lab3_code

Akshath Srinivas (akssr921) Yaning Wang (yanwa579)

```
# Question 1

#1 a
#loading data
data<-readRDS('Precipitation.rds')
ln_y<-log(data)
n<-length(ln_y)

#setting prior parameter values
mu0<-0
tau0<-1
nu0<-1
sigma0<-1

#mean and variance from data
x_bar<-mean(ln_y)
sigma2<-var(ln_y)

#initializing iterations and mu and sigma 2 samples(empty vector to store)
ndraws <- 1000
mu_samples<-rep(NA,ndraws)
sigma2_samples <- numeric(ndraws)

#gibbs sampling for ndraws(number of iteration)
for (i in 1:ndraws) {

  #calculating tau_n and mu_n to sample mu from normal distribution
  taun<-1/((n/sigma2)+(1/tau0))
  W<-(n/sigma2)/((n/sigma2)+(1/tau0))
  mun<-W*x_bar+(1-W)*mu0
  mu <- rnorm(1, mun, sqrt(taun))
  mu_samples[i] <- mu

  #calculating nu_n and sigma_n to sample sigma^2 from inverse chi distribution
  nun<-nu0+n
  sigman<-(nu0*sigma0+sum((ln_y-mu)^2))/nun
  sigma2<-LaplacesDemon::rinvchisq(1,df=nun,scale=sigman)
  sigma2_samples[i]<-sigma2
}

# Calculate the Inefficiency Factors (IFs) for mu_samples
acf_mu_IF <- 1+2*sum(acf(mu_samples,plot = FALSE)$acf[-1])

# Calculate the Inefficiency Factors (IFs) for sigma^2_samples
```

```r
acf_sigma_IF <- 1+2*sum(acf(sigma2_samples,plot = FALSE)$acf[-1])

cat(paste("The Inefficiency Factor of mu", acf_mu_IF))
cat(paste("The Inefficiency Factor of sigma^2", acf_sigma_IF))

#plotting drawn mu and sigma^2
plot(mu_samples, type="l", ylab="mu", xlab="iteration")
plot(sigma2_samples, type="l", ylab="sigma^2", xlab="iteration")


#1b

#plotting the kernel density of data
plot(density(exp(ln_y)),lwd = 4, col='blue',main = 'Original data vs Posterior simulated data',xlab='sa

#simulating posterior predictive samples from the parameteres we got from above question
simulated_posterior<-numeric(length(sigma2_samples))
for (i in 1:length(sigma2_samples)) {
  simulated_posterior[i]<-rnorm(1,mu_samples[i],sigma2_samples[i])
}

#drawing the posterior predictive density line on the existing plot
lines(density(exp(simulated_posterior)),col='green',lwd=3)

legend("topright",legend=c("Original data",'Sampled values'),col=c('blue','green'),
       lty = c(1, 1), cex=0.4)


# Question 2
#(a)
# Load data set
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))
eBay <- read.table("eBayNumberOfBidderData.dat", header = TRUE)
# delete the covariate const column
data<-eBay[,-2]
model<-glm(nBids~., family = poisson,data=data)
summary(model)
#From the result we got, and base on the P value, we can conclude that:VerifyID,
#Sealed,MajBlem,LogBook,MinBidShare. Their p values are all in the interval[0,0.05]

#(b)

# Extract the response variable and features
y<-as.matrix(eBay[,1])
x<-as.matrix(eBay[,-1])
# get the number of observations and features
n<-length(y)
n_fea<-dim(x)[2]
#get the prior variables
mu<-as.matrix(rep(0,n_fea))
sigma<-100*solve((t(x)%*%x))
#return log posterior for the poisson regression
log_postlogis <- function(betas){
```

```
  lambda <-exp(x%*%betas)
  #use the log_likelihood formula we get above
  logLik <- sum(y*log(lambda) - lambda)-sum(log(factorial(y)))
  #write the prior formula
  logPrior <- dmvnorm(t(betas), mean=mu, sigma=sigma, log=TRUE)
  return(logLik + logPrior)
}
#initial betas
init_betas <- matrix(0,n_fea,1)
#use optim function to get the posterior mode value and hessian value
OptimRes <- optim(par=init_betas,
                  fn=log_postlogis,
                  gr=NULL,
                  method=c("BFGS"),
                  control=list(fnscale=-1),
                  hessian=TRUE)

#Naming the coefficient by features
Xnames<-colnames(x)
posterior_mode<-data.frame(feature_name=Xnames,beta_mode=OptimRes$par)
print('The posterior mode is:')
print(posterior_mode)
approxpost_variance <- solve(-OptimRes$hessian)
print('The variance of the posterior is')
print(approxpost_variance)

#(c)
#set the number of draws
m<-5000
RWMSampler<-function(c){
  #set the start betas value
  sample_beta<-matrix(0,nrow = m,ncol = n_fea)
  start_betas<-sample_beta[1,]
  variance<-c*approxpost_variance
  for(i in 1:m){
    #generate new betas
    betas_new<-rmvnorm(1, mean=betas, sigma=variance)
    #change betas_new vector to matrix form with 9 rows and compute the posterior density
    postdensi_new<-log_postlogis(t(betas_new))
    postdensi_old<-log_postlogis(betas)
    #compute the ratio of posterior densities
    ratio<-exp(postdensi_new-postdensi_old)
    #compute the acceptance probability
    a<-min(1,ratio)
    u<-runif(1)
    if(u<=a){
      sample_beta[i,]<-betas_new
      }
    else{
      sample_beta[i,]<-betas
    }
    betas<-sample_beta[i,]
  }
```

```
  return(sample_beta)
}


# Now we should compare the results with the approximate results in b
RWMdraws<-RWMSampler(0.5)
mean_betas<-colMeans(RWMdraws)
compare_data<-data.frame(feature_name=Xnames,approx_b=OptimRes$par,RWM_c=mean_betas)
print(compare_data)
#From the table above, we can see the results obtained by these two methods are very close.


#The values we simulate from the actual posterior of beta using the Metropolis
# algorithm has 9 dimensions, we randomly use the first and second dimension to check if they are conve

data<-data.frame(number=c(1:m),beta1=RWMSampler(0.5)[,1],beta5=RWMSampler(0.5)[,5])

ggplot(data)+
  geom_line(aes(x=number,y=beta1),colour="blue")+
  xlab("the number of draws")+
  ylab("the simulate draws of beta1")

ggplot(data)+
  geom_line(aes(x=number,y=beta5),colour="blue")+
  xlab("the number of draws")+
  ylab("the simulate draws of beta5")

  #From the plot we can see, the values of betas are convergence, They are all around a certain value.

#(d)

#RWMdraws is the samples we get from c,and m is the rows of RWMdraws
xnew_withconst<-matrix(c(1,1,0,1,0,1,0,1.2,0.8),1,n_fea)
lambda<-exp(xnew_withconst%*%t(RWMdraws))
#use the build in function rpois to simulate the number of bidders
bids<-c()
for(i in 1:m){
  bid<-rpois(1, lambda[i])
  bids[i]<-bid
}
#plot the predictive distribution
hist(bids,main="the predict distribution")
cat("The probability of no bidders in this new auction is ",length(which(bids==0))/m)

#question3
library(rstan)
library(ggplot2)
#(a)

SimuData<-function(mu,phi,sigma2,T){
  sample<-c()
  x<-mu
  sample[1]<-x
```

```
  for(t in 2:T){
    #sample from normal distribution to get epsilon value
    epsilon<-rnorm(1,mean=0,sd=sqrt(sigma2))
    #base on the expression to get the new sample
    x_new<-mu+phi*(x-mu)+epsilon
    #collect the samples
    sample<-append(sample,x_new)
    x<-x_new
  }
  return(sample)
}

#get some phi values between -1 and 1
phis<-seq(-1,1,by=0.4)
n<-length(phis)
T<-300
data<-data.frame()
mean_sample<-c()
val_sample<-c()
#simulate samples for different phi values
for(i in 1:n){
  sample<-SimuData(mu=13,phi=phis[i],sigma2=3,T=T)
  mean_sample<-c(mean_sample,mean(sample))
  val_sample<-c(val_sample,var(sample))
  name<-rep(paste0("phi=",phis[i]),T)
  samples<-data.frame(number=c(1:300),sample=sample,phi=name)
  data<-rbind(data,samples)
}

cat("The mean of the samples base on different phi is ",mean_sample, "\n")
cat("The variance of the samples base on different phi is ",val_sample, "\n")
#plot the samples according to different phi
ggplot(data=data)+
  geom_point(aes(x=number,y=sample,color=phi))+
  labs(title="Learned NN on the test data")

#From the plot, mean and variance we got, the samples simulated based on phi between -1 and 1(not inclu
#This also proves what is said in the question that in this interval the AR process is stationary.

#(b)
### i
#draw samples from the function
samplex<-SimuData(mu=13,phi=0.2,sigma2=3,T=T)
sampley<-SimuData(mu=13,phi=0.95,sigma2=3,T=T)

StanModel <- '
data{
  int <lower=0> N;  //number of observations,it should be positive
  vector[N] y;  //obeservations at time N
}
parameters{
  real mu;
  real <lower=-1,upper=1> phi; //the constraint for phi to ensure time series stationary
```

```
  real <lower=0> sigma2; //the variance should be positive(from teachers slide)
}
model{
  mu ~ normal(0,100); // Normal prior(from teachers slide)
  phi ~ uniform(-1,1); // uniform prior
  sigma2 ~ scaled_inv_chi_square(1,2); // Scaled-inv-chiprior(from teachers slide)
  for (n in 1:N){
    y[n] ~ normal(mu + phi * (y[n-1]-mu), sqrt(sigma2));
  }
}'

datax <- list(N=300, y=samplex)
datay <- list(N=300, y=sampley)
# number of warmup iterations per chain
nwarmup <- 1000
# total number of iterations per chain
niter <- 2000
# number of Markov chains
nchains<-4
fitx <- stan(model_code=StanModel,data=datax, warmup=nwarmup,iter=niter,chains=nchains)
fity <- stan(model_code=StanModel,data=datay, warmup=nwarmup,iter=niter,chains=nchains)
print("The required values using data x are:")
print(summary(fitx)$summary[1:3,c(1,4,8,9)])
print("The required values using data y are:")
print(summary(fity)$summary[1:3,c(1,4,8,9)])
#the post means compared with the true mean using x data
post_valuex<-as.data.frame(summary(fitx)$summary[1:3,c(1,4,8,9)])
post_valuex["true_value"]<-c(13,0.2,3)
print(post_valuex)
#the post means compared with the true mean using y data
post_valuey<-as.data.frame(summary(fity)$summary[1:3,c(1,4,8,9)])
post_valuey["true_value"]<-c(13,0.95,3)
print(post_valuey)
#From the results above, we can see the true values of the three parameters are very close to the poste
# And the true values are all in the 95% credible interval which are between 2.5% and 97.5% (two equal
#So using this way we can estimate the true value.

### ii

# Do automatic traceplots of all chains of model fitx
traceplot(fitx)
#From the traceplot of all chains of the three parameters, we can see in model fitx the parameters are
# Do automatic traceplots of all chains of model fity
traceplot(fity)
#From the traceplot of all chains of the three parameters, we can see the in model fity phi and sigma2

#(ii) plot the joint posterior of mu and phi,we have the samples of mu from the posterior marginal dens

#Extract posterior samples in fitx and fity models.
x_mu<-extract(fitx)[[1]]
x_phi<-extract(fitx)[[2]]
y_mu<-extract(fity)[[1]]
y_phi<-extract(fity)[[2]]
```

```r
plot(x_mu,x_phi,xlab="mu",ylab="phi",main="the joint posteror of mu and phi using data x")
plot(y_mu,y_phi,xlab="mu",ylab="phi",main="the joint posteror of mu and phi using data y")
#check the posterior distribution
#pairs(fitx)
#pairs(fity)

#We know the posterior distribution of mu and phi in model fitx are normal distribution,
#the posterior distribution of phi in model fity is also normal distribution. But mu is not.
# so the joint distribution in model fitx is normal but in model fity is a pecial shape.
```