ComputerSol

732A91 Bayesian Learning, Linköping University

Dimitra Muni - @dimmu472

19 August 2020

Patients

(a)

```
# (a)


theta=rnorm(10000,10000,500)
cred_interval=quantile(theta,probs = c(0.05,0.95))
cat('\n 90% equal tail credible interval \n',cred_interval)
```

```
##
##  90% equal tail credible interval
##  9181.792 10816.63
```

(b)

```
pred=numeric(length = length(theta))
sum(dpois(1,lambda = theta))
```

```
## [1] 0
```

```
table(pred)
```

```
## pred
##     0
## 10000
```

(c)

Regression

(a)

```
# Defining a function that simulates from the scaled inverse Chi-square distribution
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://mattiasvillani.com
  #
  # INPUTS:
  #   y - n-by-1 vector with response data observations
  #   X - n-by-nCovs matrix with covariates, first column should be ones if 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$betaSample     - Posterior sample of beta.     nIter-by-nCovs matrix
  #   results$sigma2Sample   - Posterior sample of sigma2.   nIter-by-1 vector

  # 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%*%mu_0 - t(mu_n)%*%Omega_n%*%mu_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))
}
```

```
# Model M1
## Prior
mu_0 = rep(0,1)
Omega_0 = 0.001*diag(1)
v_0 = 1
sigma2_0 = 10
nIter=5000
#No intercept term in M1 as per formulation
```

```
M1_draws=BayesLinReg(y=muscle$Length,X=as.matrix(muscle$Conc),
            mu_0, Omega_0, v_0, sigma2_0, nIter)


# Model M1
## Prior
mu_0 = rep(0,3)
Omega_0 = 0.001*diag(3)
v_0 = 1
sigma2_0 = 10
nIter=5000

M2_draws=BayesLinReg(y=muscle$Length,X=model.matrix(~Conc+I(Conc**2),muscle),
            mu_0, Omega_0, v_0, sigma2_0, nIter)
```

```
#For M1
quantile(M1_draws$betaSample[,1],probs = c(0.025,0.975))
```

```
##      2.5%     97.5%
##  9.948169 12.721394
```

```
#For M2

quantile(M2_draws$betaSample[,1],probs = c(0.025,0.975))
```

```
##     2.5%    97.5%
## 1.421836 8.534723
```

```
quantile(M2_draws$betaSample[,2],probs = c(0.025,0.975))
```

```
##     2.5%    97.5%
## 14.31128 23.29246
```

```
quantile(M2_draws$betaSample[,3],probs = c(0.025,0.975))
```

```
##      2.5%     97.5%
## -4.503825 -2.319801
```

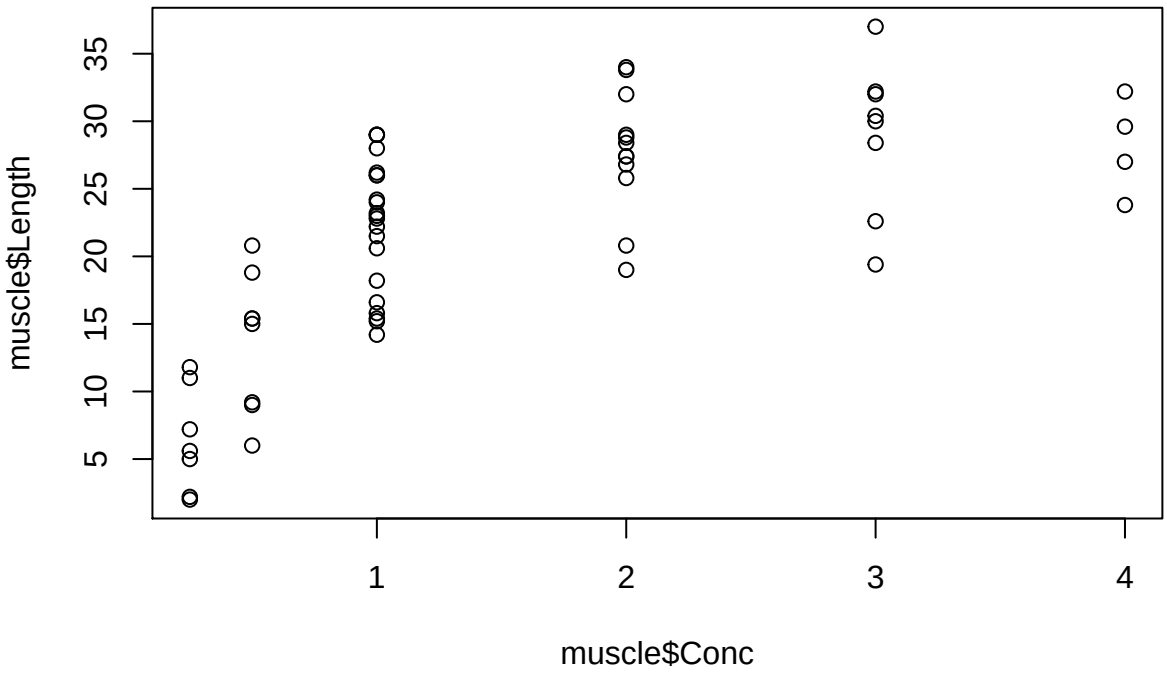
(b)

```
xgrid=seq(0,5,0.01)
for (i in 1:5000) {
  M1_length=M1_draws$betaSample[i,1]%*%muscle$Conc + rnorm(1,mean =0 ,sd=sqrt(M1_draws$betaSample[i,1]))

}



plot(x=muscle$Conc,y=muscle$Length)
```



Delivery Time

(a) On Paper

(b)

```
load("~/bayesian_exam_practice/delivery.RData")
n=length(delivery)
alpha=2
beta=2

post_modes=numeric(length = 3)
post_lambda=matrix(nrow = 1000,ncol=3)
##k=0.5
k=0.5
x=delivery
post_lambda[,1]=1/rgamma(1000,alpha+n,beta+sum(x**k))
d=density(post_lambda[,1])
cat('For k=0.5,Posterior mode using density() ',d$x[which.max(d$y)])
```

```
## For k=0.5,Posterior mode using density()  1.612669
```

```
post_modes[1]=d$x[which.max(d$y)]
#theoretical posterior model= beta/(alpha+1)
cat('\nFor k=0.5,Theoretical Posterior mode',d$x[which.max(d$y)])
```

```
##
## For k=0.5,Theoretical Posterior mode 1.612669
```

```
k=1.5
x=delivery
post_lambda[,2]=1/rgamma(1000,alpha+n,beta+sum(x**k))
d=density(post_lambda[,2])
```

```r
cat('\nFor k=1.5,Posterior mode using density() ',d$x[which.max(d$y)])
```

```
##
## For k=1.5,Posterior mode using density()  5.469534
```

```r
post_modes[2]=d$x[which.max(d$y)]
#theoretical posterior model= beta/(alpha+1)
cat('\nFor k=1.5,Theoretical Posterior mode',d$x[which.max(d$y)])
```

```
##
## For k=1.5,Theoretical Posterior mode 5.469534
```

```r
k=2.5
x=delivery
post_lambda[,3]=1/rgamma(1000,alpha+n,beta+sum(x**k))
d=density(post_lambda[,3])
cat('\nFor k=2.5,Posterior mode using density() ',d$x[which.max(d$y)])
```

```
##
## For k=2.5,Posterior mode using density()  27.15098
```

```r
post_modes[3]=d$x[which.max(d$y)]
#theoretical posterior model= beta/(alpha+1)
cat('\nFor k=2.5,Theoretical Posterior mode',d$x[which.max(d$y)])
```

```
##
## For k=2.5,Theoretical Posterior mode 27.15098
```

(c)

```r
hist(delivery,freq = FALSE,xlim=c(0,40))
hist(post_lambda[,1],add=T,col='dark grey',freq = FALSE)
abline(v=post_modes[1],col='red',lwd=4)
hist(post_lambda[,2],add=T,col=3,freq = FALSE)
abline(v=post_modes[2],col='blue',lwd=4)
hist(post_lambda[,3],add=T,col=4,freq = FALSE)
abline(v=post_modes[3],col='dark green',lwd=4)
legend('topright',c('k=0.5','k=1.5','k=2.5'),col=c('red','blue','dark green'),lwd=4)
```

From the figures above, k=1.5 seems to give better fit than k=0.5 and k=2.5, however the denity is has lower variance than the original data.

More Delivery Times

(a) On paper

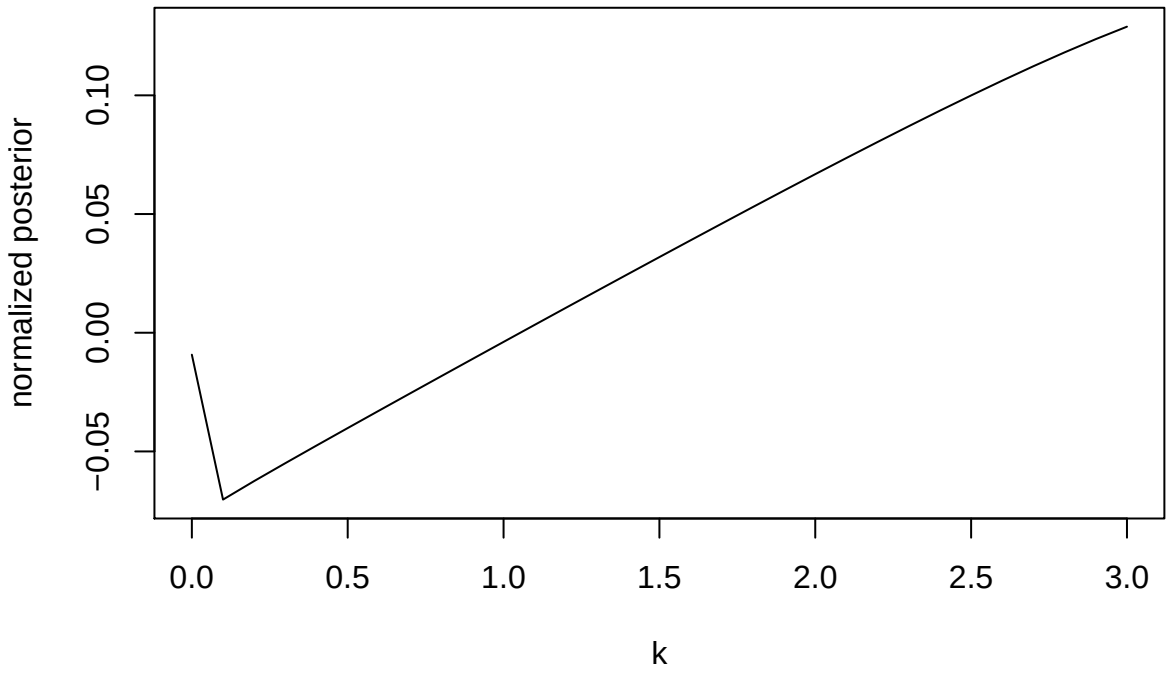
(b)

```
logpostk=function(x,k,alpha,beta){
 n=length(x)
 log_prior_k=dexp(k,rate=1,log = TRUE)
 lambdas=1/dgamma(x,alpha,beta)
 log_prior_lambda=log(lambdas)
 log_likelihood= n*log(k)-(n*log_prior_lambda)+(n*(k-1)*sum(log(x)))-(sum(x**k)/lambdas)
 return(sum(log_prior_k+log_prior_lambda+log_likelihood))
}

xgrid=seq(0,3,0.1)
posterior_dist=c()
for (k in xgrid) {
  posterior_dist=c(posterior_dist,logpostk(x=delivery,k,alpha = 2,beta = 2))

}
posterior_dist[abs(posterior_dist)==Inf]=-100000

plot(x=xgrid,y=posterior_dist/sum(posterior_dist),type='l',xlab='k',ylab='normalized posterior')
```



(c)

```
normalized=posterior_dist/sum(posterior_dist)
band=quantile(normalized,probs = c(0.025,0.975))
plot(xgrid,normalized)
abline(v=band[1])
abline(v=band[2])
```