ST308 Bayesian Inference Project

Candidate Number: 20652

April 2019

Part 1

(a) We compute fisher information from the likelihood of a Binomial distribution so as to obtain suitable Jeffrey's priors for $\theta^T and \theta^C$

Note that we will carry out computation with the Treatment group with a $Binomial(n_i^T, \theta^T)$ distribution then the Control group should follow the same procedure.

The binomial likelihood is

$$\begin{aligned} p(x_i^T|\theta^T) &= \binom{n_i^T}{x_i^T}(\theta^T)^{x_i^T}(1-\theta^T)^{n_i^T - x_i^T} \\ &\propto (\theta^T)^{x_i^T}(1-\theta^T)^{n_i^T - x_i^T} \end{aligned}$$

and the log-likelihood is

$$\begin{aligned} l &= log(p(x_i^T|\theta^T)) \\ &\propto x_i^T log(\theta^T) + (n_i^T - x_i^T) log(1-\theta^T) \end{aligned}$$

Compute first and second derivatives

$$\begin{aligned} \frac{\partial l}{\partial \theta^T} &= \frac{x_i^T}{\theta^T} - \frac{n_i^T - x_i^T}{1-\theta^T} \\ \frac{\partial^2 l}{\partial \theta^{T2}} &= -\frac{x_i^T}{\theta^T} - \frac{n_i^T - x_i^T}{(1-\theta^T)^2} \end{aligned}$$

Then the Fisher information is

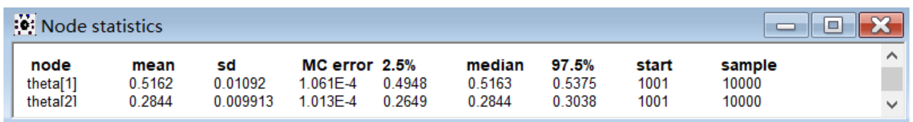
$$\begin{aligned} I(\theta^T) &= -E(\frac{\partial^2 l}{\partial \theta^{T2}}|\theta^T) \\ &= \frac{n_i^T\theta^T}{\theta^T} + \frac{n_i^T - n_i^T\theta^T}{(1-\theta^T)^2} \\ &= \frac{n_i^T}{\theta^T(1-\theta^T)} \\ &\propto (\theta^T)^{-1}(1-\theta^T)^{-1} \end{aligned}$$

Therefore the Jeffrey's prior is

$$\begin{aligned}\pi(\theta^T) &= \sqrt{I(\theta^T)} \\ &\propto (\theta^T)^{-1/2}(1-\theta^T)^{-1/2} \\ &\stackrel{\mathrm{D}}{=} Beta(\frac{1}{2},\frac{1}{2})\end{aligned}$$

Computation for the Jeffrey's prior for θ^C follows the same procedure. Consequently, $Beta(\frac{1}{2},\frac{1}{2})$ distribution is the suitable prior for θ^T and θ^C.

(b)



Node statistics

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
theta[1]	0.5162	0.01092	1.061E-4	0.4948	0.5163	0.5375	1001	10000
theta[2]	0.2844	0.009913	1.013E-4	0.2649	0.2844	0.3038	1001	10000

Figure 1: Statistics output

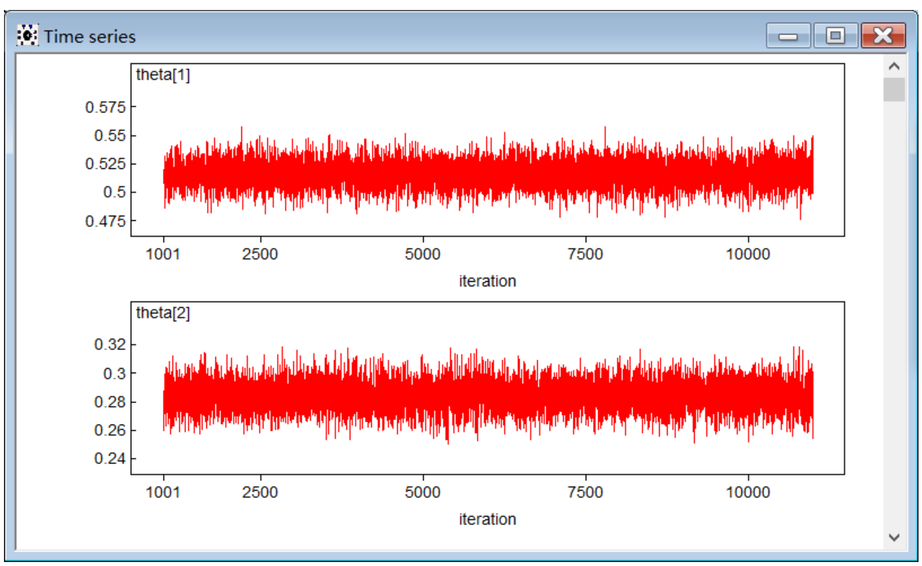


Figure 2: History output

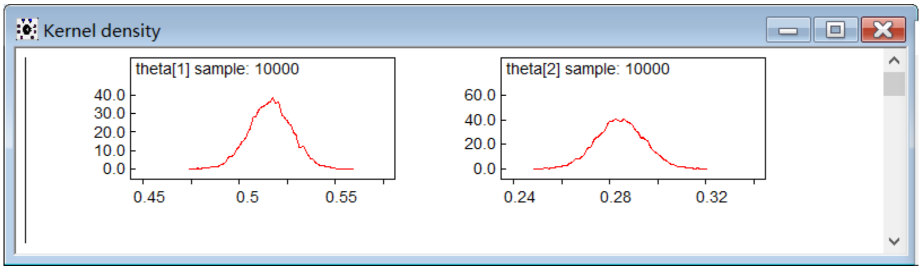


Figure 3: Density output

As per output from WinBUGS, the medians as estimates for θ^T, θ^C are 0.5163 and 0.2844. The 95-percent credible intervals for θ^T, θ^C are [0.4948, 0.5375] and [0.2649, 0.3038], which means the two parameters fall into the above interval with a probability of 0.95. Sample trace history of the two parameters are shown by Figure 2 and the posterior density of the sample is shown by Figure 3.

θ^T, θ^C are parameters in the Binomial distribution for the Treatment group and Control group and represent the probability of a patient feeling relieved after taking a pill. (a pain reliever for the Treatment and a placebo for the Control).

(c) Since there is no sign of correlations between θ^T and θ^C, we will assume them to be independent of each other.

Compute the joint prior of the two parameters

$$\pi(\theta^T, \theta^C) \propto (\theta^T)^{-1/2}(1-\theta^T)^{-1/2}(\theta^C)^{-1/2}(1-\theta^C)^{-1/2}$$

Compute the joint density

$$p(X^T, X^C|\theta^T, \theta^C) \propto (\theta^T)^{\sum_{i=1}^{30} x_i^T}(1-\theta^T)^{\sum_{i=1}^{30}(n_i^T - x_i^T)}(\theta^C)^{\sum_{i=1}^{30} x_i^C}(1-\theta^C)^{\sum_{i=1}^{30}(n_i^C - x_i^C)}$$

Compute the joint posterior

$$\begin{aligned}\pi(\theta^T, \theta^C|X^T, X^C) &\propto \pi(\theta^T, \theta^C)p(X^T, X^C|\theta^T, \theta^C) \\ &\propto (\theta^T)^{-\frac{1}{2}+\sum_{i=1}^{30} x_i^T}(1-\theta^T)^{-\frac{1}{2}+\sum_{i=1}^{30}(n_i^T - x_i^T)}(\theta^C)^{-\frac{1}{2}+\sum_{i=1}^{30} x_i^C}(1-\theta^C)^{-\frac{1}{2}+\sum_{i=1}^{30}(n_i^C - x_i^C)}\end{aligned}$$

Hence, we derive the full conditional posterior for θ^T and θ^C

$$\begin{aligned}\pi(\theta^T|X^T, X^C, \theta^C) &\propto (\theta^T)^{-\frac{1}{2}+\sum_{i=1}^{30} x_i^T}(1-\theta^T)^{-\frac{1}{2}+\sum_{i=1}^{30}(n_i^T - x_i^T)} \\ &\stackrel{D}{=} Beta(\frac{1}{2} + \sum_{i=1}^{30} x_i^T, \frac{1}{2} + \sum_{i=1}^{30}(n_i^T - x_i^T)) \\ \pi(\theta^C|X^T, X^C, \theta^T) &\propto (\theta^C)^{-\frac{1}{2}+\sum_{i=1}^{30} x_i^C}(1-\theta^C)^{-\frac{1}{2}+\sum_{i=1}^{30}(n_i^C - x_i^C)} \\ &\stackrel{D}{=} Beta(\frac{1}{2} + \sum_{i=1}^{30} x_i^C, \frac{1}{2} + \sum_{i=1}^{30}(n_i^C - x_i^C))\end{aligned}$$

(d)

```
> summary(out_theta.T[1001:10000])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.4725  0.5089  0.5163  0.5162  0.5235  0.5575 
> summary(out_theta.C[1001:10000])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.2503  0.2781  0.2845  0.2846  0.2911  0.3207 
> quantile(out_theta.T[1001:10000], probs = c(0.025, 0.975))
     2.5%     97.5% 
0.4951205 0.5374235 
> quantile(out_theta.C[1001:10000], probs = c(0.025, 0.975))
     2.5%     97.5% 
0.2660754 0.3038453 
```

Figure 4: Statistics output from R

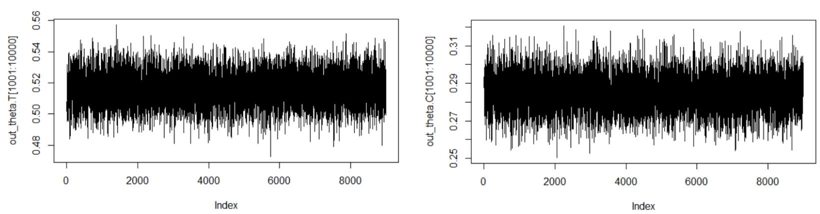


Figure 5: History output from R

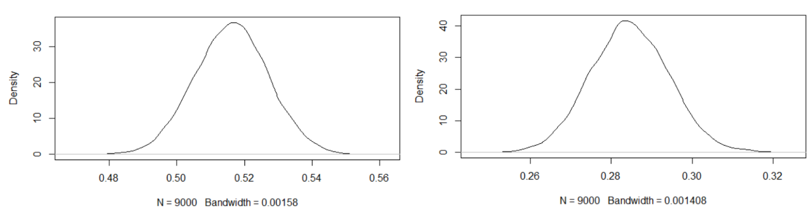


Figure 6: Density output from R

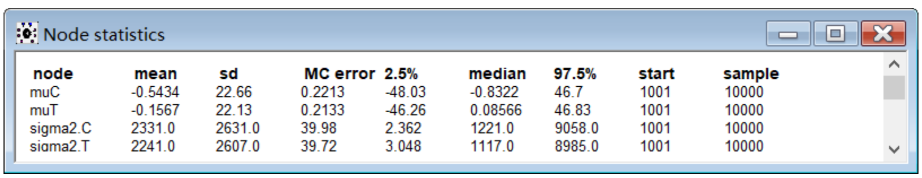
It can be observed that both numerical and graphical results from R conform with those from WinBUGS.

(e) For sampling, we have omitted the first 1000 samples and are looking at the 1001 to 10000 samples. Figure 2, the history output plots, show that the two parameters show reasonable convergence for the 1001^{th} to the 10000^{th} iteration. Figure 3, the posterior density plots demonstrate that there are perceivable differences between θ^T and θ^C. Moreover, it can be observed that the posterior density of θ^T is more centered around its mode relative to that of θ^C. Numerical results, including medians of 0.5163 and 0.2844 and 95-percent credible intervals of [0.4948, 0.5375] and [0.2649, 0.3038], are also disparate between θ^T and θ^C.

From the estimates, credible intervals and graphics from both WinBUGS and R, there appears to be a noticeable difference between θ^T and θ^C, where θ^T seems significantly higher. This result might indicate the actual effectiveness of the pain reliever.

Part 2

(b)



Node statistics

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
muC	-0.5434	22.66	0.2213	-48.03	-0.8322	46.7	1001	10000
muT	-0.1567	22.13	0.2133	-46.26	0.08566	46.83	1001	10000
sigma2.C	2331.0	2631.0	39.98	2.362	1221.0	9058.0	1001	10000
sigma2.T	2241.0	2607.0	39.72	3.048	1117.0	8985.0	1001	10000

Figure 7: Statistics output

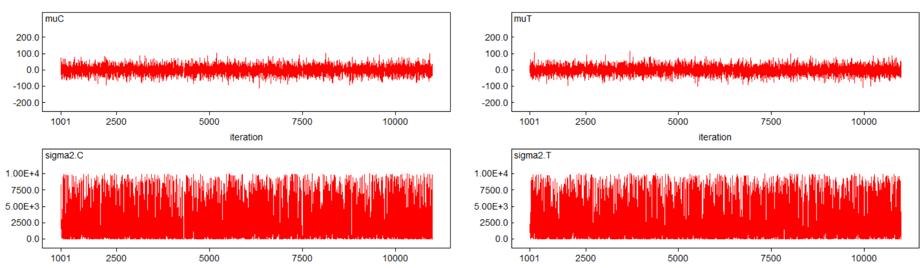


Figure 8: History output

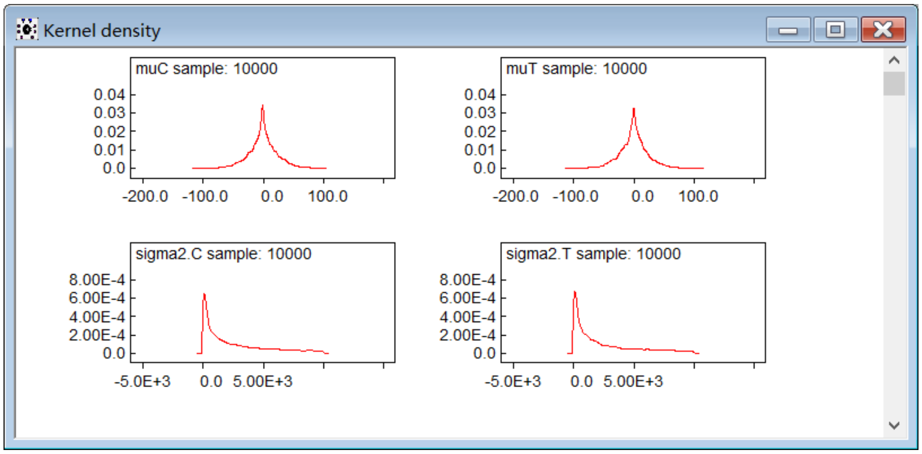


Figure 9: Density output

As per output from WinBUGS, the medians as estimates for μ_T and μ_C are 0.08566 and -0.8322, for σ_T^2 and σ_C^2 are 1117.0 and 1221.0. The 95-percent credible intervals for μ_T and μ_C are [-46.26, 46.83] and [-48.03, 46.7], for σ_T^2 and σ_C^2 are [3.048, 8985.0] and [2.362, 9058.0]. Sample trace history of the parameters are shown by Figure 8 and the posterior density of the sample is shown by Figure 9.

Estimates for μ_T and μ_C represent the expected values of logit functions of θ_i^T and θ_i^C. Estimates for σ_T^2 and σ_C^2 demonstrate how much their logit functions fluctuate about their expected levels.

(c) If we take the medians as estimates for μ_T and μ_C, which are the means for the normal distributions for the logit functions of θ_i^T and θ_i^C, this implies that

$$E[log(\frac{\theta_i^T}{1-\theta_i^T})] = 0.08566$$
$$E[log(\frac{\theta_i^C}{1-\theta_i^C})] = -0.8322$$

Then we may observe the expectations of θ_i^T and θ_i^C by less rigorous approximation

$$E(\theta_i^T) \approx 0.521$$
$$E(\theta_i^C) \approx 0.303$$

Therefore, it may still be concluded that there appears to be a noticeable difference between θ_i^T and θ_i^C, where θ_i^T seems higher. This result might indicate the effectiveness of the pain reliever. Moreover, the variance of the logit function for the Treatment group appears to be lower than that of the Control group, which indicates that θ_i^Ts fluctuate less around its expectation than θ_i^Cs do. This may further confirm the effectiveness of the pain reliever as it is intuitive to believe that hospitals' individual characteristics should have less influence if the pain reliever brings genuine effects.

(d)

```
#Loading data
nT  = c(63, 75, 61, 71, 68, 86, 76, 75, 69, 63, 67, 80, 70, 68, 57,
        68, 69, 86, 64, 75, 60, 63, 70, 77, 78, 67, 70, 73, 72, 69)

xT = c(46, 16, 43, 37, 38, 27, 68, 61, 22, 58, 25, 16, 35, 15, 36,
       25, 39, 61, 37, 35, 33, 18, 22, 66, 13, 56, 49, 38, 53, 1)

nC = c(63, 75, 61, 71, 68, 86, 76, 75, 69, 63, 67, 80, 70, 68, 57,
       68, 69, 86, 64, 75, 60, 63, 70, 77, 78, 67, 70, 73, 72, 69)

xC = c(51, 12, 30, 22, 5, 26, 33, 68, 0, 8, 57, 21, 69, 23, 0, 6,
       1, 4, 59, 4, 2, 5, 1, 1, 16, 6, 15, 42, 0, 13)

#Initialising for Metropolis
Niter = 10000
muT_init = 0
muC_init = 0
sigmaT_init = 1
sigmaC_init = 1
thetaT_init = 0.5
thetaC_init = 0.5
thetaT = matrix(nrow = 1, ncol = 30)
thetaC = matrix(nrow = 1, ncol = 30)
out_muT = matrix(nrow= Niter, ncol = 1)
out_muC = matrix(nrow= Niter, ncol = 1)
out_sigmaT = matrix(nrow= Niter, ncol = 1)
out_sigmaC = matrix(nrow= Niter, ncol = 1)
out_thetaT = matrix(nrow = 0, ncol = 30)
out_thetaC = matrix(nrow = 0, ncol = 30)
muT = muT_init
muC = muC_init
sigmaT = sigmaT_init
sigmaC = sigmaT_init
thetaT[1:30] = thetaT_init
thetaC[1:30] = thetaC_init
logit.T = log(thetaT / (1-thetaT))
logit.C = log(thetaC / (1-thetaC))
S.muT = 1.2
S.muC = 2.5
S.sigmaT = 0.85
S.sigmaC = 1.8
S.thetaT = matrix(nrow = 1, ncol = 30)
S.thetaC = matrix(nrow = 1, ncol = 30)
S.thetaT[1:30] = 1.35
S.thetaT[30] = 2.6
S.thetaC[1:30] = 1.35
S.thetaC[9] = 4
S.thetaC[13] = 2.5
S.thetaC[15] = 4
S.thetaC[17] = 2.5
S.thetaC[18] = 2
S.thetaC[20] = 2
S.thetaC[21] = 2.5
S.thetaC[23] = 3
S.thetaC[24] = 3
S.thetaC[29] = 4
```

```
#Random walk Metropolis algorithms
for (iter in 1:Niter){
out_muT[iter] = muT
out_muC[iter] = muC
out_sigmaT[iter] = sigmaT
out_sigmaC[iter] = sigmaC
out_thetaT = rbind(out_thetaT, thetaT)
out_thetaC = rbind(out_thetaC, thetaC)

#Updating muT
muT1 = rnorm(1, muT, S.muT)

logPost1 = -((10^3*sum((logit.T - muT1)^2) + (muT1^2)*(sigmaT^2))/(2*10^3*(sigmaT^2)))
logPost = -((10^3*sum((logit.T - muT)^2) + (muT^2)*(sigmaT^2))/(2*10^3*(sigmaT^2)))
logAcceptProb=logPost1-logPost

U=runif(1)
if (log(U)<logAcceptProb)
{
  muT = muT1
}

#Updating muC
muC1 = rnorm(1, muC, S.muC)

logPost1 = -((10^3*sum((logit.C - muC1)^2) + (muC1^2)*(sigmaC^2))/(2*10^3*(sigmaC^2)))
logPost = -((10^3*sum((logit.C - muC)^2) + (muC^2)*(sigmaC^2))/(2*10^3*(sigmaC^2)))
logAcceptProb=logPost1-logPost

U=runif(1)
if (log(U)<logAcceptProb)
{
  muC = muC1
}

#Updating sigmaT
a = rnorm(1, sigmaT, S.sigmaT)
if(a < 0){
  a = rnorm(1, sigmaT, S.sigmaT)
} else {
  sigmaT1 = a
}

logPost1 = -30*log(sigmaT1) - ((10^3*sum((logit.T - muT)^2) + (muT^2)*(sigmaT1^2))/(2*10^3*(sigmaT1^2)))
logPost = -30*log(sigmaT) - ((10^3*sum((logit.T - muT)^2) + (muT^2)*(sigmaT^2))/(2*10^3*(sigmaT^2)))
logAcceptProb=logPost1-logPost

U=runif(1)
if (log(U)<logAcceptProb)
{
  sigmaT = sigmaT1
}

#Updating sigmaC
b = rnorm(1, sigmaC, S.sigmaC)
if(b < 0){
  b = rnorm(1, sigmaC, S.sigmaC)
} else {
  sigmaC1 = b
}

logPost1 = -30*log(sigmaC1) - ((10^3*sum((logit.C - muC)^2) + (muC^2)*(sigmaC1^2))/(2*10^3*(sigmaC1^2)))
logPost = -30*log(sigmaC) - ((10^3*sum((logit.C - muC)^2) + (muC^2)*(sigmaC^2))/(2*10^3*(sigmaC^2)))
logAcceptProb=logPost1-logPost

U=runif(1)
if (log(U)<logAcceptProb)
{
  sigmaC = sigmaC1
}

#Updating thetaT
for (i in 1:30){
logit.Ti1 = rnorm(1, logit.T[i], S.thetaT[i])
thetaTi1 = exp(logit.Ti1) / (exp(logit.Ti1) + 1)

logPost1 = -((10^3*(logit.Ti1 - muT)^2)/(2*10^3*(sigmaT^2))) +
  xT[i]*log(thetaTi1) + (nT[i] - xT[i])*log(1 - thetaTi1)
logPost = -((10^3*(logit.T[i] - muT)^2)/(2*10^3*(sigmaT^2))) +
  xT[i]*log(thetaT[i]) + (nT[i] - xT[i])*log(1 - thetaT[i])
logAcceptProb=logPost1-logPost

U=runif(1)
if (log(U)<logAcceptProb)
{
  logit.T[i] = logit.Ti1
  thetaT[i] = thetaTi1
}
}
```

```r
#Updating thetaC
for (i in 1:30){
logit.Ci1 = rnorm(1, logit.C[i], S.thetaC[i])
thetaCi1 = exp(logit.Ci1) / (exp(logit.Ci1) + 1)

logPost1 = -((10^3*(logit.Ci1 - muC)^2)/(2*10^3*(sigmaC^2))) +
  xC[i]*log(thetaCi1) + (nC[i] - xC[i])*log(1 - thetaCi1)
logPost = -((10^3*(logit.C[i] - muC)^2)/(2*10^3*(sigmaC^2))) +
  xC[i]*log(thetaC[i]) + (nC[i] - xC[i])*log(1 - thetaC[i])
logAcceptProb=logPost1-logPost

U=runif(1)
if (log(U)<logAcceptProb)
{
  logit.C[i] = logit.Ci1
  thetaC[i] = thetaCi1
}
}
}

#Check acceptance rate and adjust standard deviation for the proposal distributions
AccRateMuT = (Niter-sum(diff(out_muT)==0))/Niter
AccRateMuC = (Niter-sum(diff(out_muC)==0))/Niter
AccRateSigmaT = (Niter-sum(diff(out_sigmaT)==0))/Niter
AccRateSigmaC = (Niter-sum(diff(out_sigmaC)==0))/Niter
AccRateThetaT = matrix(nrow = 1, ncol = 30)
AccRateThetaC = matrix(nrow = 1, ncol = 30)
for (i in 1:30){
AccRateThetaT[i] = (Niter-sum(diff(out_thetaT[1:Niter, i])==0))/Niter
}
for (i in 1:30){
AccRateThetaC[i] = (Niter-sum(diff(out_thetaC[1:Niter, i])==0))/Niter
}

#Plotting sample trace
plot(out_muT[1001:10000],type="l")
plot(out_muC[1001:10000],type="l")
plot(out_sigmaT[1001:10000],type="l")
plot(out_sigmaC[1001:10000],type="l")

#Plotting posterior sample density
plot(density(out_muT[1001:10000]))
plot(density(out_muC[1001:10000]))
plot(density(out_sigmaT[1001:10000]))
plot(density(out_sigmaC[1001:10000]))

#Checking estimates and credible intervals
summary(out_muT[1001:10000])
summary(out_muC[1001:10000])
summary(out_sigmaT[1001:10000])
summary(out_sigmaC[1001:10000])
quantile(out_muT[1001:10000], probs = c(0.025, 0.975))
quantile(out_muC[1001:10000], probs = c(0.025, 0.975))
quantile(out_sigmaT[1001:10000], probs = c(0.025, 0.975))
quantile(out_sigmaC[1001:10000], probs = c(0.025, 0.975))
```

Appendix

WinBUGS Code for Part 1

```
#Model
model
{
for(i in 1 : 30){
xT[i] ~ dbin(theta[1], nT[i])
xC[i] ~ dbin(theta[2], nC[i])
}
theta[1] ~ dbeta(0.5, 0.5)
theta[2] ~ dbeta(0.5, 0.5)
}

#Data
list(nT  = c(63, 75, 61, 71, 68, 86, 76, 75, 69, 63, 67, 80, 70, 68, 57,
68, 69, 86, 64, 75, 60, 63, 70, 77, 78, 67, 70, 73, 72, 69),

xT = c(46, 16, 43, 37, 38, 27, 68, 61, 22, 58, 25, 16, 35, 15, 36,
25, 39, 61, 37, 35, 33, 18, 22, 66, 13, 56, 49, 38, 53, 1),

nC = c(63, 75, 61, 71, 68, 86, 76, 75, 69, 63, 67, 80, 70, 68, 57,
68, 69, 86, 64, 75, 60, 63, 70, 77, 78, 67, 70, 73, 72, 69),

xC = c(51, 12, 30, 22, 5, 26, 33, 68, 0, 8, 57, 21, 69, 23, 0, 6,
1, 4, 59, 4, 2, 5, 1, 1, 16, 6, 15, 42, 0, 13))
```

R Code for Part 1

```r
#Loading data
nT  = c(63, 75, 61, 71, 68, 86, 76, 75, 69, 63, 67, 80, 70, 68, 57,
        68, 69, 86, 64, 75, 60, 63, 70, 77, 78, 67, 70, 73, 72, 69)

xT = c(46, 16, 43, 37, 38, 27, 68, 61, 22, 58, 25, 16, 35, 15, 36,
       25, 39, 61, 37, 35, 33, 18, 22, 66, 13, 56, 49, 38, 53, 1)

nC = c(63, 75, 61, 71, 68, 86, 76, 75, 69, 63, 67, 80, 70, 68, 57,
       68, 69, 86, 64, 75, 60, 63, 70, 77, 78, 67, 70, 73, 72, 69)

xC = c(51, 12, 30, 22, 5, 26, 33, 68, 0, 8, 57, 21, 69, 23, 0, 6,
       1, 4, 59, 4, 2, 5, 1, 1, 16, 6, 15, 42, 0, 13)

#Computing sums
sum.nT = sum(nT)
sum.xT = sum(xT)
sum.nC = sum(nC)
sum.xC = sum(xC)

#Initialising sampling
Niter = 10000
theta.T_init = 0.5
theta.C_init = 0.5
out_theta.T = matrix(nrow = Niter,ncol = 1)
out_theta.C = matrix(nrow = Niter,ncol = 1)
theta.T = theta.T_init
theta.C = theta.C_init

#Sampling from posterior distributions
for (iter in 1:Niter){
out_theta.T[iter] = theta.T
out_theta.C[iter] = theta.C

theta.T = rbeta(1, 0.5+sum.xT, 0.5+sum.nT-sum.xT)
theta.C = rbeta(1, 0.5+sum.xC, 0.5+sum.nC-sum.xC)
}

#Plotting sample trace (history output in WinBUGS)
plot(out_theta.T[1001:10000],type="l")
plot(out_theta.C[1001:10000],type="l")

#Plotting posterior sample density (density output in WinBUGS)
plot(density(out_theta.T[1001:10000]))
plot(density(out_theta.C[1001:10000]))

#Checking estimates and credible intervals (statistics output in WinBUGS)
summary(out_theta.T[1001:10000])
summary(out_theta.C[1001:10000])
quantile(out_theta.T[1001:10000], probs = c(0.025, 0.975))
quantile(out_theta.C[1001:10000], probs = c(0.025, 0.975))
```

WinBUGS Code for Part 2

```
#Model
model
{
for(i in 1 : 30){
logit(thetaT[i]) <- t
logit(thetaC[i]) <- c
xT[i] ~ dbin(thetaT[i], nT[i])
xC[i] ~ dbin(thetaC[i], nC[i])
}

#Priors for logit, muT, muC, sigma2.T and sigma2.C
t ~ dnorm(muT, tauT)
c ~ dnorm(muC, tauC)
muT ~ dnorm(0, 1.0E-3)
muC ~ dnorm(0, 1.0E-3)
tauT <- 1/(sigma2.T)
tauC <- 1/(sigma2.C)
sigma.T ~ dunif(0,100)
sigma.C ~ dunif(0,100)
sigma2.T <- sigma.T*sigma.T
sigma2.C <- sigma.C*sigma.C
}

#Data

list(nT  = c(63, 75, 61, 71, 68, 86, 76, 75, 69, 63, 67, 80, 70, 68, 57,
68, 69, 86, 64, 75, 60, 63, 70, 77, 78, 67, 70, 73, 72, 69),

xT = c(46, 16, 43, 37, 38, 27, 68, 61, 22, 58, 25, 16, 35, 15, 36,
25, 39, 61, 37, 35, 33, 18, 22, 66, 13, 56, 49, 38, 53, 1),

nC = c(63, 75, 61, 71, 68, 86, 76, 75, 69, 63, 67, 80, 70, 68, 57,
68, 69, 86, 64, 75, 60, 63, 70, 77, 78, 67, 70, 73, 72, 69),

xC = c(51, 12, 30, 22, 5, 26, 33, 68, 0, 8, 57, 21, 69, 23, 0, 6,
1, 4, 59, 4, 2, 5, 1, 1, 16, 6, 15, 42, 0, 13))
```

R Code for Part 2

```r
#Loading data
nT   = c(63, 75, 61, 71, 68, 86, 76, 75, 69, 63, 67, 80, 70, 68, 57,
         68, 69, 86, 64, 75, 60, 63, 70, 77, 78, 67, 70, 73, 72, 69)

xT = c(46, 16, 43, 37, 38, 27, 68, 61, 22, 58, 25, 16, 35, 15, 36,
       25, 39, 61, 37, 35, 33, 18, 22, 66, 13, 56, 49, 38, 53, 1)

nC = c(63, 75, 61, 71, 68, 86, 76, 75, 69, 63, 67, 80, 70, 68, 57,
       68, 69, 86, 64, 75, 60, 63, 70, 77, 78, 67, 70, 73, 72, 69)

xC = c(51, 12, 30, 22, 5, 26, 33, 68, 0, 8, 57, 21, 69, 23, 0, 6,
       1, 4, 59, 4, 2, 5, 1, 1, 16, 6, 15, 42, 0, 13)

#Initialising for Metropolis
Niter = 10000
muT_init = 0
muC_init = 0
sigmaT_init = 1
sigmaC_init = 1
thetaT_init = 0.5
thetaC_init = 0.5
thetaT = matrix(nrow = 1, ncol = 30)
thetaC = matrix(nrow = 1, ncol = 30)
out_muT = matrix(nrow= Niter, ncol = 1)
out_muC = matrix(nrow= Niter, ncol = 1)
out_sigmaT = matrix(nrow= Niter, ncol = 1)
out_sigmaC = matrix(nrow= Niter, ncol = 1)
out_thetaT = matrix(nrow = 0, ncol = 30)
out_thetaC = matrix(nrow = 0, ncol = 30)
muT = muT_init
muC = muC_init
sigmaT = sigmaT_init
sigmaC = sigmaT_init
thetaT[1:30] = thetaT_init
thetaC[1:30] = thetaC_init
logit.T = log(thetaT / (1-thetaT))
logit.C = log(thetaC / (1-thetaC))
S.muT = 1.2
S.muC = 2.5
S.sigmaT = 0.85
S.sigmaC = 1.8
S.thetaT = matrix(nrow = 1, ncol = 30)
S.thetaC = matrix(nrow = 1, ncol = 30)
S.thetaT[1:30] = 1.35
S.thetaT[30] = 2.6
S.thetaC[1:30] = 1.35
S.thetaC[9] = 4
S.thetaC[13] = 2.5
S.thetaC[15] = 4
S.thetaC[17] = 2.5
S.thetaC[18] = 2
S.thetaC[20] = 2
S.thetaC[21] = 2.5
S.thetaC[23] = 3
S.thetaC[24] = 3
S.thetaC[29] = 4
```

```r
#Random walk Metropolis algorithms
for (iter in 1:Niter){
out_muT[iter] = muT
out_muC[iter] = muC
out_sigmaT[iter] = sigmaT
out_sigmaC[iter] = sigmaC
out_thetaT = rbind(out_thetaT, thetaT)
out_thetaC = rbind(out_thetaC, thetaC)

#Updating muT
muT1 = rnorm(1, muT, S.muT)

logPost1 = -((10^3*sum((logit.T - muT1)^2) + (muT1^2)*(sigmaT^2))/(2*10^3*(sigmaT^2)))
logPost = -((10^3*sum((logit.T - muT)^2) + (muT^2)*(sigmaT^2))/(2*10^3*(sigmaT^2)))
logAcceptProb=logPost1-logPost

U=runif(1)
if (log(U)<logAcceptProb)
{
  muT = muT1
}

#Updating muC
muC1 = rnorm(1, muC, S.muC)

logPost1 = -((10^3*sum((logit.C - muC1)^2) + (muC1^2)*(sigmaC^2))/(2*10^3*(sigmaC^2)))
logPost = -((10^3*sum((logit.C - muC)^2) + (muC^2)*(sigmaC^2))/(2*10^3*(sigmaC^2)))
logAcceptProb=logPost1-logPost

U=runif(1)
if (log(U)<logAcceptProb)
{
  muC = muC1
}

#Updating sigmaT
a = rnorm(1, sigmaT, S.sigmaT)
if(a < 0){
  a = rnorm(1, sigmaT, S.sigmaT)
} else {
  sigmaT1 = a
}

logPost1 = -30*log(sigmaT1) - ((10^3*sum((logit.T - muT)^2) + (muT^2)*(sigmaT1^2))/(2*10^3*(sigmaT1^2)))
logPost = -30*log(sigmaT) - ((10^3*sum((logit.T - muT)^2) + (muT^2)*(sigmaT^2))/(2*10^3*(sigmaT^2)))
logAcceptProb=logPost1-logPost

U=runif(1)
if (log(U)<logAcceptProb)
{
  sigmaT = sigmaT1
}

#Updating sigmaC
b = rnorm(1, sigmaC, S.sigmaC)
if(b < 0){
  b = rnorm(1, sigmaC, S.sigmaC)
} else {
  sigmaC1 = b
}

logPost1 = -30*log(sigmaC1) - ((10^3*sum((logit.C - muC)^2) + (muC^2)*(sigmaC1^2))/(2*10^3*(sigmaC1^2)))
logPost = -30*log(sigmaC) - ((10^3*sum((logit.C - muC)^2) + (muC^2)*(sigmaC^2))/(2*10^3*(sigmaC^2)))
logAcceptProb=logPost1-logPost

U=runif(1)
if (log(U)<logAcceptProb)
{
  sigmaC = sigmaC1
}

#Updating thetaT
for (i in 1:30){
logit.Ti1 = rnorm(1, logit.T[i], S.thetaT[i])
thetaTi1 = exp(logit.Ti1) / (exp(logit.Ti1) + 1)

logPost1 = -((10^3*(logit.Ti1 - muT)^2)/(2*10^3*(sigmaT^2))) +
  xT[i]*log(thetaTi1) + (nT[i] - xT[i])*log(1 - thetaTi1)
logPost = -((10^3*(logit.T[i] - muT)^2)/(2*10^3*(sigmaT^2))) +
  xT[i]*log(thetaT[i]) + (nT[i] - xT[i])*log(1 - thetaT[i])
logAcceptProb=logPost1-logPost

U=runif(1)
if (log(U)<logAcceptProb)
{
  logit.T[i] = logit.Ti1
  thetaT[i] = thetaTi1
}
}
```

```
#Updating thetaC
for (i in 1:30){
logit.Ci1 = rnorm(1, logit.C[i], S.thetaC[i])
thetaCi1 = exp(logit.Ci1) / (exp(logit.Ci1) + 1)

logPost1 = -((10^3*(logit.Ci1 - muC)^2)/(2*10^3*(sigmaC^2))) +
  xC[i]*log(thetaCi1) + (nC[i] - xC[i])*log(1 - thetaCi1)
logPost = -((10^3*(logit.C[i] - muC)^2)/(2*10^3*(sigmaC^2))) +
  xC[i]*log(thetaC[i]) + (nC[i] - xC[i])*log(1 - thetaC[i])
logAcceptProb=logPost1-logPost

U=runif(1)
if (log(U)<logAcceptProb)
{
  logit.C[i] = logit.Ci1
  thetaC[i] = thetaCi1
}
}
}

#Check acceptance rate and adjust standard deviation for the proposal distributions
AccRateMuT = (Niter-sum(diff(out_muT)==0))/Niter
AccRateMuC = (Niter-sum(diff(out_muC)==0))/Niter
AccRateSigmaT = (Niter-sum(diff(out_sigmaT)==0))/Niter
AccRateSigmaC = (Niter-sum(diff(out_sigmaC)==0))/Niter
AccRateThetaT = matrix(nrow = 1, ncol = 30)
AccRateThetaC = matrix(nrow = 1, ncol = 30)
for (i in 1:30){
AccRateThetaT[i] = (Niter-sum(diff(out_thetaT[1:Niter, i])==0))/Niter
}
for (i in 1:30){
AccRateThetaC[i] = (Niter-sum(diff(out_thetaC[1:Niter, i])==0))/Niter
}

#Plotting sample trace
plot(out_muT[1001:10000],type="l")
plot(out_muC[1001:10000],type="l")
plot(out_sigmaT[1001:10000],type="l")
plot(out_sigmaC[1001:10000],type="l")

#Plotting posterior sample density
plot(density(out_muT[1001:10000]))
plot(density(out_muC[1001:10000]))
plot(density(out_sigmaT[1001:10000]))
plot(density(out_sigmaC[1001:10000]))

#Checking estimates and credible intervals
summary(out_muT[1001:10000])
summary(out_muC[1001:10000])
summary(out_sigmaT[1001:10000])
summary(out_sigmaC[1001:10000])
quantile(out_muT[1001:10000], probs = c(0.025, 0.975))
quantile(out_muC[1001:10000], probs = c(0.025, 0.975))
quantile(out_sigmaT[1001:10000], probs = c(0.025, 0.975))
quantile(out_sigmaC[1001:10000], probs = c(0.025, 0.975))
```