Contents

Labs

Lab 1 – One param models, simulation, direct numerical evaluation

Assignment 1 – Bernoulli posterior and log odds posterior with convergence

```r
## Assignment 1: Let y1,...,yn be Bernoulli distributed with parameter the
ta. Assume that you have obtained a sample
## with s=5 successes in n=20 trials. Assume a Beta(alpha0, beta0) prior f
or 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 t
he 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 respec
```

```
t 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 converge
s 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 standar
d deviation converges towards its true
## value of approx 0.29 and 0.09 respectively as the number of randow draw
s grows large.

## b) Use simulation (nDraws=10000) to compute the posterior probability P
r(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 probabi
lity 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 – Lognormal distribution, posterior vs real distrib, ginicoefficient, cred/HPD intervals

```
## Assignment 2: Assume that you have asked 10 randomly selected persons a
bout 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 ~ l
og 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 pr
oportional to 1/sigma^2. The posterior
## for sigma^2 is the Inv - chitwo distribution with X(n, thao^2) distribu
```

```r
tion, 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) poster
ior 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 = simulat
ed 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 d
istribution 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 co
efficient")

## As seen in the plot the gini coefficient is centered at around 0.2 whic
h means a rather inequal distribution.
```

```r
## 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 – von Mises distribution with Bessel, plot posterior (normalization), finding posterior mode

```r
## 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 th
```

```r
e 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 arou
nd 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 ove
r 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
```

```r
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 ret
rieve 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 – Polynomial regression, and classification with logistic regression

Assignment 1 – Setting proper priors, simulating from joint posterior, marginal posterior, scatter plot overlaid with posterior median and credible intervals, posterior mode prediction

```r
## Assignment 1: The dataset TempLinkoping.txt contains daily average tamp
eratures (in Celcius degrees) at
## Malmslatt, Linkoping over the course of the year 2018. The response var
iable 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, om
ega0, 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 prio
r of all parameters and for every draw
## compute the regression curve. This gives a collection of regression cur
ves, one for each draw from the prior.
## Do the collection of curves look reasonable? If not, change the prior h
yperparameters until the collection
## of prior regression curves agrees with your prior beliefs about the reg
ression 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
```

```r
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 plot
s
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 mod
els", xlab="Time", ylab="Temp")

## The collection of curves look reasonable and in line with our prior bel
iefs. The temperature rises during the
## summer months and stays low in the beginning and the end of the year re
spectively.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 pro
duce 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 lowe
r 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$time), 2)
# Retrieves the median of the response values as well as obtaining the upp
er and lower bound of credInterval
for (i in 1:length(temp$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 tempe
rature (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 c
an find a simple formula for xtilde
## given beta0, beta1 and beta2]

# Function for calculating the time-value which yields the maximum respons
e (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 temperatur
e 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 tempera
tures simulated from xtilde",
     xlab="Temperature")

## As seen in the histogram the derived highest temperature from the simul
ated models is mostly present in late
## june which seems reasonable if applying to Malmslatt in Sweden where th
e 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 Lam
bda*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 – Normal approx. through optimization, comparing with maximum likelihood, simulating from predictive distribution, multiple trials (binomial)

```
## 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
```

```r
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 th
em 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, sigm
a=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 la
rge in the negative direction it is
## reasonable to conclude that the feature NSmallChild affects the respons
e variable farily much towards the
## response 0 which means that the woman doesn't work. This seems like a r
easonable 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 si
milar 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 ch
ildren (3 and 9 years old), 8 years
## of education, 10 years of experience. and a husband with an income of 1
0. [Hints: The R package mvtnorm will
## again be handy. Remember my discussion on how Bayesian prediction can b
e 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 probabil
ities")

## As seen in the plots the calculated probabilities of the woman in quest
ion working is fairly low. The highest
## density is seen in the range between 0.2 and 0.3 approximately. This al
so makes sense if applied to a real
## situation. A woman with a small child is likely to stay at home with th
e child, i.e. not working. If the
## classification of the response variable results in "working" if the pre
dicted probabilities is above 0.5 and
## "not working" otherwise, it is clear from the distribution that the cla
ssification 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 mad
e 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 pre
dicted probabilities with the
## highest density found at 2 women. This result seems reasonable since wh
en the number of draws taken from
## the binomial distribution goes towards infinity the shape of the corres
ponding distribution will resemble
## the shape of the distribution for the probability p in the Bernoulli ca
se, more and more.
```

Lab 3 – MCMC using Gibbs sampling and Metropolis Hastings

Assignment 3 – Gibbs sampler with normal data, evaluate convergence, mixture of normals, plot of histogram kernel density estimate & normal density & mixture of normal

```
## 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 z
ero 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 distribu
tied,
## 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 pos
terior 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])
```

```r
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], pos
teriorMatrix[i,1], n)/drawX
  }
}

# The posterior coverage
plot(posteriorMatrix[1001:nrow(posteriorMatrix),1], posteriorMatrix[1001:n
row(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")
```

```r
## b) Let us now instead assume that the daily precipitation {y1,...,yn} f
ollow 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 NormalMixtureGibb
s.R (available under Lecture 7 on the
## course page) to analyze the daily precipitation data. Set the prior hyp
erparameters 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 s
um of the elements in that column.
  return(piDraws)
}
```

```r
# Simple function that converts between two different representations of t
he 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-nC
omp 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 re
presentations 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(probObsInC
omp)))
  }

  # 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 = "Fi
nal 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 densi
ty","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 inc
hes 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 i
t 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 no
rmal 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 = "Fi
nal 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 densi
ty","Normal density"),
       col=c("black","red","blue"), lwd = 2)

## As seen in the new plot, where the only difference is the blue line, th
e 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 – Linear regression with Poisson, ML-estimator, optim with Poisson, simulating through RWM Hastings, myFunction(…), simulate from predictive distribution

```
## 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-dimen
sional vector with covariate observations
## for the ith observation. Use the data set eBayNumberOfBidderData.dat. T
his 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 regre
ssion model for the eBay data
## [Hint: glm.R, don't forget that glm() adds its own intercept so don't i
nput the covariate Const]. Which
## covariates are significant?
```

```r
# 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, LogB
ook, 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 c
alled 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 post
erior mode. Beta~ and J can be obtained
## by numerical optimization (optim.R) exactly like you already did for th
e 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 th
em 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
```

```r
=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 Metr
opolis algorithm and compare with the
## approximate results in b). Program a general function that uses the Met
ropolis 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 densit
y 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 M
etropolis 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 des
cribes how to do this in R. Now, use your
## new Metropolis function to sample from the posterior of beta in the Poi
sson 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)
  }
}
```

```r
nDraws=5000
beta_matrix = matrix(0, nDraws, ncol(X))
# Setting initial values of beta to same initVals as in the optimizer (tak
en 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, logPostPoiss
on, 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 c
ovariate", covNames[i]),
       ylab=covNames[i])
}
par(mfrow=c(1,1), new=FALSE)

# Calculating distinct rows and dividing by total rows to get average acce
ptance 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 sa
me 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 sat
isfying.

## d) Use the MCMC draws from c) to simulate from the predictive distribut
ion of the number of bidders in a new
## auction with the characteristics below. Plot the predictive distributio
n. 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 converge
nce
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",
```

```r
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 bid
ders. The calculated probability for
## no bidder is 0.3581.
```

Lab 4 – HMC with Stan

Assignment 1 – AR(1)-process, stanmodel with AR(1)-process, Poisson conditioned on AR(1)-process in stan, plot with data & posterior mean & credible intervals over time

```r
## 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. Star
t 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 value
s of phi between -1 and 1 (this is the
## interval of phi where the AR-process is stable). Include a plot of at l
east 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 p
hi-values above zero the process will
## be more correlated. The correlation between the different iterations in
creases as the phi-value becomes larger.
## This causes the oscillation to slow down and the process to move more s
lowly.

## 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 us
ing 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 sam
plers 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", xla
b="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 % credibl
e interval are shown above for both of the
## simulated AR(1)-processes. It is possible to estimate the true values o
f the parameters for the sample which
## used a phi=0.3 when obtaining the dataset used in the simulation. Howev
er, it is not as obvious to estimate
## the parameters' true values for the second sample where phi=0.95 were u
sed to obtain the dataset used in this
## particular simulation. The credible intervals for the parameters in thi
s simulation are very wide and it is
## difficult to predict with certainty the true vale of the parameter. Thi
s 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 whi
ch 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 ver
y 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 i
s shown in the posterior for the first
## sampler, where phi was set to 0.3 initially, since this distribution wa
s much tighter around the value of 10
## for mu.
```

```
## c) The data campy.dat contain the number of cases of campylobacter infe
ctions 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 infection
s ct at each time point follows an
## independent Poisson distribution when conditioned on a latend AR(1)-pro
cess 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 t
ime.
## [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)
```

```r
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", "Posterio
r 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 accuratel
y. Almost all of the datapoints are
## inside the credible intervals which aren't that wide which indicates th
at the approximated posterior
## resembles the reality shown by the data well.

## d) Now, assume that we have a prior belief that the true underlying int
ensity theta_t varies more smoothly than
## the data suggests. Change the prior for sigma_sq so that it becomes inf
ormative 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(quant
iles)),]
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(quantile
s_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", "Posterio
r 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 th
e new plot that the posterior mean
## varies less and moves more smoothly. The consequence of this is that mo
re datapoints lie outside of the
## credible interval which suggests that the approximated posterior does n
ot 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.
```

Exams

2017-05-30

Assignment 1 – Plot posterior (rice function), normal approx. through optim, simulation for new obs

```
## a) Plot the posterior distribution of theta

riceData <- c(1.556, 1.861, 3.135, 1.311, 1.877, 0.622, 3.219, 0.768, 2.35
8, 2.056)

# Random number generator for the Rice distribution
rRice <-function(n = 1, theta = 1, psi = 1){
  x <- rnorm(n = n, mean = 0, sd = sqrt(psi))
  y <- rnorm(n = n, mean = theta, sd = sqrt(psi))
  return(sqrt(x^2+y^2))
}

# Function for calculating the log posterior distrib with theta prior set
to 1
logPosterior = function(data, theta, psi) {
  bessel_factor=1
  for (i in data) {
    bessel_factor=bessel_factor*besselI(i*theta/psi, nu=0)
  }
  post=-log(psi)-1/(2*psi)*sum(data^2+theta^2)+log(bessel_factor)
  return(post+0) # If prior is assumed to be constant we set the prior to
1 which in log scale yields 0
}

gridWidth=0.01
theta_grid=seq(0,3,gridWidth)
posterior_distrib_log=sapply(theta_grid, logPosterior, data=riceData, psi=
1)
posterior_distrib_norm=1/gridWidth*exp(posterior_distrib_log)/sum(exp(post
erior_distrib_log))
sum(posterior_distrib_norm)
plot(theta_grid, posterior_distrib_norm, xlab=expression(theta), ylab="Den
sity", main="Posterior density of theta",
     type="l", lwd=2)

## b) Use numerical optimization to obtain a normal approx. of the posteri
or distrib of theta. Overlay curve
## from a) with the approximated normal distribution

# Defining initial values to be passed on to the optimizer
set.seed(12345)
initVal = rnorm(1, mean=0, sd=1)

# Finding the optimized betavector
optimResult = optim(initVal, logPosterior, data=riceData, psi=1, method=c(
"L-BFGS-B"),
                    control=list(fnscale=-1), lower=0, hessian=TRUE)
```

```r
# Defining the values of interest
postMode = optimResult$par
postCov = as.numeric(-solve(optimResult$hessian))
print("The posterior mode is:")
print(postMode)
print("The approximated standard deviation is:")
print(postCov)
lines(theta_grid, dnorm(theta_grid, mean=postMode, sd=sqrt(postCov)), col="red", lwd=2)
legend(x = 1.8, y = 1, legend = c("True posterior", "Approximate posterior"),
       col = c("black","red"), lty = c(1,1), lwd = c(2,2), cex = 0.8)

## Answer: Not perfect approx but fairly good.

## c) Simulate distrib for new observation using normal approx in b)

nDraws=5000
set.seed(12345)
theta=rnorm(nDraws, mean=postMode, sd=sqrt(postCov))
pred_distrib=c()
for (i in theta) {
  pred_distrib=c(pred_distrib, rRice(theta=i))
}

hist(pred_distrib, breaks=100, xlab="Index", main="Predictive density of new obs")
```

Assignment 2 – Model posterior data, plot posterior and compare with data, Gibbs for mixture of models, graphical methods for evaluating

```r
## a) Model posterior data with prior Gamma and likelihood Poisson, plot the posterior

# We know that posterior mapping with gamma prior and poisson likelihood is gamma distributed

sumBids=sum(bids)
n=length(bids)
alpha=1
beta=1
posterior_theta=dgamma(seq(3,4,0.001), alpha+sumBids, beta+n)
plot(seq(3,4,0.001), posterior_theta, type="l", lwd=2)

# b) Investigate through graphical methods if Poisson model describes data well

xGrid=seq(min(bids), max(bids))
data_norm=bidsCounts/sum(bidsCounts)
nDraws=5000
thetaDraws=rgamma(nDraws, alpha+sumBids, beta+n)
poissonDensity=rep(0, length(xGrid))
for (i in thetaDraws) {
  poissonDensity=poissonDensity+dpois(xGrid, lambda=i)
```

```
}

avgPoissonDensity=poissonDensity/nDraws
plot(xGrid, data_norm, xlab="No. of bids", ylab="Density", main="Fitted mo
dels", type="o", cex=0.8,
     ylim=c(0,0.25), lwd=2)
lines(xGrid, avgPoissonDensity, col="red", lwd=2, type="o")
legend(x=7, y=0.2, col=c("black", "red"), legend=c("Data", "Poisson mean d
ensity"), lty=c(1,1),
       lwd=c(2,2), pch=c("o", "o"))

## Terrible fit which the plot shows

## c) Use GibbsMixPois.R. Esimate the mixture of Poissons both with K=2 an
d K=3. nIter=5000.

GibbsMixPois <- function(x, nComp, alpha, alphaGamma, betaGamma, xGrid, nI
ter){

  # Gibbs sampling for a mixture of Poissons
  # Author: Mattias Villani, IDA, Linkoping University. http://mattiasvill
ani.com
  #
  # INPUTS:
  #   x - vector with data observations (counts)
  #   nComp - Number of mixture components to be fitted
  #   alpha - The prior on the mixture component weights is w ~ Dir(alpha,
alpha,..., alpha)
  #   alphaGamma and betaGamma -
  #               The prior on the mean (theta) of the Poisson mixture comp
onents is
  #               theta ~ Gamma(alphaGamma, betaGamma) [rate parametrizatio
n of the Gamma dist]
  #   xGrid - the grid of data values over which the mixture is evaluated
and plotted
  #   nIter - Number of Gibbs iterations
  #
  # OUTPUTS:
  #   results$wSample     - Gibbs sample of mixture component weights. nIt
er-by-nComp matrix
  #   results$thetaSample - Gibbs sample of mixture component means.   nIt
er-by-nComp matrix
  #   results$mixDensMean - Posterior mean of the estimated mixture densit
y over xGrid.


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

```
Draws by the sum of the elements in that column.
    return(thetaDraws)
  }

  # 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 values for the Gibbs sampling
  nObs <- length(x)
  S <- t(rmultinom(nObs, size = 1 , prob = rep(1/nComp,nComp))) # nObs-by-
nComp matrix with component allocations.
  theta <- rep(mean(x), nComp) # Each component is initialized at the mean
of the data

  # Setting up the grid where the mixture density is evaluated.
  mixDensMean <- rep(0,length(xGrid))
  effIterCount <- 0

  # Setting up matrices to store the draws
  wSample <- matrix(0, nIter, nComp)
  thetaSample <- matrix(0, nIter, nComp)
  probObsInComp <- rep(NA, nComp)

  # Setting up the priors - the same prior for all components
  alpha <- rep(alpha, nComp)
  alphaGamma <- rep(alphaGamma, nComp)
  betaGamma <- rep(betaGamma, nComp)

  # HERE STARTS THE ACTUAL GIBBS SAMPLING

  for (k in 1:nIter){
    message(paste('Iteration number:',k))
    alloc <- S2alloc(S) # Function that converts between different represe
ntations of the group allocations
    nAlloc <- colSums(S)

    # Step 1 - Update components probabilities
    w <- rDirichlet(alpha + nAlloc)
    wSample[k,] <- w

    # Step 2 - Update theta's in Poisson components
    for (j in 1:nComp){
      theta[j] <- rgamma(1, shape = alphaGamma + sum(x[alloc == j]), rate
= betaGamma + nAlloc[j])
    }
    thetaSample[k,] <- theta
```

```
    # Step 3 - Update allocation
    for (i in 1:nObs){
      for (j in 1:nComp){
        probObsInComp[j] <- w[j]*dpois(x[i], lambda = theta[j])
      }
      S[i,] <- t(rmultinom(1, size = 1 , prob = probObsInComp/sum(probObsI
nComp)))
    }

    # Computing the mixture density at the current parameters, and averagi
ng that over draws.
    effIterCount <- effIterCount + 1
    mixDens <- rep(0,length(xGrid))
    for (j in 1:nComp){
      compDens <- dpois(xGrid, lambda = theta[j])
      mixDens <- mixDens + w[j]*compDens
    }
    mixDensMean <- ((effIterCount-1)*mixDensMean + mixDens)/effIterCount
  }
  return(results = list(wSample = wSample, thetaSample = thetaSample, mixD
ensMean = mixDensMean))
}

result_comp2=GibbsMixPois(bids, nComp=2, alpha=1, alphaGamma = alpha, beta
Gamma = beta,
                     xGrid=xGrid, nIter=500)
result_comp3=GibbsMixPois(bids, nComp=3, alpha=1, alphaGamma = alpha, beta
Gamma = beta,
                          xGrid=xGrid, nIter=500)

## c) Use graphical methods to investigate if mixture of poissons fits dat
a well. Is K=2 enough or should we
## use K=3?

plot(xGrid, data_norm, xlab="No. of bids", ylab="Density", main="Fitted mo
dels", type="o",
     ylim=c(0,0.25), lwd=2)
lines(xGrid, result_comp2$mixDensMean, col="red", lwd=2, type="o")
lines(xGrid, result_comp3$mixDensMean, col="gray", lwd=2, type="o")
legend(x=7, y=0.2, col=c("black", "red", "gray"),
       legend=c("Data", "Mixture density with 2 components", "Mixture dens
ity with 3 components"),
       lty=c(1,1,1), lwd=c(2,2, 2), pch=c("o", "o", "o"), cex=1)

## Good enough with 2 components in the mixture density
```

Assignment 3 – Linear regression (cars), marginal distributions, interpretation of cred. Interval, predictive simulation

```
################################
########## Problem 3 ##########
################################
```

```r
# Reading the cars data from file
load("cars.RData")

library(mvtnorm)

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

BayesLinReg <- function(y, X, mu_0, Omega_0, v_0, sigma2_0, nIter){
  # Direct sampling from a Gaussian linear regression with conjugate prior
:
  #
  # beta | sigma2 ~ N(mu_0, sigma2*inv(Omega_0))
  # sigma2 ~ Inv-Chi2(v_0,sigma2_0)
  #
  # Author: Mattias Villani, IDA, Linkoping University. http://mattiasvill
ani.com
  #
  # INPUTS:
  #   y - n-by-1 vector with response data observations
  #   X - n-by-nCovs matrix with covariates, first column should be ones i
f you want an intercept.
  #   mu_0 - prior mean for beta
  #   Omega_0  - prior precision matrix for beta
  #   v_0      - degrees of freedom in the prior for sigma2
  #   sigma2_0 - location ("best guess") in the prior for sigma2
  #   nIter - Number of samples from the posterior (iterations)
  #
  # OUTPUTS:
  #   results$betaSample     - Posterior sample of beta.     nIter-by-nCov
s matrix
  #   results$sigma2Sample   - Posterior sample of sigma2.   nIter-by-1 ve
ctor

  # Compute posterior hyperparameters
  n = length(y) # Number of observations
  nCovs = dim(X)[2] # Number of covariates
  XX = t(X)%*%X
  betaHat <- solve(XX,t(X)%*%y)
  Omega_n = XX + Omega_0
  mu_n = solve(Omega_n,XX%*%betaHat+Omega_0%*%mu_0)
  v_n = v_0 + n
  sigma2_n = as.numeric((v_0*sigma2_0 + ( t(y)%*%y + t(mu_0)%*%Omega_0%*%m
u_0 - t(mu_n)%*%Omega_n%*%mu_n))/v_n)
  invOmega_n = solve(Omega_n)

  # The actual sampling
  sigma2Sample = rep(NA, nIter)
  betaSample = matrix(NA, nIter, nCovs)
  for (i in 1:nIter){
```

```r
    # Simulate from p(sigma2 | y, X)
    sigma2 = rScaledInvChi2(n=1, df = v_n, scale = sigma2_n)
    sigma2Sample[i] = sigma2

    # Simulate from p(beta | sigma2, y, X)
    beta_ = rmvnorm(n=1, mean = mu_n, sigma = sigma2*invOmega_n)
    betaSample[i,] = beta_

  }
  return(results = list(sigma2Sample = sigma2Sample, betaSample=betaSample
))
}

## a) Linear regression problem with given dataset. Use Mattias function t
o derive joint posterior.
## i) Plot marginal distributions of each param
## ii) Compute point estimates for each regression coefficient assuming lo
ss function
## iii) Construct 95 % equal tail probability intervals for each parameter
and interpret them


y=cars$mpg
x=as.matrix(cars[2:ncol(cars)])
mu_0=c(0,0,0,0)
omega_0=0.01*diag(x=4)
nu_0=1
sigma_sq_0=36
jointPostDistrib=BayesLinReg(y, x, mu_0, omega_0, nu_0, sigma_sq_0, 1000)
hist(jointPostDistrib$sigma2Sample, breaks=10, main=paste("Marginal distri
bution of", expression(sigma^2)),
     xlab=expression(sigma^2))
par(mfrow=c(2,2))
for(i in 1:4) {
  hist(jointPostDistrib$betaSample[,i], breaks=10, main=paste("Marginal di
stribution of ", expression(beta), i,
                                                              sep=""), xla
b=paste(expression(beta),i, sep=""))
}
title("Marginal distributions of the different betavalues", line=-1, outer
=TRUE)
par(mfrow=c(1,1))

# Linear loss function is posterior median
median(jointPostDistrib$sigma2Sample)
median(jointPostDistrib$betaSample[,1])
median(jointPostDistrib$betaSample[,2])
median(jointPostDistrib$betaSample[,3])
median(jointPostDistrib$betaSample[,4])

# Prediction intervals for each param
quantile(jointPostDistrib$sigma2Sample, c(0.025, 0.975))
quantile(jointPostDistrib$betaSample[,1], c(0.025, 0.975))
quantile(jointPostDistrib$betaSample[,2], c(0.025, 0.975))
```

```r
quantile(jointPostDistrib$betaSample[,3], c(0.025, 0.975))
quantile(jointPostDistrib$betaSample[,4], c(0.025, 0.975))

## Answer: Interpretation of the credible interval for weight [-4.759964,
-1.531457]. A one unit increase of weight
## lowers the amount of miles per gallon between -4.759964 and -1.531457 w
ith 95 % posterior probability.

## b) Investigate if effect on mpg is different in cars with six cylinders
compared to cars with 8 cylinders

hist(jointPostDistrib$betaSample[,4]-jointPostDistrib$betaSample[,3], 50)
quantile(jointPostDistrib$betaSample[,4]-jointPostDistrib$betaSample[,3],
c(0.025, 0.975))

## Answer: Since 0 is present in interval we can not say that there is a d
ifference between 8 and 6 cylinders
## with 95 % posterior probability.

## c) Compute by simulation predictive distrib for a new car 4 cylinders a
nd weight=3.5

new_x=c(1,3.5,0,0)
pred_y=rep(0,nIter)
for (i in 1:nIter) {
  pred_y[i]=sum(new_x*jointPostDistrib$betaSample[i,])+rnorm(1,sd=sqrt(joi
ntPostDistrib$sigma2Sample[i]))
}
hist(pred_y, breaks=40, freq=FALSE)
```

Assignment 4 – Maximizing posterior expected utility

```r
## Maximizing posterior expected utility.

post_dens = function(x) {
  return(gamma(6+x)/gamma(13+x))
}

barplot(post_dens(seq(0,10,1)), type="l")

# x6=10 seems to yield a low enough probability to be an upper bound for s
um

posterior_prob=post_dens(seq(0,10))
posterior_prob=posterior_prob/sum(posterior_prob)
exp_util=c()
for (k in 0:10) {
  exp_util=c(exp_util,(2^k-3))
}

exp_post_dens=sum(posterior_prob*exp_util)
print(exp_post_dens)
```

2017-08-16

Assignment 1 – Plot posterior density (Cauchy), normal approx. through optimization, marginal posterior

```
## a) Plot posterior density of theta, with normal prior and cauchy distri
b as likelihood

# Reading the data vector yVect from file
load(file = 'CauchyData.RData')
cauchydata=yVect

dCauchy <- function(x, theta = 0, gamma = 1){
  return(dens = (1/(pi*gamma))*(1/(1+((x-theta)/gamma)^2)))
}

dlognormal <- function(x, mu, sigma2){
  return(dens = (1/(sqrt(2*pi*sigma2)*x))*exp((-1/(2*sigma2))*(log(x)-mu)^
2))
}

logPrior_theta = function(theta, mu, sigma_sq) {
  return(dnorm(theta, mean=mu, sd=sqrt(sigma_sq), log=TRUE))
}

logPosterior = function(data, mu, sigma_sq, theta=0, gamma=1) {
  prior=logPrior(theta, mu, sigma_sq)
  likelihood=dCauchy(data, theta, gamma)
  likelihood=sum(log(likelihood))
  return(likelihood + prior)
}

mu=0
sigma_sq=100
gamma=1
gridWidth=0.01
theta_grid=seq(0,8,gridWidth)
posterior_distrib=sapply(theta_grid, logPosterior, data=cauchydata, mu=mu,
sigma_sq=sigma_sq, gamma=1)
posterior_distrib=1/gridWidth*exp(posterior_distrib)/sum(exp(posterior_dis
trib))
plot(theta_grid, posterior_distrib, type="l", lwd=2, main="Posterior densi
ty for theta", xlab=expression(theta),
     ylab="Density")

## b) gamma is unknown with prior lognormal.

set.seed(12345)
initVal = c(0,0)

logJointPosterior = function(joint, data, mu, sigma_sq) {
  prior_theta=logPrior(joint[1], mu, sigma_sq)
  prior_gamma=log(dlognormal(joint[2], mu, sigma_sq))
  likelihood=dCauchy(data, joint[1], joint[2])
  likelihood=sum(log(likelihood))
```

```
  return(likelihood + prior_theta + prior_gamma)
}

# Finding the optimized theta and gamma
optimResult = optim(initVal, logJointPosterior, data=cauchydata, mu=mu, si
gma_sq=sigma_sq, method=c("L-BFGS-B"),
                    control=list(fnscale=-1), lower=c(-Inf, 0.001), upper=
c(Inf, Inf), hessian=TRUE)

# Defining the values of interest
postMode = optimResult$par
postCov = -solve(optimResult$hessian)
names(postMode)=c("Theta", "Gamma")
print("The posterior mode is:")
print(postMode)
print("The approximated standard deviation is:")
print(postCov)

## c) Use normal approx in 1b) to obtain marginal posterior for the 99 % p
ercentile of the caucy distrib
## theta + gamma * tan(pi(0.99-0.5))

library(rmvnorm)
normal_approx=rmvnorm(5000, mean=postMode, sigma=postCov)
cauchy_distrib=normal_approx[,1]+normal_approx[,2]*tan(pi*(0.99-0.5))
hist(cauchy_distrib, breaks=50, main="Marginal distribution of special cas
e of caucby", xlab="Function value")
```

Assignment 2 – Bayes Linear Regression, decision with loss function, predictive distribution

```
# Reading the data from file
library(MASS)
BostonHousing = Boston
y = BostonHousing$medv
X = cbind(1,BostonHousing[,1:13]) # Adding a column of ones for the interc
ept
names(X)[1] <- "intercept"
covNames <- names(X)
y <- as.numeric(y)
X <- as.matrix(X)

library(mvtnorm)

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

BayesLinReg <- function(y, X, mu_0, Omega_0, v_0, sigma2_0, nIter){
  # Direct sampling from a Gaussian linear regression with conjugate prior
:
  #
  # beta | sigma2 ~ N(mu_0, sigma2*inv(Omega_0))
  # sigma2 ~ Inv-Chi2(v_0,sigma2_0)
```

```r
  #
  # Author: Mattias Villani, IDA, Linkoping University. http://mattiasvill
ani.com
  #
  # INPUTS:
  #   y - n-by-1 vector with response data observations
  #   X - n-by-nCovs matrix with covariates, first column should be ones i
f you want an intercept.
  #   mu_0 - prior mean for beta
  #   Omega_0  - prior precision matrix for beta
  #   v_0      - degrees of freedom in the prior for sigma2
  #   sigma2_0 - location ("best guess") in the prior for sigma2
  #   nIter - Number of samples from the posterior (iterations)
  #
  # OUTPUTS:
  #   results$betaSample     - Posterior sample of beta.     nIter-by-nCov
s matrix
  #   results$sigma2Sample   - Posterior sample of sigma2.   nIter-by-1 ve
ctor

  # Compute posterior hyperparameters
  n = length(y) # Number of observations
  nCovs = dim(X)[2] # Number of covariates
  XX = t(X)%*%X
  betaHat <- solve(XX,t(X)%*%y)
  Omega_n = XX + Omega_0
  mu_n = solve(Omega_n,XX%*%betaHat+Omega_0%*%mu_0)
  v_n = v_0 + n
  sigma2_n = as.numeric((v_0*sigma2_0 + ( t(y)%*%y + t(mu_0)%*%Omega_0%*%m
u_0 - t(mu_n)%*%Omega_n%*%mu_n))/v_n)
  invOmega_n = solve(Omega_n)

  # The actual sampling
  sigma2Sample = rep(NA, nIter)
  betaSample = matrix(NA, nIter, nCovs)
  for (i in 1:nIter){

    # Simulate from p(sigma2 | y, X)
    sigma2 = rScaledInvChi2(n=1, df = v_n, scale = sigma2_n)
    sigma2Sample[i] = sigma2

    # Simulate from p(beta | sigma2, y, X)
    beta_ = rmvnorm(n=1, mean = mu_n, sigma = sigma2*invOmega_n)
    betaSample[i,] = beta_

  }
  return(results = list(sigma2Sample = sigma2Sample, betaSample=betaSample
))
}

mu_0=rep(0, ncol(X))
omega_0=0.01*diag(ncol(X))
v_0=1
sigma2_0=36
```

```r
nIter=5000

bayes_lin_results=BayesLinReg(y, X, mu_0, omega_0, v_0, sigma2_0, nIter)
# Under quadratic loss, posterior mean is point estimate
beta_estimates=rep(0,ncol(X))
beta_credIntervals=matrix(0, ncol(X), 2)
for (i in 1:ncol(X)) {
  beta_estimates[i]=mean(bayes_lin_results$betaSample[,i])
  beta_credIntervals[i,]=quantile(bayes_lin_results$betaSample[,i], c(0.02
5, 0.975))
}
sigma_estimate=mean(bayes_lin_results$sigma2Sample)
sigma_credInterval=quantile(bayes_lin_results$sigma2Sample, c(0.025, 0.975
))
rownames(beta_credIntervals)=covNames
beta_credIntervals[which(rownames(beta_credIntervals)=="rm"), ]

## Interpretation: for one unit increase of rooms the hosing prices will r
ise between 3991,475 and 5009,826 dollars
## with 95 % posterior probability.

## b) Owner of house 381 is considering selling their house. Bought house
for 10400

old_obs=as.vector(X[381,])
new_obs=old_obs
new_obs[2]=10
pred_draw=rep(0,nIter)
for (i in 1:nIter) {
  pred_draw[i]=bayes_lin_results$betaSample[i,]%*%new_obs+rnorm(1, mean=0,
                                                                sd=sqrt(ba
yes_lin_results$sigma2Sample[i]))
}
pred_mean=mean(pred_draw)
hist(pred_draw, breaks=50)
quantile(posterior_prices, c(0.025, 0.975))
sum(pred_draw>=30)/nIter

## c) See paper.
```

Assignment 4 – Simulation from posterior (Poisson with Gamma prior), simulation predictive distribution, maximizing posterior utility

```r
## a) Simulate 1000 draws from the posterior distrib of theta using conjug
ate prior for theta with mean 250
## and std = 50. Poisson likelihood.

data=c(220,323,174,229)
alpha=25
beta=0.1
n=length(data)

logPriorGamma = function(theta, alpha, beta) {
  return(dgamma(theta, 50, beta, log=TRUE))
```

```r
}

logLike = function(data, theta) {
  n=length(data)
  first=sum(data)*log(theta)
  second=theta*n
  third=0
  for (i in data) {
    for (j in 1:i) {
      third=third+log(j)
    }
  }
  return(first-second-third)
}

logPosterior = function(data, theta, alpha, beta) {
  prior=logPriorGamma(theta, alpha, beta)
  likelihood=logLike(data, theta)
  return(likelihood + prior)
}

# Conjugate prior for poisson is Gamma(alpha, beta), we know that posterio
r is Gamma(alpha + sum(data), beta+n)
post_draws=rgamma(1000, alpha+sum(data), beta+n)
hist(post_draws, main="Posterior distribution of theta", xlab=expression(t
heta))

## b) Simulate 1000 draws from the predictive distrib of next quarter's de
mand, X5, and plot the draws
## in histogram.

q5=rpois(1000, post_draws)
hist(q5, breaks=50, main="Predictive distribution of quarter 5", xlab="Qty
")
sum(q5<=200)/1000

## c)


utility <- function(a,X5){
  util = rep(0,length(X5))
  util[X5<=a] = 10*X5[X5<=a]-(a-X5[X5<=a])
  util[X5>a] = 10*a-0.05*(X5[X5>a]-a)^2
  return(util)
}

mean(q5)
a=seq(136,336,1)
results = matrix(0,length(q5),length(a))
count=1
nameVec=rep(0,length(a))
for (i in a) {
  results[,count]=utility(i,q5)
  nameVec[count]=as.character(i)
```

```
  count=count+1
}
opt_vector=matrix(0,1,length(a))
for (i in 1:length(a)) {
  opt_vector[i]=mean(results[,i])
}
colnames(opt_vector)=nameVec
opt_decision=as.numeric(opt_vector[,which(opt_vector==max(opt_vector))])
names(opt_vector[,which(opt_vector==max(opt_vector))])
plot(a, opt_vector, type="l", lwd=1, col="red")
abline(v=as.numeric(names(opt_vector[,which(opt_vector==max(opt_vector))])
), col="blue")
```

2017-10-27

Assignment 1 – Plot posterior distrib (beta symmetric prior, exponential likelihood), joint posterior distribution through optim, how to choose model?

```
## a) Likelihood: Beta symmetric, prior, expon(1). Plot posterior distrib.

thetaGrid=seq(0.01, 15, length=1000)
data=yProp
lambda=1

logPriorExp = function(theta, lambda) {
  return(dexp(theta, rate=lambda, log=TRUE))
}

logPosterior = function(x, theta, lambda) {
  prior=logPriorExp(theta, lambda)
  likelihood=sum(dbeta(x, theta, theta, log=TRUE))
  return(likelihood+prior)
}

theta_post=sapply(thetaGrid, logPosterior, x=data, lambda=lambda)
theta_post_norm=1/((15-0.01)/1000)*exp(theta_post)/sum(exp(theta_post))
plot(thetaGrid, theta_post_norm, type="l", lwd=2, xlab=expression(theta),
ylab="Posterior density")

# Zero to 1 loss means posterior mode is the optimal point estimator

index=which(theta_post_norm==max(theta_post_norm))
opt_theta=thetaGrid[index]
print(opt_theta)

## Optimal theta is around 4.481491

## b) Theta1 and theta2 are independent apriori. Plot joint posterior dist
rib

logPosteriorMult = function(theta, x, lambda) {
  theta1=theta[1]
  theta2=theta[2]
  prior1=logPriorExp(theta1, lambda)
  prior2=logPriorExp(theta2, lambda)
```

```
  likelihood=sum(dbeta(x, theta1, theta2, log=TRUE))
  return(likelihood+prior1+prior2)
}

# Defining initial values to be passed on to the optimizer
initVal = c(1,1)

# Finding the optimized betavector
optimResult = optim(initVal, logPosteriorMult, x=data, lambda=1, method=c(
"L-BFGS-B"),
                    control=list(fnscale=-1), lower=c(0.01,0.01), upper=c(
Inf, Inf), hessian=TRUE)

# Defining the values of interest
postMode = optimResult$par
postCov = -solve(optimResult$hessian)
names(postMode)=c("Theta1", "Theta2")
rownames(postCov)=c("Theta1", "Theta2")
colnames(postCov)=c("Theta1", "Theta2")
print("The posterior mode is:")
print(postMode)
print("The approximated standard deviation is:")
print(postCov)

## c) Discuss how a Bayesian can determine if the symmetric model in 1a) o
r the non-symmetric model in 1b)
## is most appropriate for this data. No need to compute anything here, ju
st discuss.

## By calculating marginal likelihood for each model and check which has t
he highest. One can also calculate
## the bayes factor or the posterior model probabilities and choose the mo
del with the highest probability.
```

Assignment 2 – Bayes Linear Regression, HPD interval, predictive distribution house prices for specific house type

```
## a) Use conjugate priors, standard normal and invchisq and use BayesLinR
eg to simulate 5000 draws from posterior
## distrib

# Reading the data from file
library(MASS)
BostonHousing = Boston
y = BostonHousing$medv
X = cbind(1,BostonHousing[,1:13]) # Adding a column of ones for the interc
ept
names(X)[1] <- "intercept"
covNames <- names(X)
y <- as.numeric(y)
X <- as.matrix(X)

library(mvtnorm)
```

```r
# Defining a function that simulates from the scaled inverse Chi-square di
stribution
rScaledInvChi2 <- function(n, df, scale){
  return((df*scale)/rchisq(n,df=df))
}

BayesLinReg <- function(y, X, mu_0, Omega_0, v_0, sigma2_0, nIter){
  # Direct sampling from a Gaussian linear regression with conjugate prior
:
  #
  # beta | sigma2 ~ N(mu_0, sigma2*inv(Omega_0))
  # sigma2 ~ Inv-Chi2(v_0,sigma2_0)
  #
  # Author: Mattias Villani, IDA, Linkoping University. http://mattiasvill
ani.com
  #
  # INPUTS:
  #   y - n-by-1 vector with response data observations
  #   X - n-by-nCovs matrix with covariates, first column should be ones i
f you want an intercept.
  #   mu_0 - prior mean for beta
  #   Omega_0  - prior precision matrix for beta
  #   v_0      - degrees of freedom in the prior for sigma2
  #   sigma2_0 - location ("best guess") in the prior for sigma2
  #   nIter - Number of samples from the posterior (iterations)
  #
  # OUTPUTS:
  #   results$betaSample     - Posterior sample of beta.     nIter-by-nCov
s matrix
  #   results$sigma2Sample   - Posterior sample of sigma2.   nIter-by-1 ve
ctor

  # Compute posterior hyperparameters
  n = length(y) # Number of observations
  nCovs = dim(X)[2] # Number of covariates
  XX = t(X)%*%X
  betaHat <- solve(XX,t(X)%*%y)
  Omega_n = XX + Omega_0
  mu_n = solve(Omega_n,XX%*%betaHat+Omega_0%*%mu_0)
  v_n = v_0 + n
  sigma2_n = as.numeric((v_0*sigma2_0 + ( t(y)%*%y + t(mu_0)%*%Omega_0%*%m
u_0 - t(mu_n)%*%Omega_n%*%mu_n))/v_n)
  invOmega_n = solve(Omega_n)

  # The actual sampling
  sigma2Sample = rep(NA, nIter)
  betaSample = matrix(NA, nIter, nCovs)
  for (i in 1:nIter){

    # Simulate from p(sigma2 | y, X)
    sigma2 = rScaledInvChi2(n=1, df = v_n, scale = sigma2_n)
    sigma2Sample[i] = sigma2

    # Simulate from p(beta | sigma2, y, X)
```

```
    beta_ = rmvnorm(n=1, mean = mu_n, sigma = sigma2*invOmega_n)
    betaSample[i,] = beta_

  }
  return(results = list(sigma2Sample = sigma2Sample, betaSample=betaSample
))
}

mu_0=rep(0,ncol(X))
omega_0=1/100*diag(ncol(X))
v_0=1
sigma2_0=36
nIter=5000
post_distrib=BayesLinReg(y,X, mu_0, omega_0, v_0, sigma2_0, nIter)
post_beta=post_distrib$betaSample
colnames(post_beta)=covNames
lstat_post=subset(post_beta, select="lstat")
par(mfrow=c(1,1))
plot(density(lstat_post), main="Posterior density of lstat", lwd=2)
credInterval=quantile(lstat_post, probs=c(0.05, 0.95))
abline(v=credInterval[1], col="grey", lwd=3, lty=3)
abline(v=credInterval[2], col="grey", lwd=3, lty=3)

# Since posterior of beta is the student t-distrib the distrib is symmetri
c and therefore HPD interval is the same
# as equal tail interval

new_obs=X[9,]
names(new_obs)=covNames
new_obs_2=new_obs
new_obs_2[which(names(new_obs)=="lstat")]=new_obs_2[which(names(new_obs)==
"lstat")]*0.7
post_sigma2=post_distrib$sigma2Sample
pred_price1=post_beta%*%new_obs+rnorm(nIter, mean=0, sd=sqrt(post_sigma2))
pred_price2=post_beta%*%new_obs_2+rnorm(nIter, mean=0, sd=sqrt(post_sigma2
))
hist(pred_price1, breaks=50, main="Histogram of predicted price before cha
nge")
hist(pred_price2, breaks=50, main="Histogran of predicted price after chan
ge")
pred_price_house9=post_beta[,14]*(new_obs[14]*0.7-new_obs[14])
mean(pred_price_house9)
quantile(pred_price_house9, probs=c(0.025, 0.975))

# For a house like number 9 it will increase the house price with high pos
terior probability.
```

Assignment 4 – Simulation of predDraw from normal, approximated probability weight larger than value, decision making (linear loss function)

```
## a) Simulate 1000 draws from predictive distrib of the maximal weight on
a given future day, model: y=10*a where
## y is the weight and a is the build cost. y~N(theta, sigma2). Noninforma
tive prior assumed.
```

```r
y=c(191, 196, 197, 189)
sigma2=10^2
# Noninformative prior assumed to be constant

yPred_post=rnorm(1000, mean=mean(y), sd=sqrt(sigma2*(1+1/length(y))))

## b) Use simulation to approximate the predictive probability that weight
higher than 230

pred_max365=rep(0,1000)
for (i in 1:1000) {
  pred_max365[i]=max(rnorm(365, mean=mean(y), sd=sqrt(sigma2*(1+1/length(y
)))))
}
prob_yPred365=sum(pred_max365>230)/1000
print(prob_yPred365)

## The probability is 0.157

## c) The loss function is linear

expectedLoss = function(a, maxWeight) {
  probCollapse=sum(maxWeight>10*a)/1000
  return(a*(1-probCollapse)+probCollapse*(a+100))
}

a=seq(20,30,0.01)
plot(a, sapply(a, expectedLoss, maxWeight=pred_max365), type="l", lwd=2, x
lab="a", ylab="EL",
     main="Loss function")
aOpt=a[which(min(sapply(a, expectedLoss, maxWeight=pred_max365)))]
print(aOpt)

## The answer is 23.89
```

2018-06-01

Assignment 1 – Plot posterior using samples and expression, simulate from predictive

```r
## a) Draw 1000 samples from prior (Gamma) and 1000 samples from posterior
(Gamma). Plot prior and posterior using
## both samples and their analytical expressions.

n=50
x_mean=10
beta=2
nDraws=1000

# We know that posterior distribution is the Gamma(alpha+sum(data), beta+n
). Mean for Gamma distrib is alpha/beta.
## If beta=2 then beta+n for posterior is 52. alpha/2=(alpha+500)/52 which
yields 50*alpha=1000 and alpha=20
## Check: 20/2=10, (20+500)/52=10 OK!
```

```r
alpha=20 # According to motivation above
post_draws=rgamma(nDraws, alpha+n*x_mean, beta+n)
prior_draws=rgamma(nDraws, alpha, beta)
gridWidth=0.01
muGrid_post=seq(7,12, gridWidth) # Range taken with inspiration from histo
gram
muGrid_prior=seq(4,20,gridWidth)
par(mfrow=c(2,1))
hist(post_draws, breaks=50, main="Posterior", xlab=expression(mu),
     freq=FALSE)
lines(muGrid_post, dgamma(muGrid_post, alpha+n*x_mean, beta+n), lwd=2, xla
b=expression(mu))
hist(prior_draws, breaks=50, main="Prior", xlab=expression(mu),
     freq=FALSE)
lines(muGrid_prior, dgamma(muGrid_prior, alpha, beta), lwd=2, xlab=express
ion(mu))

## As seen in the plots the distributions resemble each other.

## b) Simulate 1000 draws from predictive distribution of new observation
and plot distribution.

par(mfrow=c(1,1))
x_pred=rpois(1000, lambda=post_draws)
hist(x_pred, breaks=50, main="Histogram, approximated posterior predictive
distribution", xlab=expression(mu),
     freq=FALSE)

## c) Prob that x51=10 based on posterior predictive distribution

sum(x_pred==10)/nDraws

## [1] 0.132
```

Assignment 2 – Bayes Linear Regression (fish), marginal posterior, 90 % equal tail with interpretation, new experiment with two different new_obs with likelihoods (Bayesian analysis)

```r
## Linear regression model for fish with 3 covariates.

# Reading the data from file
load(file = 'fish.RData')

library(mvtnorm)

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

BayesLinReg <- function(y, X, mu_0, Omega_0, v_0, sigma2_0, nIter){
  # Direct sampling from a Gaussian linear regression with conjugate prior
:
  #
```

```r
  # beta | sigma2 ~ N(mu_0, sigma2*inv(Omega_0))
  # sigma2 ~ Inv-Chi2(v_0,sigma2_0)
  #
  # Author: Mattias Villani, IDA, Linkoping University. http://mattiasvill
ani.com
  #
  # INPUTS:
  #   y - n-by-1 vector with response data observations
  #   X - n-by-nCovs matrix with covariates, first column should be ones i
f you want an intercept.
  #   mu_0 - prior mean for beta
  #   Omega_0  - prior precision matrix for beta
  #   v_0      - degrees of freedom in the prior for sigma2
  #   sigma2_0 - location ("best guess") in the prior for sigma2
  #   nIter - Number of samples from the posterior (iterations)
  #
  # OUTPUTS:
  #   results$betaSample     - Posterior sample of beta.     nIter-by-nCov
s matrix
  #   results$sigma2Sample   - Posterior sample of sigma2.   nIter-by-1 ve
ctor

  # Compute posterior hyperparameters
  n = length(y) # Number of observations
  nCovs = dim(X)[2] # Number of covariates
  XX = t(X)%*%X
  betaHat <- solve(XX,t(X)%*%y)
  Omega_n = XX + Omega_0
  mu_n = solve(Omega_n,XX%*%betaHat+Omega_0%*%mu_0)
  v_n = v_0 + n
  sigma2_n = as.numeric((v_0*sigma2_0 + ( t(y)%*%y + t(mu_0)%*%Omega_0%*%m
u_0 - t(mu_n)%*%Omega_n%*%mu_n))/v_n)
  invOmega_n = solve(Omega_n)

  # The actual sampling
  sigma2Sample = rep(NA, nIter)
  betaSample = matrix(NA, nIter, nCovs)
  for (i in 1:nIter){

    # Simulate from p(sigma2 | y, X)
    sigma2 = rScaledInvChi2(n=1, df = v_n, scale = sigma2_n)
    sigma2Sample[i] = sigma2

    # Simulate from p(beta | sigma2, y, X)
    beta_ = rmvnorm(n=1, mean = mu_n, sigma = sigma2*invOmega_n)
    betaSample[i,] = beta_

  }
  return(results = list(sigma2Sample = sigma2Sample, betaSample=betaSample
))
}

## a) Plot marginal posterior for each param
```

```r
y=as.matrix(subset(fish, select="length"))
X=as.matrix(fish[,2:ncol(fish)])
covNames=colnames(X)
mu_0=rep(0,3)
omega_0=0.01*diag(1,3)
v_0=1
sigma2_0=100^2
nIter=5000

linPost=BayesLinReg(y, X, mu_0, omega_0, v_0, sigma2_0, nIter)
betaPost=linPost$betaSample
colnames(betaPost)=covNames
sigma2Post=linPost$sigma2Sample
par(mfrow=c(2,2))
for (i in 1:ncol(betaPost)) {
  hist(betaPost[,i], xlab=paste("Beta",i,sep=""), main=paste("Marginal pos
terior distribution of beta", i, sep=""))
}
hist(sigma2Post, xlab=expression(sigma), main="Marginal posterior distribu
tion of sigma2")

par(mfrow=c(1,1))

## Construct 90 % equal tail interval for beta1 and interpret it.

quantile(subset(betaPost, select="age"), probs=c(0.05, 0.95))

## It can be concluded that when the age of the fish increases with one un
it the length of the fish increases
## with approximately between 2.284 and 2.960 mm with 90 % posterior proba
bility.

## d) New experiment fish has been grown in water tank with water temp 30
degrees celsius. Newborn fish have
## have been inserted into the tank at two time points, 30 days ago and 10
0 days ago. Equal amount of fish
## in the two different ages. You pick up fish randomly from water tank. D
o bayesian analysis (using sim methods)
## to determine predictive distrib of the length of the picked up fish.

x1=c(1,30,30)
x2=c(1,100,30)
x_pred=rep(0,nIter)
for (i in 1:nIter) {
  prob=runif(1)
  if(prob>0.5) {
    x_pred[i]=betaPost[i,]%*%x1+rnorm(1, mean=0, sd=sqrt(sigmaPost[i]))
  } else {
    x_pred[i]=betaPost[i,]%*%x2+rnorm(1, mean=0, sd=sqrt(sigmaPost[i]))
  }
}
hist(x_pred, main="Histogram of predictive distribution of length of fish"
,
     xlab="Length in mm", freq=FALSE, breaks=50)
```

Assignment 3 – Choosing between 3 models, marginal likelihood

```
## c) Choose between three models where two of them use Beta prior and the
last one assumes p=0.5. Which model
## should be chosen?

model1=choose(10,3)*gamma(4)*gamma(8)*gamma(2)/gamma(12)
model2=choose(10,3)*gamma(7)*gamma(11)*gamma(8)/(gamma(4)*gamma(4)*gamma(1
8))
model3=choose(10,3)*0.5^10
model1_norm=model1/sum(c(model1, model2, model3))
model2_norm=model2/sum(c(model1, model2, model3))
model3_norm=model3/sum(c(model1, model2, model3))
```

Assignment 4 – Truncated normal distrib, stan with time series model, plot of data & posterior mean & 95 % credible intervals over time

```
## a) Consider observations with values above 200. Remaining datapoints as
sumed to be indep. and follow a
## truncated normal distribution with density specified. L=200 lower trunc
ation point. Write a function in R
## that computes the (unnormalized) log posterior distribution of mu. Use
function to plot the posterior distrib
## of mu for the observations greater than 200 in the data vector sulfur.
For the plot, use a grid constructed
## in R with seq(100,400,1)

# Reading the data from file
load(file = 'sulfur.RData')

muGrid=seq(100,400,1)
sigma=100
data=sulfur[sulfur>200]

# Constant prior for mu is assumed

logPost = function(data, mu, sigma, L=200) {
  nominator=dnorm((data-mu)/sigma, mean=0, sd=1, log=TRUE)
  denominator=log(sigma)+log(1-pnorm((L-mu)/sigma))
  return(sum(nominator-denominator+0)) # Assumed constant prior which can
be set to 1 which in log scale is 0
}

post_mu=exp(sapply(muGrid, logPost, data=data, sigma=sigma))
post_mu_norm=post_mu/sum(post_mu) # Since gridwidth is 1 we don't have to
compensate for it
plot(muGrid, post_mu_norm, type="l", lwd=2, main="Posterior distribution o
f mu", xlab=expression(mu))

library(rstan)
T = length(sulfur)
T_cens = sum(sulfur <= 200)
censData <- list(T=T, T_cens = T_cens, x=sulfur, L=200)

# Model
```

```
censModel <- '
data {
  int<lower=0> T;       // Total number of time points
  int<lower=0> T_cens;  // Number of censored time points
  real x[T];            // Partly censored data
  real<upper=max(x)> L; // Lower truncation point
}

parameters {
  real mu;
  real<lower=0> sigma;
  real<upper=L> x_cens[T_cens]; // Censored values
}

model {
  int t_cens = 0;
  for (t in 1:T){
    if (x[t] > L)
      x[t] ~ normal(mu,sigma);
    else {
      t_cens += 1;
      x_cens[t_cens] ~ normal(mu,sigma);
    }
  }
}
'

## b) Now condiser all data points. Values below 200 being censored.

fit=stan(model_code=censModel, data=censData)
print(fit)
post_draws=extract(fit)
grid=seq(1,4000,1)
plot(grid, post_draws$mu, type="l", main="Traceplot of mu", xlab=expressio
n(mu), ylab="Value")
plot(grid, post_draws$sigma, type="l", main="Traceplot of sigma", xlab=exp
ression(sigma), ylab="Value")
par(mfrow=c(4,2))
for (i in 1:8) {
  plot(grid, post_draws$x_cens[,i], type="l", main=paste("Traceplot of ",
i, "th obs of obs below 200", sep=""),
       xlab=i, ylab="Value")
}
par(mfrow=c(1,1))

plot(post_draws$mu, post_draws$sigma, type="p", col="grey", main="Joint po
sterior of mu and sigma",
     xlab=expression(mu), ylab=expression(sigma))

## c) Instead consider time series model. Assume that observations follow
an independent normal distrib
## when conditioned on a latent AR(1) process z, but with values of xi bel
ow 200 being censored and set to 200.
## Modify the stan code in order to do inference for this model instead. A
```

```
lso put a normal prior on
## mu~N(300,100^2) Plot the posterior of phi. Also produce a plot that con
tains both the data and the posterior
## mean and 95 % credible intervals for the latent intensity z over time.

StanModel_AR = '
data {
  int<lower=0> T;       // Total number of time points
  int<lower=0> T_cens;  // Number of censored time points
  real x[T];            // Partly censored data
  real<upper=max(x)> L; // Lower truncation point
}

parameters {
  real mu;
  real phi;
  real<lower=0> sigma;
  real<upper=L> x_cens[T_cens]; // Censored values
  vector[T] z;
}

model {
  // Prior
  int t_cens = 0;
  phi ~ uniform(-1,1);
  mu ~ normal(300, 100);
  for (n in 2:T)
    z[n] ~ normal(mu + phi * (z[n-1]-mu), sigma);


  // Model/likelihood
  for (t in 1:T){
    if (x[t] > L)
      x[t] ~ normal(z[t],20);
    else {
      t_cens += 1;
      x_cens[t_cens] ~ normal(z[t],20);
    }
  }
}

generated quantities {
  vector[T] post_mean;
  post_mean = z;
}
'
fitAR=stan(model_code=StanModel_AR, data=censData)
print(fitAR)
post_draws_AR=extract(fitAR)
postPhi=post_draws_AR$phi
postZ=post_draws_AR$post_mean
hist(postPhi, breaks=50, main="Approximated posterior density of phi", xla
b=expression(phi), freq=FALSE)
grid=seq(1,31)
```

```r
plot(grid, sulfur, col="blue", main="Emissions of sulfur dioxide", xlab="D
ay of month", ylab="mg/Nm^3",
     ylim=c(0,500))
postMean=rep(0,ncol(postZ))
credIntervals=matrix(0,ncol(postZ),2)
for (i in 1:ncol(postZ)) {
  postMean[i]=mean(postZ[,i])
  credIntervals[i,]=quantile(postZ[,i], probs=c(0.025, 0.975))
}
lines(grid, postMean, type="l", col="red", lwd=2)
lines(grid, credIntervals[,1], col="grey", lwd=1, lty=2)
lines(grid, credIntervals[,2], col="grey", lwd=1, lty=2)
legend("topleft", legend=c("Data", "Posterior mean", "95 % cred intervals"
), lwd=c(NaN, 2, 1), lty=c(NaN,1,2),
       pch=c(1,NaN, NaN), col=c("blue", "red", "grey"))
```

2019-08-21

Assignment 1 – Bayes Linear Regression, point estimates, 95 % tail intervals, posterior mode and HPD 90 % intervals, predictive Bayesian analysis

```r
## a) Use BayesLinReg to sim 5000 draws from posterior distrib of all coef
f coefficients. Summarize posterior
## with point estimate under quadratic loss function and 95 % equal tail i
ntervals. Interpret cred intervals for
## regression coefficient on nitrogen oxides concentration.

################################
########## Problem 1 ##########
################################

# Reading the data from file
library(MASS)
BostonHousing = Boston
y = BostonHousing$medv
X = cbind(1,BostonHousing[,1:13]) # Adding a column of ones for the interc
ept
names(X)[1] <- "intercept"
covNames <- names(X)
y <- as.numeric(y)
X <- as.matrix(X)
XNewHouse <- c(1,0.03,40,1.5,0,0.5,6,30,5,3,300,17,390,4)

if(length((grep("mvtnorm",installed.packages()[,1])))==0)
  install.packages("mvtnorm")
library(mvtnorm)

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

BayesLinReg <- function(y, X, mu_0, Omega_0, v_0, sigma2_0, nIter){
  # Direct sampling from a Gaussian linear regression with conjugate prior
```

```
:
  #
  # beta | sigma2 ~ N(mu_0, sigma2*inv(Omega_0))
  # sigma2 ~ Inv-Chi2(v_0,sigma2_0)
  #
  # Author: Mattias Villani, IDA, Linkoping University. http://mattiasvill
ani.com
  #
  # INPUTS:
  #   y - n-by-1 vector with response data observations
  #   X - n-by-nCovs matrix with covariates, first column should be ones i
f you want an intercept.
  #   mu_0 - prior mean for beta
  #   Omega_0  - prior precision matrix for beta
  #   v_0      - degrees of freedom in the prior for sigma2
  #   sigma2_0 - location ("best guess") in the prior for sigma2
  #   nIter - Number of samples from the posterior (iterations)
  #
  # OUTPUTS:
  #   results$betaSample     - Posterior sample of beta.     nIter-by-nCov
s matrix
  #   results$sigma2Sample   - Posterior sample of sigma2.   nIter-by-1 ve
ctor

  # Compute posterior hyperparameters
  n = length(y) # Number of observations
  nCovs = dim(X)[2] # Number of covariates
  XX = t(X)%*%X
  betaHat <- solve(XX,t(X)%*%y)
  Omega_n = XX + Omega_0
  mu_n = solve(Omega_n,XX%*%betaHat+Omega_0%*%mu_0)
  v_n = v_0 + n
  sigma2_n = as.numeric((v_0*sigma2_0 + ( t(y)%*%y + t(mu_0)%*%Omega_0%*%m
u_0 - t(mu_n)%*%Omega_n%*%mu_n))/v_n)
  invOmega_n = solve(Omega_n)

  # The actual sampling
  sigma2Sample = rep(NA, nIter)
  betaSample = matrix(NA, nIter, nCovs)
  for (i in 1:nIter){

    # Simulate from p(sigma2 | y, X)
    sigma2 = rScaledInvChi2(n=1, df = v_n, scale = sigma2_n)
    sigma2Sample[i] = sigma2

    # Simulate from p(beta | sigma2, y, X)
    beta_ = rmvnorm(n=1, mean = mu_n, sigma = sigma2*invOmega_n)
    betaSample[i,] = beta_

  }
  return(results = list(sigma2Sample = sigma2Sample, betaSample=betaSample
))
}
```

```r
mu_0=rep(0, ncol(X))
omega_0=1/10^2*diag(ncol(X))
v_0=1
sigma2_0=5^2
nIter=5000
linPost=BayesLinReg(y, X, mu_0, omega_0, v_0, sigma2_0, nIter)
betaPost=linPost$betaSample
sigma2Post=linPost$sigma2Sample
results=matrix(0,ncol(X)+1,3)
results_names=covNames
results_names=append(results_names, "sigma2")
rownames(results)=results_names
colnames(results)=c("Point estimator", "2,5%", "97,5%")
for (i in 1:ncol(X)) {
  results[i,1]=mean(betaPost[,i])
  results[i,-1]=quantile(betaPost[,i], probs=c(0.025, 0.975))
}
results[(ncol(X)+1),1]=mean(sigma2Post)
results[(ncol(X)+1),-1]=quantile(sigma2Post, probs=c(0.025, 0.975))
results

## b) Kernel density estimates. Compute posterior mode and HPD 90 % for si
gma2

sigma2_kernel=density(sigma2Post)
sigma2_kernel.df=data.frame(sigma2=sigma2_kernel$x, density=sigma2_kernel$
y)
sigma2_kernel.df=sigma2_kernel.df[order(-sigma2_kernel.df[,2]),]
index=dim(sigma2_kernel.df)[1]
sigma2_kernel.df$density=cumsum(sigma2_kernel.df$density)/sum(sigma2_kerne
l.df$density)
sigma2Cred=sigma2_kernel.df[sigma2_kernel.df$density<0.9,]
credInterval=c(min(sigma2Cred$sigma2), max(sigma2Cred$sigma2))
sigma2Mode=sigma2_kernel.df[1,]$sigma2

plot(sigma2_kernel, type="l", lwd=2, main="Kernel density estimate of sigm
a2", xlab=expression(sigma^2))
abline(v=sigma2Mode, col="red", lwd=1, lty=2)
abline(v=credInterval[1], col="grey", lwd=1, lty=3)
abline(v=credInterval[2], col="grey", lwd=1, lty=3)
legend("topright", legend=c("Kernel density estimate", "Posterior mode", "
90 % HPD Interval"), lty=c(1,2,3),
       lwd=c(2,1,1), col=c("black", "red", "grey"))

## c) Construction company planning to build a new house with covariates g
iven in XNewHouse. Cost is 20000 dollars
## and the company is planning to sell the house when finished. Do Bayesia
n analysis to determine how probable
## it is that the company will make money (that the house will sell for mo
re than 20000 dollars).

XNewHouse <- c(1,0.03,40,1.5,0,0.5,6,30,5,3,300,17,390,4)
profitVec=rep(0,nIter)
for (i in 1:nIter) {
```

```
  profitVec[i]=-20+betaPost[i]%*%XNewHouse+rnorm(1, mean=0, sd=sqrt(sigma2
Post[i]))
}
hist(profitVec)
probProfit=sum(profitVec>0)/nIter
print(probProfit)
quantile(profitVec, probs=c(0.025, 0.975))

## Very probable that the company will make a profit since 98.82 % of the
posterior draws are above zero. Negative
## values are also not present in the 95 % equal tail interval which also
indicates that the company will make
## a profit.
```

Assignment 2 – Predictive draw earthquakes (math calculations as basis)

```
## b) Simulate predictive draw of max no. of years until next earthquake o
ccurs, 95 % prob. alpha=1, beta=1.

alpha=1
beta=1
xObs=c(35, 14, 4, 10, 2)
n=length(xObs)i
nIter=5000
predDistrib=rep(0,nIter)
for(i in 1:nIter) {
  posteriorDraw=rbeta(1,alpha+n, beta+sum(xObs))
  predDistrib[i]=rgeom(1,posteriorDraw)
}
predDistrib_maxYear=quantile(predDistrib, probs=0.95)
predDistrib_maxYear
```

Assignment 3 – Calc unnormalized posterior and plot normalized posterior

```
## c) Calc unnormalized posterior and plot normalized posterior. Gamma pri
or and indep likelihoods.

gridWidth=0.01
thetaGrid=seq(0,2,gridWidth)
xData <- c(1.888, 2.954, 0.364, 0.349, 1.090, 7.237)
yData <- c(-1.246, -1.139, -0.358, -1.308, -0.930, -0.157, -0.111, -0.635)
alpha=3
beta=2

logPosteriorX = function(theta, alpha, beta) {
  return(dgamma(theta, alpha, beta, log=TRUE))
}

likeY = function(y, theta) {
  return(-3*sum(log(1+(1/5)*(y-log(theta))^2)))
}

logPosterior = function(theta, alpha, beta, xDat, yDat) {
  likelihoodY=likeY(yDat, theta)
  logPostX=logPosteriorX(theta, length(xDat+3), sum(xDat)+2)
  return(likelihoodY+logPostX)
```

```
}

post_theta=sapply(thetaGrid, logPosterior, alpha=alpha, beta=beta, xDat=xD
ata, yDat=yData)
post_theta_norm=1/gridWidth*exp(post_theta)/sum(exp(post_theta))
plot(thetaGrid, post_theta_norm, type="l", lwd=2, main="Posterior of theta
", xlab=expression(theta),
     ylab="Density")
```

Assignment 4 – Simulate using RWM Hastings, simulation using metropolis hastings with gamma proposal density function, suggestions how to improve sampler, traceplots

```
## Aircraft incidents assumed to be independent, follow negative binomial
distrib. Assume joint prior
## 1/phi^2
## a) Simulate from posterior using Metropolis algorithm. Denote theta=c(m
u, phi) and use as proposal dens
## the multivariate normal density (random walk metropolis).

# Load airline incidents data
load(file = 'incidents.RData')
data=incidents$incidents
library(mvtnorm)

nIter=1000
burnIn=50
theta_0=c(200,20)
c=0.1
postCov=diag(c(100,5))

# Defining function for sampling through metropolishastings
RVMSampler = function(previousVal, postCov, c, myFunction, ...) {
  proposalVal=rmvnorm(1, mean=previousVal, sigma=c*postCov)
  proposalVal[proposalVal<=0]=1e-6
  alpha=min(1, exp(myFunction(proposalVal,...)-myFunction(previousVal, ...
)))
  u=runif(1)
  if(u < alpha) {
    return(list(theta=proposalVal, acceptProb=alpha))
  } else {
    return(list(theta=previousVal, acceptProb=alpha))
  }
}

logPrior = function(phi) {
  return(-2*log(phi))
}

logLike <- function(param, x){
  theta1 = param[1]
  theta2 = param[2]
  logPost = sum(logdNegBin(x, theta1, theta2))  - 2*log(theta2)
  return(logPost)
}
```

```r
logPost = function(theta, data) {
  log_Prior=logPrior(theta[2])
  log_Like=logLike(theta, data)
  return(log_Prior+log_Like)
}

post_matrix = matrix(0, nIter+burnIn, 2)
# Setting initial values of beta to same initVals as in the optimizer (tak
en randomly from normal distrib)
post_matrix[1,]=theta_0
accProb=rep(0, nIter)
set.seed(12345)

for(i in 1:(nIter+burnIn)) {
  if(i<(nIter+burnIn)) {
    draw=RVMSampler(post_matrix[i,], postCov, c, logPost, data)
    post_matrix[i+1,]=draw$theta
    accProb[i+1]=draw$acceptProb
  }
}

iter=seq(1,nIter+burnIn,1)
plot(iter[-(1:burnIn)], post_matrix[-(1:burnIn),1], type="l", lwd=1, col="
grey", main="Traceplot of mu in RVM",
     xlab=expression(mu), ylab="Value")
plot(iter[-(1:burnIn)], post_matrix[-(1:burnIn),2], type="l", lwd=1, col="
grey", main="Traceplot of phi in RVM",
     xlab=expression(phi), ylab="Value")
mean(accProb)

## This MCMC sampler is not efficient since it moves very slowly and is th
erefore probably not exploring
## the whole posterior distribution.We can also see that the acceptance pr
obability for this algorithm
## is around 84,4 % and it should be around 30 %. Once could tune the c pa
ram to lower the acceptance probability.
## One example is to increase c to a value of 3 which would yield in appro
ximately 30 % acceptance rate.

## b) Instead simulate from posterior using metropolis hastings.

c=0.8

MHSampler = function(previousVal, postCov, c, myFunction, ...) {
  proposalVal_mu=rgamma(1, c*previousVal[1], c)
  proposalVal_phi=rgamma(1, c*previousVal[2], c)
  proposalVal=c(proposalVal_mu, proposalVal_phi)
  proposalVal[proposalVal<=0]=1e-6
  alpha=min(1, exp(myFunction(proposalVal,...)-myFunction(previousVal, ...
)+
                     dgamma(previousVal[1], c*proposalVal[1], c)+dgamma(pr
eviousVal[2],c*proposalVal[2],c)-
                     dgamma(proposalVal[1], c*previousVal[1], c)-dgamma(pr
```

```
oposalVal[2], c*proposalVal[2],c)))
  u=runif(1)
  if(u < alpha) {
    return(list(theta=proposalVal, acceptProb=alpha))
  } else {
    return(list(theta=previousVal, acceptProb=alpha))
  }
}

post_matrix2 = matrix(0, nIter+burnIn, 2)
theta_0=c(200,10)
post_matrix2[1,]=theta_0
accProb2=rep(0, nIter)
set.seed(12345)

for(i in 1:(nIter+burnIn)) {
  if(i<(nIter+burnIn)) {
    draw=MHSampler(post_matrix2[i,], postCov, c, logPost, data)
    post_matrix2[i+1,]=draw$theta
    accProb2[i+1]=draw$acceptProb
  }
}

plot(iter[-(1:burnIn)], post_matrix2[-(1:burnIn),1], type="l", lwd=1, col=
"grey", main="Traceplot of mu in MH",
     xlab=expression(mu), ylab="Value")
plot(iter[-(1:burnIn)], post_matrix2[-(1:burnIn),2], type="l", lwd=1, col=
"grey", main="Traceplot of phi in MH",
     xlab=expression(phi), ylab="Value")
mean(accProb2)

## The new algorithm seems to rapidly explore the posterior which is good.
The acceptance probability is also lower
## around 30 % which also indicates that this algorithm is better than the
previous one.
```

2019-10-31

Assignment 1 – Expected utility, calculations from math

```
## a) Theta is known

theta=0.6
eu_buy=0.6*30-0.4*10
eu_nobuy=0.6*90-120*0.4

## Answer: Should buy option

## c) Compute bayesian decision for day 101 based on information in b).

theta_new=13/21
eu_buy_new=theta_new*30-(1-theta_new)*10
eu_nobuy_new=theta_new*90-(1-theta_new)*120

## Answer: Should buy since utility higher.
```

Assignment 2 – Poisson likelihood with gamma prior, plot posterior, separation of data, now two indep poisson models, comparison

```
## a) Consider poisson likelihood model. Use conjugate prior and plot post
erior in given interval.
## Compute posterior probability that theta is smaller than 21.

# Calculations show that alpha=20, beta=1

data=Traffic$y
alpha=20
beta=1
n=length(data)

# We know that Poisson with gamma prior is gamma distributed with alphaNew
=alpha+sum(data), betaNew=beta+n

grid=seq(18,24,0.01)
post_distrib=dgamma(grid, shape=alpha+sum(data), rate=beta+n)
plot(grid, post_distrib, type="l", lwd=2, main="Posterior distrib. of thet
a", xlab=expression(theta))
post_prob=pgamma(21, shape=alpha+sum(data), rate=beta+n)

## Answer: Probability is 0.0557

## b) Two independent poisson models.

data_model1=Traffic[which(Traffic[,3]=="yes"),]$y
data_model2=Traffic[which(Traffic[,3]=="no"),]$y

alpha_1=20+sum(data_model1)
alpha_2=20+sum(data_model2)
beta_1=1+length(data_model1)
beta_2=1+length(data_model2)
post_distrib_1=rgamma(5000, shape=alpha_1, rate=beta_1)
post_distrib_2=rgamma(5000, shape=alpha_2, rate=beta_2)
hist(post_distrib_1, breaks=50)
hist(post_distrib_2, breaks=50)
post_diff=post_distrib_2-post_distrib_1
hist(post_diff,
     main="Posterior distribution of difference between no speedlimit and
speedlimit", xlab="No. of accidents")
quantile(post_diff, prob=c(0.025, 0.975))
mean(post_diff)

## We can see that the difference between the two distributions is larger
than 0 with high probability. In this
## case we can say that the difference in traffic accidents between when n
o speed limit were applied and
## when a speed limit were applied is between 2.82 and 5.53 approximately
with 95 % posterior probability.
## The conclusion from this is that yes, a speed limit leads to a lower am
ount of accidents.
```

```r
## c) A politician claims that the experiment proves that introducing spee
d limit decreases the number
## of accidents by at least 15 %.

mean(0.85*post_distrib_2>post_distrib_1)

## Likely that the decrease yields 15 % but 86 % probable and not 95 % pro
bability which is commonly used
## in statistical experiments.
```

Assignment 3 – Simulation of joins posteriors with Gibbs sampling, traceplots

```r
## c) Make simulations of joins posterior of v and pi using Gibbs sampling
.

x=20
lambda=10
alpha=2
beta=2
nIter=2000
burnIn=500

results=matrix(0,burnIn+nIter,2)
initVal=lambda # Since lambda=30
results[1,1]=initVal
results[1,2]=rnorm(1)
for (i in 1:(nIter+burnIn-1)) {
  z=rpois(1, lambda*(1-results[i,2]))
  results[i+1,1]=z+x
  results[i+1,2]=rbeta(1, alpha+x, beta+results[i+1,1]-x)
}

grid=seq(burnIn+1, nIter+burnIn)
barplot(table(results[(burnIn+1):(nIter+burnIn),1]), main="Marginal poster
ior of nu", xlab=expression(nu))
hist(results[(burnIn+1):(nIter+burnIn),2], breaks=50, main="Marginal poste
rior of pi", xlab=expression(pi))
plot(grid, results[(burnIn+1):(nIter+burnIn),2],type="l")
plot(grid, results[(burnIn+1):(nIter+burnIn),1], type="l")

## Convergence seems good since markov chain is exploring full posterior a
nd have good mixing.
```

Assignment 4 – Stan, plot scatter plot & mean of posterior predictive distrib & 90 % equal tail intervals, stanmodel with heteroscedastic variance (different variance over time)

```r
## a) Use supplied stan model to do Bayesian inference. Draw 2000 posterio
r samples and use 500 for burnin.
## Produce figure with scatter plot, overlay curve for mean of posterior p
redictive distrib, in range [0,25].
## Also overlay curves 90 % equal tail interval for same posterior predict
ive distrib given values of x in range [0,25]

# Load data
cars = cars
```

```
library(rstan)
LinRegModel <- '
data {
  int<lower=0> N;
  vector[N] x;
  vector[N] y;
}
parameters {
  real alpha;
  real beta;
  real<lower=0> sigma2;
}
model {
  sigma2 ~ scaled_inv_chi_square(5,10);
  for (n in 1:N)
    y[n] ~ normal(alpha + beta * x[n], sqrt(sigma2));
}
'
x=cars$speed
y=cars$dist
nIter=2000
burnIn=500
N=dim(cars)[1]
data=list(N=N,x=x,y=y)
fit=stan(model_code=LinRegModel, data=data, iter=nIter, warmup = 500, chai
ns=1)
print(fit)
postDraws=extract(fit)
alpha_draws=postDraws$alpha
beta_draws=postDraws$beta
sigma_draws=postDraws$sigma2
xGrid=seq(0,25)
n=length(alpha_draws)
mean_credInt=matrix(0,length(xGrid),3)
count=1
for (i in 1:length(xGrid)) {
  ysim=rep(0,length(nIter-burnIn))
  ysim=alpha_draws+beta_draws*xGrid[i]+rnorm(nIter-burnin, mean=0, sd=sqrt
(sigma_draws))
  mean_credInt[count,1]=mean(ysim)
  mean_credInt[count,-1]=quantile(ysim, probs = c(0.05, 0.95))
  count=count+1
}

plot(x,y,xlab="Speed", ylab="Distance", col="blue", main="Plot for model w
ith constant sigma prior")
lines(xGrid, mean_credInt[,1], lwd=2, col="red")
lines(xGrid, mean_credInt[,2], lwd=1, lty=2)
lines(xGrid, mean_credInt[,3], lwd=1, lty=2)
legend("topleft", legend=c("Data", "Posterior mean", "90 % cred interval")
, col=c("blue", "red", "grey"),
       pch=c(1, NaN, NaN), lty=c(NaN, 1, 2), lwd=c(NaN, 2, 1))
```

```r
## b) Compute 95 % equal tail credible interval for alpha. Give real-world
interpret of the interval.

quantile(alpha_draws, probs=c(0.025, 0.975))

## The interpretation of the credible interval for alpha is that if the ca
r has no speed it travels a negative
## distance between -31 and 4.25 approximately with 95 % posterior probabi
lity. This is not realistic. To prevent this
## a prior can be set to alpha with a mean around zero which however would
make the linear prediction worse.
## One can also use the log Normal distribution for y to force it to have
a value above zero.

## c) Reproduce results in b) with heteroscadastic variance.

LinRegModel_hetero <- '
data {
  int<lower=0> N;
  vector[N] x;
  vector[N] y;
}
parameters {
  real alpha;
  real beta;
  real<lower=0> sigma2[N];
  real gamma;
  real phi;
}
model {
  for (n in 1:N)
    sigma2[n] ~ scaled_inv_chi_square(5,exp(gamma+phi*x[n]));
  for (n in 1:N)
    y[n] ~ normal(alpha + beta * x[n], sqrt(sigma2[n]));
}
'
data=list(N=N,x=x,y=y)
fit2=stan(model_code=LinRegModel_hetero, data=data, iter=nIter, warmup = 5
00, chains=1)
print(fit2)
postDraws2=extract(fit2)
alpha_draws=postDraws2$alpha
beta_draws=postDraws2$beta
sigma_draws=postDraws2$sigma2
xGrid=seq(0,25)
n=length(alpha_draws)
mean_credInt=matrix(0,length(xGrid),3)
count=1
for (i in 1:length(xGrid)) {
  rinv=rchisq(nIter-burnIn, 5)
  sigma_draw=5*exp(postDraws2$gamma + xgrid[i] * postDraws2$phi)^2/rinv
  ysim=rep(0,length(nIter-burnIn))
  ysim=alpha_draws+beta_draws*xGrid[i]+rnorm(nIter-burnin, mean=0, sd=sqrt
(sigma_draw))
```

```r
  mean_credInt[count,1]=mean(ysim)
  mean_credInt[count,-1]=quantile(ysim, probs = c(0.05, 0.95))
  count=count+1
}

plot(x,y,xlab="Speed", ylab="Distance", col="blue", main="Plot of model wi
th heteroscadastic sigma prior")
lines(xGrid, mean_credInt[,1], lwd=2, col="red")
lines(xGrid, mean_credInt[,2], lwd=1, lty=2)
lines(xGrid, mean_credInt[,3], lwd=1, lty=2)
legend("topleft", legend=c("Data", "Posterior mean", "90 % cred interval")
, col=c("blue", "red", "grey"),
       pch=c(1, NaN, NaN), lty=c(NaN, 1, 2), lwd=c(NaN, 2, 1))

## The new model seems to capture the data better than the old one.
```