Computational Statistics CW 3
MSc in Statistics, Imperial College London

CID: 01855742

Contents

1 Question 1 **1**
1.1 a 1
1.2 b 2
1.3 Part c 2
1.4 Part d: 9

2 Question 2 **14**
2.1 a 14
2.2 b 15
2.3 c 15
2.4 d: 19

3 Code appendix **21**

4 References **29**

I, CID: 01855742, certify that this assessed coursework is my own work, unless otherwise acknowledged, and includes no plagiarism. I have not discussed my coursework with anyone else except when seeking clarification with the module lecturer via email or on MS Teams. I have not shared any code underlying my coursework with anyone else prior to submission.

1 Question 1

1.1 a

Using Bayes theorem, the posterior distribution can be written as:

$$\pi(p_1, p_2|Y) \propto \left(\prod_{i=1}^{n} P(Y_i|p_1, p_2)\right) \cdot \pi(p_1) \cdot \pi(p_2)$$

Substituting the prior and likelihood:

$$\pi(p_1, p_2|Y) \propto \left(\prod_{i=1}^{n}(p_1p_2)^{Y_i} \cdot (1-p_1p_2)^{1-Y_i}\right) \cdot 1_{[0,1]}(p_1) \cdot 1_{[0,1]}(p_2)$$

Using rules of exponents:

$$\pi(p_1, p_2|Y) \propto (p_1p_2)^{\sum_{i=1}^{n} Y_i} \cdot (1-p_1p_2)^{\sum_{i=1}^{n}(1-Y_i)} \cdot 1_{[0,1]}(p_1) \cdot 1_{[0,1]}(p_2)$$

1.2 b

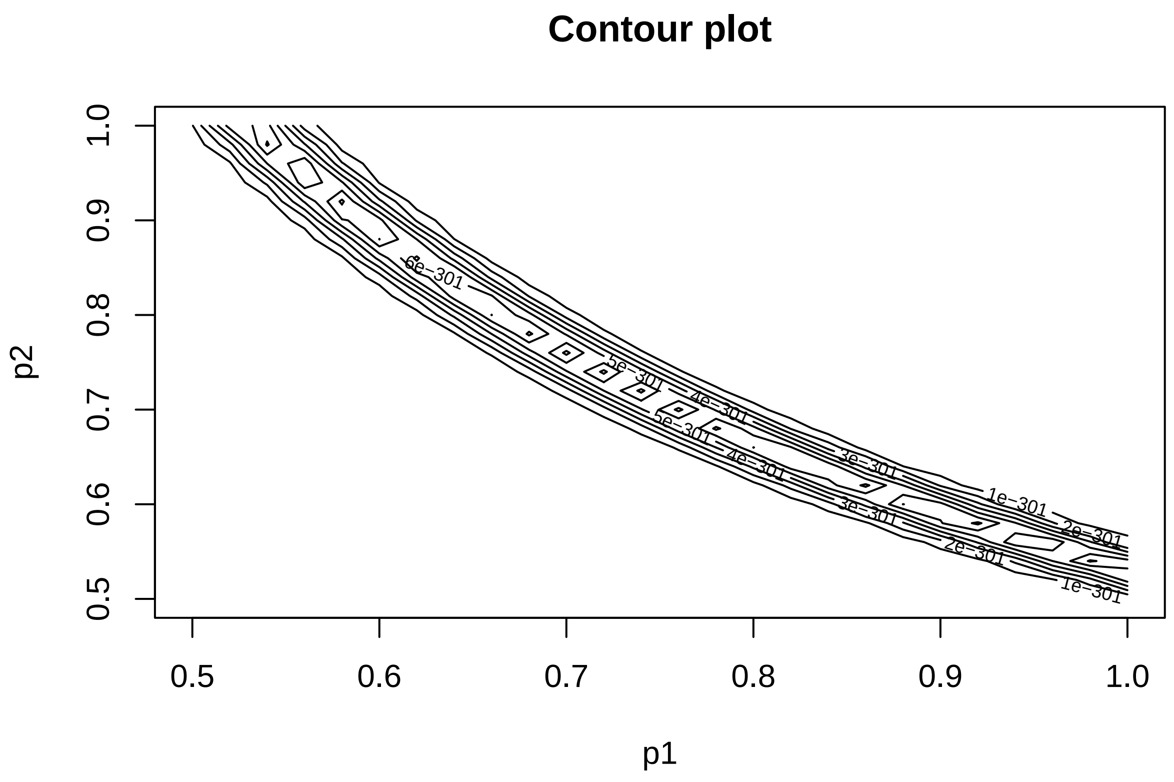
Note that we only need to know the value of $\sum_{i=1}^{n} Y_i$ from the data, since the posterior can be written as:

$$\pi(p_1, p_2|Y) \propto (p_1p_2)^{\sum_{i=1}^{n} Y_i} \cdot (1-p_1p_2)^{n-\sum_{i=1}^{n} Y_i} \cdot 1_{[0,1]}(p_1) \cdot 1_{[0,1]}(p_2)$$

Therefore, by creating a variable that stores this value: $Y_{sum} = \sum_{i=1}^{n} Y_i$, we only need to iterate through the dataset once.

Note that we also use in ifelse() statement in R to implement the indicator functions.

Implementing the posterior in R and producing the contour plot:



1.3 Part c

Implementing Random Walk Metropolis Hastings algorithm:

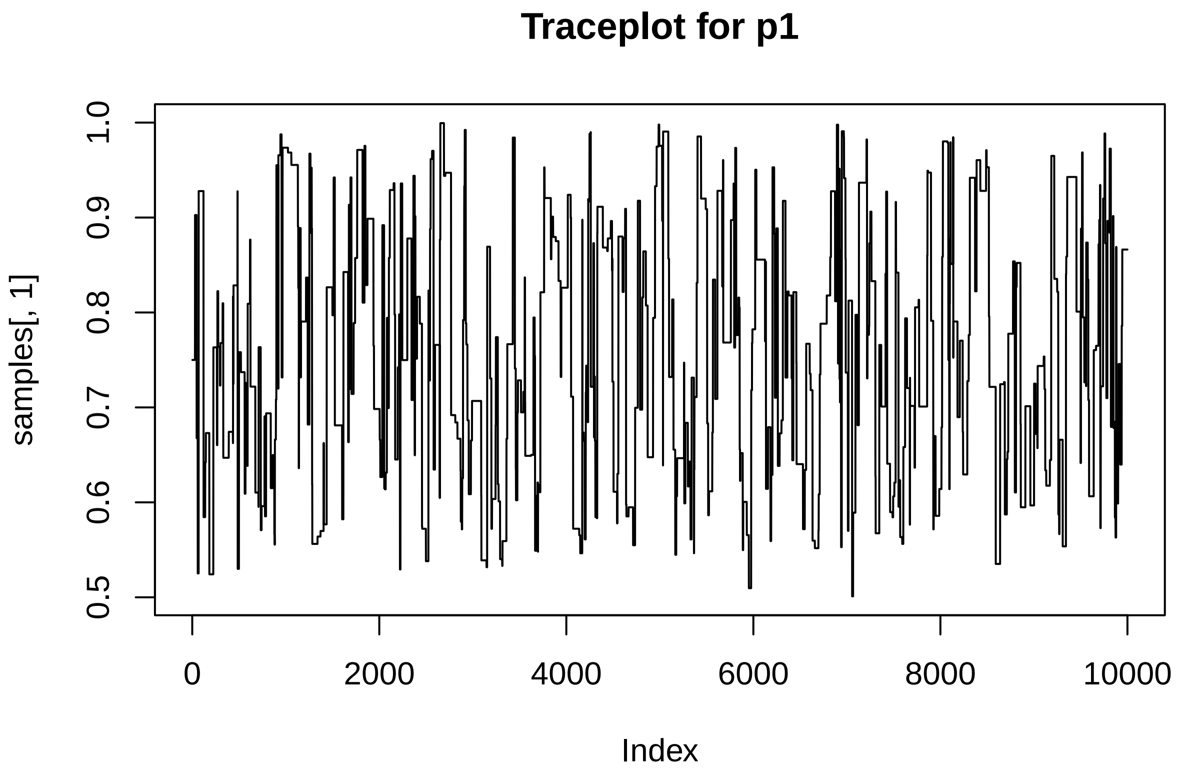
We run the RWMH Algorithm to generate 10000 samples and starting point $(p_1, p_2) = (0.75, 0.75)$ for different values of σ, and decide to use the σ that yields the highest Effective Sample Size (ESS).

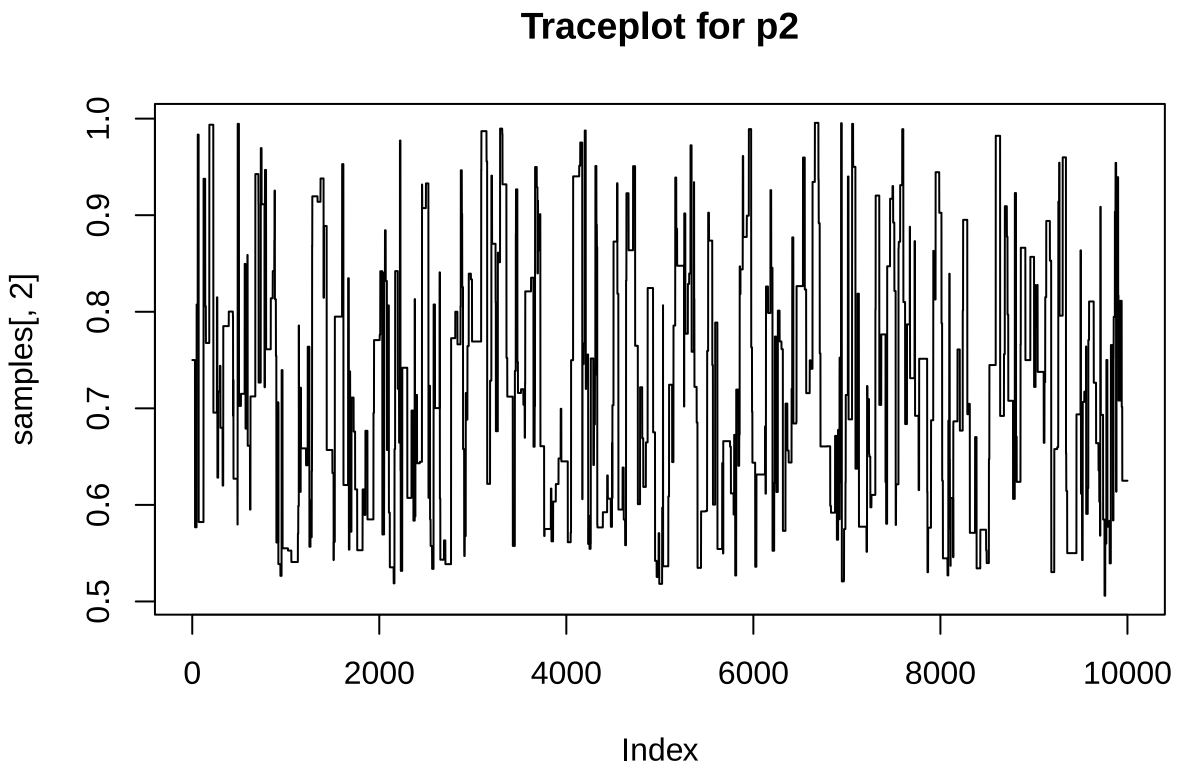
```
#> sigma: 0.01 ESS: 5.442034 6.799118
#> sigma: 0.05 ESS: 44.64741 47.8569
#> sigma: 0.1 ESS: 109.737 109.6309
#> sigma: 0.5 ESS: 87.08746 93.22978
#> sigma: 1 ESS: 30.12719 27.16286
#> sigma: 5 ESS: 1.541835 1.541835
#> sigma: 10 ESS: 1.659139 1.659139
```

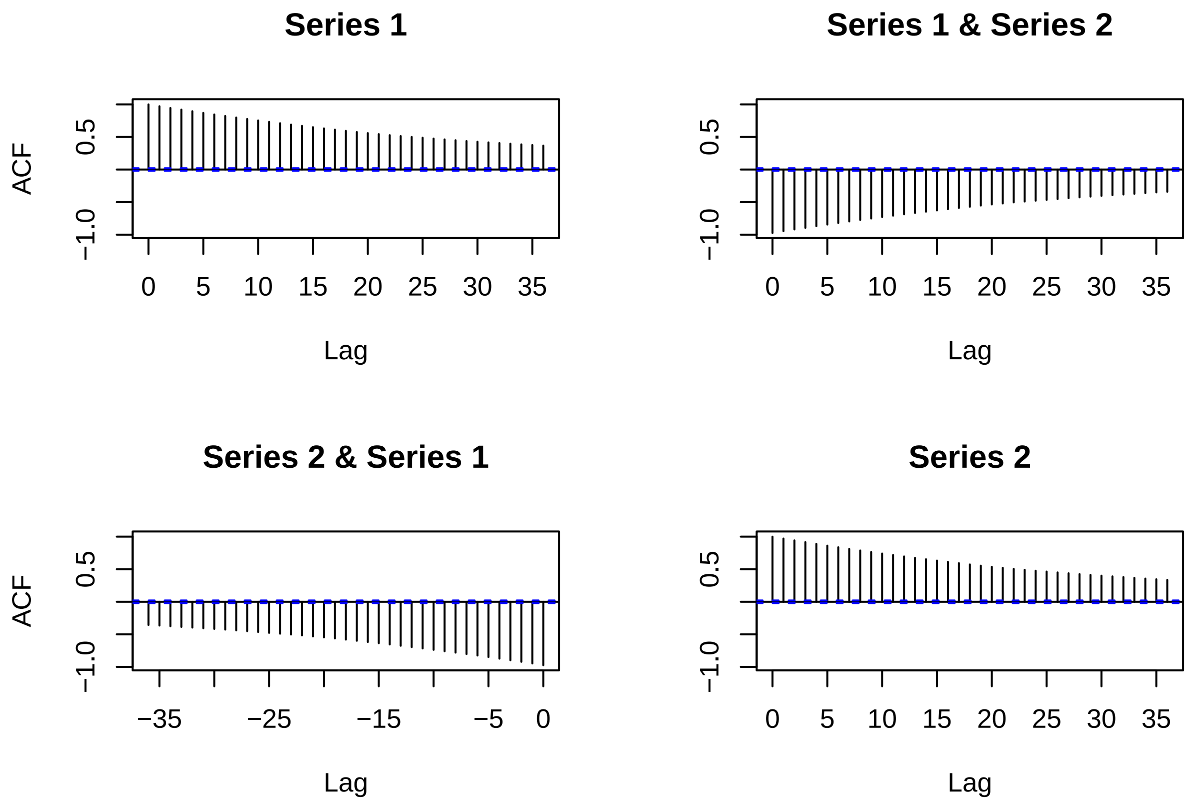
Exploring a finer grid of σ values between 0.1 and 0.5, with 0.05 spaces:

```
#> sigma: 0.1 ESS: 106.6131 107.4323
#> sigma: 0.15 ESS: 132.4121 134.7102
#> sigma: 0.2 ESS: 125.6539 119.5613
#> sigma: 0.25 ESS: 134.8419 137.208
#> sigma: 0.3 ESS: 108.9687 108.0862
#> sigma: 0.35 ESS: 111.7884 91.52556
#> sigma: 0.4 ESS: 99.72068 97.76231
#> sigma: 0.45 ESS: 82.18426 77.57989
#> sigma: 0.5 ESS: 45.55832 48.79359
```

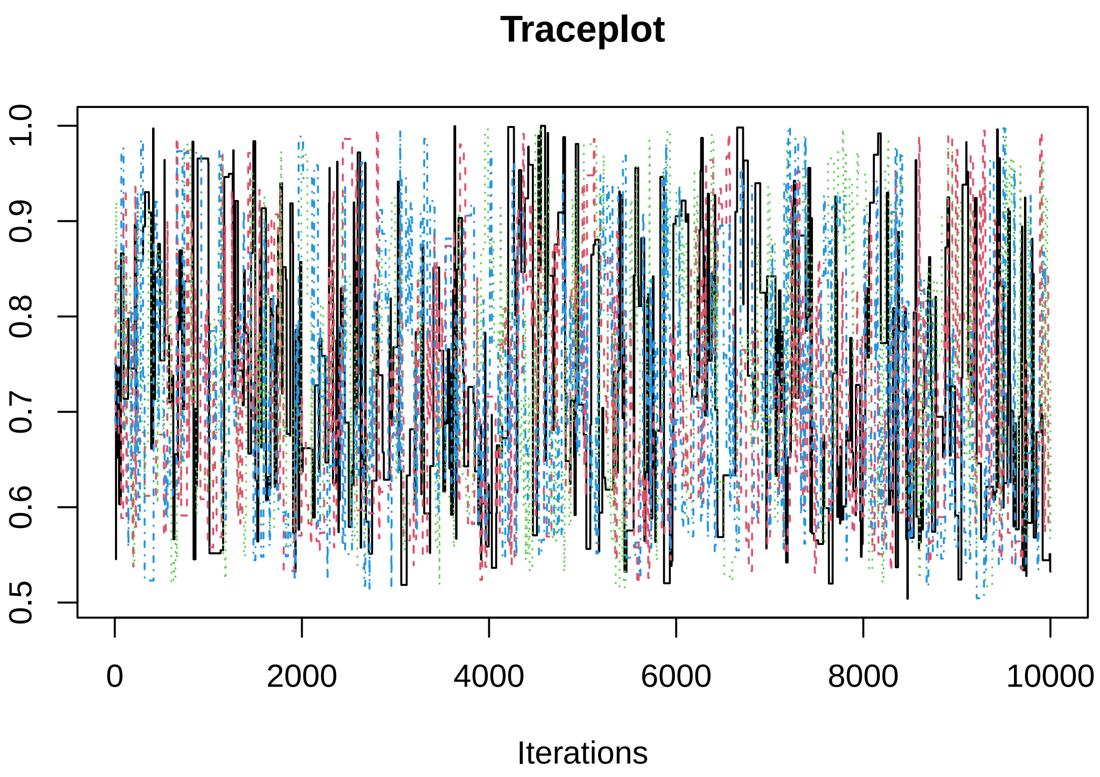
Choosing the σ with largest Effecive sample size, we shoose $\sigma = 0.25$. For this $\sigma = 0.25$, we see the following traceplots and acf:



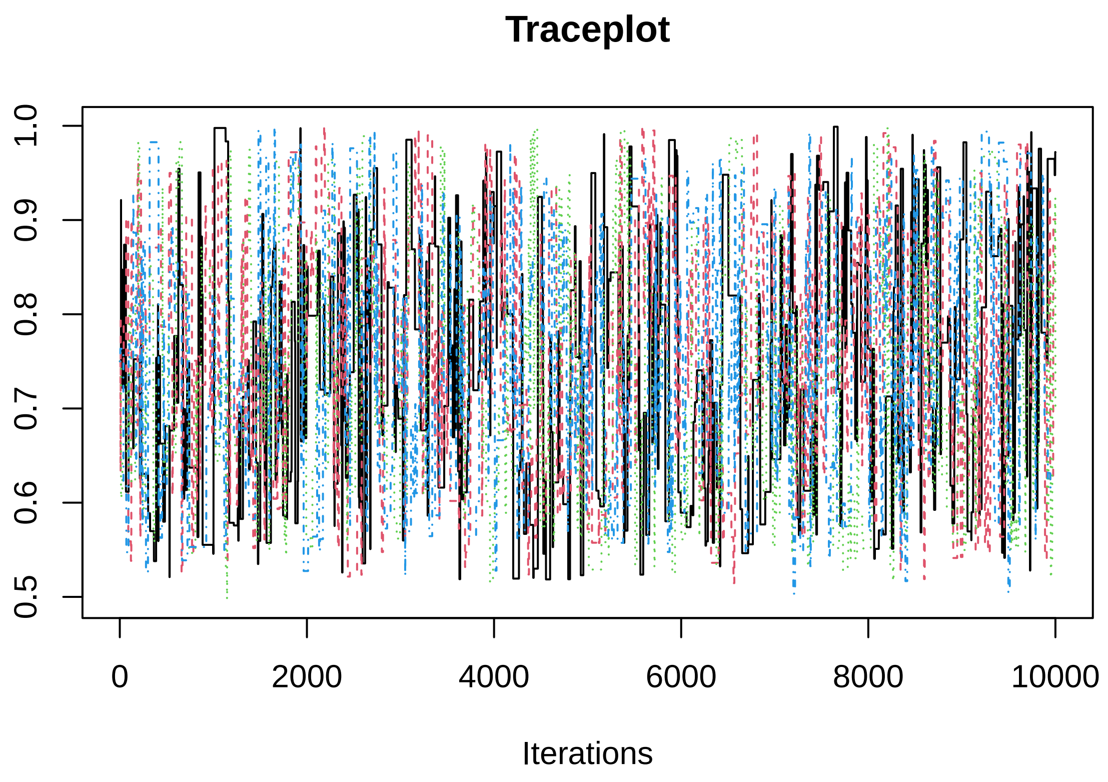

Traceplot for p2
samples[, 2]
1.0
0.9
0.8
0.7
0.6
0.5
0
2000
4000
6000
8000
10000
Index



Now Generating 4 independent runs of the Markov chain of 10000 samples each with $\sigma = 0.25$ and starting point $(p_1, p_2) = (0.75, 0.75)$, we get the traceplots for p_1 and p_2 respectively.:

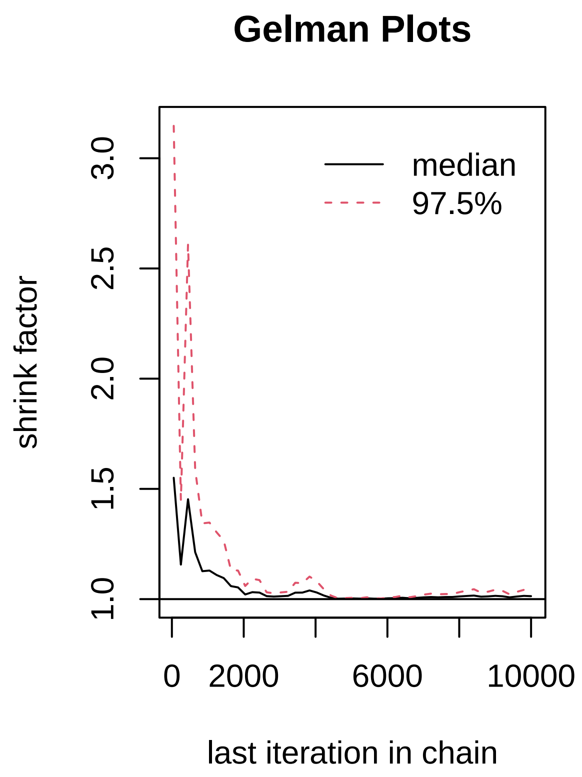


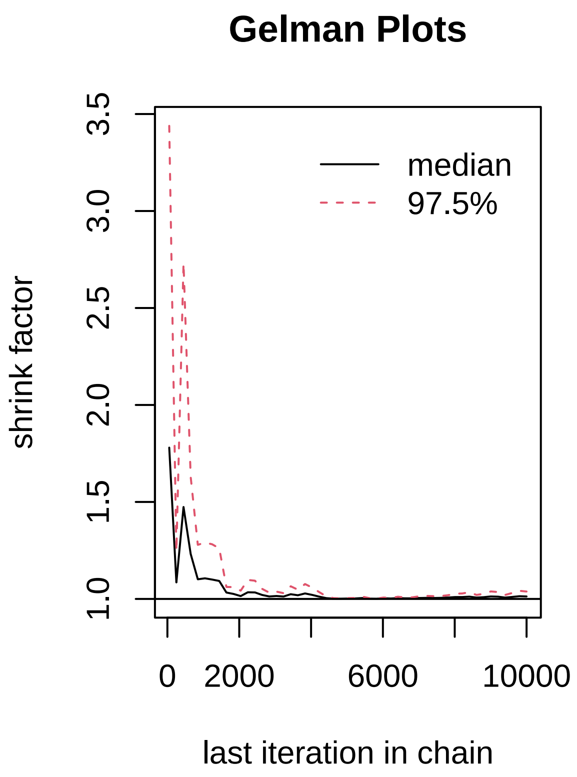
Traceplot
1.0
0.9
0.8
0.7
0.6
0.5
0
2000
4000
6000
8000
10000
Iterations



We report the R-hat (Gelman) statistic of our chains, and also plot the Gelman statistics for p_1 and p_2:

```
#> Potential scale reduction factors:
#>
#>      Point est. Upper C.I.
#> [1,]       1.01       1.04
#> [2,]       1.01       1.04
#>
#> Multivariate psrf
#>
#> 1.01
```

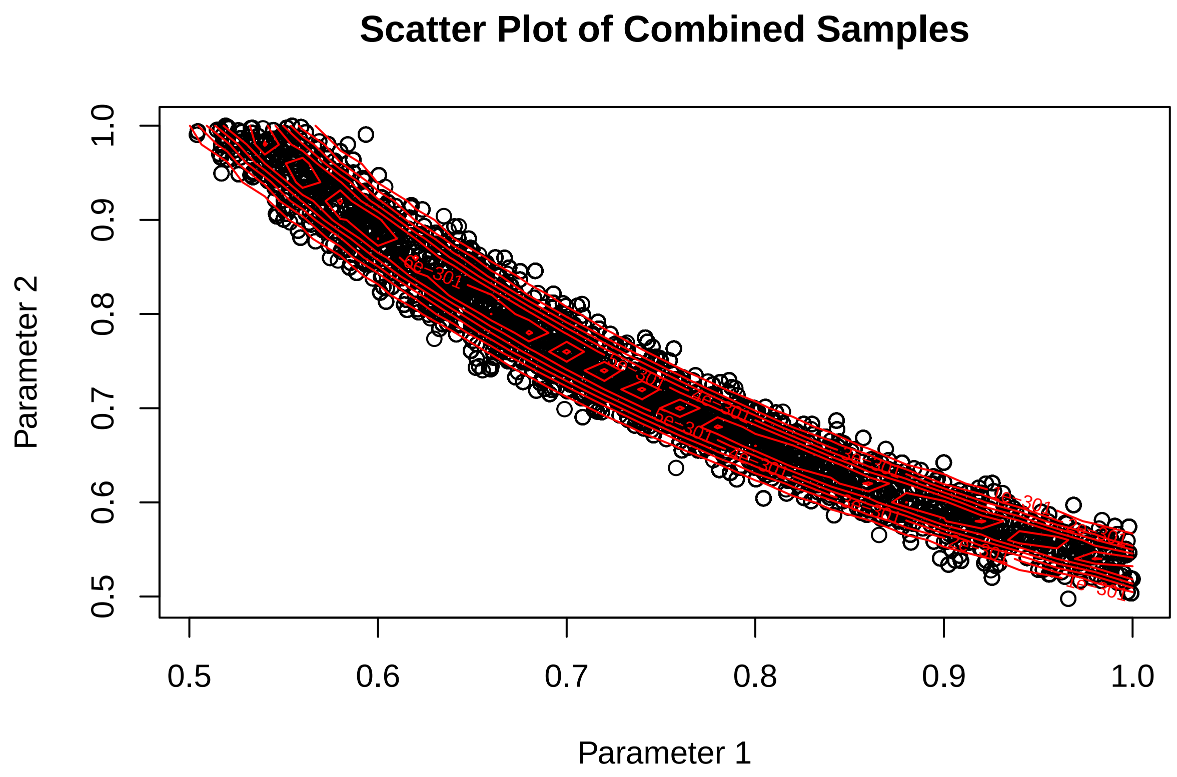




Combining the samples from the four chains, we see that the effective sample size of our markov chains are:

```
#>      var1     var2
#> 562.2395 558.7052
```

Generating a scatter plot from the combined samples and plotting over the contour plot:



1.4 Part d:

From part c, we note that for our samples of (p_1, p_2), we have the following statistics:

Standard deviation of p_1 is: $\hat{\sigma_{p_1}} = 0.1373411$

Standard deviation of p_2 is: $\hat{\sigma_{p_2}} = 0.1354375$

Correlation of p_1 and p_2 is $\hat{\rho_{p_1,p_2}} = -0.9726867$

Therefore, we now use the anisotropic Gaussian random walk proposal: .

$$Y = X_n + \lambda \cdot \epsilon, \text{where } \epsilon \sim N\left(0, \begin{bmatrix} \hat{\sigma_{p_1}}^2 & \hat{\sigma_{p_1}}\hat{\sigma_{p_2}}\hat{\rho}_{p_1,p_2} \\ \hat{\sigma_{p_1}}\hat{\sigma_{p_2}}\hat{\rho}_{p_1,p_2} & \hat{\sigma_{p_2}}^2 \end{bmatrix}\right)$$

λ is some step size which we will tune.

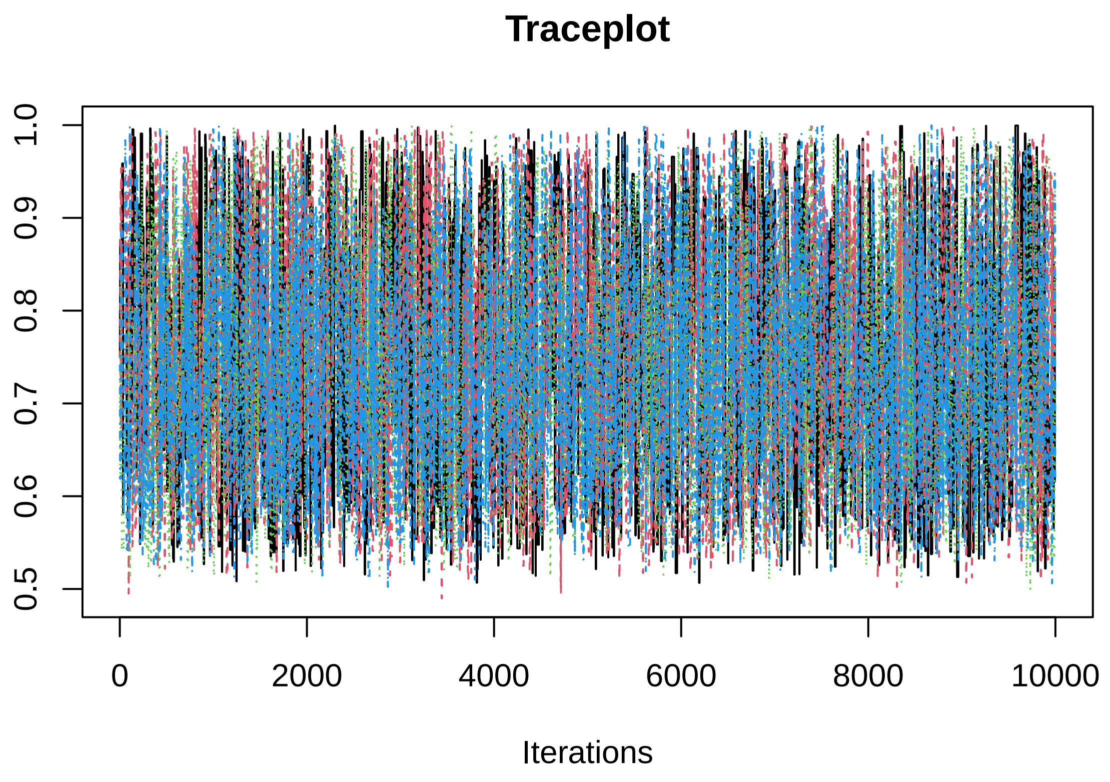
```
#> Standard deviation of p1:  0.1318669
#> Standard deviation of p2:  0.1299218
#> Correlation of p1 and p2:  -0.9741479
```

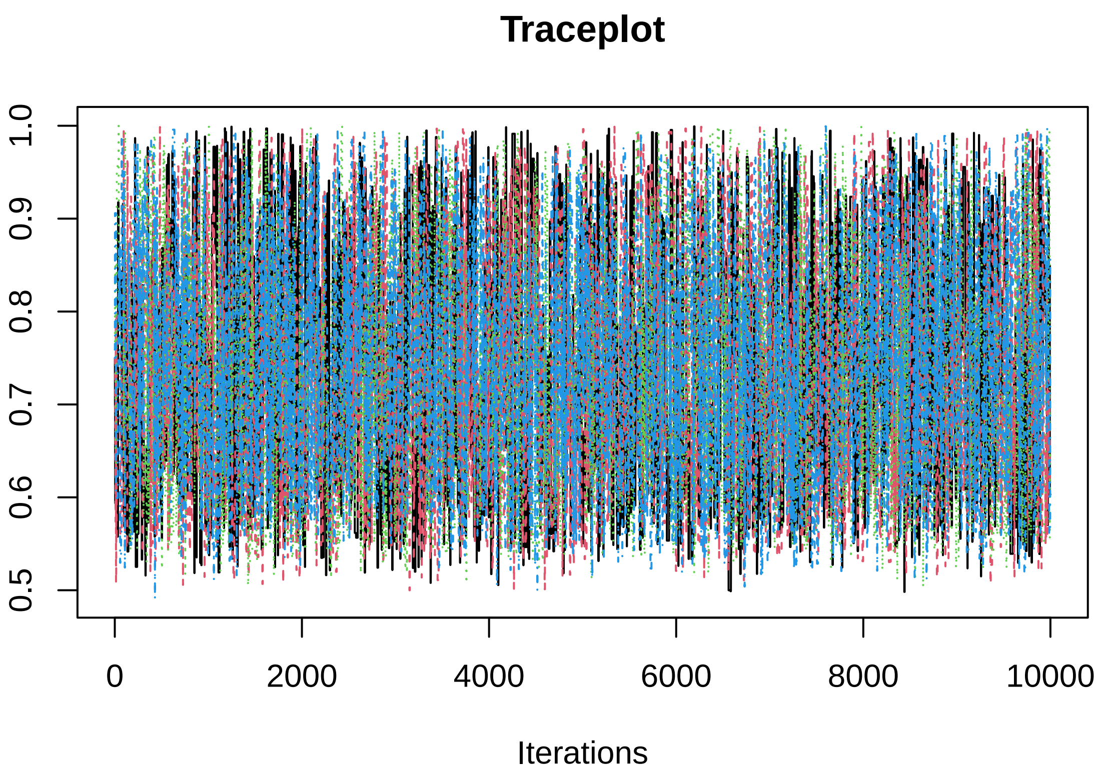
We choose the λ so that our markov chain of 10,000 samples has the highest Effective Sample Size:

```
#> lambda: 0.5 ESS: 194.5388 181.8308
#> lambda: 1 ESS: 656.7355 635.6553
#> lambda: 2 ESS: 771.8325 757.5987
#> lambda: 3 ESS: 505.7238 512.1831
#> lambda: 5 ESS: 233.9578 236.2966
```

We see that $\lambda = 2$ yields the highest ESS, so this is the value we will use to construct 4 markov chains, each of 10,000 samples and starting value $(p_1, p_2) = (0.75, 0.75)$.

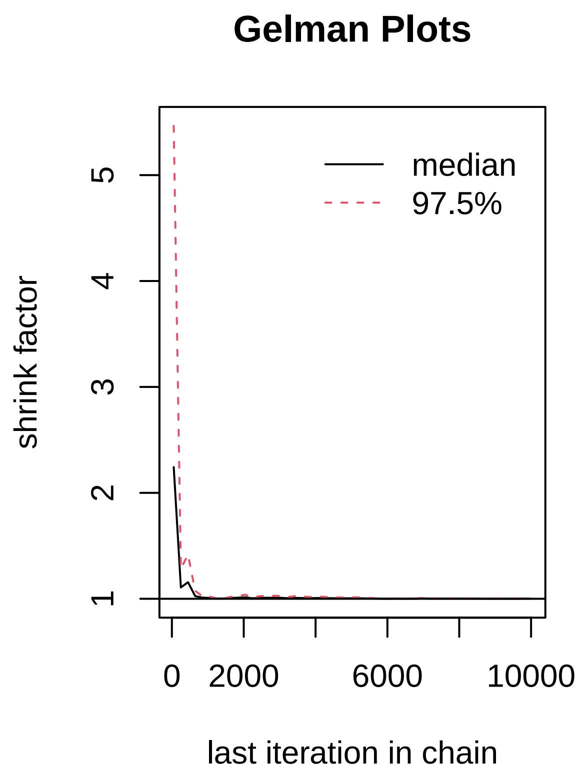
Plotting the traceplots for p_1 and p_2 respectively for these 4 independent Markov Chains:

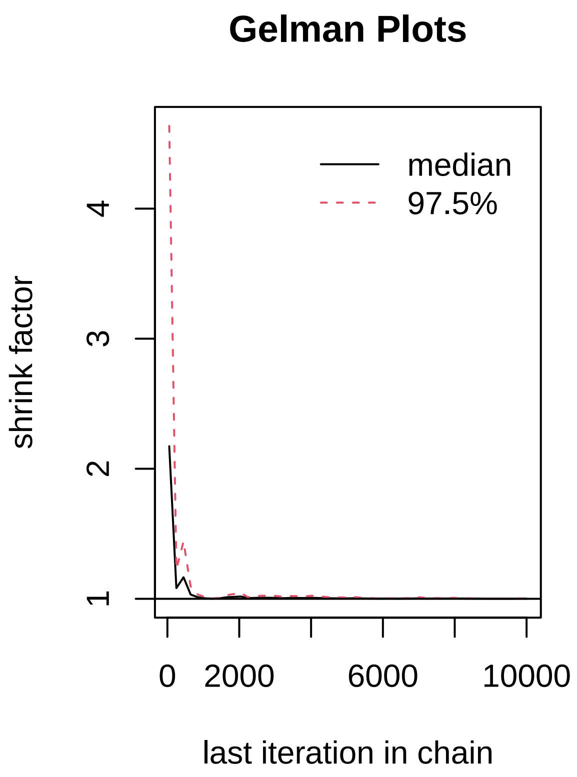




R hat value and Gelman plot:

```
#> Potential scale reduction factors:
#>
#>      Point est. Upper C.I.
#> [1,]          1          1
#> [2,]          1          1
#>
#> Multivariate psrf
#>
#> 1
```





Note that the R hat estimate has a lower upper confidence interval than the R hat estimate for part c, suggesting a better proposal.

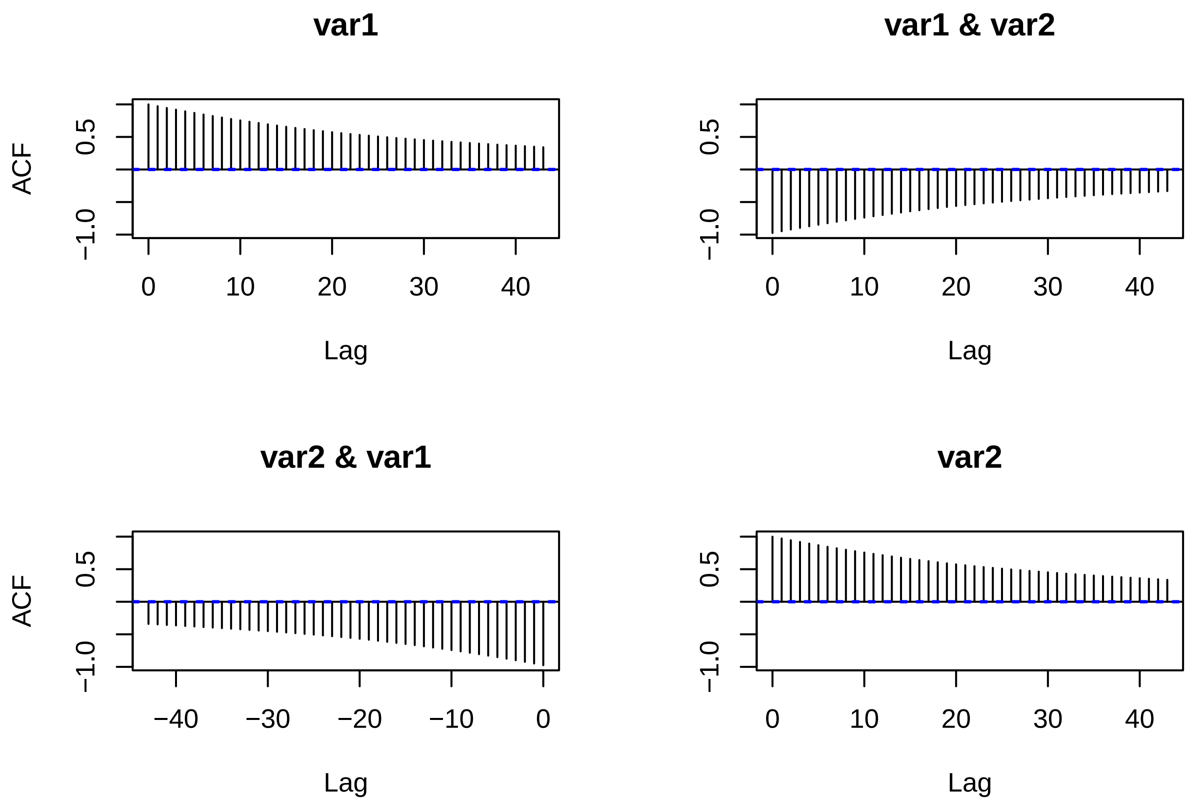
Let us also compare the Effective Sample Sizes in parts c and d, using the combined sample from the 4 Markov Chains in both:

```
#> ESS for combined sample in (c):  562.2395 558.7052
#> ESS for combined sample in (d):  3031.669 2993.665
```

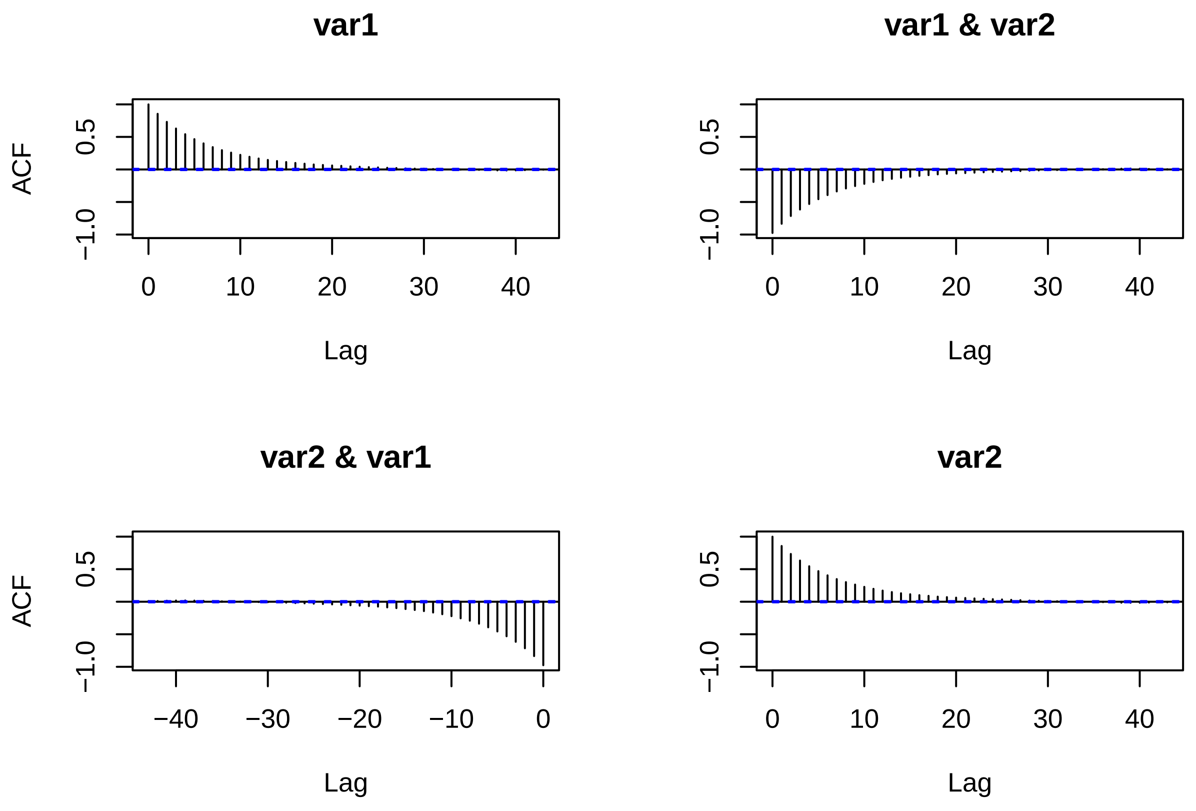
We can clearly see that the Effective Sample Size in part (d) is much larger compared to part (c), suggesting that our new proposal is more efficient.

Let us look at the acf for the samples in part c and d:

ACF of combined samples in part (c)



ACF of combined samples in part (d)



We can see that the acf goes to 0 much quicker than in part (d), than part (c), suggesting that our new proposal leads to our samples converging quicker to the stationary distribution.

2 Question 2

2.1 a

The posterior is given by:

$$p(\alpha, \beta, \tau|S) \propto p(S|\alpha, \beta, \tau) \cdot \pi(\alpha) \cdot \pi(\beta) \cdot \pi(\tau)$$

Plugging in expressions for priors and likelihood:

$$p(\alpha, \beta, \tau|S) \propto \left(\prod_{i=1}^{\tau} \frac{\alpha^{S_i} e^{-\alpha}}{(S_i)!} \cdot \prod_{i=\tau+1}^{N_{\text{days}}} \frac{\beta^{S_i} e^{-\beta}}{(S_i)!} \right) \cdot \frac{1}{\bar{S}} exp\left(-\frac{\alpha}{\bar{S}}\right) \cdot \frac{1}{\bar{S}} exp\left(-\frac{\beta}{\bar{S}}\right) \cdot \frac{1}{N_{days}-1}$$

Simplifying the above expression:

$$p(\alpha, \beta, \tau|S) \propto \left(\prod_{i=1}^{\tau} \frac{1}{(S_i)!} \right) \cdot \alpha^{\sum_{i=1}^{\tau} S_i} \cdot e^{-\alpha\tau} \cdot \beta^{\sum_{i=\tau+1}^{N_{\text{days}}} S_i} \cdot e^{-\beta(N_{days}-\tau)} \cdot \frac{1}{\bar{S}} exp\left(-\frac{\alpha}{\bar{S}}\right) \cdot \frac{1}{\bar{S}} exp\left(-\frac{\beta}{\bar{S}}\right) \cdot \frac{1}{N_{days}-1}$$

2.2 b

Conditional distributions:

Ignoring terms which do not contain α:

$$p(\alpha|\beta,\tau,S) \propto \alpha^{\sum_{i=1}^{\tau} S_i} \cdot e^{-\alpha\tau} \cdot exp\left(-\frac{\alpha}{\bar{\bar{S}}}\right)$$

Using properties of exponentials:

$$p(\alpha|\beta,\tau,S) \propto \alpha^{\sum_{i=1}^{\tau} S_i} \cdot exp\left(-\alpha(\tau + \frac{1}{\bar{\bar{S}}})\right)$$

and so

$$\alpha|\beta,\tau,S \sim Gamma\left(1 + \sum_{i=1}^{\tau} S_i, \tau + \frac{1}{\bar{\bar{S}}}\right)$$

Similarly, ignoring terms which do not contain β:

$$p(\beta|\alpha,\tau,S) \propto \beta^{\sum_{i=\tau+1}^{N_{\text{days}}} S_i} \cdot e^{-\beta(N_{days}-\tau)} \cdot exp\left(-\frac{\beta}{\bar{\bar{S}}}\right)$$

$$p(\beta|\alpha,\tau,S) \propto \beta^{\sum_{i=\tau+1}^{N_{\text{days}}} S_i} \cdot exp\left(-\beta(N_{days} - \tau + \frac{1}{\bar{\bar{S}}})\right)$$

and so

$$\beta|\alpha,\tau,S \sim Gamma\left(1 + \sum_{i=\tau+1}^{N_{\text{days}}} S_i, N_{days} - \tau + \frac{1}{\bar{\bar{S}}}\right)$$

Finally, considering only τ terms in the posterior, we have:

$$p(\tau = t|\alpha,\beta,S) \propto e^{-\alpha t} \cdot \alpha^{\sum_{i=1}^{t} S_i} \cdot e^{-\beta(N_{days}-t)} \cdot \beta^{\sum_{i=t+1}^{N_{\text{days}}} S_i}$$

for $t = 1, ...N_{days} - 1$as required.

2.3 c

We now implement our Gibbs Sampler in R.

Note that at each iteration of our loop, we sample from each of the conditional distributions above $p(\alpha|\beta,\tau,S)$, $p(\beta|\alpha,\tau,S)$, $p(\tau = t|\alpha,\beta,S)$, using the values from the previous iteration if required.

The conditional distributions $\alpha|\beta,\tau,S$ and $\beta|\alpha,\tau,S$ are easy to implement in R using the rgamma() function.

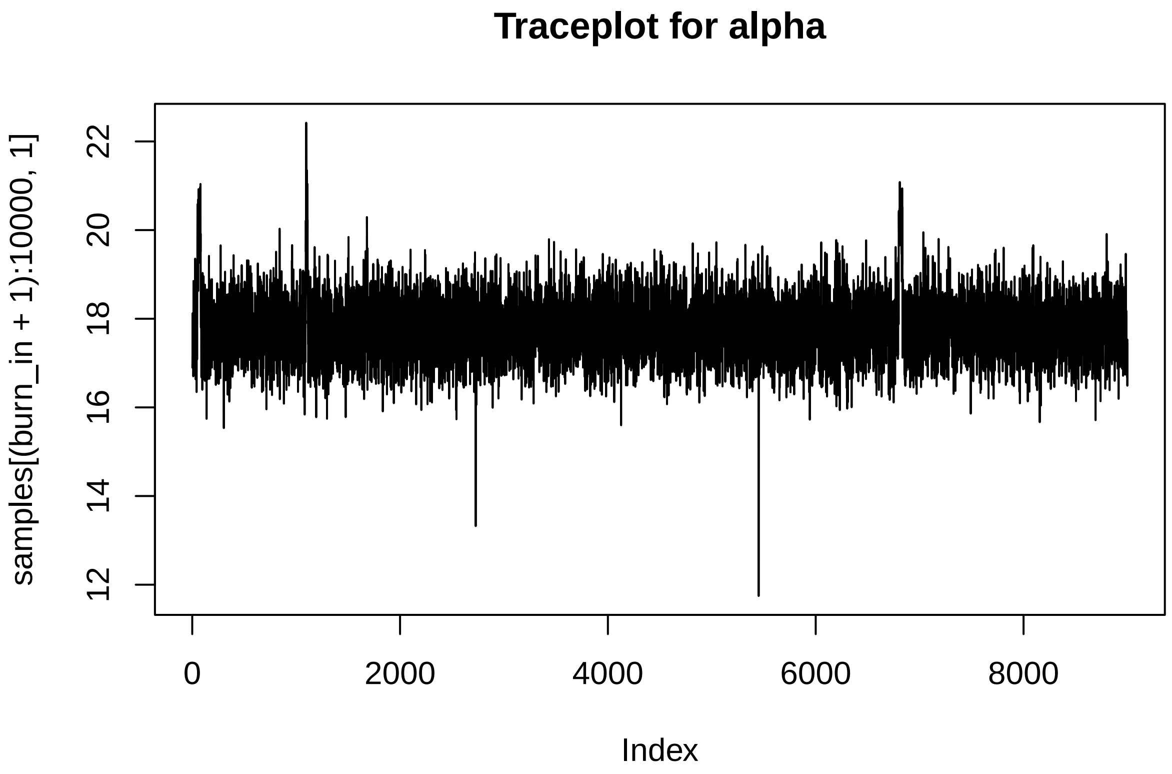
Note that $\tau|\alpha,\beta,S$ is a discrete random variable, and so by re-weighting the below probabilities so they sum to 1,

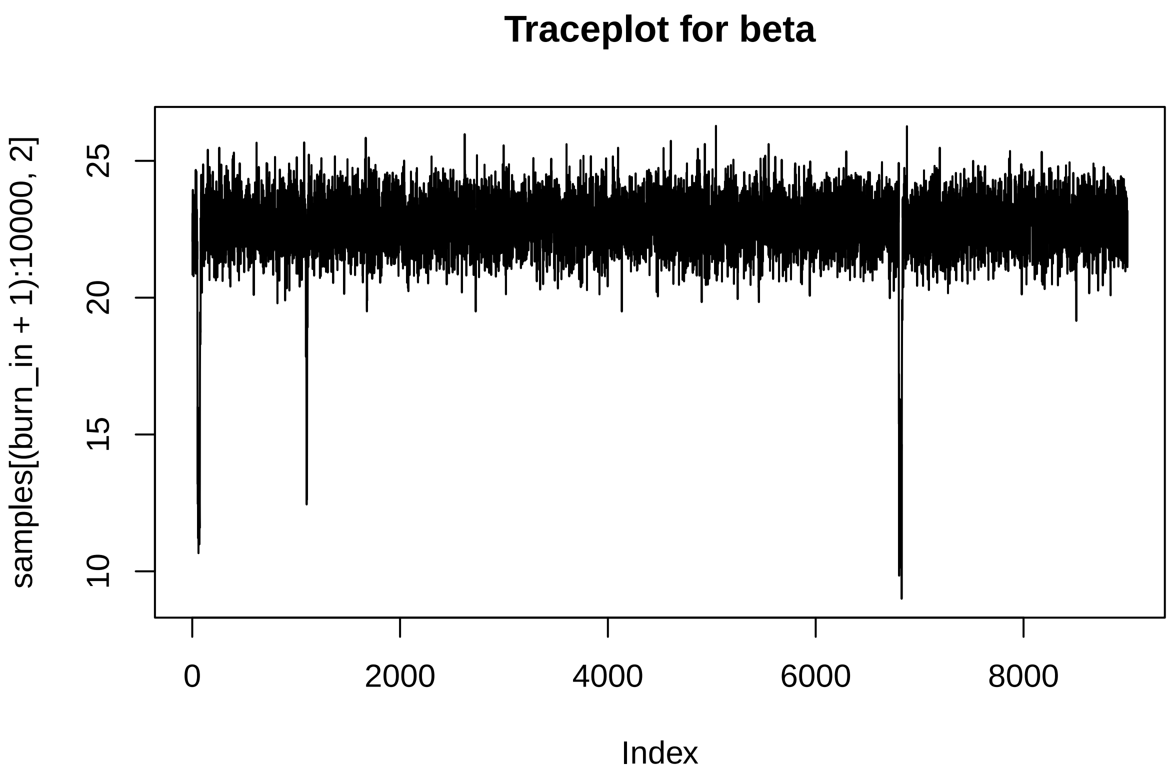
$$p(\tau = t|\alpha,\beta,S) \propto e^{-\alpha t} \cdot \alpha^{\sum_{i=1}^{t} S_i} \cdot e^{-\beta(N_{days}-t)} \cdot \beta^{\sum_{i=t+1}^{N_{\text{days}}} S_i} \ , \ t = 1, ...N_{days} - 1$$

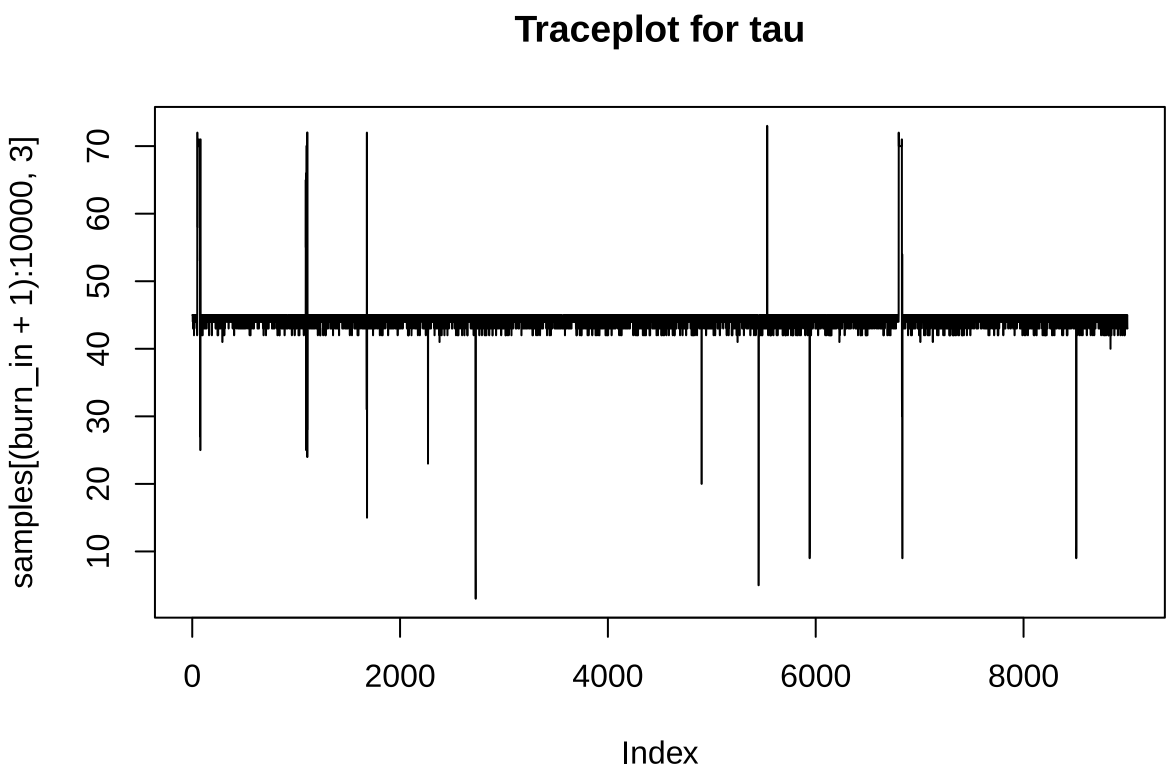
we can use the sample() function in R to random sample from $t = 1, ...N_{days} - 1$ with the corresponding re-weighted probabilities from above.

We use starting values: $(\alpha_0, \beta_0, \tau_0) = (\bar{S}, \bar{S}, 10)$

We generate 10,000 samples and use the first 1,000 as the burn-in period.




Traceplot for beta
samples[(burn_in + 1):10000, 2]
Index
0
2000
4000
6000
8000
10
15
20
25



Approximation for expected number of calls per day:

We use a Monte Carlo Estimate to approximate the expected number of calls per days as follows:

$$E[\lambda_i|S] = \int \lambda(i;\tau,\alpha,\beta)p(d\alpha,d\beta,d\tau|S) \approx \frac{1}{N-n_0}\sum_{i=n_0+1}^{N}\lambda(i;\tau^{(i)},\alpha^{(i)},\beta^{(i)})$$

where N is the total number of samples, 10000 in our code,

n_0 is the burn-in period, 1000 in our code,

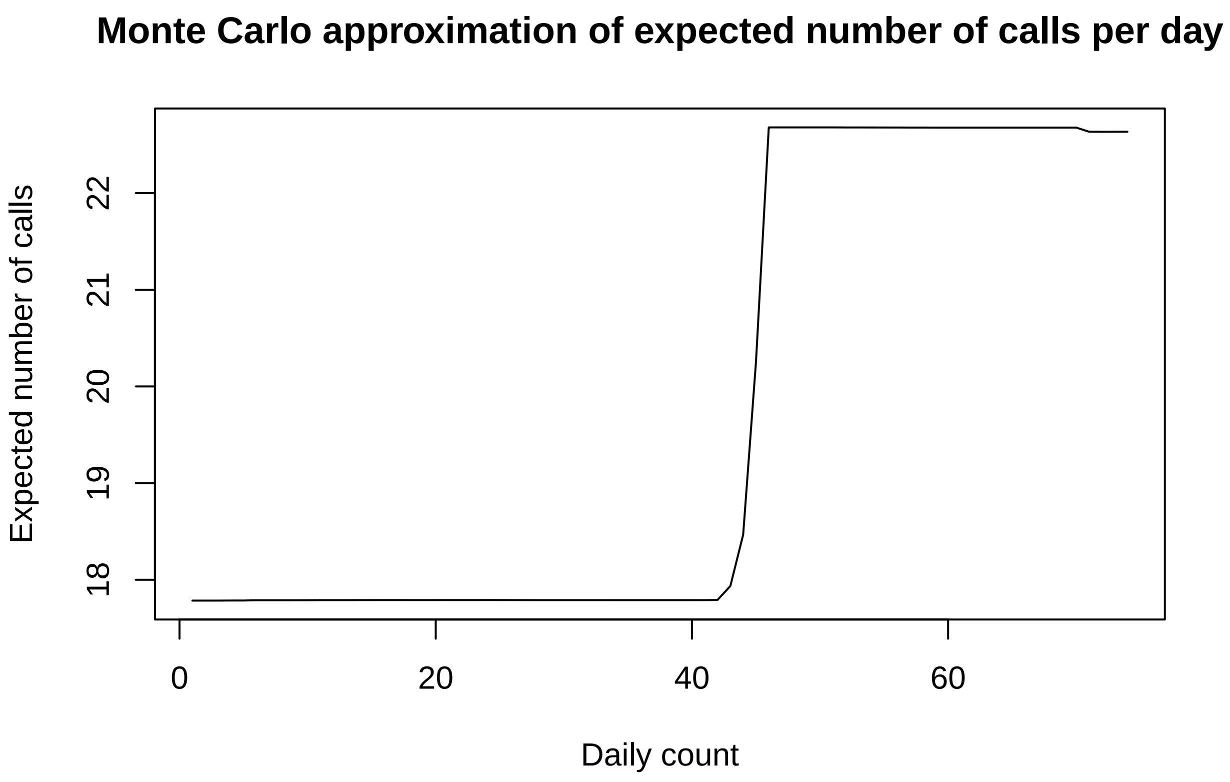
and $(\tau^{(i)},\alpha^{(i)},\beta^{(i)})$ represents the i^{th} values in our Markov chain for each of the parameters.

This can implemented in R using a sample mean of $\lambda()$ for the different values of i.

For $i = 1,...,N_{days} - 1$, we obtain the approximations of $E[\lambda_i|S]$ as:

```
#>  [1] 17.78442 17.78442 17.78442 17.78511 17.78511 17.78765 17.78765 17.78765
#>  [9] 17.78765 17.78823 17.78925 17.78942 17.78942 17.78997 17.78997 17.79002
#> [17] 17.79002 17.78995 17.78995 17.78995 17.79033 17.79033 17.79033 17.79056
#> [25] 17.79040 17.78990 17.78954 17.78937 17.78923 17.78923 17.78923 17.78934
#> [33] 17.78934 17.78905 17.78905 17.78905 17.78905 17.78905 17.78905 17.78905
#> [41] 17.78928 17.79128 17.93518 18.46528 20.25368 22.67969 22.67969 22.67969
#> [49] 22.67969 22.67969 22.67969 22.67922 22.67922 22.67885 22.67874 22.67855
#> [57] 22.67814 22.67814 22.67796 22.67796 22.67796 22.67782 22.67782 22.67782
#> [65] 22.67782 22.67792 22.67788 22.67788 22.67788 22.67788 22.63508 22.63418
#> [73] 22.63439 22.63486
```

Plotting these over the daily counts:



Interpretation:

The changepoint appears to occur at $\tau \approx 45$, and it appears that $S_i \sim$ Poisson(17.8) for $i = 1, ...\tau$ and $S_i \sim$ Poisson(22.7) for $i = \tau + 1, ...N_{days}$

2.4 d:

Suppose S has missing values at $25, 42, 55$, and let the unobserved values of the missing data be $S^* = (S_{25}, S_{42}, S_{55})$

Note that in our new Gibbs Sampler, the conditional distribution $\alpha|\beta, \tau, S, S^*$ is still gamma distributed, except now we plug in the values $S^* = (S_{25}, S_{42}, S_{55})$ from the previous iteration. Therefore,

$$\alpha|\beta, \tau, S, S^* \sim Gamma\left(1 + \sum_{i=1}^{\tau} S_i, \tau + \frac{1}{\bar{S}}\right)$$

Similarly, we have that

$$\beta|\alpha, \tau, S \sim Gamma\left(1 + \sum_{i=\tau+1}^{N_{\text{days}}} S_i, N_{days} - \tau + \frac{1}{\bar{S}}\right)$$

and:

$$p(\tau = t|\alpha, \beta, S, S^*) \propto e^{-\alpha t} \cdot \alpha^{\sum_{i=1}^{t} S_i} \cdot e^{-\beta(N_{days}-t)} \cdot \beta^{\sum_{i=t+1}^{N_{\text{days}}} S_i} \text{ , } t = 1, ...N_{days} - 1$$

Note that the conditional distributions for the missing data S_i for i = 25, 42, 55 are:

$$S_i|\alpha, \beta, \tau, S \sim \text{Poisson}(\alpha), \text{ if } i \leq \tau,$$

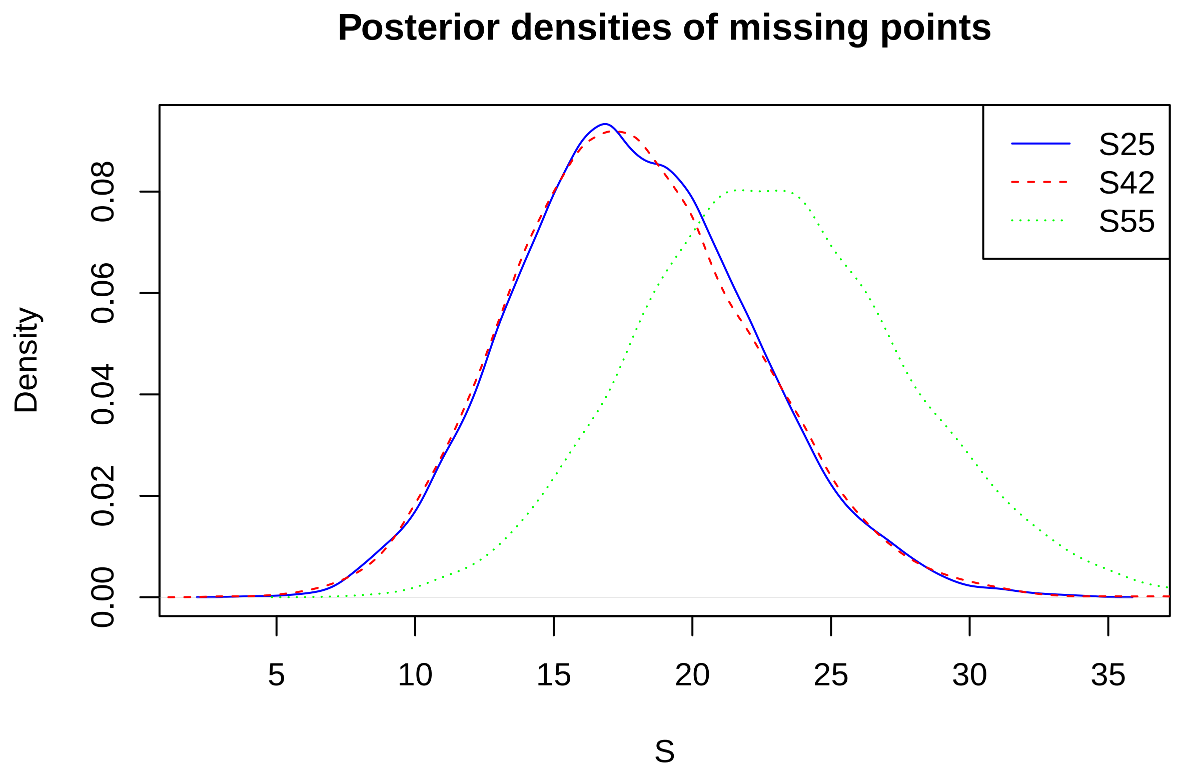
or

$$S_i|\alpha, \beta, \tau, S \sim \text{Poisson}(\beta), \text{ if } i > \tau,$$

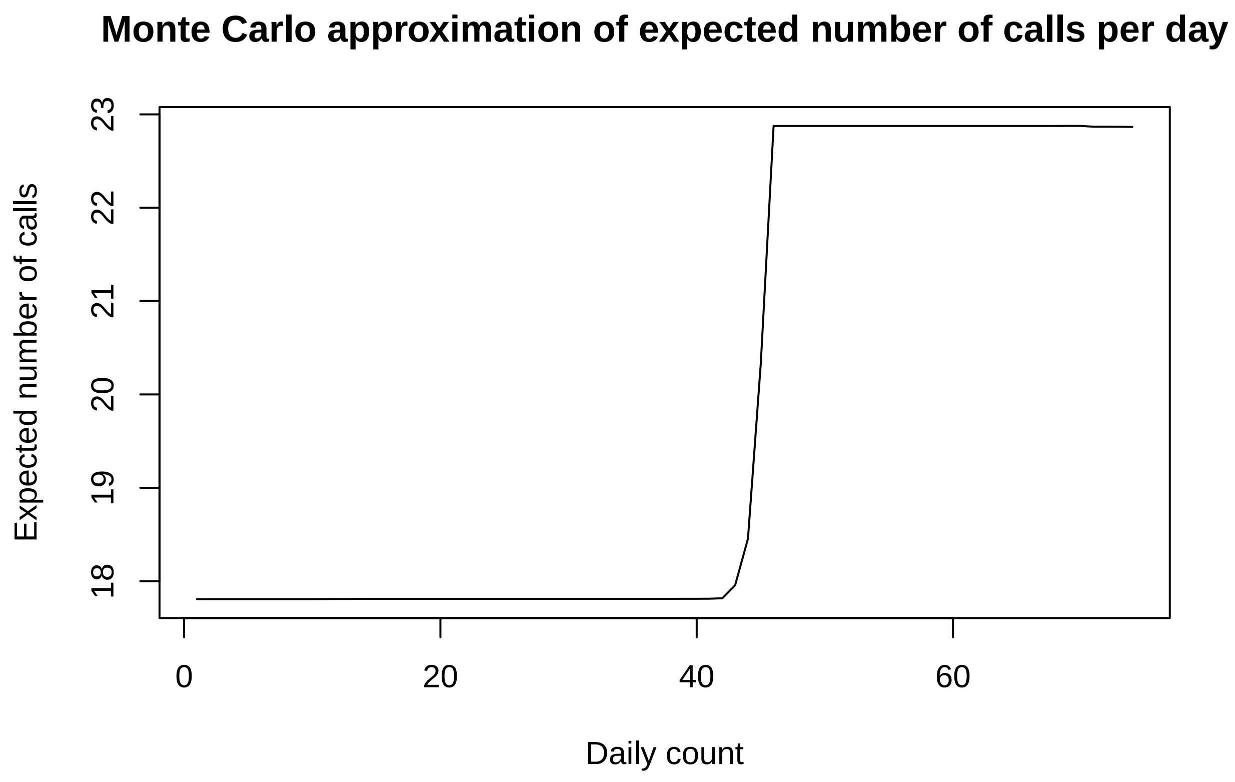
This can be implemented in R using an if statement and using the rpois() function.

Implementing in R using the starting values: $(\alpha_0, \beta_0, \tau_0, S_{25}, S_{42}, S_{55}) = (\bar{S}, \bar{S}, 10, \bar{S}, \bar{S}, \bar{S})$ where we are using $\bar{S}$ as the sample mean of the observed values in S:

We plot the posterior distributions for each missing point:



Repeating steps for part c to compute an approximation of the expected number of calls per day using Monte Carlo, we get the following plot:



3 Code appendix

Rather than re-paste all the code to the appendix, here is a trick which makes the markdown file output all the code (without) execution in the appendix, without any duplication.

Please keep in mind to format the code so that the entire code is clearly visible and does not run into the margins of the pdf version.

```
knitr::opts_chunk$set(
  collapse = TRUE,
  comment = "#>"
)
include_solutions <- TRUE
require(rmarkdown)
require(knitr)
require(kableExtra)
library(coda)
# Put any library imports and other preamble here.

### Question 1b

trials <- read.csv(file="trials.csv", header = TRUE)

Y_sum <- sum(trials$Y)
n <- nrow(trials)
```

```r
#Posterior density:
post <- function(p1, p2){
  ifelse( 0<=p1 & p1 <= 1 & 0 <= p2 & p2 <= 1,
         (p1*p2)^(Y_sum) * (1 - p1*p2)^(n - Y_sum),
         0)

}

p1 <- seq(0.5, 1, 0.02)
p2 <- seq(0.5, 1, 0.02)

contour(p1, p2, outer(p1,p2, post), main="Contour plot", xlab="p1", ylab="p2")


### Question 1c - RWMH Algorithm:

# RWMH

do.the.rwmh <- function(num.steps, sigma, X0) {
    X <- X0
    samples <- matrix(NA, nrow=num.steps, ncol=2)

    for (i in 1:num.steps) {

        samples[i,] = X
        Y <- X + sigma*rnorm(2)

        logalpha <-  log(post(Y[1], Y[2]))  - log(post(X[1], X[2]))

        if(log(runif(1)) <= logalpha) {
          X<-Y
        }
    }

    return(samples)
}

num.steps <- 10000
sigma <- 0.25
X0 <- c(0.75,0.75)

for (sigma in c(0.01,0.05, 0.1, 0.5,1, 5,10)){
  samples <- do.the.rwmh(10000, sigma,X0)
  ESS <- effectiveSize(samples)
  cat("sigma:",sigma,"ESS:",ESS,"\n")
}

set.seed(27)
for (sigma in seq(0.1, 0.5, 0.05)){
  samples <- do.the.rwmh(10000, sigma,X0)
  ESS <- effectiveSize(samples)
  cat("sigma:",sigma,"ESS:",ESS,"\n")
}
```

```
num.steps <- 10000
sigma <- 0.25
X0 <- c(0.75,0.75)

samples <- do.the.rwmh(num.steps, sigma=0.25, X0)

plot(samples[,1], type = 'l', main = "Traceplot for p1")
plot(samples[,2], type = 'l', main = "Traceplot for p2")

#p1 <- seq(0.5, 1, 0.02)
#p2 <- seq(0.5, 1, 0.02)

#contour(p1, p2, outer(p1,p2, post), add = TRUE, col='red')


acf(samples)



### Traceplots - 4 chains:

# Define initial values for the chains
X0s <- list(c(0.75, 0.75), c(0.75, 0.75), c(0.75, 0.75), c(0.75, 0.75))  # Four different initial value


num_steps <- 10000
sigma_value <- 0.25

# Run RWMH for each initial value using lapply
set.seed(123)
chains <- lapply(X0s, function(X0) mcmc(do.the.rwmh(num_steps, sigma_value, X0)))

traceplot(chains, main="Traceplot")




chains <- mcmc.list(chains)
gelman.diag(chains)

gelman.plot(chains, main="Gelman Plots")




# Combine samples from all chains
combined_samples <- do.call(rbind, lapply(chains, as.matrix))

effectiveSize(combined_samples)


# Generate scatter plot
```

```r
plot(combined_samples[, 1], combined_samples[, 2],
     xlab = "Parameter 1", ylab = "Parameter 2",
     main = "Scatter Plot of Combined Samples")

p1 <- seq(0.5, 1, 0.02)
p2 <- seq(0.5, 1, 0.02)

contour(p1, p2, outer(p1,p2, post), add = TRUE, col='red')




### Q1 Part d

sd1 <- sd(samples[,1])
sd2 <- sd(samples[,2])
corr12 <- cor(samples[,1], samples[,2])

cat("Standard deviation of p1: ", sd1, "\n")
cat("Standard deviation of p2: ", sd2, "\n")
cat("Correlation of p1 and p2: ", corr12, "\n")


# Question 1d
# RWMH

library('MASS')

do.the.rwmh2 <- function(num.steps, sigma, X0) {
    X <- X0
    samples <- matrix(NA, nrow=num.steps, ncol=2)

    for (i in 1:num.steps) {

        samples[i,] = X

        covMatrix <- matrix(c(sd1^2, sd1*sd2*corr12,
                              sd1*sd2*corr12, sd2^2),
                            nrow = 2, ncol = 2, byrow = TRUE)

        Y <- X + sigma * mvtnorm::rmvnorm(1, mean=c(0,0), covMatrix)

        logalpha = log(post(Y[1], Y[2]))  - log(post(X[1], X[2]))

        if(log(runif(1)) <= logalpha) {
          X<-Y
        }
    }

    return(samples)
}
```

```r
num.steps <- 10000
sigma <- 2
X0 <- c(0.75,0.75)

for (sigma in c(0.5,1, 2, 3, 5)){
  samples <- do.the.rwmh2(10000, sigma,X0)
  ESS <- effectiveSize(samples)
  cat("lambda:",sigma,"ESS:",ESS,"\n")
}



### Traceplots - 4 chains:

# Define initial values for the chains
X0 <- list(c(0.75, 0.75), c(0.75, 0.75), c(0.75, 0.75), c(0.75, 0.75))  # Four different initial values

# Set parameters
num_steps <- 10000  # Number of steps
sigma_value <- 2  # Sigma value for the proposal distribution

# Run RWMH for each initial value using lapply
chains2 <- lapply(X0s, function(X0) mcmc(do.the.rwmh2(num_steps, sigma_value, X0)))

traceplot(chains2, main="Traceplot")


set.seed(150)
chains2 <- mcmc.list(chains2)
gelman.diag(chains2)

gelman.plot(chains2, main="Gelman Plots")




# Combine samples from all chains
combined_samples2 <- do.call(rbind, lapply(chains2, as.matrix))

ESS2 <- effectiveSize(combined_samples2)

cat("ESS for combined sample in (c): ", effectiveSize(combined_samples), "\n")
cat("ESS for combined sample in (d): ", ESS2)


acf(combined_samples)

acf(combined_samples2)

### Importing data

sms <- read.csv(file='sms.csv', header = FALSE)
```

```r
S <- sms$V1


### Gibbs Sampler:

Ndays = 74

# Define a function to calculate the probability for a given t
calculate_probability <- function(t, alpha, beta, Ndays=74) {
  exp(-alpha * t) * alpha^(sum(S[1:t])) * exp(-beta * (Ndays - t)) * beta^(sum(S[(t+1):Ndays]))
}


calculate_log_probability <- function(t, alpha, beta, Ndays=74) {
  (-alpha * t) + (sum(S[1:t]))*log(alpha) - (beta * (Ndays - t)) +  sum(S[(t+1):Ndays]) * log(beta)
}


do.the.gibbs <- function(niter, alpha0, beta0, tau0) {

  result <- matrix(nrow = niter, ncol = 3)
  alpha <- alpha0
  beta <- beta0
  tau <- tau0

  for (i in 1:niter) {
    alpha <- rgamma(1, 1 + sum(S[1:tau]), tau + (1/mean(S)))

    beta <- rgamma(1, 1 + sum(S[(tau+1):74]), 74 - tau + (1/mean(S)) )

  log_probabilities <- rep(0, Ndays - 1)

  # Calculate probabilities for each t
  for (t in 1:(Ndays - 1)) {
    log_probabilities[t] <- calculate_log_probability(t, alpha, beta, Ndays=74)
  }

  max_log_prob <- max(log_probabilities)
  probabilities <- exp(log_probabilities - max_log_prob)
  probabilities <- probabilities / sum(probabilities)

  tau <- sample(1:73, size=1, prob = probabilities)

  result[i, ] <- c(alpha, beta, tau)
}
  result
}

set.seed(156)
samples <- do.the.gibbs(10000, mean(S), mean(S), 10)
```

```r
###

burn_in <- 1000

plot(samples[(burn_in + 1):10000, 1], type='l', main="Traceplot for alpha")
plot(samples[(burn_in + 1):10000,2], type='l', main= "Traceplot for beta")
plot(samples[(burn_in + 1):10000,3], type='l', main= "Traceplot for tau")




# Define the lambda function
lambda <- function(alpha, beta, tau, i) ifelse(i <= tau, alpha, beta)

# Generate a sequence from 1 to 74 for i
i_values <- 1:74

# Calculate lambda(i; tau, alpha, beta) for each i using the samples
results <- sapply(i_values, function(i) {
  mean(apply(samples[(burn_in + 1):10000, ], 1, function(row) lambda(row[1], row[2], row[3], i)))
})
results

plot(results, type='l', xlab = "Daily count", ylab= "Expected number of calls"
     ,main="Monte Carlo approximation of expected number of calls per day")

### Question 2d - Data Augmentation:


# Define a function to calculate the probability for a given t
calculate_probability <- function(t, alpha, beta, Ndays=74) {
  exp(-alpha * t) * alpha^(sum(S[1:t])) * exp(-beta * (Ndays - t)) * beta^(sum(S[(t+1):Ndays]))
}


calculate_log_probability <- function(t, alpha, beta, Ndays=74) {
  (-alpha * t) + (sum(S[1:t]))*log(alpha) - (beta * (Ndays - t)) +  sum(S[(t+1):Ndays]) * log(beta)
}


do.the.gibbs2 <- function(niter, alpha0, beta0, tau0, S25, S42, S55) {

  result <- matrix(nrow = niter, ncol = 6)
  alpha <- alpha0
  beta <- beta0
  tau <- tau0
  S[25] <- S25
  S[42] <- S42
  S[55] <- S55

  for (i in 1:niter) {
    alpha <- rgamma(1, 1 + sum(S[1:tau]), tau + (1/mean(S)))
    beta <- rgamma(1, 1 + sum(S[(tau+1):74]), 74 - tau + (1/mean(S)) )

  log_probabilities <- rep(0, Ndays - 1)

  # Calculate probabilities for each t
  for (t in 1:(Ndays - 1)) {
    log_probabilities[t] <- calculate_log_probability(t, alpha, beta, Ndays=74)
  }

  max_log_prob <- max(log_probabilities)
  probabilities <- exp(log_probabilities - max_log_prob)
  probabilities <- probabilities / sum(probabilities)


  tau <- sample(1:73, size=1, prob = probabilities)


  for (j in c(25,42,55)) {
    if (j <= tau) {
      S[j] <- rpois(1, alpha)
    }
    else{
      S[j] <- rpois(1, beta)

    }
}


  result[i, ] <- c(alpha, beta, tau, S[25], S[42], S[55])
}
  result
}

samples2 <- do.the.gibbs2(10000, mean(S), mean(S), 10,
                         mean(S),  mean(S),  mean(S))



#plot(density(samples2[, 4]), type='l', main = "Posterior densities of missing points")
#lines(density(samples2[, 5]))
#lines(density(samples2[, 6]))





plot(density(samples2[(burn_in + 1):10000, 4]), type = 'l', col = 'blue', lty = 1,
     main = "Posterior densities of missing points", xlab = "S", ylab = "Density")

lines(density(samples2[(burn_in + 1):10000, 5]), col = 'red', lty = 2)
lines(density(samples2[(burn_in + 1):10000, 6]), col = 'green', lty = 3)

legend("topright", legend = c("S25", "S42", "S55"),
       col = c("blue", "red", "green"), lty = c(1, 2, 3))
```

```r
#Repeating steps in part c

# Define the lambda function
lambda <- function(alpha, beta, tau, i) ifelse(i <= tau, alpha, beta)

# Generate a sequence from 1 to 74 for i
i_values <- 1:74

# Calculate lambda(i; tau, alpha, beta) for each i using the samples
results2 <- sapply(i_values, function(i) {
  mean(apply(samples2[(burn_in + 1):10000, ], 1, function(row) lambda(row[1], row[2], row[3], i)))
})

plot(results2, type='l', , xlab = "Daily count", ylab= "Expected number of calls",
     main="Monte Carlo approximation of expected number of calls per day")
```

4 References