Untitled

# 2019-10-31

# Compute the bayecian decision   
p\_up = 0.6  
p\_down = 1 - p\_up  
EU\_buy = p\_up\*30 + p\_down\*(-10)  
EU\_notBuy = p\_up\*90 + p\_down\*(-120)  
  
#EU\_byy > nnot byt => should buy  
  
p\_posterior\_up = 65/105  
p\_posterior\_down = 1-p\_posterior\_up  
  
EU\_buy2 = p\_posterior\_up\*30 + p\_posterior\_down\*(-10)  
EU\_notBuy2 = p\_posterior\_up\*90 + p\_posterior\_down\*(-120)  
  
  
#2  
n = nrow(Traffic)  
sumY = sum(Traffic$y)  
  
  
thetaGrid = seq(18,24,.01)  
alpha = 20  
beta = 1  
alpha\_posterior = sumY + alpha  
beta\_posterior = n + beta  
plot(thetaGrid, dgamma(thetaGrid, alpha\_posterior, beta\_posterior))  
  
prob = pgamma(21, alpha\_posterior, beta\_posterior)  
  
#b  
data = Traffic  
yes\_subset <- subset(data, limit == 'yes')  
no\_subset = subset(data, limit == 'no')  
  
draws\_yes = rgamma(5000, sum(yes\_subset$y) + alpha, nrow(yes\_subset) + beta)  
draws\_no = rgamma(5000, sum(no\_subset$y) + alpha, nrow(no\_subset) + beta)  
difference = draws\_no - draws\_yes   
hist(difference)  
  
mean(draws\_no>draws\_yes)  
#andelen av fall där no limit kommer att resultera att number off accidents (y ) är större än antalet accidents när speed limit fanns  
  
#tar antalet accidents när inget limit fanns \* 0.85 => ex 12 accidents \* 0.85. Kollar nu andelen där de fortfarande skulle vara större n där det fanns limit  
# Ser då att andelen minskar till 0.869. Det är är därmed rimligt att   
mean(draws\_no\*0.9>draws\_yes)  
  
#c  
set.seed(12345)  
x = 20  
lambda = 30  
alpha = 2  
beta = 2  
nDraws = 2000  
  
  
v\_draws = c(30)  
pi\_draws = c(0.5)  
pi\_draw = 1  
for (i in 1:nDraws){  
 #Compute Full conditional posterior (Normal model with conditionally conjugate prior)  
 #my <- rnorm(1, mean = myn, sd = sqrt(taonSquared))  
 z <- rpois(1, lambda\*(1-pi\_draw))  
 v = z + x  
 v\_draws = append(v\_draws, v)  
   
 pi\_draw = rbeta(1, alpha + x, beta + v -x)  
 pi\_draws = append(pi\_draws, pi\_draw)  
}  
  
v\_draws = v\_draws[500:nDraws]  
pi\_draws = pi\_draws[500:nDraws]  
par(mfrow=c(2,1))  
#plot(v\_draws, type = 'l')  
#plot(pi\_draws, type = 'l')  
  
hist(pi\_draws[500:nDraws], 30)  
hist(v\_draws[500:nDraws], 30)  
  
set.seed(1235)  
# start values  
burnin = 500  
niter = 2000  
nu <- 30  
pi <- .5  
nu\_vec <- rep(0,burnin+niter)  
pi\_vec <- rep(0,burnin+niter)  
nu\_vec[1] <- nu  
pi\_vec[1] <- pi  
for(i in 2:(burnin+niter)){  
 z <- rpois(1,lambda\*(1-pi))  
 nu = z + x  
 nu\_vec[i] <- nu  
 pi <- rbeta(1,alpha+x,beta+nu-x)  
 pi\_vec[i] <- pi  
}  
  
  
par(mfrow=c(2,1))  
hist(nu\_vec[(burnin+1):(burnin+niter)],30,prob=TRUE,main="Posterior",xlab = "nu")  
hist(pi\_vec[(burnin+1):(burnin+niter)],30,prob=TRUE,main="Posterior",xlab = "pi")  
par(mfrow=c(2,1))  
plot(nu\_vec[(burnin+1):(burnin+niter)],type="l",main="Traceplot",ylab = "nu",xlab="Iteration")  
plot(pi\_vec[(burnin+1):(burnin+niter)],type="l",main="Traceplot",ylab = "pi",xlab="Iteration")  
  
#The Markov chain seems to have good mixing, since it rapidly explores the posterior, so the convergence is good.  
  
  
#4  
  
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));  
}'  
  
DataRStan<-  
 list(N = nrow(cars),  
 x = cars$speed,  
 y = cars$dist)   
  
fit\_Model<-stan(model\_code=LinRegModel,  
 data=DataRStan,  
 warmup=500,  
 iter=2000,  
 chains = 1)  
  
print(fit\_Model,digits=4)  
res<-extract(fit\_Model)  
res  
  
  
plot(DataRStan)  
plot(cars)  
  
xGrid = seq(0,25)  
y\_prediction\_mean = c()  
lower = c()  
upper = c()  
for (x in xGrid){  
 ypred = res$alpha + res$beta\*x + rnorm(1500, 0, sqrt(res$sigma2))  
 y\_prediction\_mean = append(y\_prediction\_mean, mean(ypred))  
 qc = quantile(ypred, probs = seq(0,1, 0.05))  
 lower = append(lower, qc[2])  
 upper = append(upper, qc[20])  
}  
  
qc = quantile(ypred, probs = seq(0,1, 0.05))  
qc[20]  
lines(y\_prediction\_mean)  
lines(upper)  
lines(lower)  
  
quantile(res$alpha, probs = c(0.05, 0.95))  
res$sigma2  
lines(res$sigma2)  
  
  
  
#C  
  
LinRegModel2 <- '  
data {  
 int<lower=0> N;  
 vector[N] x;  
 vector[N] y;  
}  
parameters {  
 real alpha;  
 real beta;  
 real gamma;  
 real phi;  
 real<lower=0> sigma2[N];  
}  
model {  
   
 for (n in 1:N){  
 sigma2[n] ~ scaled\_inv\_chi\_square(5, exp(gamma + phi \* x[n]));  
 y[n] ~ normal(alpha + beta \* x[n], sqrt(sigma2[n]));  
 }  
}'  
  
fit\_Model2<-stan(model\_code=LinRegModel2,  
 data=DataRStan,  
 warmup=500,  
 iter=2000,  
 chains = 1)  
  
print(fit\_Model2,digits=4)  
res<-extract(fit\_Model2)  
res$sigma2  
res  
res  
  
plot(DataRStan)  
plot(cars)  
  
xGrid = seq(0,25)  
y\_prediction\_mean = c()  
lower = c()  
upper = c()  
for (x in xGrid){  
 ypred = res$alpha + res$beta\*x + rnorm(1500, 0, sqrt(res$sigma2))  
 y\_prediction\_mean = append(y\_prediction\_mean, mean(ypred))  
 qc = quantile(ypred, probs = seq(0,1, 0.05))  
 lower = append(lower, qc[2])  
 upper = append(upper, qc[20])  
}  
  
qc = quantile(ypred, probs = seq(0,1, 0.05))  
qc[20]  
lines(y\_prediction\_mean)  
lines(upper)  
lines(lower)  
  
quantile(res$alpha, probs = c(0.05, 0.95))  
res$sigma2  
lines(res$sigma2)

#2019-08-21

nCovs = dim(X)[2]  
mu\_0 = rep(0,nCovs)  
Omega\_0 = (1/100)\*diag(nCovs)  
v\_0 = 1  
sigma2\_0 = 5^2   
  
joint\_posterior = BayesLinReg(y, X, mu\_0, Omega\_0, v\_0, sigma2\_0, nIter = 5000)  
betas = joint\_posterior$betaSample  
sigmas = joint\_posterior$sigma2Sample  
  
  
Bmean = colMeans(joint\_posterior$betaSample)  
Bq025 = apply(joint\_posterior$betaSample,2,quantile,.025)  
Bq975 = apply(joint\_posterior$betaSample,2,quantile,.975)  
print(data.frame(round(cbind(Bmean,Bq025,Bq975),3)),row.names=covNames)#B compute density   
sigma\_density = density(joint\_posterior$sigma2Sample)  
plot(sigma\_density)  
summary(sigma\_density)  
  
  
sorted\_normalized\_y = sort(sigma\_density$y, decreasing = TRUE)/sum(sigma\_density$y)  
sorted\_x = sigma\_density$x[order(-sigma\_density$y)]  
  
  
count = 0  
summa = 0  
while(summa <= 0.95){  
 count = count + 1  
 summa = sorted\_normalized\_y[count] + summa  
}  
a = min(sorted\_x[1:count-1])  
b = max(sorted\_x[1:count-1])  
  
mode = sorted\_x[which.max(sorted\_normalized\_y)]  
  
XNewHouse  
nSim <- dim(joint\_posterior$betaSample)[1]  
  
ypred = c()  
  
for(i in 1:5000){  
 ypred = append(ypred, XNewHouse%\*%betas[i,]) + rnorm(1, 0, sqrt(sigmas[i]))  
}  
  
sum(ypred>20)/nSim  
  
#2  
alpha = 1  
beta = 1  
n = 5  
sumx = 65  
nsim = 5000  
  
theta = rbeta(5000, n + alpha, sumx + beta)  
predicted\_observations = rgeom(5000, theta)  
  
#Using quantile extraciting 0.95 we get the value for x where we with 95% probability can say that the next earthwake will have occured  
quantile(predicted\_observations, 0.95)  
  
  
# 3  
yData  
xData  
m = length(xData)  
sumx = sum(xData)  
log.posterior = function(xData, yData, theta){  
 m = length(xData)  
 sumx = sum(xData)  
 log.theta\_given\_x = dgamma(theta, m + 3, sumx + 2, log = TRUE)  
 #log.likelihood = sum(-3\*log(1+(1/5)\*(yData-log(theta))^2))  
 log.likelihood = sum(dt(yData-log(theta), 5, log = TRUE))  
 log.post = log.theta\_given\_x + log.likelihood  
 return(exp(log.post))  
}  
  
posterior = c()  
thetaGrid = seq(0,2, 0.01)  
for(theta in thetaGrid){  
 posterior = append(posterior, log.posterior(xData, yData, theta))  
}  
  
  
posterior\_normalized = (1/0.01)\*posterior/sum(posterior)  
plot(thetaGrid, posterior\_normalized)  
  
logdNegBin <- function(x,mu,phi){  
 lchoose(x+phi-1,x) + x\*log(mu/(mu+phi)) + phi\*log(phi/(mu+phi))  
}  
  
log.posterior = function(param, x){  
 #initval[1] = my  
 #initval[2] = phi  
 #temp = factorial(x+phi-1)/(factorial(phi-1)\*factorial(x))\*(x\*log(my/(my+phi))\*phi\*log(phi/(my+phi)))  
 theta1 = param[1]  
 theta2 = param[2]  
 logPost = sum(logdNegBin(x, theta1, theta2)) - 2\*log(theta2)  
 return(logPost)  
 #return(temp)  
}  
  
  
#  
  
library(mvtnorm)  
metropolis = function(n, c, initval, hessian, posterior\_density, x){  
   
 # this step depends on previous position. Previous position becomes this turns mean.   
 proposal\_draws\_previous = initval;  
   
 acceptedDraws = matrix(0, ncol=2,nrow=n)  
 accprobvec <- rep(0,n)  
   
 set.seed(12345)  
 for(i in 1:n){  
 # draws (theta\_p) from the proposal distribution ~ N(theta\_p-1, c\*hessian)  
 proposal\_draws = rmvnorm(1, proposal\_draws\_previous, c\*hessian)  
 proposal\_draws[proposal\_draws <= 0] = 1e-6  
 # create a ratio depending on if this draw is better than previous, take exp to remove logarithm (logposterior)  
 # posterior\_density = log.posterior => exp of the division => logA -logB   
 acceptance\_ratio = min(1,exp(posterior\_density(proposal\_draws, x)-posterior\_density(proposal\_draws\_previous, x)))  
 # draw a random uniformed variable to compare wiht acceptance ratio  
 random\_acceptance = runif(1,0,1)  
 # if acceptance ratio is bigger than random variable than we move to the new position, otherwise we stay  
 accprobvec[i] <- min(acceptance\_ratio,1)  
 if(acceptance\_ratio >= random\_acceptance){  
 proposal\_draws\_previous = proposal\_draws  
 params = proposal\_draws  
 }  
 acceptedDraws[i,] = params  
   
 }  
 return(list(draws = acceptedDraws, prob = accprobvec))  
}  
  
c = 0.1  
initval =c(200,20)  
hessian = diag(100,2)  
hessian = c\*diag(c(100,5))  
x = incidents$incidents  
  
testDraws = metropolis(10000, c=0.1, initval, hessian, log.posterior, x)  
  
for(i in 1:2){  
 plot(testDraws[,i], type='s')  
 #a = c(rep(betas\_posterior[i],nrow(testDraws)))  
 #lines(a, col='red')  
}