Bayesian Analysis of Categorical Data - A Revisit

Samyajoy Pal, M. Subbiah, M. R. Srinivasan

Web Appendix for Technical Report

A comprehensive study of two-dimensional categorical data from a Bayesian perspective has been attempted in this study. Two categorical variables with levels I and J are considered with the assumption that the underlying model follows multinomial distribution. The entire study embraces the essential components of Bayesian approach; prior construction, computations, and appropriate inferences using posterior distributions of the parameters. Conjugate prior distribution with symmetric and asymmetric hyper parameters are considered.Point and Interval estimation along with different types of probability computation from posterior distribution have been done using closed form integration, Monte-Carlo integration and MCMC methods. A Data on voters' preference for the 2016 presidential election has been collected from GSS.Bayesian computation has been done using above techniques. The codes and outputs are displayed below.

The Packages Required

```
knitr::opts_chunk$set(echo = TRUE)
library(MCMCpack)
library(pracma)
require(R2WinBUGS)
require(coda)
```

Data Model

```
knitr::opts_chunk$set(echo = TRUE)
x=c(5203,   3966,   5610,   3862,1685,3288) #Given Multinomial data
k=length(x)

n_r=100000      ##NO OF RUNS FOR SIMULATION OF POSTERIOR

po_un=x+1       ##POSTERIOR BASED ON UNIFORM PRIOR ON DIRIC PARAMETERS
po_jf=x+0.5     ##POSTERIOR BASED ON JEFFREYS PRIOR ON DIRIC PARAMETERS

ac=runif(k,0,1) #ARBITRARY CHOICE ON DIRICHLET PARAMETERS
po_ac=x+ac  ##POSTERIOR BASED ON ARBITRARY CHOICE ON DIRICHLET PARAMETERS

dr1=rdirichlet(n_r,po_un) #POSTERIOR SIMULATION BASED ON UNIFORM PRIOR
a1=0
l1=0
u1=0
dif1=0
for(j in 1:k)
{
  a1[j]=round(mean(dr1[,j]),4)      #POINT ESTIMATE
  l1[j]=round(quantile(dr1[,j],0.025),4) #INTERVAL ESTIMATES
  u1[j]=round(quantile(dr1[,j],0.975),4)
  dif1[j]=u1[j]-l1[j]                    #LENGTH OF INTERVAL
}
```

```r
dr2=rdirichlet(n_r,po_jf)           #POSTERIOR SIMULATION BASED ON JEFFREYS PRIOR
a2=0
l2=0
u2=0
dif2=0
for(j in 1:k)
{
  a2[j]=round(mean(dr2[,j]),4)                    #POINT ESTIMATE
  l2[j]=round(quantile(dr2[,j],0.025),4)          #INTERVAL ESTIMATES
  u2[j]=round(quantile(dr2[,j],0.975),4)
  dif2[j]=u2[j]-l2[j]                               #LENGTH OF INTERVAL
}


dr3=rdirichlet(n_r,po_ac)               #POSTERIOR SIMULATION BASED ON JEFFREYS PRIOR
a3=0
l3=0
u3=0
dif3=0
for(j in 1:k)
{
  a3[j]=round(mean(dr3[,j]),4)               #POINT ESTIMATE
  l3[j]=round(quantile(dr3[,j],0.025),4)     #INTERVAL ESTIMATES
  u3[j]=round(quantile(dr3[,j],0.975),4)
  dif3[j]=u3[j]-l3[j]                            #LENGTH OF INTERVAL
}
```

Monte-Carlo Simulation

Point and Interval Estimation Using Monte-Carlo Simulation

```r
knitr::opts_chunk$set(echo = TRUE)

#-------------------------------------------------------------------------------
#RESULTS
ans_PE=cbind(a1,a2,a3)
colnames(ans_PE)=c("Unif","Jeff","ArCh")
ans_CI=cbind(cbind(l1,u1),cbind(l2,u2),cbind(l3,u3))
colnames(ans_CI)=c("Un_LL","Un_UL","Je_LL","Je_UL","Ac_LL","Ac_UL")
ans_PR=cbind(prod(dif1),prod(dif2),prod(dif3))
colnames(ans_PR)=c("Unif","Jeff","ArCh")

ans_PE
```

```
##        Unif   Jeff   ArCh
## [1,] 0.2203 0.2203 0.2203
## [2,] 0.1680 0.1679 0.1679
## [3,] 0.2376 0.2376 0.2376
## [4,] 0.1636 0.1636 0.1636
## [5,] 0.0714 0.0714 0.0714
## [6,] 0.1392 0.1393 0.1393
```

```r
ans_CI
```

```
##       Un_LL  Un_UL  Je_LL  Je_UL  Ac_LL  Ac_UL
## [1,] 0.2150 0.2256 0.2151 0.2257 0.2150 0.2256
## [2,] 0.1632 0.1728 0.1632 0.1728 0.1632 0.1727
## [3,] 0.2321 0.2430 0.2321 0.2430 0.2322 0.2430
## [4,] 0.1589 0.1683 0.1588 0.1683 0.1588 0.1683
## [5,] 0.0681 0.0747 0.0681 0.0746 0.0681 0.0747
## [6,] 0.1349 0.1437 0.1349 0.1437 0.1349 0.1437
```

```r
ans_PR
```

```
##              Unif         Jeff         ArCh
## [1,] 6.055612e-13 6.027306e-13 6.000721e-13
```

```r
#___________________________
```

Probability From Posterior Distribution

```r
knitr::opts_chunk$set(echo = TRUE)
#This section illustrates how probabilities can be computed from the posterior simulation

#1a. p(theta1>theta2) based on three priors
length(which(dr1[,1]>dr1[,2]))/n_r
```

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

```r
length(which(dr2[,1]>dr2[,2]))/n_r
```

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

```r
length(which(dr3[,1]>dr3[,2]))/n_r
```

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

```r
#1b. p(theta4>theta5) based on three priors
length(which(dr1[,4]>dr1[,5]))/n_r
```

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

```r
length(which(dr2[,4]>dr2[,5]))/n_r
```

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

```r
length(which(dr3[,4]>dr3[,5]))/n_r
```

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

```r
#1c. p(theta1>theta4) based on three priors
length(which(dr1[,1]>dr1[,4]))/n_r
```

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

```r
length(which(dr2[,1]>dr2[,4]))/n_r
```

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

```r
length(which(dr3[,1]>dr3[,4]))/n_r
```

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

```r
#1d. p(theta2>theta5) based on three priors
length(which(dr1[,2]>dr1[,5]))/n_r
```

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

```r
length(which(dr2[,2]>dr2[,5]))/n_r
```

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

```r
length(which(dr3[,2]>dr3[,5]))/n_r
```

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

```r
#1e. p(theta3>theta6) based on three priors
length(which(dr1[,3]>dr1[,6]))/n_r
```

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

```r
length(which(dr2[,3]>dr2[,6]))/n_r
```

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

```r
length(which(dr3[,3]>dr3[,6]))/n_r
```

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

```r
#2a. p(theta2+theta3>theta1) based on three priors
length(which((dr1[,2]+dr1[,3])>dr1[,1]))/n_r
```

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

```r
length(which((dr2[,2]+dr2[,3])>dr2[,1]))/n_r
```

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

```r
length(which((dr3[,2]+dr3[,3])>dr2[,1]))/n_r
```

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

```r
#2b. p(theta5+theta6>theta4) based on three priors
length(which((dr1[,5]+dr1[,6])>dr1[,4]))/n_r
```

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

```r
length(which((dr2[,5]+dr2[,6])>dr2[,4]))/n_r
```

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

```r
length(which((dr3[,5]+dr3[,6])>dr2[,4]))/n_r
```

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

```r
#3a. p(theta2+theta3>0.40|theta1>0.23) based on three priors
length(which((dr1[,2]+dr1[,3]>0.40)&dr1[,1]>0.23))/length(which(dr1[,1]>0.23))
```

```
## [1] 0.4814815
```

```r
length(which((dr2[,2]+dr2[,3]>0.40)&dr2[,1]>0.23))/length(which(dr2[,1]>0.23))
```

```
## [1] 0.4347826
```

```r
length(which((dr3[,2]+dr3[,3]>0.40)&dr3[,1]>0.23))/length(which(dr3[,1]>0.23))
```

```
## [1] 0.6153846
```

```r
#3b. p(theta5+theta6>0.21|theta4>0.16) based on three priors
length(which((dr1[,5]+dr1[,6]>0.21)&dr1[,4]>0.16))/length(which(dr1[,4]>0.16))
```

```
## [1] 0.5799843
```

```r
length(which((dr2[,5]+dr2[,6]>0.21)&dr2[,4]>0.16))/length(which(dr2[,4]>0.16))
```

```
## [1] 0.5789615
```

```r
length(which((dr3[,5]+dr3[,6]>0.21)&dr3[,4]>0.16))/length(which(dr3[,4]>0.16))
```

```
## [1] 0.5780647
```

```r
#4. p(0.16<theta1<0.32,0.16<theta4<0.32)
length(which(dr1[,1]<0.32&dr1[,1]>0.16&dr1[,4]<0.32&dr1[,4]>0.16))/n_r
```

```
## [1] 0.93087
```

```r
length(which(dr2[,1]<0.32&dr2[,1]>0.16&dr2[,4]<0.32&dr2[,4]>0.16))/n_r
```

```
## [1] 0.93077
```

```r
length(which(dr3[,1]<0.32&dr3[,1]>0.16&dr3[,4]<0.32&dr3[,4]>0.16))/n_r
```

```
## [1] 0.93083
```

```r
#5. p(0.16<theta1<0.32|0.16<theta4<0.32)
length(which(dr1[,1]>0.16&dr1[,1]<0.32&dr1[,4]>0.16&dr1[,4]<0.32))/length(which(dr1[,4]>0.16&dr1[,4]<0.3
```

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

```r
length(which(dr2[,1]>0.16&dr2[,1]<0.32&dr2[,4]>0.16&dr2[,4]<0.32))/length(which(dr2[,4]>0.16&dr2[,4]<0.3
```

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

```r
length(which(dr3[,1]>0.16&dr3[,1]<0.32&dr3[,4]>0.16&dr3[,4]<0.32))/length(which(dr3[,4]>0.16&dr3[,4]<0.3
```

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

```r
#6.p(theta2>0.16,theta3>0.2|theta1>0.2)
length(which(dr1[,2]>0.16&dr1[,3]>0.2&dr1[,1]>0.2))/length(which(dr1[,1]>0.2))
```

```
## [1] 0.99942
```

```r
length(which(dr2[,2]>0.16&dr2[,3]>0.2&dr2[,1]>0.2))/length(which(dr2[,1]>0.2))
```

```
## [1] 0.99946
```

```r
length(which(dr3[,2]>0.16&dr3[,3]>0.2&dr3[,1]>0.2))/length(which(dr3[,1]>0.2))
```

```
## [1] 0.99954
```

```r
#7a. p(theta2+theta3/5>theta1) based on three priors
length(which((dr1[,2]+dr1[,3]/5)>dr1[,1]))/n_r
```

```
## [1] 0.11993
```

```r
length(which((dr2[,2]+dr2[,3]/5)>dr2[,1]))/n_r
```

```
## [1] 0.11781
```

```r
length(which((dr3[,2]+dr3[,3]/5)>dr2[,1]))/n_r
```

```
## [1] 0.08626
```

```r
#7b. p(theta2+theta3/4>theta1) based on three priors
length(which((dr1[,2]+dr1[,3]/4)>dr1[,1]))/n_r
```

```
## [1] 0.95589
```

```r
length(which((dr2[,2]+dr2[,3]/4)>dr2[,1]))/n_r
```

```
## [1] 0.95455
```

```r
length(which((dr3[,2]+dr3[,3]/4)>dr2[,1]))/n_r
```

```
## [1] 0.97492
```

```r
#7c. p(theta5+theta6/1.428>theta4) based on three priors
length(which((dr1[,5]+dr1[,6]/1.428)>dr1[,4]))/n_r
```

```
## [1] 0.93178
```

```r
length(which((dr2[,5]+dr2[,6]/1.428)>dr2[,4]))/n_r
```

```
## [1] 0.93102
```

```r
length(which((dr3[,5]+dr3[,6]/1.428)>dr2[,4]))/n_r
```

```
## [1] 0.9502
```

```r
#7d. p(theta5+theta6/1.66>theta4) based on three priors
length(which((dr1[,5]+dr1[,6]/1.66)>dr1[,4]))/n_r
```

```
## [1] 0.00834
```

```r
length(which((dr2[,5]+dr2[,6]/1.66)>dr2[,4]))/n_r
```

```
## [1] 0.0082
```

```r
length(which((dr3[,5]+dr3[,6]/1.66)>dr2[,4]))/n_r
```

```
## [1] 0.00386
```

MCMC Method

Data Model

```r
knitr::opts_chunk$set(echo = TRUE)
rc=2;cc=3
datx=c(5203,    3966,   5610,   3862,1685,3288)
#c(21,2,7,10,5,15,8,12,10) #c(604,245,67,130,235,76,63,180,252)#Example Agresti 1990 book 2.10 p32
k=length(datx)
T=sum(datx)

MUL01<-function()
{
  for (i in 1:1)
  {
    n[i,1:k] ~ dmulti(p[i,1:k], T)
    for(j in 1:k)
    {alpha[i,j]<-0.5
      #alpha[i,j]<-1
```

```
      #alpha[i,j] ~ dunif(0,1)
      p[i,j] <- delta[i,j] / (sum(delta[i,]))
      delta[i,j] ~ dgamma(alpha[i,j], 1)

    }
  }
}
```

WinBUGS File

```
knitr::opts_chunk$set(echo = TRUE)
#Writing BUGS File
MULT_1<- file.path(tempdir(), "MUL01.bug")
write.model(MUL01, MULT_1)
#file.show(MULT_1) #optional
#Data part - make sure your BUGS variable list is matching here
dat_MUL01<-list(n = structure(.Data = datx, .Dim = c(1, k)), k=k,T = T)

ns=15000 #Number of simulations in BUGS
al1=as.vector(runif(k,1,2))
del1=as.vector(runif(k,1,2))
inits <- function()
{
  list(alpha = structure(.Data = al1, .Dim = c(1,k)),
   delta = structure(.Data =del1, .Dim = c(1,k)))
}
parameters <- c("p")

post_MUL01.sim <- bugs(dat_MUL01, inits, parameters, model.file=MULT_1,
                       n.chains=3,n.thin=1, n.iter=ns,bugs.directory="D:/WinBUGS14/",debug =TRUE,digits=
```

Point and Interval Estimates

```
knitr::opts_chunk$set(echo = TRUE)
#Results
#Recommended
print(post_MUL01.sim,digits=5)
```

```
## Inference for Bugs model at "C:\Users\Samyajoy\AppData\Local\Temp\Rtmpucpxkh/MUL01.bug", fit using Wi
##  3 chains, each with 15000 iterations (first 7500 discarded)
##  n.sims = 22500 iterations saved
##               mean      sd      2.5%       25%       50%       75%
## p[1,1]     0.22034 0.00270   0.21510   0.21850   0.22030   0.22220
## p[1,2]     0.16796 0.00244   0.16320   0.16630   0.16800   0.16960
## p[1,3]     0.23757 0.00278   0.23210   0.23570   0.23760   0.23940
## p[1,4]     0.16353 0.00242   0.15880   0.16190   0.16350   0.16520
## p[1,5]     0.07136 0.00168   0.06807   0.07024   0.07135   0.07249
## p[1,6]     0.13924 0.00225   0.13480   0.13770   0.13920   0.14070
## deviance 133.96099 3.19999 129.80000 131.60000 133.30000 135.60000
##              97.5%    Rhat n.eff
## p[1,1]     0.22570 1.00122  7000
## p[1,2]     0.17280 1.00103 20000
## p[1,3]     0.24300 1.00107 15000
## p[1,4]     0.16830 1.00100 22000
## p[1,5]     0.07468 1.00094 22000
```

```
## p[1,6]     0.14360 1.00099 22000
## deviance 141.80000 1.00099 22000
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = Dbar-Dhat)
## pD = 5.0 and DIC = 139.0
## DIC is an estimate of expected predictive error (lower deviance is better).
```

```
#plots may be obtained from storing from chain
#Know your runs
mmm=post_MUL01.sim$sims.matrix
head(mmm)
```

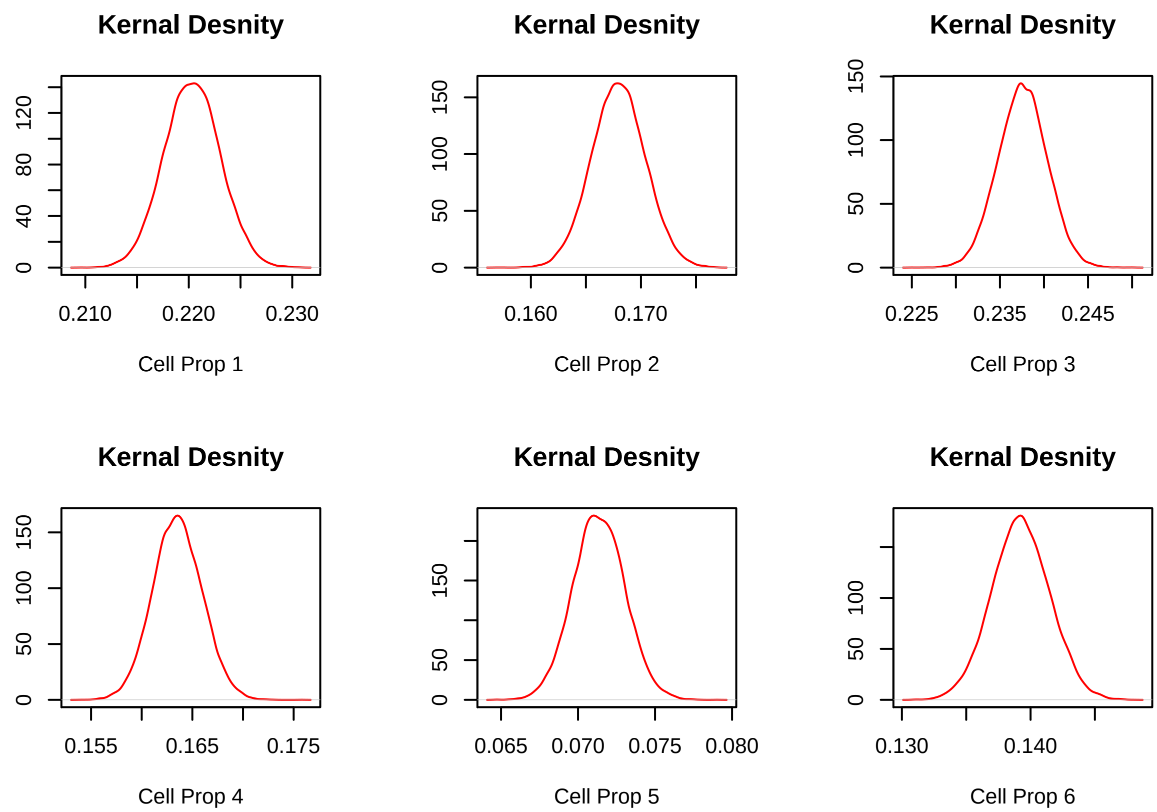
```
##      p[1,1] p[1,2] p[1,3] p[1,4]  p[1,5] p[1,6] deviance
## [1,] 0.2214 0.1687 0.2378 0.1659 0.06865 0.1375    132.9
## [2,] 0.2201 0.1678 0.2392 0.1619 0.07036 0.1407    130.2
## [3,] 0.2187 0.1653 0.2365 0.1670 0.07171 0.1408    132.5
## [4,] 0.2186 0.1706 0.2383 0.1638 0.07066 0.1381    130.7
## [5,] 0.2192 0.1706 0.2417 0.1618 0.06809 0.1386    135.9
## [6,] 0.2234 0.1664 0.2352 0.1643 0.07160 0.1390    130.9
```

```
me_theta=0;theta_LL=0;theta_UL=0
for (t in 1:k)
  {
me_theta[t]=mean(mmm[,t])
theta_LL[t]=quantile(mmm[,t],0.025)
theta_UL[t]=quantile(mmm[,t],0.975)
}
###############RESULTS
res=cbind(me_theta,theta_LL,theta_UL)

colnames(res)<-c("mean","95% CrI_LL","95% CrI_UL")
res
```

```
##            mean 95% CrI_LL 95% CrI_UL
## [1,] 0.22033570    0.21510    0.22570
## [2,] 0.16796042    0.16320    0.17280
## [3,] 0.23757159    0.23210    0.24300
## [4,] 0.16353402    0.15880    0.16830
## [5,] 0.07136222    0.06807    0.07468
## [6,] 0.13923610    0.13480    0.14360
```

```
#Kernal Densities
windows()
L=seq(1:k)
par(mfrow=c(rc,cc))
for (t in 1:k)
{
plot(density(mmm[,t]),main="Kernal Desnity",ylab=" ",col=2,xlab=paste("Cell Prop",L[t]))
}
```



Probabilities From Psterior Distributions

```
knitr::opts_chunk$set(echo = TRUE)
#Probabilities
#1
length(which(mmm[,1]>mmm[,2]))/post_MUL01.sim$n.sims
```

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

```
length(which(mmm[,4]>mmm[,5]))/post_MUL01.sim$n.sims
```

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

```
length(which(mmm[,1]>mmm[,4]))/post_MUL01.sim$n.sims
```

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

```
length(which(mmm[,2]>mmm[,5]))/post_MUL01.sim$n.sims
```

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

```
length(which(mmm[,3]>mmm[,6]))/post_MUL01.sim$n.sims
```

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

```
#2
length(which((mmm[,2]+mmm[,3]>mmm[,1])))/post_MUL01.sim$n.sims
```

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

```
length(which((mmm[,5]+mmm[,6]>mmm[,4])))/post_MUL01.sim$n.sims
```

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

```
#7
length(which((mmm[,2]+mmm[,3]/5>mmm[,1])))/post_MUL01.sim$n.sims
```

```
## [1] 0.1175111
```

```
length(which((mmm[,2]+mmm[,3]/4>mmm[,1])))/post_MUL01.sim$n.sims
```

```
## [1] 0.9545333
```

```
length(which((mmm[,5]+mmm[,6]/1.66>mmm[,4])))/post_MUL01.sim$n.sims
```

```
## [1] 0.0096
```

```
length(which((mmm[,5]+mmm[,6]/1.428>mmm[,4])))/post_MUL01.sim$n.sims
```

```
## [1] 0.9295111
```

```
#3
length(which((mmm[,2]+mmm[,3]>0.40)&mmm[,1]>0.23))/length(which(mmm[,1]>0.23))
```

```
## [1] 0.5
```

```
length(which((mmm[,5]+mmm[,6]>0.21)&mmm[,4]>0.16))/length(which(mmm[,4]>0.16))
```

```
## [1] 0.5735054
```

```
#4
length(which(mmm[,1]>0.16&mmm[,1]<0.32&mmm[,4]>0.16&mmm[,4]<0.32))/post_MUL01.sim$n.sims
```

```
## [1] 0.9263111
```

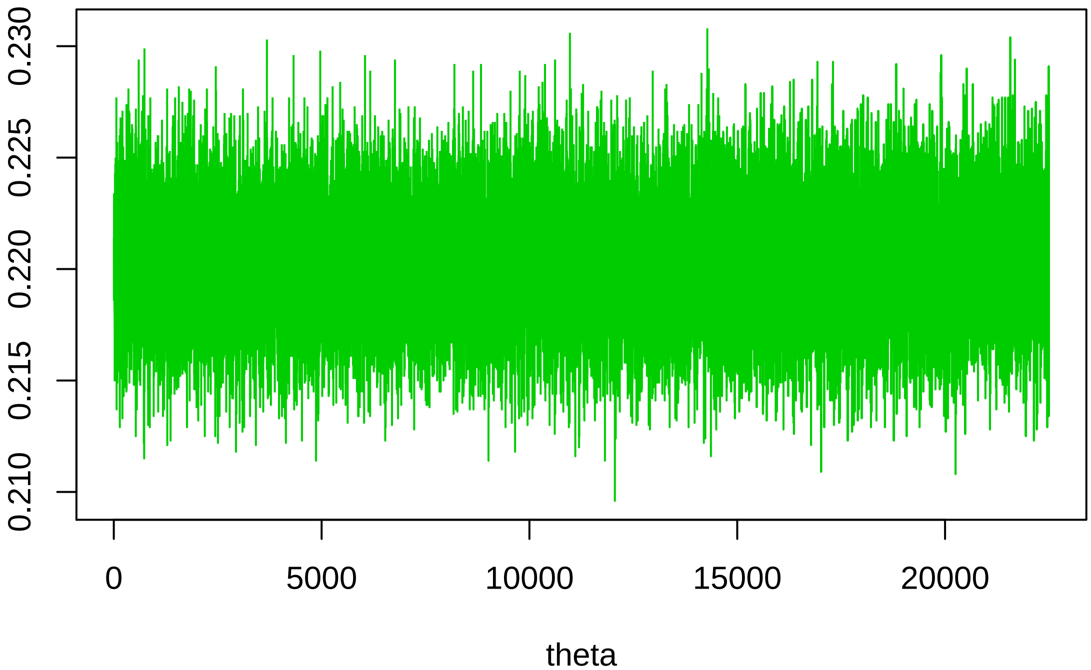
```
#5
length(which(mmm[,1]<0.32&&mmm[,1]>0.16&&mmm[,4]<0.32&&mmm[,4]>0.16))/length(which(mmm[,4]>0.16&&mmm[,4]
```

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

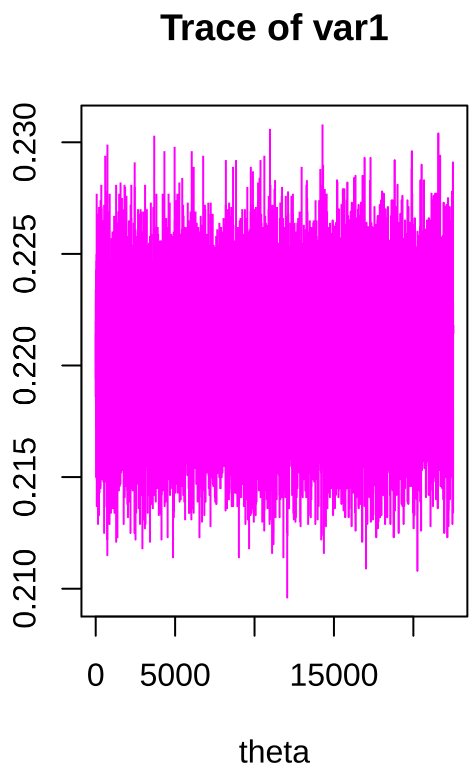
```
#6.p(theta2>0.16,theta3>0.2|theta1>0.2)
length(which(mmm[,2]>0.16&mmm[,3]>0.2&mmm[,1]>0.2))/length(which(mmm[,1]>0.2))
```

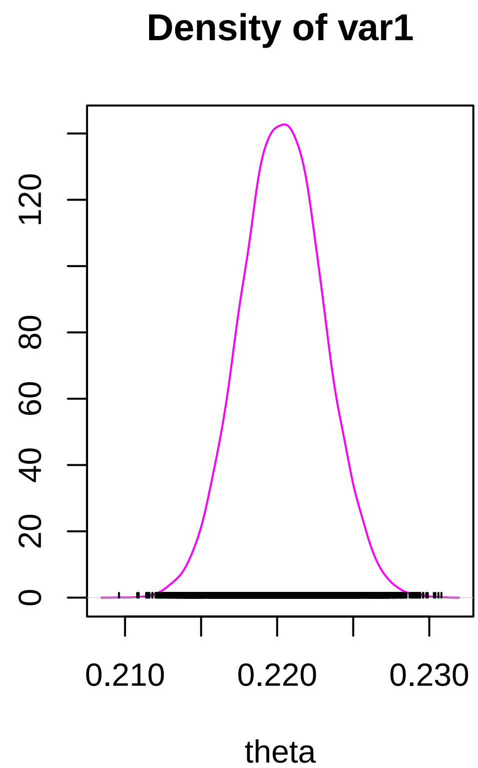
```
## [1] 0.9994222
```

```
######################################################################
#CODA BASED - Creating mcmc objects - optinal
mc_theta=mcmc(mmm[,1])
windows()
traceplot(mc_theta, smooth = TRUE,col=3,xlab = "theta")
```




```r
windows()
plot(mc_theta,trace = TRUE, density = TRUE, smooth = TRUE, col=6,xlab = "theta")
```






```
##########################################################################
```

Closed Form Integration

Probabilities can be estimated from the posterior distrubution using closed form integration. For large data integration becomes troublesome. Sample codes are given below

$p(\theta_{ij} + \theta_{kl} \geq \theta_{mn})$

```
knitr::opts_chunk$set(echo = TRUE)
f<-function(x,y,z)
{(gamma(94.5)/(gamma(5.5)*gamma(15.5)*gamma(21.5)*gamma(52)))*x^4.5*y^14.5*z^20.5*(1-x-y-z)^51}
zmin1<-0
zmax1<-function(x,y){x+y}
ymin1<-0
ymax1<-function(x){0.5-x}
a1<-0
b1<-0.5
zmin2<-0
zmax2<-function(x,y){1-x-y}
ymin2<-0
ymax2<-function(x){1-x}
a2<-0.5
b2<-1
p<-integral3(f,a1,b1,ymin1,ymax1,zmin1,zmax1)
```

```
q<-integral3(f,a2,b2,ymin2,ymax2,zmin2,zmax2)
p
```

```
## [1] 0.4691324
```

```
q
```

```
## [1] 9.734816e-22
```

```
p+q
```

```
## [1] 0.4691324
```

$p(\theta_{ij} + \theta_{kl} \geq \ a|\theta_{mn} \geq \ b)$

```
knitr::opts_chunk$set(echo = TRUE)
f<-function(x,y,z)
{(gamma(94.5)/(gamma(5.5)*gamma(15.5)*gamma(21.5)*gamma(52)))*x^4.5*y^14.5*z^20.5*(1-x-y-z)^51}
zmin<-0.11
zmax<-function(x,y)1-x-y
ymax<-function(x)0.15-x
ymin<-0
xmin<-0
xmax<-0.15
p<-integral3(f,xmin,xmax,ymin,ymax,zmin,zmax)
1-p
```

```
## [1] 0.9646275
```

$p(\theta_{ij} \geq \theta_{kl})$

```
knitr::opts_chunk$set(echo = TRUE)
f<-function(x,y)
{(gamma(94.5)/(gamma(21.5)*gamma(15.5)*gamma(57.5)))*x^20.5*y^14.5*(1-x-y)^56.5}
ymin1<-0
ymax1<-function(x)x
xmin1<-0
xmax1<-0.5
ymin2<-0
ymax2<-function(x)1-x
xmin2<-0.5
xmax2<-1
p<-integral2(f,xmin1,xmax1,ymin1,ymax1)
q<-integral2(f,xmin2,xmax2,ymin2,ymax2)
p
```

```
## $Q
## [1] 0.8413225
##
## $error
## [1] 5.585099e-08
```

```
q
```

```
## $Q
## [1] 2.214095e-08
##
## $error
## [1] 6.211761e-17
```

```r
r<-as.matrix(p)
r
```

```
##       [,1]
## Q     0.8413225
## error 5.585099e-08
```

```r
s<-as.matrix(q)
s
```

```
##       [,1]
## Q     2.214095e-08
## error 6.211761e-17
```

```r
as.numeric(r[1,])+as.numeric(s[1,])
```

```
## [1] 0.8413225
```

$p(a \leq \theta_{ij} \leq \ b, c \leq \theta_{kl} \leq \ d)$

```r
knitr::opts_chunk$set(echo = TRUE)
f<-function(x,y)
{(gamma(94.5)/(gamma(21.5)*gamma(15.5)*gamma(57.5)))*x^20.5*y^14.5*(1-x-y)^56.5}
a<-0.2
b<-0.3
c<-0.15
d<-0.25
integral2(f,a,b,c,d)
```

```
## $Q
## [1] 0.3967451
##
## $error
## [1] 2.183194e-14
```

$p(a \leq \theta_{ij} \leq \ b|c \leq \theta_{kl} \leq \ d)$

```r
knitr::opts_chunk$set(echo = TRUE)
f<-function(x,y)
{(gamma(94.5)/(gamma(21.5)*gamma(15.5)*gamma(57.5)))*x^20.5*y^14.5*(1-x-y)^56.5}
a<-0.2
b<-0.3
c<-0.15
d<-0.25
p<-integral2(f,a,b,c,d)
p
```

```
## $Q
## [1] 0.3967451
##
## $error
## [1] 2.183194e-14
```

```r
q<-pbeta(d,15.5,79)-pbeta(c,15.5,79)
q
```

```
## [1] 0.6040014
```

```
r<-as.matrix(p)
r
```

```
##       [,1]
## Q     0.3967451
## error 2.183194e-14
```

```
as.numeric(r[1,])/q
```

```
## [1] 0.6568612
```

$p(a \leq \theta_{ij} \leq \ b, c \leq \theta_{kl} \leq \ d | e \leq \theta_{mn} \leq \ f)$

```
knitr::opts_chunk$set(echo = TRUE)
fun<-function(x,y,z)
{(gamma(94.5)/(gamma(5.5)*gamma(15.5)*gamma(21.5)*gamma(52)))*x^4.5*y^14.5*z^20.5*(1-x-y-z)^51}
a<-0.05
b<-0.2
c<-0.11
d<-0.25
e<-0.2
f<-0.33
p<-integral3(fun,a,b,c,d,e,f)
p
```

```
## [1] 0.3704418
```

```
q<-pbeta(f,21.5,73)-pbeta(e,21.5,75)
q
```

```
## [1] 0.6803392
```

```
p/q
```

```
## [1] 0.5444957
```