assignment 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<=y<=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 given my,k) = exp(k\*cos(y-u))/(2\*pi\*I0(k)), -pi<=y<=pi, where I0(k) is the modified Bessel function of the   
## first kind of order zero (see ?besselI in R). The parameter my (-pi<=my<=pi) is the mean direction and k>0 is  
## called the concentration parameter. Large k gives a small variance around my, and vice versa. Assume that my is  
## known to be 2.39. Let K ~ Exponential(Lambda=1) a priori, where lambda is the rate parameter of the exponential  
## distribution (so that the mean is 1/lambda).  
  
## a) Plot the posterior distribution of k for the wind direction data over a fine grid of k values.   
  
data\_radian=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  
  
# Function for computing the vonMisesDistrib for a given dataset  
vonMisesDistrib = function(kappa, data, my){  
 likelihood=1  
 for (i in data) {  
 likelihood=likelihood\*exp(kappa\*cos(i-my))/(2\*pi\*besselI(kappa, 0))  
 }  
 return(likelihood)  
}  
  
# Function for computing the exponential distribution  
exponDistrib = function(data, lambda) {  
 return(lambda\*exp(-lambda\*data))  
}  
  
kappa\_values=seq(0,10,0.01)  
  
# Function for computing the posterior distribution  
posteriorDistrib = function(kappa, lambda, data, my) {  
 likelihood=vonMisesDistrib(kappa, data, my)  
 prior=exponDistrib(kappa, lambda)  
 return(likelihood\*prior)  
}  
  
posteriorLikelihood=posteriorDistrib(kappa\_values, lambda, data\_radian, my)  
posterior.df=data.frame(kappa=kappa\_values, likelihood=posteriorLikelihood)  
sumOfPosterior=sum(posterior.df$likelihood)  
posterior.df$likelihood=posterior.df$likelihood\*(1/sumOfPosterior)  
final\_sum=sum(posterior.df$likelihood)  
plot(kappa\_values, posterior.df$likelihood, xlab="Kappa", ylab="Likelihood",  
 main="Posterior likelihood for different kappavalues", type="l", col="blue")  
  
## As seen in the plot the likelihood of the posterior peaks between 2 and 4 and then dies off for larger  
## kappa-values.  
  
## b) Find the (approximate) posterior mode of k from the information in a).  
  
# Puts likelihood values with corresponding kappa-values to be able to retrieve the kappa-value corresponding to  
## the highest likelihood (mode)  
  
posteriorMode=subset(posterior.df, likelihood==max(likelihood), kappa)  
print(posteriorMode$kappa)  
  
## The approximated posterior mode is found to be 2.12.

#Lab 2

##Assignment 1

## Assignment 1: The dataset TempLinkoping.txt contains daily average tamperatures (in Celcius degrees) at   
## Malmslatt, Linkoping over the course of the year 2018. The response variable is temp and the covariate is   
## time=(the number of days since beginning of year)/365  
## You're task is to perform a Bayesian analysis of a quadratic regression  
## temp=beta0+beta1\*time+beta2\*time^2+epsilon, epsilon~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 my0, omega0, v0 and sigma0^2 to sensible   
## values. Start with my0=(-10,100,-100)T, omega0=0.01\*I3, v0=4 and sigma0^2=1. 0 = 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-chisquared simulator from Lab1.  
  
# Read file  
temp = read.table("TempLinkoping.txt", header=TRUE)  
  
## install.packages("mvtnorm")  
library(mvtnorm)  
# Defining the parameters for the prior distribution  
# Switched to beta0=0 since it seems more reasonable and -10 seems too low.  
my0=c(-10,100,-100)  
omega0=0.5\*diag(3)  
# Using v0 = 365 since we have 365 observations  
v0=365  
sigma0\_sq=0.5  
omega0Inv=solve(omega0)  
  
# Function for returning the response variable  
calcRegr = function(betaMatrix, row, x) {  
 return(betaMatrix[row,1]+betaMatrix[row,2]\*x+betaMatrix[row,3]\*x^2)  
}  
  
# Function for drawing simulated betavalues  
drawBeta = function(my, sigma\_sq, omegaInv) {  
 return(rmvnorm(1, mean=my, sigma=sigma\_sq\*omegaInv))  
}  
  
nDraws=1000  
set.seed(12345)  
drawX=rchisq(nDraws, v0)  
sigma\_sq=(v0)\*sigma0\_sq/drawX  
betaMatrix=matrix(0,nDraws,3)  
# Create new plot with specific settings so that the loop can overlay plots  
plot.new()  
plot.window(xlim=c(0,1), ylim=c(-50,50))  
axis(side=1)  
axis(side=2)  
set.seed(12345)  
for (i in 1:nDraws) {  
 betaMatrix[i,]=drawBeta(my0, sigma\_sq[i], omega0Inv)  
 lines(temp$time, calcRegr(betaMatrix, i, temp$time), col=rgb(0,0,0,0.2))  
}  
title(main="Temps depending on different times for different simulated models", xlab="Time", ylab="Temp")  
  
## The collection of curves look reasonable and in line with our prior beliefs. The temperature rises during the  
## summer months and stays low in the beginning and the end of the year respectively.However, the value of -10  
## were switched to 0 since it seems more reasonable with a measurement of the temperature 0 on the 1st of  
## January than a measurement of -10.  
  
## b) Write a program that simulates from the joint posterior distribution of beta0, beta1, beta2 and sigma^2.  
## Plot the marginal posteriors of 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)=beta0+beta1\*time+beta2\*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?  
  
# Calculating the parameters for the posterior distribution  
v\_n=v0+length(temp$temp)  
X=cbind(1, temp$time, temp$time^2)  
Y=temp$temp  
beta\_hat=solve(t(X)%\*%X)%\*%t(X)%\*%Y  
my\_n=solve(t(X)%\*%X+omega0)%\*%(t(X)%\*%X%\*%beta\_hat+omega0%\*%my0)  
omega\_n=t(X)%\*%X+omega0  
omega\_n\_Inv=solve(omega\_n)  
sigma\_sq\_n=(v0\*sigma0\_sq+(t(Y)%\*%Y+t(my0)%\*%omega0%\*%my0-t(my\_n)%\*%omega\_n%\*%my\_n))/v\_n  
  
# Simulate the joint posterior  
sigma\_sq\_post=(v\_n)\*c(sigma\_sq\_n)/drawX  
betaMatrix\_post=matrix(0,nDraws,3)  
response\_post\_temp=matrix(0,nDraws,length(temp$time))  
for (i in 1:nDraws) {  
 betaMatrix\_post[i,]=drawBeta(my\_n, sigma\_sq\_post[i], omega\_n\_Inv)  
}  
# Plots the marginal distributions for the different beta-values  
hist(betaMatrix\_post[,1], breaks=100, main="Marginal posterior for beta0")  
hist(betaMatrix\_post[,2], breaks=100, main="Marginal posterior for beta1")  
hist(betaMatrix\_post[,3], breaks=100, main="Marginal posterior for beta2")  
hist(sigma\_sq\_post, breaks=100, main="Marginal posterior for sigmasq")  
  
plot(temp$time, Y, main="Plot of the temp data for different times", col="blue",   
 xlab="Time coefficient", ylab="Temp")  
# Applies function calcRegr to the time-values for each of the drawn betas and stores the results in matrix  
for (i in 1:nDraws) {  
 betaTemp=sapply(temp$time, calcRegr, betaMatrix=betaMatrix\_post, row=i)  
 response\_post\_temp[i,]=betaTemp  
}  
  
response\_post=c()  
credInterval=matrix(0, length(temp$time), 2)  
# Retrieves the median of the response values as well as obtaining the upper and lower bound of credInterval  
for (i in 1:length(temp$time)) {  
 sortedTemp=sort(response\_post\_temp[,i])  
 response\_post=c(response\_post, (sortedTemp[500]+sortedTemp[501])/2)  
 credInterval[i,]=quantile(response\_post\_temp[,i], probs=c(0.025, 0.975))  
}  
  
lines(temp$time, response\_post)  
lines(temp$time, credInterval[,1], lty=21, col="gray")  
lines(temp$time, credInterval[,2], lty=21, col="gray")  
title(sub="Grey = 95 % credible intervals, Black = Median")  
  
## The interval bands contain most of the data points. They should contain most of the data points if the model  
## is accurate in terms of describing the reality. In this case, it seems like the model has captured most of  
## the data points which means that the model describes the reality fairly well.   
  
## 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 xtilde. Use the simulations in b) to simulate from posterior  
## distribution of xtilde. [Hint: The regression curve is quadratic. You can find a simple formula for xtilde  
## given beta0, beta1 and beta2]  
  
# Function for calculating the time-value which yields the maximum response (the derivative of response function)  
calcMaxTemp = function(betaMatrix, row) {  
 return(-betaMatrix[row,2]/(2\*betaMatrix[row,3]))  
}  
  
# For each of the draws the time-value which yields the maximum temperature is stored in a vector  
time\_max\_temp=c()  
for (i in 1:nDraws) {  
 time\_max\_temp=c(time\_max\_temp, calcMaxTemp(betaMatrix\_post, i))  
}  
  
hist(time\_max\_temp, breaks=10, xlim=c(0,1), main="Frequency of max temperatures simulated from xtilde",  
 xlab="Temperature")  
  
## As seen in the histogram the derived highest temperature from the simulated models is mostly present in late  
## june which seems reasonable if applying to Malmslatt in Sweden where the temperature is the highest during the   
## summer time.   
  
## 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  
## my0 and omega0 in a smart way.]  
  
## A suitable prior for this task would be to set my0 to 0 since you want most of the coefficients close to zero  
## to obtain increased shrinkage. You would also want to set omega0 to Lambda\*IdentityMatrix. This would mean  
## that for larger values of lambda more and more of the beta values would be close to zero since the spread of  
## the distribution of the beta values would decrease. In this case, where there is a worry about overfitting,  
## it might be a good idea to choose a large lambda to decrease the spread of the beta values and increase the  
## probability that most of the beta values are around 0.

##Assignment 2

## Assignment 2: Consider the logistic regression Pr(y=1 given x)=exp(xT\*Beta)/(1+exp(xT\*Beta)), where y is the  
## binary variable with y = 1 if the woman works and y = 0 if she does not. 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 beta with multivariate normal distribution.  
## Beta given y and x ~ N(~Beta, JY(~Beta)^(-1)), where ~Beta is the posterior mode and J(~Beta) is the second  
## derivative, the observed Hessian evaluated at the posterior mode. 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 ~Beta and  
## J(~Beta) are computed by the optim function in R. I want you to implement an own version of my example code at  
## the website. 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 ~ N(0, thao^2\*I) with thao=10. Your report  
## should include your code as well as numerical values for ~Beta and JY(~Beta)^(-1) 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)  
  
# Use of libraries  
library(mvtnorm)  
  
# Read data  
WomenWork = read.table("WomenWork.dat", header=TRUE)  
  
# User input  
tau = 10  
  
# Defining vectors X and Y  
X = as.matrix(WomenWork[,2:ncol(WomenWork)])  
Y = WomenWork[,1]  
nFeatures = dim(X)[2]  
covNames=names(WomenWork[,2:ncol(WomenWork)])  
  
# Constructing prior  
mu\_prior = rep(0,nFeatures)  
sigma\_prior = tau^2\*diag(nFeatures)   
  
# Defining function for returning the log posterior  
  
logPostLogistic = function(beta, Y, X, mu, sigma) {  
 nFeat = length(beta)  
 XBeta=X%\*%beta  
 # Defining loglikelihood  
 logLike = sum(Y\*XBeta-log(1+exp(XBeta)))  
 # Defining prior  
 prior = dmvnorm(beta, mean=mu, sigma=sigma, log=TRUE)  
 # Adding loglikelihood and logprior together. Since it is log both of them are added instead of multiplied  
 return(logLike + prior)  
}  
  
# Defining initial values to be passed on to the optimizer  
set.seed(12345)  
initVals = rnorm(dim(X)[2])  
  
# Finding the optimized betavector  
optimResult = optim(initVals, logPostLogistic, Y=Y, X=X, mu=mu\_prior, sigma=sigma\_prior, method=c("BFGS"),  
 control=list(fnscale=-1), hessian=TRUE)  
  
# Defining the values of interest  
postMode = optimResult$par  
postCov = -solve(optimResult$hessian)  
names(postMode) = covNames  
approx\_PostStd = sqrt(diag(postCov))  
names(approx\_PostStd) = covNames  
print("The posterior mode is:")  
print(postMode)  
print("The approximated standard deviations are:")  
print(approx\_PostStd)  
  
# Compute marginal distribution for nSmallChild  
NSmallChild\_mode = as.numeric(postMode["NSmallChild"])  
NSmallChild\_std = as.numeric(approx\_PostStd["NSmallChild"])  
credInterval\_NSmallChild = qnorm(p=c(0.025, 0.975), mean=NSmallChild\_mode, sd=NSmallChild\_std)  
print(paste("The lower bound of the 95 % credible interval for the feature NSmallChild is",  
 round(credInterval\_NSmallChild[1], 6), "and the upper bound is",   
 round(credInterval\_NSmallChild[2], 6)))  
  
# Control that the calculations have been made correctly  
glmModel = glm(Work ~ 0+., data=WomenWork, family=binomial)  
print(glmModel$coefficients)  
print(postMode)  
  
## Since the values for the credible interval for NSmallChild are quite large in the negative direction it is  
## reasonable to conclude that the feature NSmallChild affects the response variable farily much towards the   
## response 0 which means that the woman doesn't work. This seems like a reasonable conclusion in terms of how  
## it is in reality as well. When checking if the results are reasonable, a comparison was made with an   
## estimation using the maximum likelihood method. The results was very similar which strongly suggests that   
## the results obtained from the code are reasonable.   
  
## 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.]  
  
sigmoid = function(value) {  
 return (exp(value)/(1+exp(value)))  
}  
  
makePredLogReg = function(data, mean, sigma, nDraws) {  
 betaPred = rmvnorm(nDraws, mean=mean, sigma=sigma)  
 linearPred = betaPred %\*% data  
 logPred = sigmoid(linearPred)  
 return(logPred)  
}  
  
nDraws=10000  
woman=c(1, 10, 8, 10, (10/10)^2, 40, 1, 1)  
set.seed(12345)  
womanWorkPred=makePredLogReg(woman, postMode, postCov, nDraws)  
logistic\_distrib=c()  
for (i in womanWorkPred) {  
 logistic\_distrib=c(logistic\_distrib, rbinom(1, 1, i))  
}   
barplot(table(logistic\_distrib), main="Histogram of the predicted probabilities")  
  
## As seen in the plots the calculated probabilities of the woman in question working is fairly low. The highest  
## density is seen in the range between 0.2 and 0.3 approximately. This also makes sense if applied to a real  
## situation. A woman with a small child is likely to stay at home with the child, i.e. not working. If the   
## classification of the response variable results in "working" if the predicted probabilities is above 0.5 and  
## "not working" otherwise, it is clear from the distribution that the classification of a woman working, with  
## the parameters inputted, is very unlikely to happen.   
  
## 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?]  
  
makePredLogRegMultiple = function(data, mean, sigma, nDraws, n) {  
 multiplePred=c()  
 for (i in 1:nDraws) {  
 betaDraw = makePredLogReg(data, mean, sigma, 1)  
 multiplePred=c(multiplePred, rbinom(1, n, betaDraw))  
 }  
 barplot(table(multiplePred), main=paste("Distribution for prediction made on", n, "women"),   
 xlab="No. of women")  
}  
  
makePredLogRegMultiple(woman, postMode, postCov, 10000, 10)  
  
## As seen in the histogram the binomial case resembles the density of predicted probabilities with the  
## highest density found at 2 women. This result seems reasonable since when the number of draws taken from   
## the binomial distribution goes towards infinity the shape of the corresponding distribution will resemble  
## the shape of the distribution for the probability p in the Bernoulli case, more and more.

#Lab 3

##Assignment 1

## Assignment 1: 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 1/100 inch, and records of zero precipitation are exluded) at   
## Snoqualmie Falls Washington. Analyze the data using the following two models.  
  
## a) Assume the daily precipitation (y1,...,yn) are iid normally distributied,   
## y1,...,yn given mu and sigma^2 ~ N(mu,sigma^2) where both mu and sigma^2 are unknown. Let mu ~ N(mu0, tao0^2)  
## independently of sigma^2 ~ Inv chisquare(v0, sigma0^2)  
## i) Implement (code!) a Gibbs sampler that simulates from the joint posterior p(mu, sigma^2 given y1,...,yn).  
## The full conditional posteriors are given on the slides from Lecture 7.   
  
library(mvtnorm)  
# Read data  
Rainfall = read.table("rainfall.dat")  
  
# Setup  
# Prior knowledge of mu0 taken from Google  
mu0=14.79  
mean\_rainfall=mean(Rainfall[,1])  
tao0sq=100  
v0=1  
sigma0sq=1  
# Initial sigma value for Gibbs sampling  
n=dim(Rainfall)[1]  
vn=v0+n  
nDraws=5000  
  
# Function for calculating tao\_n which is used as argument for the std dev for the normal distribution of mu  
calcTaoN = function(sigmasq,tao0sq,n){  
 return(1/(n/sigmasq+1/tao0sq))  
}  
  
calcMuN = function(sigmasq, tao0sq, mu0, mean, n) {  
 w=(n/sigmasq)/(n/sigmasq+1/tao0sq)  
 return(w\*mean+(1-w)\*mu0)  
}  
  
calcSigmaHat = function(v0, sigma0sq, data, mu, n) {  
 return((v0\*sigma0sq+sum((data-mu)^2))/(n+v0))  
}  
posteriorMatrix = matrix(0, nDraws, 2)  
# Setting initial value of sigma^2 to 1  
posteriorMatrix[1,2]=1  
for (i in 1:nDraws) {  
 posteriorMatrix[i,1] = rnorm(1, calcMuN(posteriorMatrix[i,2],tao0sq, mu0, mean\_rainfall, n),  
 calcTaoN(posteriorMatrix[i,2], tao0sq, n))  
 if(i<nDraws) {  
 drawX=rchisq(1,vn)  
 posteriorMatrix[i+1,2]=vn\*calcSigmaHat(v0, sigma0sq, Rainfall[,1], posteriorMatrix[i,1], n)/drawX  
 }  
}  
  
# The posterior coverage  
plot(posteriorMatrix[1001:nrow(posteriorMatrix),1], posteriorMatrix[1001:nrow(posteriorMatrix),2], xlab="Mu",   
 ylab="Sigma^2")  
  
## ii) AnalyzethedailyprecipitationusingyourGibbssamplerin(a)-i. Evaluate the convergence of the Gibbs sampler  
## by suitable graphical methods, for example by plotting the trajectories of the sampled Markov chains.   
  
iter=seq(1001,5000,1)  
plot(iter, posteriorMatrix[1001:nrow(posteriorMatrix),1], type="l", xlab="Iteration",   
 ylab="Mu", main="Marginal posterior for mu")  
plot(iter, posteriorMatrix[1001:nrow(posteriorMatrix),2], type="l", xlab="Iteration",   
 ylab="Sigma", main="Marginal posterior for sigma")  
  
## b) Let us now instead assume that the daily precipitation {y1,...,yn} follow an iid two-component mixture  
## of normals model: p(yi given mu, sigma^2, pi)=pi\*N(yi given my1, sigma1^2)+(1-pi)\*N(yi given mu2, sigma2^2)  
## where mu=(mu1, mu2) and sigma^2=(sigma1^2, sigma2^2)  
## 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.  
  
# NormalMixtureGibbs.R with modifications  
  
########## BEGIN USER INPUT #################  
# Data options  
x <- as.matrix(Rainfall[,1])  
  
# Model options  
nComp <- 2 # Number of mixture components  
  
# Prior options  
alpha <- rep(1,nComp) # Dirichlet(alpha)  
# Obtained from Google, prior knowledge  
muPrior <- c(14.79, 17.6) # Prior mean of mu  
tau2Prior <- rep(100,nComp) # Prior std of mu  
sigma2\_0 <- rep(var(x),nComp) # s20 (best guess of sigma2)  
nu0 <- rep(1,nComp) # degrees of freedom for prior on sigma2  
  
# MCMC options  
nIter <- 1000 # Number of Gibbs sampling draws  
  
# Plotting options  
plotFit <- TRUE  
lineColors <- c("blue", "green", "magenta", 'yellow')  
sleepTime <- 0.01 # 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(Rainfall[,1])  
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))  
param\_matrix=matrix(0,4,nIter)  
rownames(param\_matrix)=c("Mu1", "Mu2", "Sigma1", "Sigma2")  
  
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))  
 }  
 param\_matrix[1,k]=mu[1]  
 param\_matrix[2,k]=mu[2]  
   
 # 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]))  
 }  
 param\_matrix[3,k]=sigma2[1]  
 param\_matrix[4,k]=sigma2[2]  
   
 # Update allocation  
 for (i in 1:nObs){  
 for (j in 1:nComp){  
 probObsInComp[j] <- pi[j]\*dnorm(x[i], mean = mu[j], sd = sqrt(sigma2[j]))  
 }  
 S[i,] <- t(rmultinom(1, size = 1 , prob = probObsInComp/sum(probObsInComp)))  
 }  
   
 # Printing the fitted density against data histogram  
 if (plotFit && (k%%1 ==0)){  
 effIterCount <- effIterCount + 1  
 hist(x, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax), main = paste("Iteration number",k),   
 ylim = ylim)  
 mixDens <- rep(0,length(xGrid))  
 components <- c()  
 for (j in 1:nComp){  
 compDens <- dnorm(xGrid,mu[j],sd = sqrt(sigma2[j]))  
 mixDens <- mixDens + pi[j]\*compDens  
 lines(xGrid, compDens, type = "l", lwd = 2, col = lineColors[j])  
 components[j] <- paste("Component ",j)  
 }  
 mixDensMean <- ((effIterCount-1)\*mixDensMean + mixDens)/effIterCount  
   
 lines(xGrid, mixDens, type = "l", lty = 2, lwd = 3, col = 'red')  
 legend("topright", box.lty = 1, legend = c("Data histogram",components, 'Mixture'),   
 col = c("black",lineColors[1:nComp], 'red'), lwd = 2)  
 }  
   
}  
  
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)  
plot(param\_matrix[1,200:ncol(param\_matrix)], type="l")  
plot(param\_matrix[2,200:ncol(param\_matrix)], type="l")  
plot(param\_matrix[3,200:ncol(param\_matrix)], type="l")  
plot(param\_matrix[4,200:ncol(param\_matrix)], type="l")  
  
## It seems like the sampler has converged towards a mixture distribution which resembles the histogram of   
## the data. The mode of the distribution is approximately at 20\*1/100 inches per day. The mixture density  
## function seems to resemble the reality more accurately than the normal density function. It seems reasonable  
## to apply a mixture distribution to this type of data since rain is not a constant occurance but can happen  
## on some days, and on some days not. When going through the iterations it is apparent that the the mixture  
## distribution converges quite quickly.   
  
## c) Plot the following densities in one figure: 1) a histogram or kernel density estimate of the data.   
## 2) Normal density of N(yi given mu and sigma^2) in a); 3) Mixture of normal density   
## p(yi given mu, sigma^2, pi) in b). Base your plots on the mean over all posterior draws.  
  
hist(x, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax), main = "Final density")  
lines(xGrid, mixDensMean, type = "l", lwd = 2, lty = 4, col = "red")  
lines(xGrid, dnorm(xGrid, mean = mean(posteriorMatrix[,1]), sd = mean(sqrt(posteriorMatrix[,2]))),   
 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)  
  
## As seen in the new plot, where the only difference is the blue line, the resembles to the previous plot is   
## obvious. The blue curve has not changed at all which is due to the fact that the mean of Gibbs sampled data,  
## when iterations go towards infinity, converges to the real mean of the data.

##Assignment 2

## Assignment 2: Consider the following Poisson regression model yi given Beta ~ Poisson(exp(xiT\*Beta)), i=1,...,n  
## where yi isthecountforthe ithobservationinthesampleand xi isthe 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 (x):   
  
## a) Obtain the maximum likelihood estimator of Beta 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?  
  
# Read data  
ebay = read.table("ebayNumberOfBidderData.dat", header=TRUE)  
data = ebay[, -2]  
  
# Create model  
model = glm(nBids~., family="poisson", data=data)  
print(model$coefficients)  
summary(model)  
  
## The covariates that are significant are VerifyID, Sealed, MajBlem, LogBook, MinBidShare.  
  
## b) Let's now do a Bayesian analysis of the Poisson regression. Let the prior be Beta~N(0,100\*(XTX)^(-1)) where X  
## is the n x 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 given y ~ N(Beta~, Jy(Beta~)^(-1)) where Beta~  
## is the posterior mode and Jy(Beta~) is the negative Hessian at the posterior mode. Beta~ and J can be obtained  
## by numerical optimization (optim.R) exactly like you already did for the logistic regression in Lab2.   
  
library(mvtnorm)  
# Defining constants  
X = as.matrix(ebay[,2:ncol(ebay)])  
Y = ebay[,1]  
nFeatures = dim(X)[2]  
covNames=names(ebay[,2:ncol(ebay)])  
  
# Constructing prior  
mu\_prior = rep(0,nFeatures)  
sigma\_prior = 100\*solve(t(X)%\*%X)   
  
# Defining function for returning the log posterior  
logPostPoisson = function(beta, Y, X, mu, sigma) {  
 n=length(Y)  
 XBeta=beta%\*%t(X)  
 # Defining loglikelihood  
 logLike <- sum(-log(factorial(Y))+XBeta\*Y-exp(XBeta))  
 # Defining prior  
 prior=dmvnorm(beta, mean=mu, sigma=sigma, log=TRUE)  
 # Adding loglikelihood and logprior together. Since it is log both of them are added instead of multiplied  
 return(logLike + prior)  
}  
  
# Defining initial values to be passed on to the optimizer  
set.seed(12345)  
initVals = rnorm(dim(X)[2])  
  
# Finding the optimized betavector  
optimResult = optim(initVals, logPostPoisson, Y=Y, X=X, mu=mu\_prior, sigma=sigma\_prior, method=c("BFGS"),  
 control=list(fnscale=-1), hessian=TRUE)  
  
# Defining the values of interest  
postMode = optimResult$par  
postCov = -solve(optimResult$hessian)  
names(postMode) = covNames  
approx\_PostStd = sqrt(diag(postCov))  
names(approx\_PostStd) = covNames  
print("The posterior mode is:")  
print(postMode)  
print("The approximated standard deviations are:")  
print(approx\_PostStd)  
  
## Through optimization we have obtained the optimal betavector as well as the hessian evaluated at the posterior  
## mode.   
  
## c) Now, let's simulate from the actual posterior of beta 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 theta. Let the proposal density be the multivariate normal density  
## mentioned in Lecture 8 (random walk Metropolis): Theta\_p given Theta(i-1) ~ N(Theta(i-1), c\*Cov) where   
## Cov = Jy(Beta~)^(-1) 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 beta in the Poisson regression for the eBay dataset.   
## Assess MCMC convergence by graphical methods.   
  
# Defining function for sampling through metropolishastings  
RVMSampler = function(previousVal, postCov, c, myFunction, ...) {  
 proposalVal=rmvnorm(1, mean=previousVal, sigma=c\*postCov)  
 alpha=min(1, exp(myFunction(proposalVal,...)-myFunction(previousVal, ...)))  
 u=runif(1)  
 if(u < alpha) {  
 return(proposalVal)  
 } else {  
 return(previousVal)  
 }  
}  
  
nDraws=5000  
beta\_matrix = matrix(0, nDraws, ncol(X))  
# Setting initial values of beta to same initVals as in the optimizer (taken randomly from normal distrib)  
beta\_matrix[1,]=initVals  
c=0.5  
set.seed(12345)  
  
for(i in 1:nDraws) {  
 if(i<nDraws) {  
 beta\_matrix[i+1,]=RVMSampler(beta\_matrix[i,], postCov, c, logPostPoisson, Y, X, mu\_prior, sigma\_prior)  
 }  
}  
  
iter=seq(1,nDraws,1)  
par(mfrow=c(3,3))  
for (i in 1:9) {  
 plot(iter, beta\_matrix[,i], type="l", main=paste("Convergence plot for covariate", covNames[i]),  
 ylab=covNames[i])  
}  
par(mfrow=c(1,1), new=FALSE)  
  
# Calculating distinct rows and dividing by total rows to get average acceptance probability  
avg\_alpha=dim(beta\_matrix[!duplicated(beta\_matrix),])[1]/dim(beta\_matrix)[1]  
  
## As seen in the convergence plots the covariates oscillate around the same value which was found in the previous  
## problem where the optimal beta values were found through optimization. Since the variable c should be chosen   
## in a way to acquire an average acceptance rate of approximately 25-30%, the average acceptance rate were   
## calculated to approximately 33 % which is deemed to be sufficiently satisfying.   
  
## 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? Use vector x=c(1,1,1,1,0,0,0,1,0.5)  
  
obs\_X=c(1,1,1,1,0,0,0,1,0.5)  
# Removing first 1000 rows since they are before the start of the convergence  
approx\_post\_beta=beta\_matrix[1001:nrow(beta\_matrix),]  
mean\_vector=exp(approx\_post\_beta%\*%obs\_X)  
set.seed(12345)  
pred\_distrib\_bidder=rpois(10000, mean\_vector)  
barplot(table(pred\_distrib\_bidder),  
 main="Histogram of the predictive distribution of no. of bidders", xlab="No. of bidders")  
# Calculating the probability of no bidders with the given characteristics  
prob\_noBidders=sum(pred\_distrib\_bidder==0)/length(pred\_distrib\_bidder)  
print(prob\_noBidders)  
  
## As seen in the predictive distribution the majority of cases given the specified characteristics, will result in  
## either 0 or 1 bidder with the probability decreasing for additional bidders. The calculated probability for  
## no bidder is 0.3581.

#Lab 4

##Assignment 1

## Assignment 1:  
## a) Write a function in R that simulate data from the AR(1)-process: xt=mu+phi(x(t-1)-mu) + epsilon(t),   
## epsilon(t)~N(0,sigma^2), for given values of mu, phi, and sigma^2. Start the process at x1=mu and then simulate  
## values for xt for t=2,3,...,T and return the vector x1:T containing all time points. Use mu=10, sigma^2=2 and  
## T=200 and look at different realizations (simulation) of x1:T for values of phi between -1 and 1 (this is the  
## interval of phi where the AR-process is stable). Include a plot of at least one realization in the report. What  
## effect does the value of phi have on x1:t  
  
#install.packages("rstan")  
mu=10  
sigma\_sq=2  
T=200  
x\_init=mu  
phi\_vector=seq(-0.9,0.9,0.1)  
results\_matrix=matrix(0,200,length(phi\_vector))  
results\_matrix[1,]=x\_init  
counter=1  
set.seed(12345)  
  
AR\_process\_function=function(mu, sigma\_sq, T, phi) {  
 x\_init=mu  
 result=rep(0,T)  
 result[1]=x\_init  
 for (i in 2:T) {  
 epsilon=rnorm(1,0,sqrt(sigma\_sq))  
 result[i]=mu+phi\*(result[i-1]-mu)+epsilon  
 }  
 return(result)  
}  
  
results\_matrix=matrix(0,T,length(phi\_vector))  
counter=1  
for (phi in phi\_vector) {  
 results\_matrix[,counter]=AR\_process\_function(mu,sigma\_sq,T,phi)  
 counter=counter+1  
}  
iter=seq(1,200,1)  
counter=1  
for (i in 1:length(phi\_vector)) {  
 if (counter %% 6 == 0) {  
 plot(iter, results\_matrix[,i], main="Plot of realization of AR-process",   
 sub=paste("Phi =", phi\_vector[i]),  
 xlab="Iteration", ylab="Value", type="l", col="grey")  
 }  
 counter=counter+1  
}  
  
## With phi-values below zero the process will oscillate faster but with phi-values above zero the process will  
## be more correlated. The correlation between the different iterations increases as the phi-value becomes larger.   
## This causes the oscillation to slow down and the process to move more slowly.   
  
## b) Use your function from a) to simulate two AR(1)-processes, x1:T with phi=0.3 and y1:T with phi=0.95. Now,  
## treat the values of mu, phi and sigma^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  
## parametizations 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  
## mu and phi. Comments?  
  
library(rstan)  
  
x=rep(0,T)  
y=rep(0,T)  
set.seed(12345)  
x=AR\_process\_function(mu, sigma\_sq, T, 0.3)  
set.seed(12345)  
y=AR\_process\_function(mu, sigma\_sq, T, 0.95)  
  
StanModel= '  
data {  
 int<lower=0> N;  
 vector[N] y;  
}  
  
parameters {  
 real mu;  
 real phi;  
 real<lower=0> sigma;  
}  
model {  
 for (n in 2:N)  
 y[n] ~ normal(mu + phi \* (y[n-1]-mu), sigma);  
}  
'  
  
data\_x=list(N=T, y=x)  
data\_y=list(N=T, y=y)  
fit\_x=stan(model\_code=StanModel, data=data\_x)  
fit\_y=stan(model\_code=StanModel, data=data\_y)  
postDraws\_x <- extract(fit\_x)  
postDraws\_y <- extract(fit\_y)  
print(fit\_x)  
print(fit\_y)  
  
# Do traceplots of the first chain  
plot(postDraws\_x$mu[1000:2000], postDraws\_x$phi[1000:2000],ylab="phi", xlab="mu", main="Traceplot")  
  
# Do traceplots of the first chain  
plot(postDraws\_y$mu[1000:2000],postDraws\_y$phi[1000:2000],ylab="mu", xlab="mu",main="Traceplot")  
  
## The posterior mean, number of effective samples as well as 95 % credible interval are shown above for both of the  
## simulated AR(1)-processes. It is possible to estimate the true values of the parameters for the sample which  
## used a phi=0.3 when obtaining the dataset used in the simulation. However, it is not as obvious to estimate   
## the parameters' true values for the second sample where phi=0.95 were used to obtain the dataset used in this  
## particular simulation. The credible intervals for the parameters in this simulation are very wide and it is  
## difficult to predict with certainty the true vale of the parameter. This might be due to the higher correlation  
## between the lags caused by the higher value of phi.   
  
## The convergence of the samplers are different. For the first sample which used phi=0.3, the convergence is  
## evident whilst for the second sample the posterior distribution is not obvious. This correlates with the fact  
## the credible intervals for the parameters on the second sample were very wide. What we can see from the   
## posterior distribution obtained by the second sampler is that for lower values of phi the distribution centers  
## around a value between 10 and 20. This is a behaviour similar to what is shown in the posterior for the first  
## sampler, where phi was set to 0.3 initially, since this distribution was much tighter around the value of 10  
## for mu.  
  
## 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 ct at each time point follows an   
## independent Poisson distribution when conditioned on a latend AR(1)-process xt, that is  
## ct given xt ~ Poisson(exp(xt)), where xt 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(xt) over time.   
## [Hint: Should xt be seen as data or parameters]  
  
campy=read.table("campy.dat", header=TRUE)  
library(rstan)  
  
StanModel\_Pois = '  
data {  
 int<lower=0> T;  
 int c[T];  
}  
  
parameters {  
 real mu;  
 real phi;  
 real<lower=0> sigma;  
 vector[T] x;  
}  
  
model {  
 // Prior  
 phi ~ uniform(-1,1);  
 for (n in 2:T)  
 x[n] ~ normal(mu + phi \* (x[n-1]-mu), sigma);  
   
  
 // Model/likelihood  
 for (n in 1:T)  
 c[n] ~ poisson(exp(x[n]));  
   
}  
  
generated quantities {  
 vector[T] post\_mean;  
 post\_mean = exp(x);  
}  
'  
  
data=list(T=dim(campy)[1], c=campy$c)  
fit\_pois=stan(model\_code=StanModel\_Pois, data=data)  
print(fit\_pois)  
pois\_mean\_list=fit\_pois@.MISC$summary$msd  
post\_mean=pois\_mean\_list[grep("post\_mean", rownames(pois\_mean\_list)),]  
  
plot(campy$c, col="blue", ylab="No. of infected", xlab="Time")  
points(post\_mean[,1], col="black", type="l")  
  
quantiles=fit\_pois@.MISC$summary$quan  
quantiles\_post\_mean=quantiles[grep("post\_mean", rownames(quantiles)),]  
cred\_interval\_post\_mean=matrix(0,dim(quantiles\_post\_mean)[1], 2)  
cred\_interval\_post\_mean[,1]=quantiles\_post\_mean[,1]  
cred\_interval\_post\_mean[,2]=quantiles\_post\_mean[,ncol(quantiles\_post\_mean)]  
  
lines(cred\_interval\_post\_mean[,1], col="gray", lty=1)  
lines(cred\_interval\_post\_mean[,2], col="gray", lty=1)  
title(main="Plot of data vs approximated posterior")  
legend("topleft", box.lty= 1, pch=c(1,NaN,NaN), legend=c("Data", "Posterior mean", "95 % cred. interval"),  
 col=c("blue", "black", "gray"), lwd=c(NaN,1,1), lty=c(NaN, 1, 1))  
  
## As seen in the plot above the posterior mean follows the data accurately. Almost all of the datapoints are  
## inside the credible intervals which aren't that wide which indicates that the approximated posterior  
## resembles the reality shown by the data well.   
  
## d) Now, assume that we have a prior belief that the true underlying intensity theta\_t varies more smoothly than  
## the data suggests. Change the prior for sigma\_sq so that it becomes informative about that the AR(1)-process   
## increments epsilon\_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 theta\_t changed?  
  
StanModel\_Pois\_Prior = '  
data {  
 int<lower=0> T;  
 int c[T];  
}  
  
parameters {  
 real mu;  
 real phi;  
 real<lower=0> sigma;  
 vector[T] x;  
}  
  
model {  
 // Prior  
 phi ~ uniform(-1,1);  
 sigma ~ scaled\_inv\_chi\_square(140, 0.15);  
 for (n in 2:T)  
 x[n] ~ normal(mu + phi \* (x[n-1]-mu), sigma);  
   
  
 // Model/likelihood  
 for (n in 1:T)  
 c[n] ~ poisson(exp(x[n]));  
   
}  
  
generated quantities {  
 vector[T] post\_mean;  
 post\_mean = exp(x);  
}  
'  
fit\_pois\_prior=stan(model\_code=StanModel\_Pois\_Prior, data=data)  
print(fit\_pois\_prior)  
pois\_mean\_list\_prior=fit\_pois\_prior@.MISC$summary$msd  
post\_mean\_prior=pois\_mean\_list\_prior[grep("post\_mean", rownames(pois\_mean\_list)),]  
  
plot(campy$c, col="blue", ylab="No. of infected", xlab="Time")  
points(post\_mean\_prior[,1], col="black", type="l")  
  
quantiles\_prior=fit\_pois\_prior@.MISC$summary$quan  
quantiles\_post\_mean\_prior=quantiles\_prior[grep("post\_mean", rownames(quantiles)),]  
cred\_interval\_post\_mean\_prior=matrix(0,dim(quantiles\_post\_mean)[1], 2)  
cred\_interval\_post\_mean\_prior[,1]=quantiles\_post\_mean\_prior[,1]  
cred\_interval\_post\_mean\_prior[,2]=quantiles\_post\_mean\_prior[,ncol(quantiles\_post\_mean)]  
  
lines(cred\_interval\_post\_mean\_prior[,1], col="gray", lty=1)  
lines(cred\_interval\_post\_mean\_prior[,2], col="gray", lty=1)  
title(main="Plot of data vs approximated posterior")  
legend("topleft", box.lty= 1, pch=c(1,NaN,NaN), legend=c("Data", "Posterior mean", "95 % cred. interval"),  
 col=c("blue", "black", "gray"), lwd=c(NaN,1,1), lty=c(NaN, 1, 1))  
  
## Now when we have specified a small prior for sigma it is noteable in the new plot that the posterior mean  
## varies less and moves more smoothly. The consequence of this is that more datapoints lie outside of the   
## credible interval which suggests that the approximated posterior does not resemble the reality described by  
## the data as accurately as before. However, by doing this one can avoid overfitting when the model is applied  
## to a new dataset.