SDS Page No.
Date

Bayesian Computing

Name :- Preksha A Patel
sapid :- 60004210126
Branch :- Computer Engineering
Div :- C2

Experiment no. 8

Aim :- To implement a program for learning about a normal population from grouped data, using height and frequency data from student's dataset.

Theory :-

Markov chain Monte Carlo (MCMC) Technique :-

Markov chain Monte Carlo (MCMC) serves as a pivotal technique in Bayesian computing, offering a systematic approach to sample from intricate probability distributions. At its ~~error~~ core, MCMC constructs a Markov chain, where each subsequent sample depends on the current state, allowing for a controlled exploration of parameter spaces. This iterative process proves invaluable in Bayesian analysis, particularly when dealing with high-dimensional or complex posterior distributions.

Metropolis Random walk Algorithms :-

The metropolis Random walk algorithm is a cornerstone in Bayesian computing, specifically within the realm of markov chain Monte Carlo (MCMC) methods. It addresses the challenge of sampling from complex and high-dimensional probability distributions, a common scenario in Bayesian statistical inference.

1. **Initialization** :- Commence the algorithm by providing an initial estimate for the parameters, denoted as θ.
2. **Proposal distribution** :- Define a proposal distribution $q(\theta'/\theta)$ that suggests a new parameter value θ' given the ~~to~~ current value θ.
3. **Generate a Proposed state** :-

Draw a sample from the proposed distribution ~~to~~ get a proposed state θ'

P.T.O.

4. Acceptance Probability :
Calculate the acceptance probability $\alpha = \min\left(1, \frac{P(\theta')}{P(\theta)}\right)$, where $p(\theta)$ is the posterior probability.

5. Accept or Reject :
Accept the proposed state with probability α; otherwise stay at the current state.

6. Repeat :
Repeat steps 3-5 for a predefined no. of iterations or until convergence.

Conclusion :
In this experiment, we used the LearnBayes library in R to perform Bayesian inference on a normal population from grouped data. we used both a normal approximation using laplace method, and a MCMC algorithm using metropolis Random walks to obtain posterior distribution of parameters.

SAP ID.: 60004210126 Name: Preksha Ashok Patel

Batch: C2-1

Subject: Bayesian Computing Laboratory Semester: VII

Experiment No. 8

Aim:

Implement a program for learning about a normal population from grouped data, using height and frequency data from student's dataset.

Code:

Import libraries:

```
Library(LearnBayes)
```

Observe normally distributed data in grouped form. Consider the posterior of (**μ**, log(**σ**)):

```
d <- list(int.lo=c(-Inf, seq(66, 74, by=2)),

int.hi=c(seq(66, 74, by=2), Inf),

f=c(14, 30, 49, 70, 33, 15))


y <- c(rep(65,14), rep(67,30), rep(69,49),

rep(71,70), rep(73,33), rep(75,15)) mean(y)


log(sd(y))
```

Obtain normal approximation to posterior:

```
start <- c(70, 1) fit <-

laplace(groupeddatapost, start, d)

fit
```

Use Metropolis (random walk) MCMC algorithm:

```
modal.sds <- sqrt(diag(fit$var))

proposal <- list(var=fit$var, scale=2)

fit2 <- rwmetrop(groupeddatapost,

                proposal,

start,

                10000, d)


fit2$accept


post.means <- apply(fit2$par, 2, mean)

post.sds <- apply(fit2$par, 2, sd)

cbind(c(fit$mode), modal.sds)


cbind(post.means, post.sds)


mycontour(groupeddatapost,

         c(69, 71, .6, 1.3), d,
xlab="mu",ylab="log sigma")
points(fit2$par[5001:10000, 1],
fit2$par[5001:10000, 2])
```

Output:

Observe normally distributed data in grouped form. Consider the posterior of μ, $\log(\sigma)$):

```
[1] 70.16588
```

```
[1] 0.9504117
```

Obtain normal approximation to posterior:

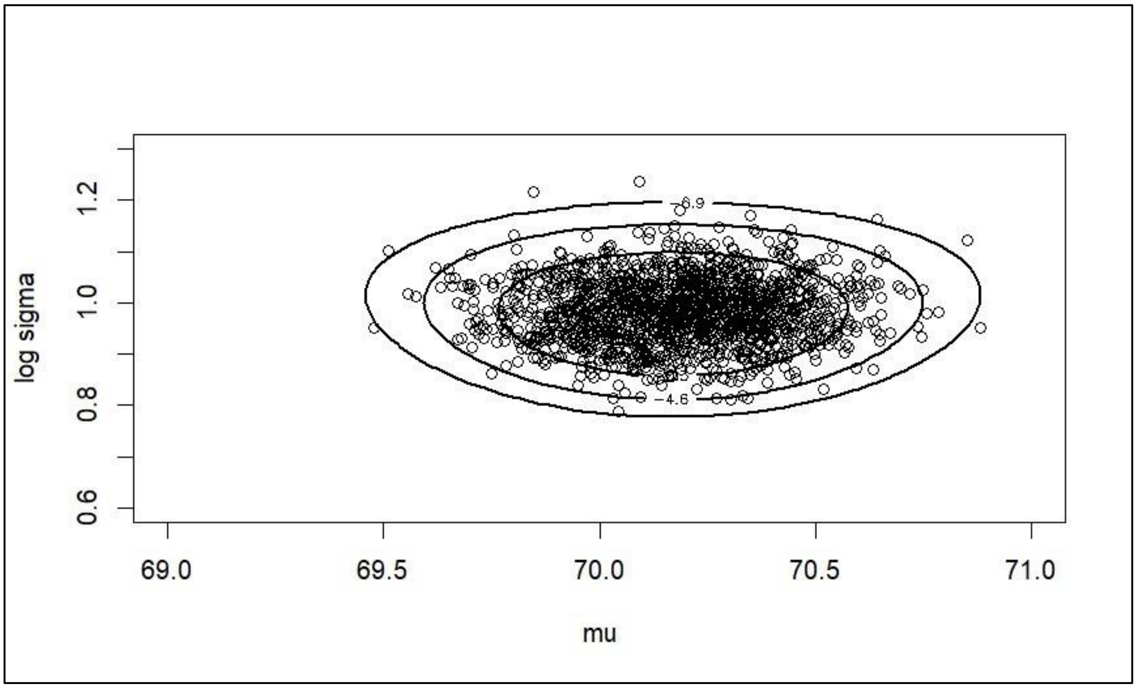
```
$mode
[1] 70.169880  0.973644

$var
             [,1]         [,2]
[1,] 3.534713e-02 3.520776e-05
[2,] 3.520776e-05 3.146470e-03

$int
[1] -350.6305

$converge
[1] TRUE
```

Use Metropolis (random walk) MCMC algorithm:

```
[1] 0.293
```

```
                modal.sds
[1,] 70.169880 0.18800834
[2,]  0.973644 0.05609341
```

```
     post.means   post.sds
[1,] 70.1737319 0.19051458
[2,]  0.9815361 0.05686768
```


-6.9
-4.6
log sigma
0.6
0.8
1.0
1.2
69.0
69.5
70.0
70.5
71.0
mu