Untitled

# 191031

# 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)