# ST-540 Assignment-4

Tilekbek Zhoroev

**Problem 1** Give an advantage and a disadvantage of the following methods:

a. Maximum a posterior estimation: *Fast but doesn't quantify uncertainty*

b. Numerical integration: *Accurate but not feasible in high dimensions*

c. Bayesian central limit theorem: *Fast but need large sample size.*

d. Gibbs sampling: *Works in high dimensions and can be accurate but requires known conjugate priors.*

e. Metropolis Hastings sampling: *Flexible but requires tuning.*

**Problem 2**

Consider the model $Y_i|\mu,\sigma^2 \sim Normal(\mu,\sigma^2)$ for $i = 1,...,n$ and $Y_i|\mu,\delta,\sigma^2 \sim Normal(\mu+\delta,\sigma^2)$ for $i = n+1,...,n+m$, where $\mu,\delta \sim Normal(\mu,100^2)$ and $\sigma^2 \sim InvGamma(0.01,0.01)$

a. Give an example of a real experiment for which this would be an appropriate model.

*$Y_i$ are the average monthly amount of money one person consumes for daily expenses. After COVID-19, we would like to analyse the effect of pandemic in m month to person's daily expenses.*

*Another example: Weights of two groups of people, n people with age 10-20 and m people with age 20-30*

b. Derive the full conditional posterior distributions for $\mu,\delta$ and $\sigma^2$

*First,by given expressions we can obtain that*

$$\begin{aligned}
f(Y_1,Y_2,...,Y_{m+n}|\mu,\delta,\sigma^2) &= \prod_{i=1}^{n} f(Y_i|\mu,\sigma^2) \prod_{i=n+1}^{n+m} f(Y_i|\mu,\delta\sigma^2) \\
&= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{(Y_i-\mu)^2}{2\sigma^2}) \prod_{i=n+1}^{n+m} \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{(Y_i-\delta-\mu)^2}{2\sigma^2}) \\
&\propto exp(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(Y_i-\mu)^2) \times exp(-\frac{1}{2\sigma^2}\sum_{i=n+1}^{n+m}(Y_i-\mu-\delta)^2) \\
&\propto exp(-\frac{(m+n)\mu - 2\mu(\sum_{i=1}^{n} Y_i + \sum_{i=1}^{n}(Y_i-\delta)}{2\sigma^2})
\end{aligned}$$

*Then, from section 2.1.3 we have,*

$$\begin{aligned} f(\mu|Y_1,...,Y_{n+m},\delta,\sigma^2) &= \frac{f(Y_1,...,Y_{n+m},\mu,\delta,\sigma^2)}{f(Y_1,...,Y_{n+m},\mu,\delta,\sigma^2)} \\ &\propto f(Y_1,...,Y_{n+m},\mu,\delta,\sigma^2) \\ &\propto f(Y_1,...,Y_{n+m}|\mu,\delta,\sigma^2)f(\mu) \\ &\propto Normal\left(\frac{\sum_{i=1}^{n} Y_i + \sum_{i=n+1}^{m+n}(Y_i-\delta)}{n+m+\sigma^2/100^2}, \frac{1}{(n+m)/\sigma^2+1/100^2}\right). \end{aligned}$$

*Since $\delta$ involved only after $i = n+1$,*

$$\delta|Y,\mu,\sigma^2 \propto Normal\left(\frac{\sum_{i=n+1}^{m+n}(Y_i-\mu)}{m+\sigma^2/100^2}, \frac{1}{m/\sigma^2+1/100^2}\right).$$

*To find the full condition distribution for $\sigma^2$ we would use the fact from section 2.1.4 and we get*

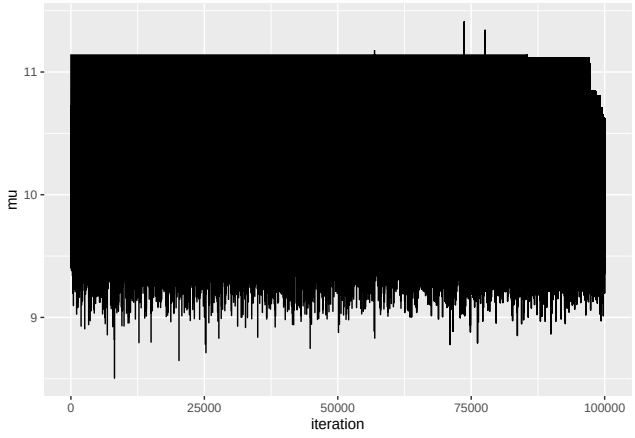
$$\sigma^2|Y,\mu,\delta \propto InvGamma\left(0.01+(n+m)/2, \sum_{i=1}^{n}(Y_i-\mu)^2/2 + \sum_{i=n+1}^{n+m}(Y_i-\mu-\delta)^2/2 + 0.01\right)$$

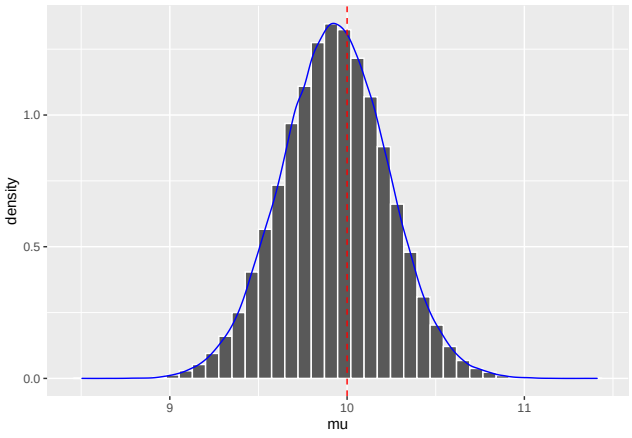
c. Simulate a dataset from this model with $n = m = 50, \mu = 10$, $\delta = 1$, and $\sigma = 2$. Write your own Gibbs sampling code (not in JAGS) to fit the model above to the simulated data and plot the marginal posterior density for each parameter. Are you able to recover the true values reasonably well?

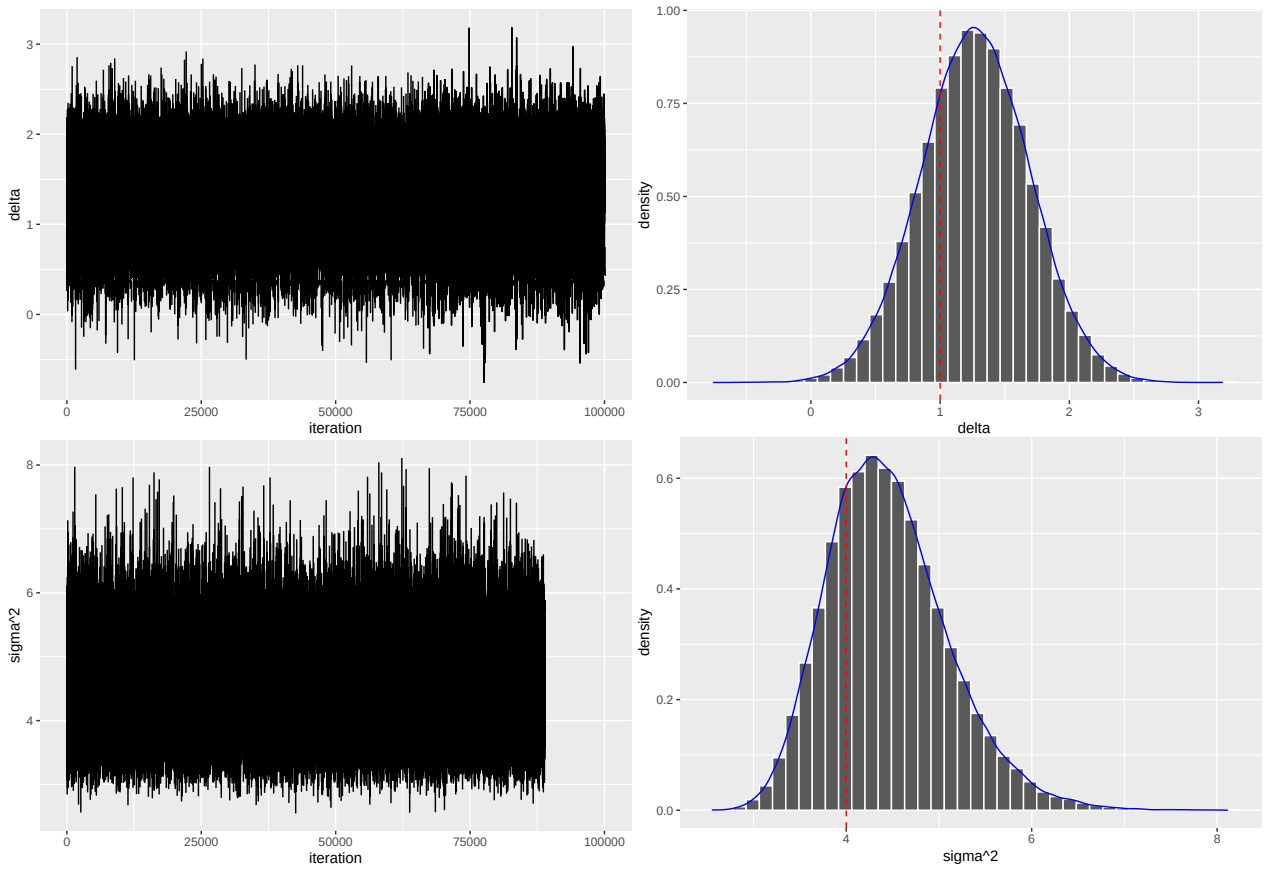
```
set.seed(42)
# Given parameters
  n <- 50; m <- 50; mu_0 <- 10; delta_0 <- 1; sigma_0 <- 2
 #generate likelihoods
  Y1 <- rnorm(n,mu_0,sigma_0); Y2 <- rnorm(m,mu_0+delta_0,sigma_0); Y <- c(Y1,Y2)

  # Prep for Gibbs sampling
  S       <- 10^5  # number of iterations
  #priors
  pri_var <- 100^2;   a <- b <- 0.01; mu <- mean(Y); delta <- mean(Y)-mean(Y2); sigma2 <- var(Y)

  Gibss<-matrix(0,S,3)
  for(iter in 1:S){
    # mu
    prec  <- (n+m)/sigma2 + 1/pri_var
    mn    <- sum(Y1)/sigma2 + sum(Y2-delta)/sigma2
    mu    <- rnorm(1,mn/prec,1/sqrt(prec))

    # delta
    prec  <- m/sigma2 + 1/pri_var
    mn    <- sum(Y2-mu)/sigma2
    delta <- rnorm(1,mn/prec,1/sqrt(prec))

    # sigma2
```

```
    sigma2 <- rinvgamma(1,a+(n+m)/2,(sum((Y1-mu)^2) + sum((Y2-mu-delta)^2))/2+b)

    Gibss[iter,] <- c(mu,delta,sigma2)
  }
Gibss<-data.frame(Gibss)
colnames(Gibss) = c("mu","delta","sigma^2")
```

```
Gibss%>%
  ggplot(aes(x=seq(1,S), y = mu ))+geom_line()+xlab("iteration")

Gibss%>%
  ggplot(aes(mu))+
  geom_histogram(aes(y = ..density..), colour = "white", bins = 40) +
  geom_density(colour ="blue")+
  geom_vline(xintercept=mu_0, linetype="dashed", colour = "red")


Gibss%>%
  ggplot(aes(x=seq(1,S), y = delta ))+geom_line()+xlab("iteration")

Gibss%>%
  ggplot(aes(delta))+
  geom_histogram(aes(y = ..density..), color = "white", bins = 40) +
  geom_density(colour ="blue")+
  geom_vline(xintercept=delta_0, linetype="dashed", color = "red")
Gibss%>%
  ggplot(aes(x=seq(1,S), y = `sigma^2` ))+geom_line()+xlab("iteration")

Gibss%>%
  ggplot(aes(`sigma^2`)) +
  geom_histogram(aes(y = ..density..), color = "white", bins = 40) +
  geom_density(colour ="blue") +
  geom_vline(xintercept=sigma_0^2, linetype="dashed", color = "red")
```






**Problem 3**

Fit the following model to the NBA free throw data in the table in the Exercise 17 in Chapter 1:

| Player | Overall proportion | Clutch makes | Clutch attempts |
|---|---|---|---|
| Russell Westbrook | 0.845 | 64 | 75 |
| James Harden | 0.847 | 72 | 95 |
| Kawhi Leonard | 0.880 | 55 | 63 |
| LeBron James | 0.674 | 27 | 39 |
| Isaiah Thomas | 0.909 | 75 | 83 |
| Stephen Curry | 0.898 | 24 | 26 |
| Giannis Antetokounmpo | 0.770 | 28 | 41 |
| John Wall | 0.801 | 66 | 82 |
| Anthony Davis | 0.802 | 40 | 54 |
| Kevin Durant | 0.875 | 13 | 16 |

$Y_i|\theta_i \sim Binomial(n_i;\theta_i)$ and $\theta_i|m \sim Beta[exp(m)q_i, exp(m)(1-q_i)]$, where $Y_i$ is the number of made clutch shots for player $i = 1, ..., 10$, $n_i$ is the number of attempted clutch shots, $q_i \in (0,1)$ is the overall proportion, and $m \sim Normal(0, 10)$.

a. Explain why this is a reasonable prior for $\theta_i$.

*Since the domain of the beta distribution covers 0 to 1 and the mean of the beta distribution is*

$$\frac{e^m q_i}{e^m q_i + e^m(1-q_i)} = q_i$$

*so distribution is centered at $q_i$. and takes values on [0,1].*

b. Explain the role of $m$ in the prior.

*This determines the sread of the distribution around $q_i$.*

c. Derive the full conditional posterior for $\theta_1$.

*Since the number of the clutches of each player is independent we have*

$$\begin{aligned} f(Y_1, ..., Y_{10}, \theta_1, ..., \theta_{10}|m) &= \prod_{i=1}^{10} f(Y_i|\theta_i) f(\theta_i|m) \\ &\propto \prod_{i=1}^{10} Beta(Y_i + exp(m)q_i, n_i - Y_i + exp(m)(1-q_i)) \end{aligned}$$

*where we employ the Beta-Binomial conjugacy. Next, the full conditional posterior of the $\theta_1$ is*

$$\begin{aligned} f(\theta_1|Y_1, ..., Y_10, \theta_2, ...\theta_{10}, m) &= \frac{f(Y_1, ..., Y_{10}, \theta_1, ..., \theta_{10}|m)}{f(Y_1, ..., Y_{10}, \theta_2, ..., \theta_{10}|m)} \\ &\propto Beta(Y_1 + exp(m)q_1, n_1 - Y_1 + exp(m)(1-q_1)) \end{aligned}$$

d. Write your own MCMC algorithm to compute a table of posterior means and 95% credible intervals for all 11 model parameters $(\theta_1, ..., \theta_{10}, m)$. Turn in commented code.

*Similarly as previous part we obtain the full conditional posterior of each $\theta_i$,*

$$\theta_i|rest \propto Beta(Y_i + exp(m)q_i, n_i - Y_i + exp(m)(1-q_i)) \qquad i = 1, ..., 10.$$

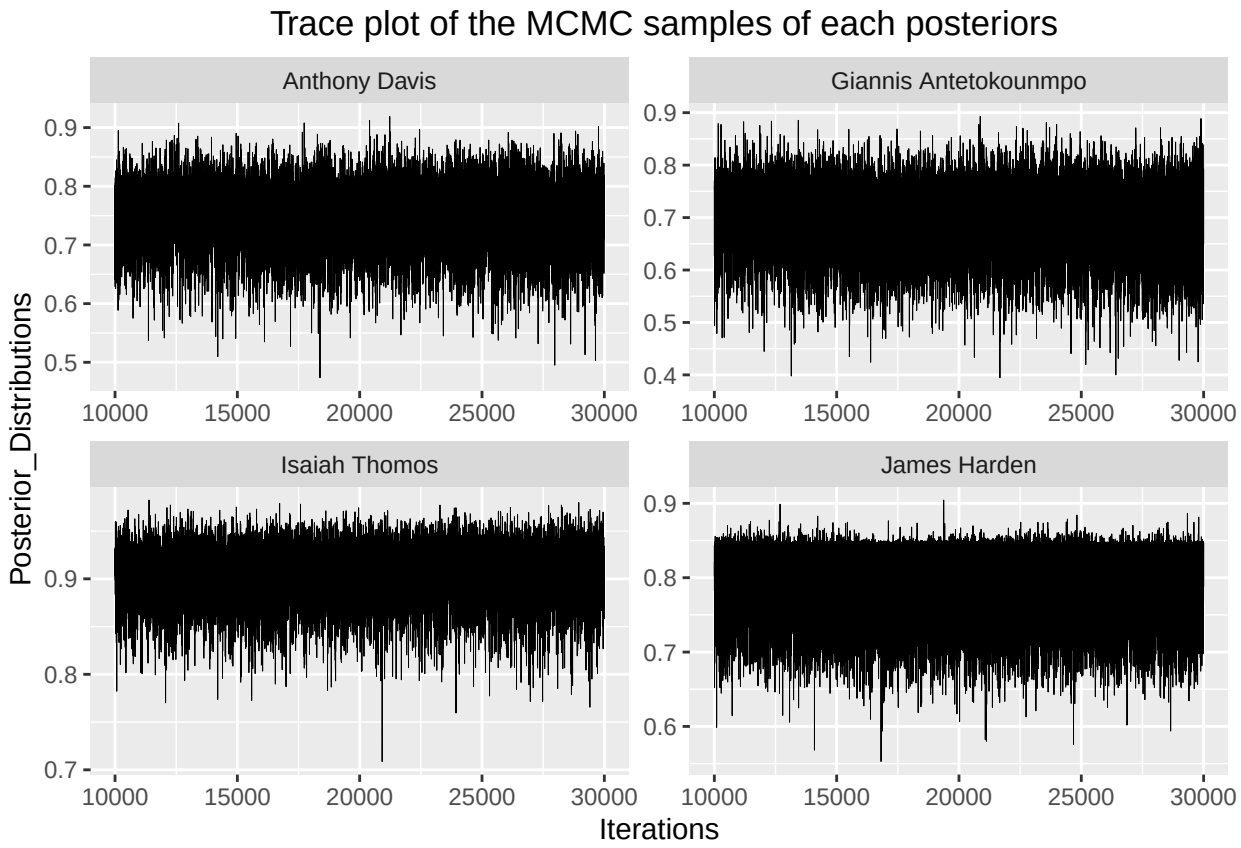
*However for m we don't have nice form of the full conditional distribution so we combine Gibbs and Metropolis algorithm*

```
set.seed(420)
#given values in table
Ys <- c(64,72,55,27,75,24,28,66,40,13)
ns <- c(75,95,63,39,83,26,41,82,54,16)
qs <- c(0.845, 0.847, 0.880, 0.674, 0.909, 0.898, 0.770, 0.801, 0.802, 0.875)
#parameters to start MCMC
m <- 0;
thetas <- qs;
S <- 3*10^4;
burn_in = 10^4;
MCMC<-matrix(0,S-burn_in,11); # save only after burn-in
can_sd <- 0.3
#log posterior for m
log_post_m <- function(theta, Ys, ns, qs, m){
  like <- 0
  for(i in 1:10){
  like = like + dbeta(theta[i], Ys[i] + exp(m)*qs[i], ns[i] - Ys[i] + exp(m)*(1-qs[i]),log = TRUE )
  }
  prior = dnorm(m,0,10,log=TRUE)
  return(like + prior)}

for(iter in 1:S){
   # Gibbs for each theta
  for(i in 1:10){
    alpha = Ys[i] + exp(m) * qs[i]
    beta = ns[i] - Ys[i] + exp(m) * (1 - qs[i])
```

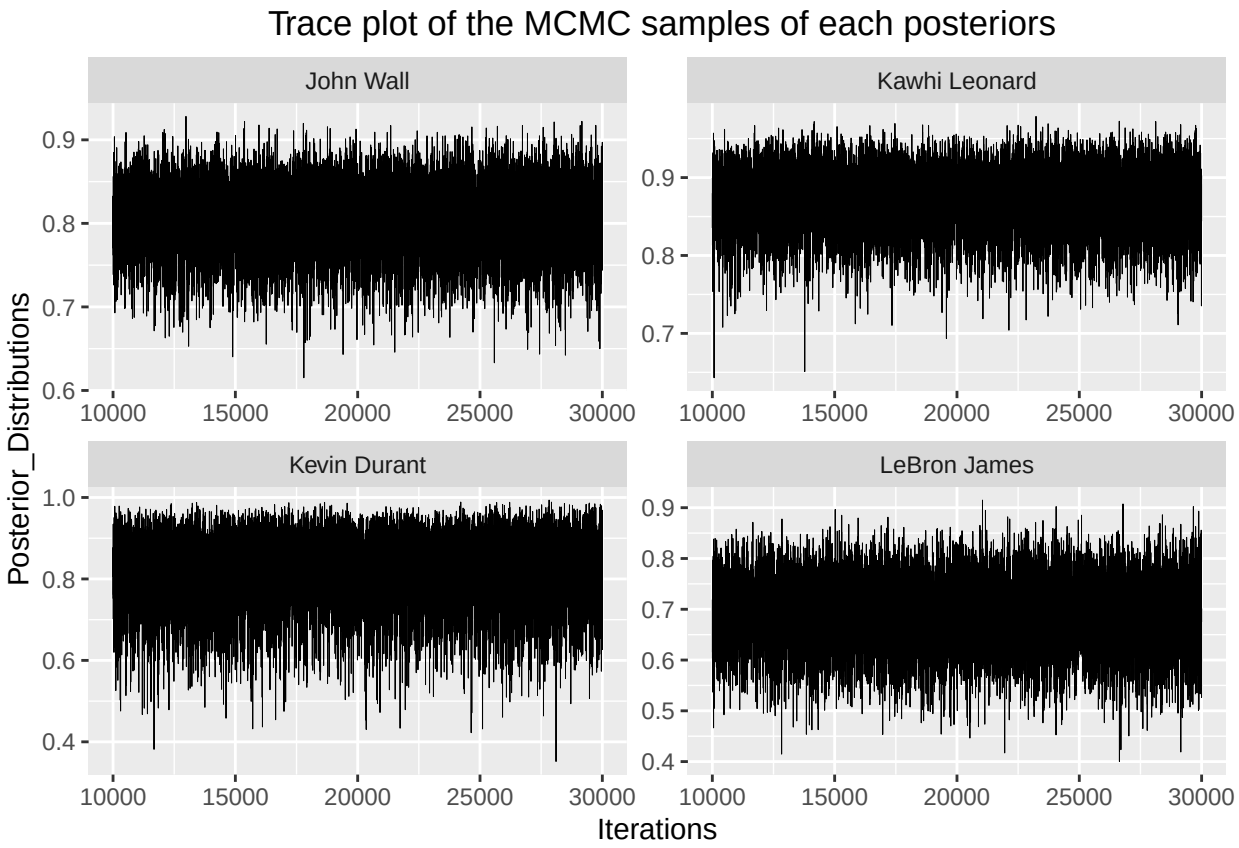
```
    thetas[i] = rbeta(1, alpha, beta)
  }
  # Metropolis for m
  can <- rnorm(1, m, can_sd) #proposal distribution
  logR <- log_post_m(thetas, Ys, ns, qs, can) - log_post_m(thetas, Ys, ns, qs, m)
  R<-exp(logR)
  if(runif(1) < R){
       m <- can
  }
  if(iter>burn_in){
  MCMC[iter - burn_in,] = c(thetas,m)}
}


MCMC<-data.frame(MCMC)


colnames(MCMC)<-c("Russell Westbrook","James Harden","Kawhi Leonard","LeBron James",
"Isaiah Thomos","Stephen Curry","Giannis Antetokounmpo","John Wall",
"Anthony Davis","Kevin Durant","m")
```

The trace plots of MCMC sample of parameters are presented below.

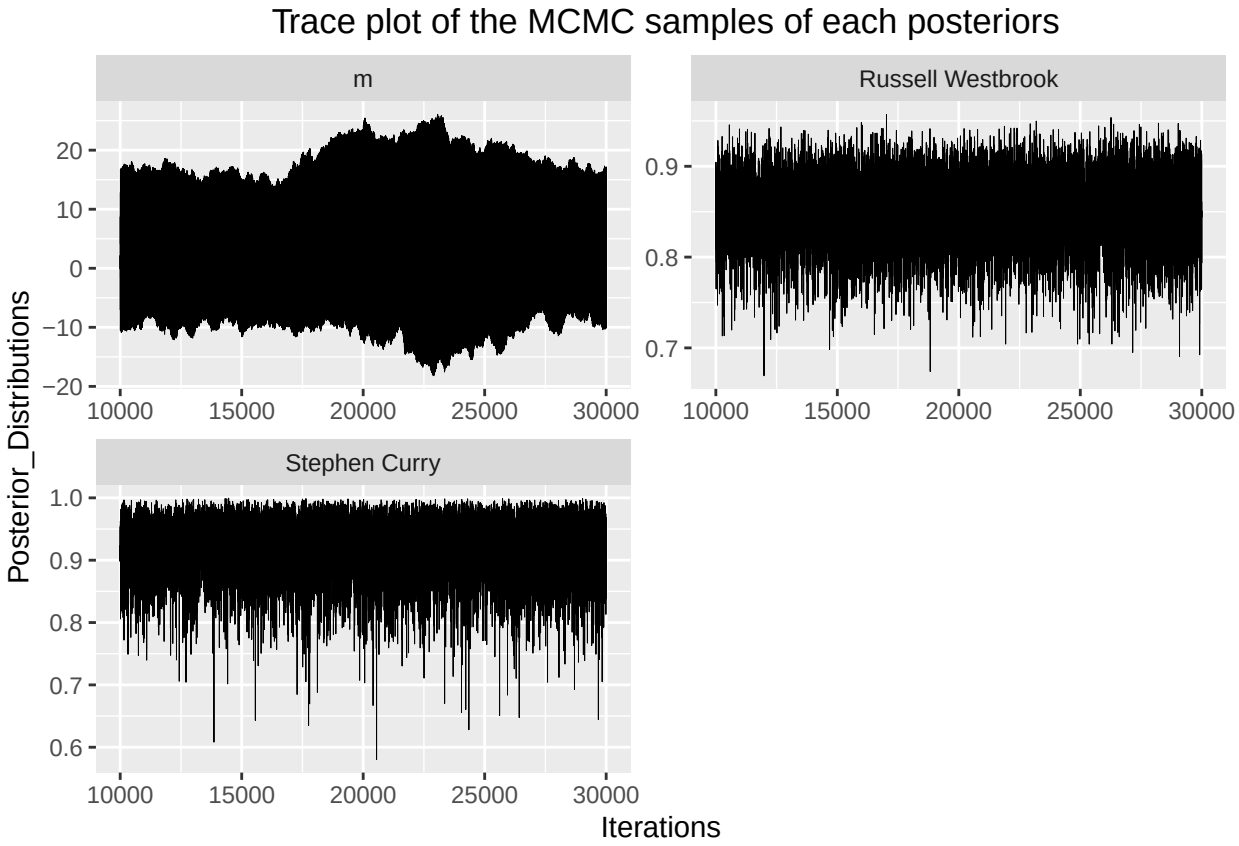
```
MCMC%>%
pivot_longer(cols = "Russell Westbrook":"m", names_to = "Parameters",
             values_to = "Posterior_Distributions")%>%
  ggplot(aes(x=rep(seq(burn_in+1,S), 11), y = Posterior_Distributions))+
  xlab("Iterations")+
  geom_line(size=.1)+theme(plot.title = element_text(hjust = 0.5))+
  ggtitle(" Trace plot of the MCMC samples of each posteriors")+
  facet_wrap_paginate(~Parameters,scales = "free", ncol = 2, nrow = 2, page = 1)
```




```
MCMC%>%
pivot_longer(cols = "Russell Westbrook":"m", names_to = "Parameters",
             values_to = "Posterior_Distributions")%>%
  ggplot(aes(x=rep(seq(burn_in+1,S), 11), y = Posterior_Distributions))+
  xlab("Iterations")+
  geom_line(size=.1)+theme(plot.title = element_text(hjust = 0.5))+
  ggtitle(" Trace plot of the MCMC samples of each posteriors")+
  facet_wrap_paginate(~Parameters,scales = "free", ncol = 2, nrow = 2, page = 2)
```




```
MCMC%>%
pivot_longer(cols = "Russell Westbrook":"m", names_to = "Parameters",
             values_to = "Posterior_Distributions")%>%
  ggplot(aes(x=rep(seq(burn_in+1,S), 11), y = Posterior_Distributions))+
  xlab("Iterations")+
  geom_line(size=.09)+theme(plot.title = element_text(hjust = 0.5))+
  ggtitle(" Trace plot of the MCMC samples of each posteriors")+
  facet_wrap_paginate(~Parameters,scales = "free", ncol = 2, nrow = 2, page = 3)
```




```
table=sapply(MCMC, quantile,  probs = c(.5, 0.025, 0.975))
rownames(table) = c("Means","2.5 %","97.5 %")
knitr::kable(t(table))
```

| | Means | 2.5 % | 97.5 % |
|---|---|---|---|
| Russell Westbrook | 0.8451769 | 0.7782915 | 0.9143715 |
| James Harden | 0.8016995 | 0.6826536 | 0.8507454 |
| Kawhi Leonard | 0.8799938 | 0.7967172 | 0.9347815 |
| LeBron James | 0.6742497 | 0.5654131 | 0.8053789 |
| Isaiah Thomos | 0.9089932 | 0.8430956 | 0.9509257 |
| Stephen Curry | 0.9001120 | 0.8230233 | 0.9853191 |
| Giannis Antetokounmpo | 0.7482889 | 0.5582152 | 0.8054475 |
| John Wall | 0.8010575 | 0.7276388 | 0.8738146 |
| Anthony Davis | 0.7885786 | 0.6405154 | 0.8385163 |
| Kevin Durant | 0.8736614 | 0.6461045 | 0.9452855 |
| m | 2.2746495 | -11.5461571 | 21.4155681 |

e. Fit the same model in JAGS. Turn in commented code, and comment on whether the two algorithms returned the same results.

```
#given data
Ys <- c(64,72,55,27,75,24,28,66,40,13)
Ns <- c(75,95,63,39,83,26,41,82,54,16)
qs <- c(0.845, 0.847, 0.880, 0.674, 0.909, 0.898, 0.770, 0.801, 0.802, 0.875)
n = 10
# define string model
model_string <- textConnection("model{
```

```
   # Likelihood
    for(i in 1:n){
      Y[i] ~ dbin(theta[i], N[i])
    }
   # Priors
    for(i in 1:n){
      theta[i] ~ dbeta(exp(m)*qs[i], exp(m)*(1-qs[i]))
    }

    m   ~  dnorm(0, 10)
 }")
# Load the data and compile the MCMC code
 inits <- list(theta=qs, m = 0)
 data  <- list(Y = Ys,N = Ns, qs = qs, n = n)
 model <- jags.model(model_string,data = data, inits=inits, n.chains=2)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 10
##    Unobserved stochastic nodes: 11
##    Total graph size: 76
##
## Initializing model
```

```
 #Burn-in for 10000 samples
update(model, 10000, progress.bar="none")
# Generate 20000 post-burn-in samples
```

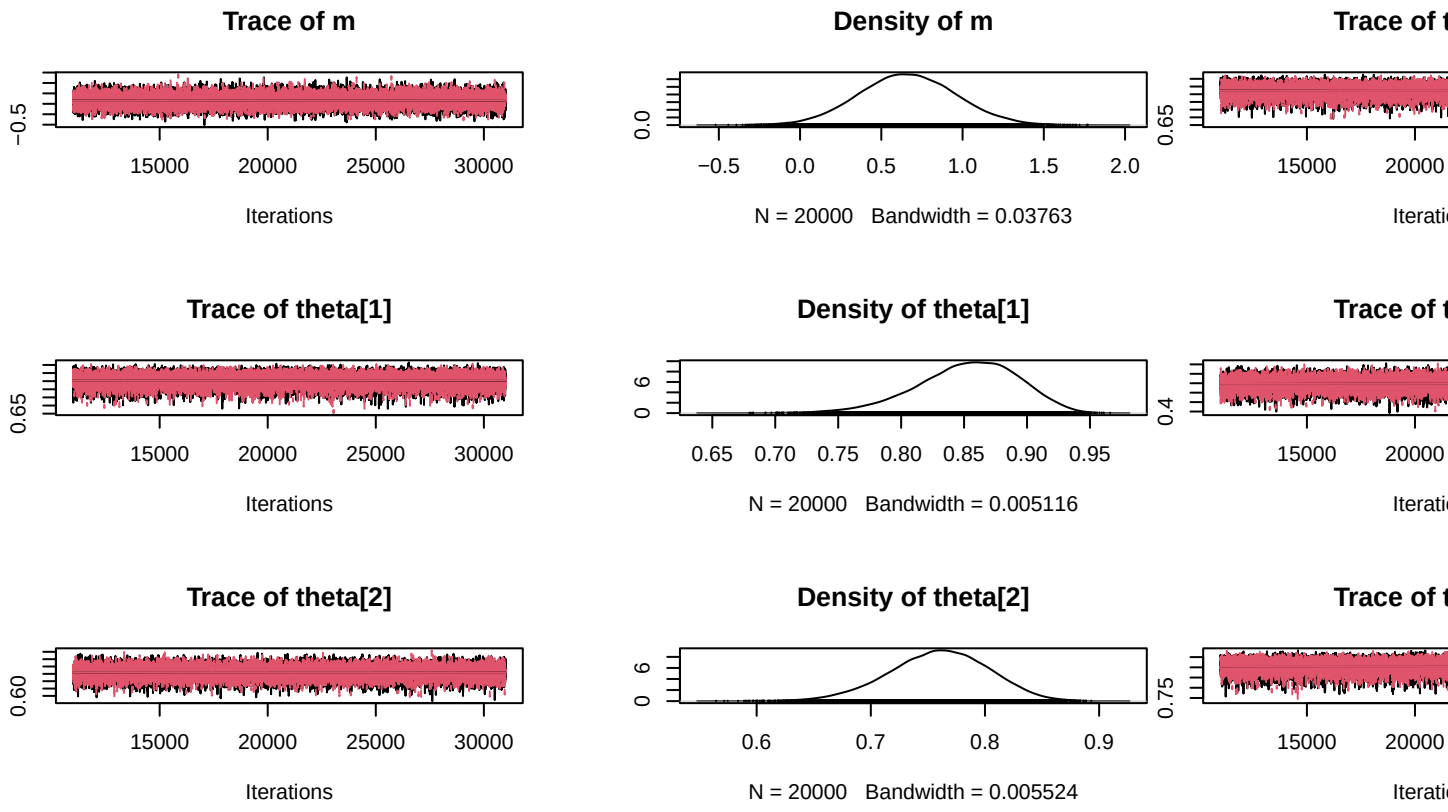
```
 params  <- c("theta","m")
 samples <- coda.samples(model,
            variable.names=params,
            n.iter=20000, progress.bar="none")
```

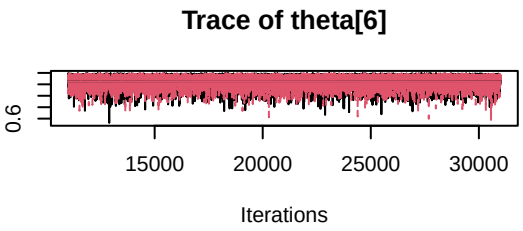
```
 summary(samples)
```

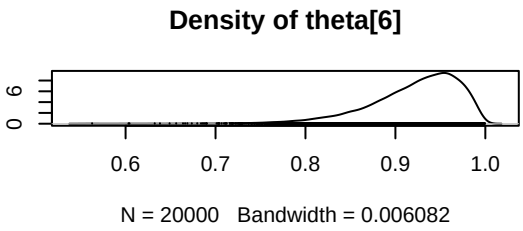
```
##
## Iterations = 11001:31000
## Thinning interval = 1
## Number of chains = 2
## Sample size per chain = 20000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##              Mean      SD  Naive SE Time-series SE
## m          0.6619 0.29559 0.0014779      0.0018409
## theta[1]   0.8528 0.04018 0.0002009      0.0002670
## theta[2]   0.7596 0.04339 0.0002169      0.0002802
## theta[3]   0.8732 0.04088 0.0002044      0.0002797
## theta[4]   0.6912 0.07113 0.0003557      0.0004594
## theta[5]   0.9039 0.03166 0.0001583      0.0002165
## theta[6]   0.9212 0.05005 0.0002503      0.0004110
## theta[7]   0.6871 0.06953 0.0003477      0.0004457
```

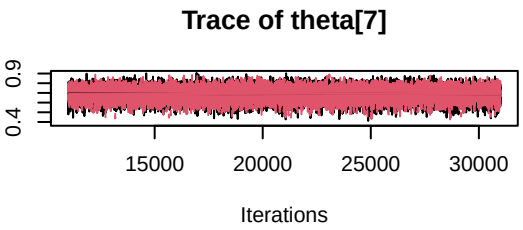
```
## theta[8]  0.8043 0.04312 0.0002156      0.0002834
## theta[9]  0.7437 0.05739 0.0002869      0.0003600
## theta[10] 0.8193 0.08812 0.0004406      0.0006229
##
## 2. Quantiles for each variable:
##
##              2.5%    25%    50%    75%  97.5%
## m         0.08821 0.4607 0.6592 0.8624 1.2487
## theta[1]  0.76598 0.8273 0.8557 0.8816 0.9225
## theta[2]  0.67031 0.7312 0.7610 0.7897 0.8399
## theta[3]  0.78215 0.8480 0.8770 0.9026 0.9416
## theta[4]  0.54436 0.6447 0.6943 0.7416 0.8210
## theta[5]  0.83364 0.8843 0.9070 0.9269 0.9564
## theta[6]  0.79901 0.8945 0.9310 0.9585 0.9882
## theta[7]  0.54301 0.6413 0.6899 0.7360 0.8144
## theta[8]  0.71352 0.7767 0.8068 0.8350 0.8809
## theta[9]  0.62524 0.7063 0.7465 0.7845 0.8471
## theta[10] 0.61713 0.7659 0.8312 0.8856 0.9541
```

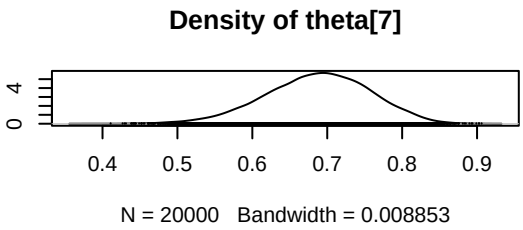
```
plot(samples)
```

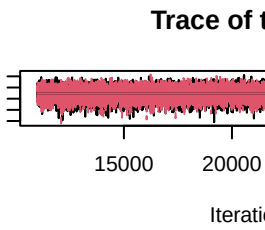


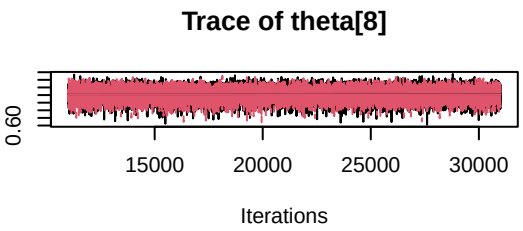


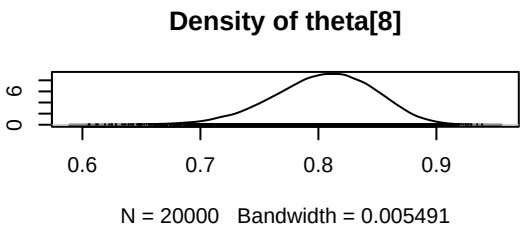



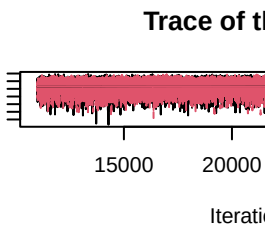




f. What are the advantages and disadvantages of writing your own code as opposed to using JAGS in this problem and in general?

*In general using build-in codes are safest since it considered all optimization aspects in terms of time and communication algorithm. Moreover, I may do type when while writing code.*