STAT 601 A3: SIR Individual-Level Models Epidemic Modelling

Hao Nan Wang

2023-04-18

part a

```
data = read.csv("A3EpidemicSheep.csv")
head(data)
```

```
##   X    V1    V2 V3 V4 V5 V6         V7
## 1 1  93.7 109.1  6  8  1  1 0.93280870
## 2 2 101.8 103.8  5  7  1  1 0.20198302
## 3 3 121.6 126.8  6  9  1  1 0.79237876
## 4 4 116.0  93.6  7  9  0  1 0.22463051
## 5 5 103.3  95.4  4  5  0  0 0.03075657
## 6 6 121.8 124.3  6  8  0  1 0.86203404
```

```
library(EpiILM)
```

```
## Loading required package: coda
```

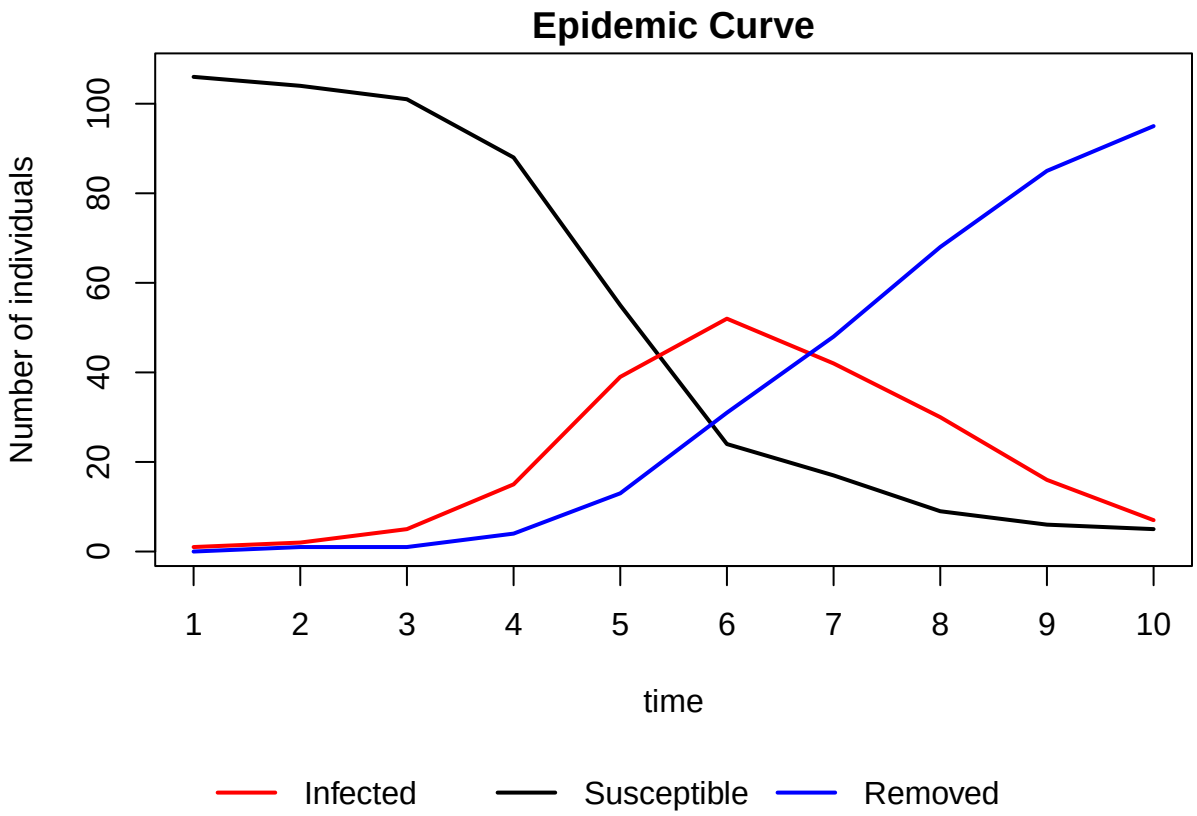
```
## Loading required package: adaptMCMC
```

```
## Loading required package: parallel
```

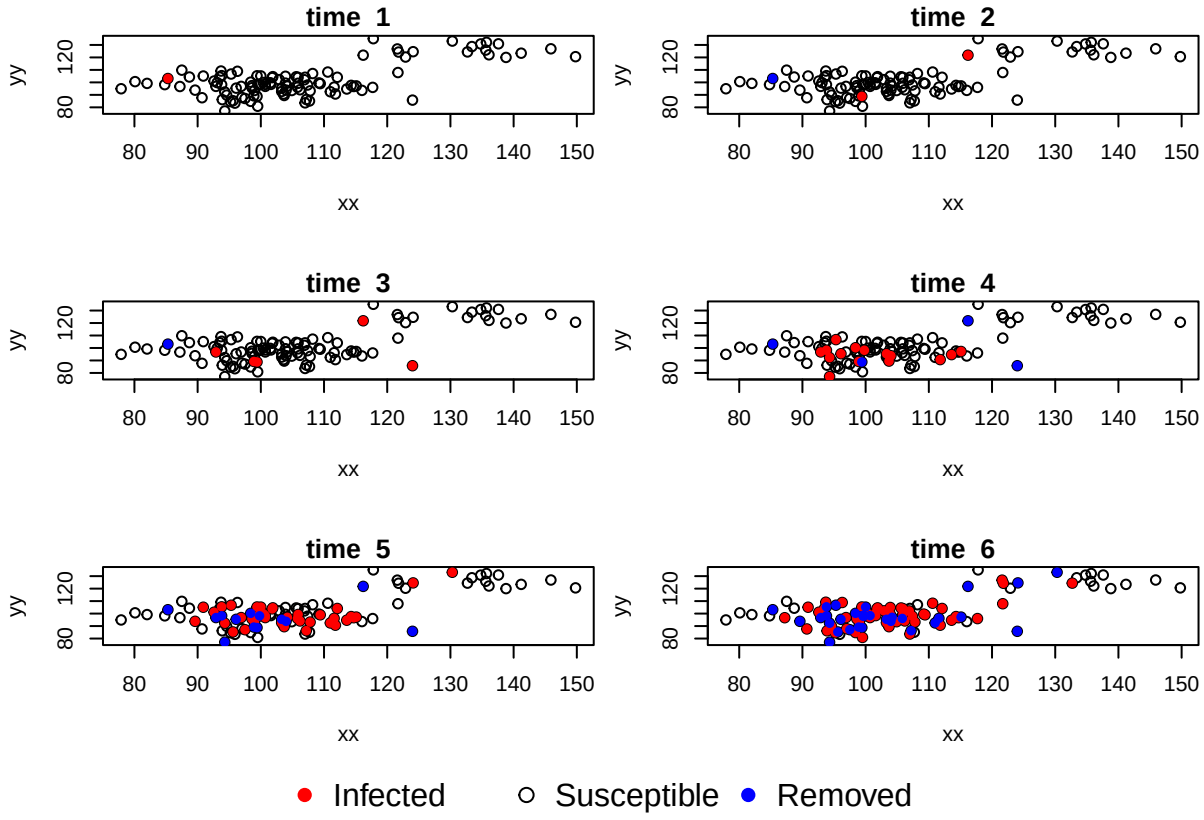
```
## Loading required package: Matrix
```

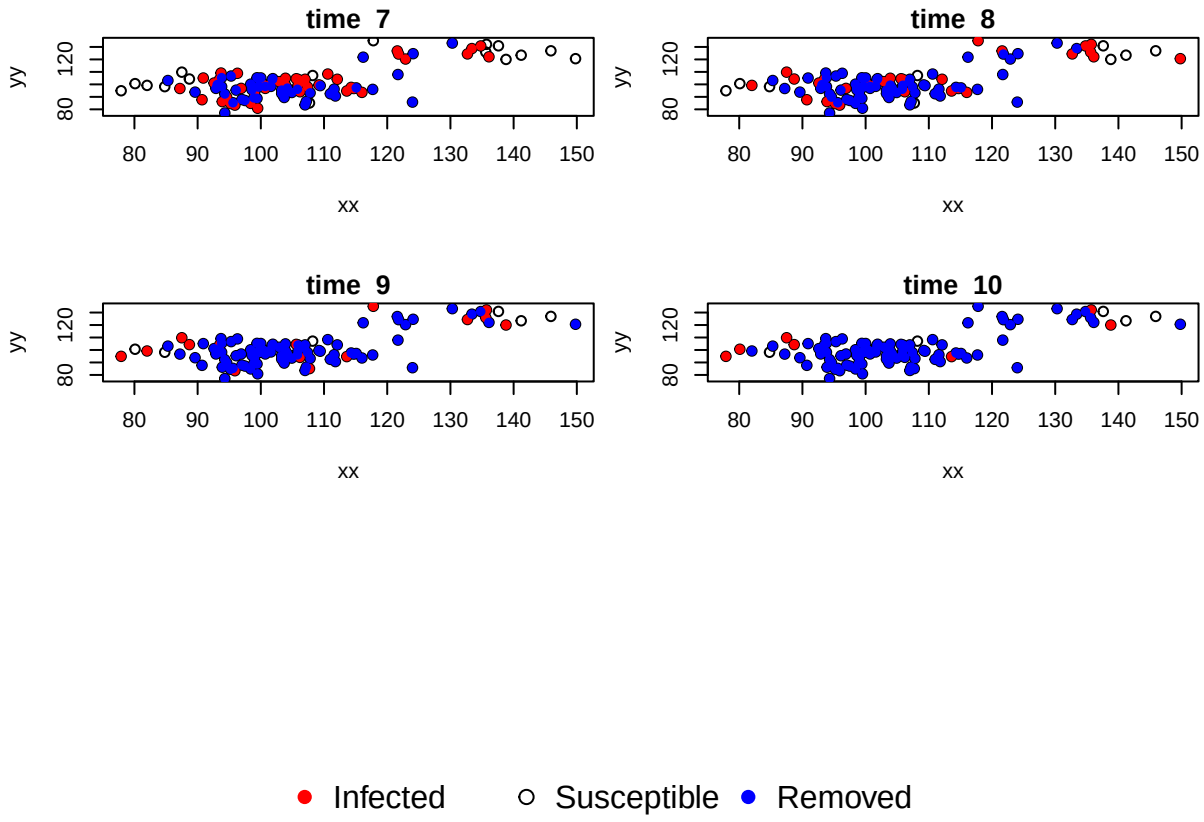
```
## Loading required package: LaplacesDemon
```

```
epidat = as.epidata(type = "SIR", n =107, x=data$V1, y=data$V2,
                    inftime = data$V3, infperiod = (data$V4-data$V3))
plot(epidat, plottype = "curve", curvetype = "complete")
```




```
plot(epidat, plottype = "spatial")
```


time 1
time 2
time 3
time 4
time 5
time 6
yy
xx
80
90
100
110
120
130
140
150
Infected
Susceptible
Removed


time 7
time 8
time 9
time 10
yy
xx
80
90
100
110
120
130
140
150
Infected
Susceptible
Removed

part b

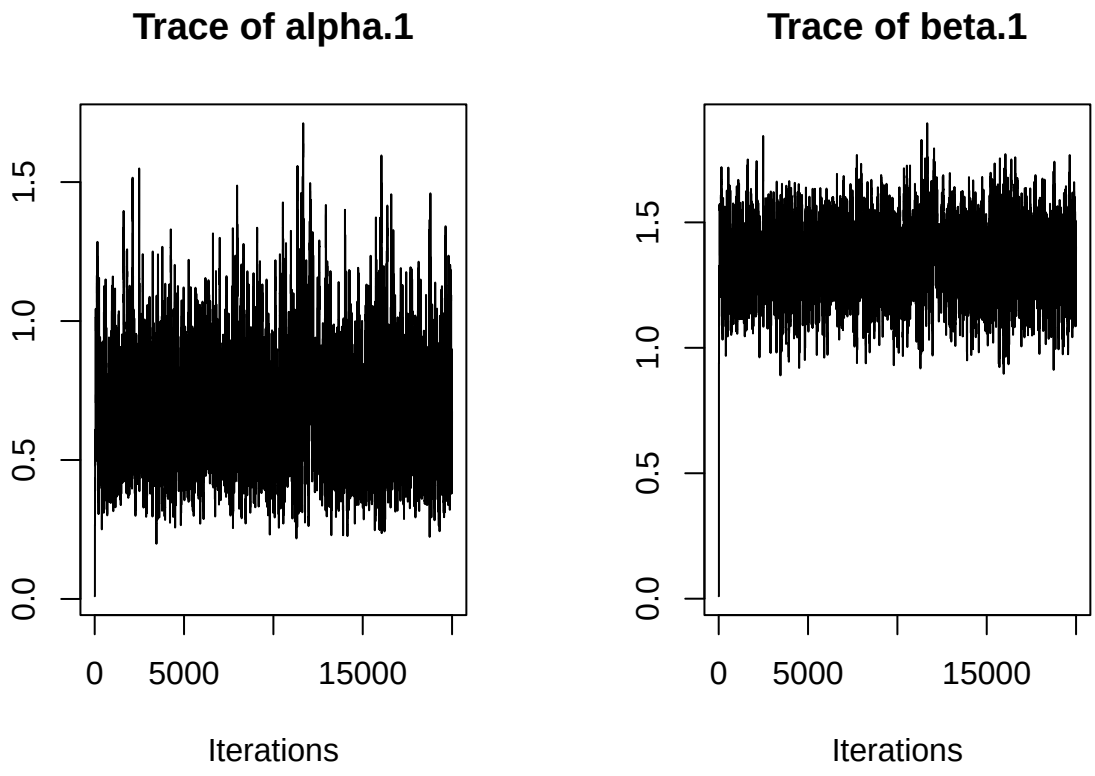
Here we have two parameters $\theta = (\alpha, \beta)$

```
mcmcout <- epimcmc(epidat, tmax = max(epidat$inftime), niter=20000, sus.par.ini=0.01,
                   beta.ini=0.01, adapt=TRUE, Sformula = NULL, pro.sus.var = 0.1,
                   pro.beta.var = 0.1, prior.sus.dist="gamma", prior.sus.par=c(1,1),
                   prior.beta.dist="uniform", prior.beta.par =c(0,10000), acc.rate = 0.5)
```

```
##   generate 20000 samples
```

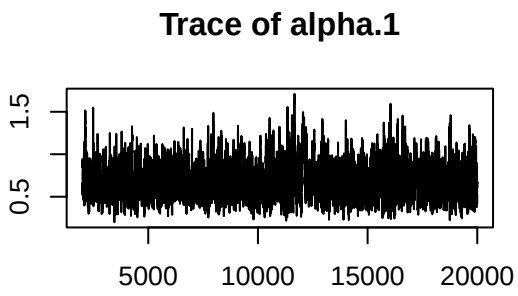
Trace plots including burn-in

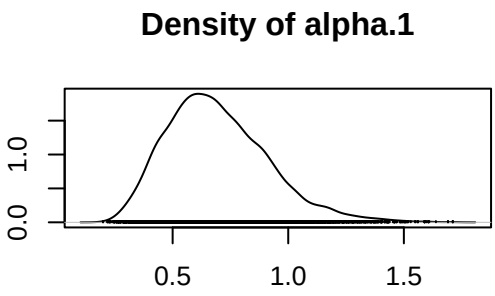
```
plot(mcmcout, partype = "parameter", start = 1, density=FALSE)
```

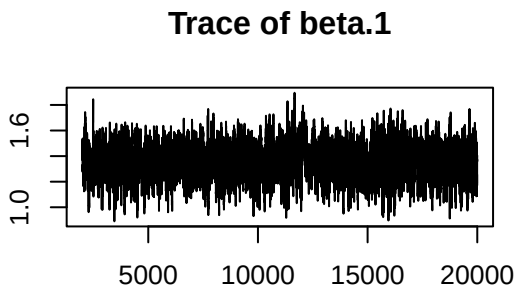


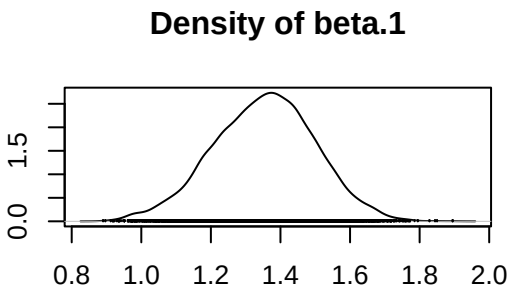
Trace plots with posterior density but this time we remove burn-in, say 2000.

```
plot(mcmcout, partype = "parameter", start = 2000, density=TRUE)
```









Posterior Summary Statistics (means and 95% credible intervals)

```
summary(mcmcout, start=2000)
```

```
## Model: SIR distance-based discrete-time ILM
## Method: Markov chain Monte Carlo (MCMC)

##
## Iterations = 2000:20000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 18001
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean     SD Naive SE Time-series SE
## alpha.1 0.6937 0.2186 0.001629       0.006735
## beta.1  1.3532 0.1452 0.001082       0.004081
##
## 2. Quantiles for each variable:
##
##           2.5%    25%    50%    75% 97.5%
## alpha.1 0.3414 0.5349 0.6683 0.8261 1.192
## beta.1  1.0584 1.2540 1.3582 1.4514 1.632
```

We can see that the posterior mean estimate of spatial parameter is around 1.357 and the posterior mean estimate of the baseline susceptibility parameter is around 0.697. Also, from the summary table, we can see that the 95% credible interval of the spatial parameter is (1.073, 1.624) and the 95% credible interval of the baseline susceptibility parameter is (0.346, 1.192)

DIC Calculation (burn-in removed)

```
sus.parameters = mean(unlist(mcmcout$Estimates[2000:20000,1]))
beta.par = mean(unlist(mcmcout$Estimates[2000:20000,2]))
loglike <- epilike(epidat, tmax = max(epidat$inftime), Sformula = NULL,
                   sus.par = sus.parameters, beta = beta.par)
dic <- epidic(burnin = 2000, niter = 20000, LLchain = mcmcout$Loglikelihood, LLpostmean = loglike)
dic
```

```
## [1] 387.5303
```

part c

We now have three candidate variables: V5 (indicator of relative size of the farm), V6 (indicator of sheep alone) and V7 (herd density). So we have the following seven combinations:

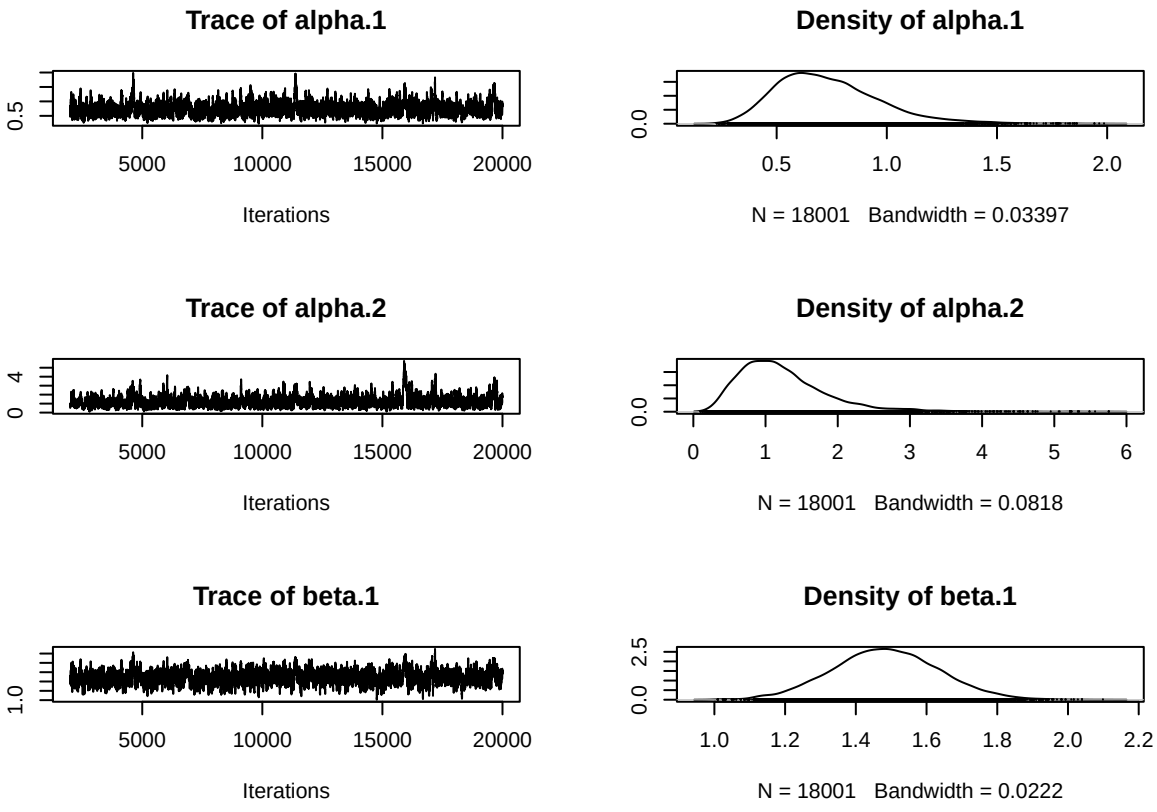
- V5
- V6
- V7
- V5+V6
- V5+V7
- V6+V7
- V5+V6+V7

Note: the parameters/coefficients to be estimated $\alpha_0, ..., \alpha_{ns} \geq 0$

```
# V5
mcmcout_v5 <- epimcmc(epidat, tmax = max(epidat$inftime), niter=20000, sus.par.ini=c(0.01,0.05),
                      beta.ini=0.01, adapt=TRUE, Sformula = ~data$V5, pro.sus.var = c(0.1,0.1),
                      pro.beta.var = 0.1, prior.sus.dist=c("uniform", "uniform"),
                      prior.sus.par=matrix(c(0,0,10000,10000), ncol = 2, nrow = 2),
                      prior.beta.dist="uniform", prior.beta.par =c(0,10000), acc.rate = 0.5)
```

```
##   generate 20000 samples
```

```
# Trace plots with posterior density but this time we remove burn-in, say 2000.
plot(mcmcout_v5, partype = "parameter", start = 2000, density=TRUE)
```

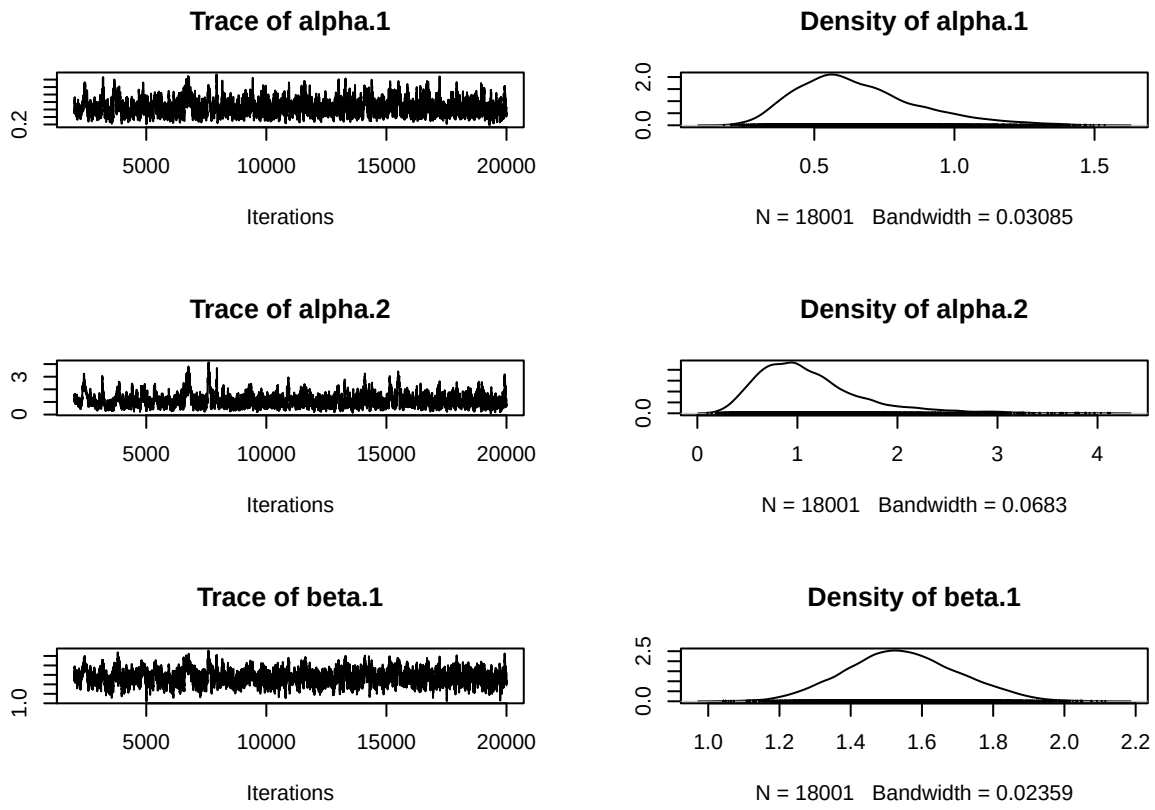


```
# DIC computation
sus.parameters = c(mean(unlist(mcmcout_v5$Estimates[2000:20000,1])),
                   mean(unlist(mcmcout_v5$Estimates[2000:20000,2])))
beta.par = mean(unlist(mcmcout_v5$Estimates[2000:20000,3]))
loglike <- epilike(epidat, tmax = max(epidat$inftime), Sformula = ~data$V5,
                   sus.par = sus.parameters, beta = beta.par)
dic_V5 <- epidic(burnin = 2000, niter = 20000, LLchain = mcmcout_v5$Loglikelihood, LLpostmean = loglike
```

```
# V6
mcmcout_v6 <- epimcmc(epidat, tmax = max(epidat$inftime), niter=20000, sus.par.ini=c(0.01,0.05),
                      beta.ini=0.01, adapt=TRUE, Sformula = ~data$V6, pro.sus.var = c(0.1,0.1),
                      pro.beta.var = 0.1, prior.sus.dist=c("uniform", "uniform"),
                      prior.sus.par=matrix(c(0,0,10000,10000), ncol = 2, nrow = 2),
                      prior.beta.dist="uniform", prior.beta.par =c(0,10000), acc.rate = 0.5)
```

```
##   generate 20000 samples
```

```
# Trace plots with posterior density but this time we remove burn-in, say 2000.
plot(mcmcout_v6, partype = "parameter", start = 2000, density=TRUE)
```



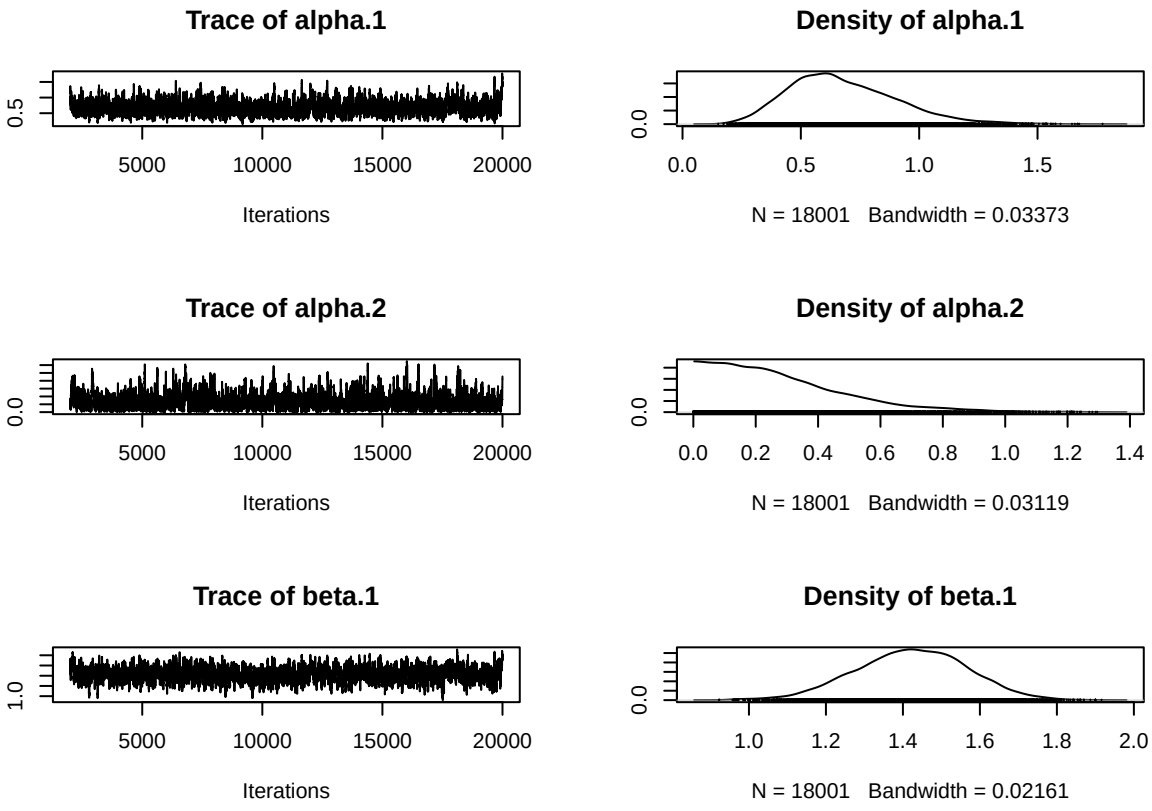

```
# DIC computation
sus.parameters = c(mean(unlist(mcmcout_v6$Estimates[2000:20000,1])),
                   mean(unlist(mcmcout_v6$Estimates[2000:20000,2])))
beta.par = mean(unlist(mcmcout_v6$Estimates[2000:20000,3]))
```

```
loglike <- epilike(epidat, tmax = max(epidat$inftime), Sformula = ~data$V6,
                   sus.par = sus.parameters, beta = beta.par)
dic_V6 <- epidic(burnin = 2000, niter = 20000, LLchain = mcmcout_v6$Loglikelihood, LLpostmean = loglike
```

```
# V7
mcmcout_v7 <- epimcmc(epidat, tmax = max(epidat$inftime), niter=20000, sus.par.ini=c(0.01,0.05),
                      beta.ini=0.01, adapt=TRUE, Sformula = ~data$V7, pro.sus.var = c(0.1,0.1),
                      pro.beta.var = 0.1, prior.sus.dist=c("uniform", "uniform"),
                      prior.sus.par=matrix(c(0,0,10000,10000), ncol = 2, nrow = 2),
                      prior.beta.dist="uniform", prior.beta.par =c(0,10000), acc.rate = 0.5)
```

```
##   generate 20000 samples
```

```
# Trace plots with posterior density but this time we remove burn-in, say 2000.
plot(mcmcout_v7, partype = "parameter", start = 2000, density=TRUE)
```




```
# DIC computation
sus.parameters = c(mean(unlist(mcmcout_v7$Estimates[2000:20000,1])),
                   mean(unlist(mcmcout_v7$Estimates[2000:20000,2])))
beta.par = mean(unlist(mcmcout_v7$Estimates[2000:20000,3]))
loglike <- epilike(epidat, tmax = max(epidat$inftime), Sformula = ~data$V7,
                   sus.par = sus.parameters, beta = beta.par)
dic_V7 <- epidic(burnin = 2000, niter = 20000, LLchain = mcmcout_v7$Loglikelihood, LLpostmean = loglike
```

```
# V5+V6
mcmcout_v5_v6 <- epimcmc(epidat, tmax = max(epidat$inftime), niter=20000,
                      sus.par.ini=c(0.01,0.05,0.07),
                      beta.ini=0.01, adapt=TRUE, Sformula = ~data$V5+data$V6,
                      pro.sus.var = c(0.1,0.1,0.1),
                      pro.beta.var = 0.1, prior.sus.dist=c("uniform", "uniform", "uniform"),
                      prior.sus.par=matrix(c(0,0,0,10000,10000,10000), ncol = 2, nrow = 3),
                      prior.beta.dist="uniform", prior.beta.par =c(0,10000), acc.rate = 0.5)
```

```
##   generate 20000 samples
```

```
# Trace plots with posterior density but this time we remove burn-in, say 2000.
dev.new(width=10, height=10)
plot(mcmcout_v5_v6, partype = "parameter", start = 2000, density=TRUE)
# DIC computation
sus.parameters = c(mean(unlist(mcmcout_v5_v6$Estimates[2000:20000,1])),
                   mean(unlist(mcmcout_v5_v6$Estimates[2000:20000,2])),
                   mean(unlist(mcmcout_v5_v6$Estimates[2000:20000,3])))
beta.par = mean(unlist(mcmcout_v5_v6$Estimates[2000:20000,4]))
loglike <- epilike(epidat, tmax = max(epidat$inftime), Sformula = ~data$V5+data$V6,
                   sus.par = sus.parameters, beta = beta.par)
dic_V5_V6 <- epidic(burnin = 2000, niter = 20000, LLchain = mcmcout_v5_v6$Loglikelihood,
                 LLpostmean = loglike)
```

```
# V5+V7
mcmcout_v5_v7 <- epimcmc(epidat, tmax = max(epidat$inftime), niter=20000,
                      sus.par.ini=c(0.01,0.05,0.07),
                      beta.ini=0.01, adapt=TRUE, Sformula = ~data$V5+data$V7,
                      pro.sus.var = c(0.1,0.1,0.1),
                      pro.beta.var = 0.1, prior.sus.dist=c("uniform", "uniform", "uniform"),
                      prior.sus.par=matrix(c(0,0,0,10000,10000,10000), ncol = 2, nrow = 3),
                      prior.beta.dist="uniform", prior.beta.par =c(0,10000), acc.rate = 0.5)
```

```
##   generate 20000 samples
```

```
# Trace plots with posterior density but this time we remove burn-in, say 2000.
dev.new(width=10, height=10)
plot(mcmcout_v5_v7, partype = "parameter", start = 2000, density=TRUE)
# DIC computation
sus.parameters = c(mean(unlist(mcmcout_v5_v7$Estimates[2000:20000,1])),
                   mean(unlist(mcmcout_v5_v7$Estimates[2000:20000,2])),
                   mean(unlist(mcmcout_v5_v7$Estimates[2000:20000,3])))
beta.par = mean(unlist(mcmcout_v5_v7$Estimates[2000:20000,4]))
loglike <- epilike(epidat, tmax = max(epidat$inftime), Sformula = ~data$V5+data$V7,
                   sus.par = sus.parameters, beta = beta.par)
dic_V5_V7 <- epidic(burnin = 2000, niter = 20000, LLchain = mcmcout_v5_v7$Loglikelihood,
                 LLpostmean = loglike)
```

```
# V6+V7
mcmcout_v6_v7 <- epimcmc(epidat, tmax = max(epidat$inftime), niter=20000,
                      sus.par.ini=c(0.01,0.05,0.07),
                      beta.ini=0.01, adapt=TRUE, Sformula = ~data$V6+data$V7,
```

```
                      pro.sus.var = c(0.1,0.1,0.1),
                      pro.beta.var = 0.1, prior.sus.dist=c("uniform", "uniform", "uniform"),
                      prior.sus.par=matrix(c(0,0,0,10000,10000,10000), ncol = 2, nrow = 3),
                      prior.beta.dist="uniform", prior.beta.par =c(0,10000), acc.rate = 0.5)
```

```
##   generate 20000 samples
```

```
# Trace plots with posterior density but this time we remove burn-in, say 2000.
dev.new(width=10, height=10)
plot(mcmcout_v6_v7, partype = "parameter", start = 2000, density=TRUE)
# DIC computation
sus.parameters = c(mean(unlist(mcmcout_v6_v7$Estimates[2000:20000,1])),
                   mean(unlist(mcmcout_v6_v7$Estimates[2000:20000,2])),
                   mean(unlist(mcmcout_v6_v7$Estimates[2000:20000,3])))
beta.par = mean(unlist(mcmcout_v6_v7$Estimates[2000:20000,4]))
loglike <- epilike(epidat, tmax = max(epidat$inftime), Sformula = ~data$V6+data$V7,
                   sus.par = sus.parameters, beta = beta.par)
dic_V6_V7 <- epidic(burnin = 2000, niter = 20000, LLchain = mcmcout_v6_v7$Loglikelihood,
                 LLpostmean = loglike)
```

```
# V5+V6+V7
mcmcout_v5_v6_v7 <- epimcmc(epidat, tmax = max(epidat$inftime), niter=20000,
                      sus.par.ini=c(0.01,0.05,0.07,0.09),
                      beta.ini=0.01, adapt=TRUE, Sformula = ~data$V5+data$V6+data$V7,
                      pro.sus.var = c(0.1,0.1,0.1,0.1),
                      pro.beta.var = 0.1, prior.sus.dist=c("uniform", "uniform", "uniform", "uniform"),
                      prior.sus.par=matrix(c(0,0,0,0,10000,10000,10000,10000), ncol = 3, nrow = 4),
                      prior.beta.dist="uniform", prior.beta.par =c(0,10000), acc.rate = 0.5)
```

```
## Warning in matrix(c(0, 0, 0, 0, 10000, 10000, 10000, 10000), ncol = 3, nrow =
## 4): data length [8] is not a sub-multiple or multiple of the number of columns
## [3]
```

```
##   generate 20000 samples
```

```
dev.new(width=10, height=10)
# Trace plots with posterior density but this time we remove burn-in, say 2000.
plot(mcmcout_v5_v6_v7, partype = "parameter", start = 2000, density=TRUE)
# DIC computation
sus.parameters = c(mean(unlist(mcmcout_v5_v6_v7$Estimates[2000:20000,1])),
                   mean(unlist(mcmcout_v5_v6_v7$Estimates[2000:20000,2])),
                   mean(unlist(mcmcout_v5_v6_v7$Estimates[2000:20000,3])),
                   mean(unlist(mcmcout_v5_v6_v7$Estimates[2000:20000,4])))
beta.par = mean(unlist(mcmcout_v5_v6_v7$Estimates[2000:20000,5]))
loglike <- epilike(epidat, tmax = max(epidat$inftime), Sformula = ~data$V5+data$V6+data$V7,
                   sus.par = sus.parameters, beta = beta.par)
dic_V5_V6_V7 <- epidic(burnin = 2000, niter = 20000, LLchain = mcmcout_v5_v6_v7$Loglikelihood,
                 LLpostmean = loglike)
```

```
name = c("V5","V6","V7","V5+V6","V5+V7","V6+V7","V5+V6+V7")
DIC = c(dic_V5,dic_V6,dic_V7,dic_V5_V6,dic_V5_V7,dic_V6_V7,dic_V5_V6_V7)
result = data.frame(name, DIC)
knitr::kable(result, "pipe", col.names = c("Model Name", "DIC Value"))
```

Model Name	DIC Value
V5	446.8235
V6	441.4782
V7	459.6482
V5+V6	468.1143
V5+V7	485.0456
V6+V7	478.2151
V5+V6+V7	506.0390

We want the model with the smallest DIC value, therefore, the final model contains one covariate only, which is V6 (sheep)

Use Posterior Predictive Approach to Judge ...

- We will employ a posterior predictive approach to make predictions based on data up to
  - time 3 (the number of newly infected individuals skyrocketed after this time point)
  - time 5 (peak)
  - time 7 (after a sudden drop in the number of newly infected individuals, the number of new infections is expected to remain stable).

This approach utilizes the available data up to these time points to estimate the posterior distribution of parameters and subsequently generate predictions for future outcomes.

```
# Posterior predictive forecasting
# Posterior prediction starting from time point 5
pred.SIR.spatial.point.3 <- pred.epi(epidat, xx = mcmcout_v6, tmin = 3,
    Sformula = ~data$V6, Tformula = NULL,
    burnin = 2000, criterion = "newly infectious",
    n.samples = 500)
```

```
## generate 500 epidemics
```

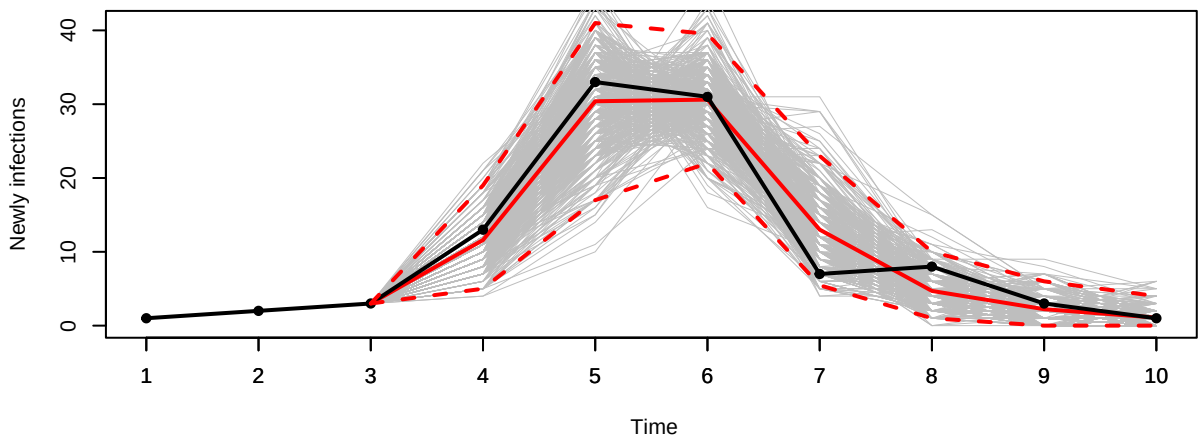
```
pred.SIR.spatial.point.5 <- pred.epi(epidat, xx = mcmcout_v6, tmin = 5,
    Sformula = ~data$V6, Tformula = NULL,
    burnin = 2000, criterion = "newly infectious",
    n.samples = 500)
```

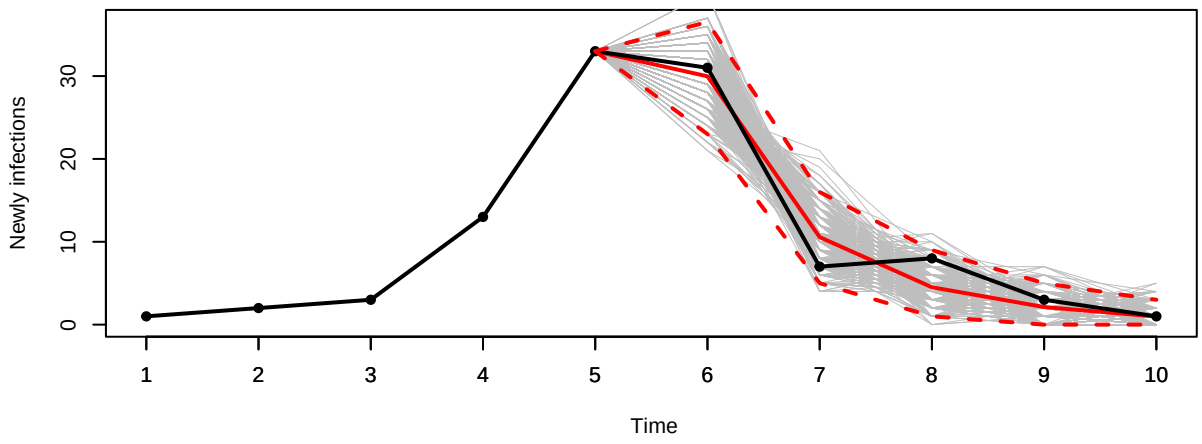
```
## generate 500 epidemics
```

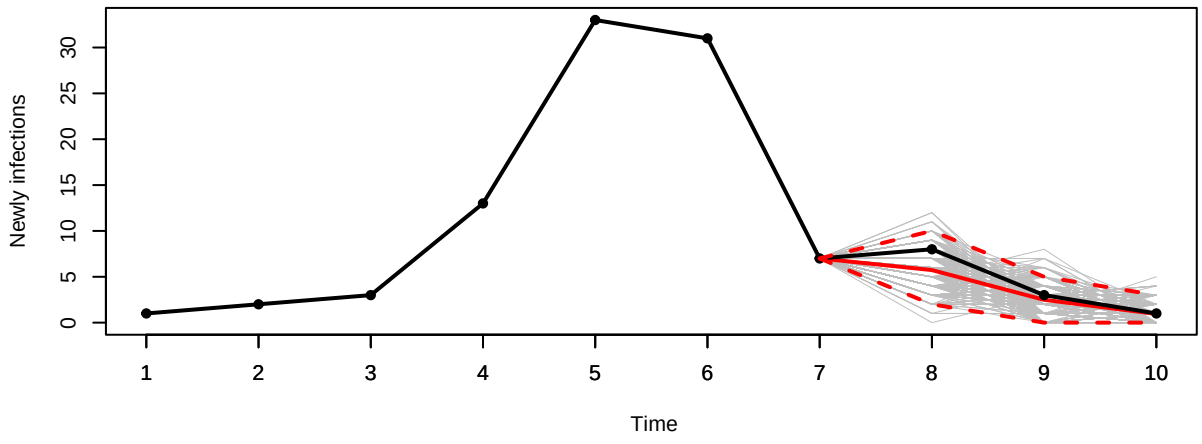
```
pred.SIR.spatial.point.7 <- pred.epi(epidat, xx = mcmcout_v6, tmin = 7,
    Sformula = ~data$V6, Tformula = NULL,
    burnin = 2000, criterion = "newly infectious",
    n.samples = 500)
```

```
## generate 500 epidemics
```

```r
# plot predictions:
par(mfrow = c(3,1))
plot(pred.SIR.spatial.point.3, col = "red", lwd = 2, pch = 19)
plot(pred.SIR.spatial.point.5, col = "red", lwd = 2, pch = 19)
plot(pred.SIR.spatial.point.7, col = "red", lwd = 2, pch = 19)
```







One can see that there are no obvious deficiencies between the original data and the simulation data (from

the posterior predictive distribution); however, one may argue that the simulated data at time point 7 is overestimated, and the simulated data at time point 8 is underestimated. The big drop from time point 6 to 7 is something that should be explored more. My hypothesis is that covariate V6/sheep may not contribute the most to this drop. Also, there is a slight bounceback between time point 7 and 8, which requires more exploration. My hypothesis is that covariate V6/sheep may not contribute the most to this bounceback. Overall, I believe the model I proposed could have good predictive accuracy using the posterior predictive approach, and the credible interval contains the real data, with low variance in most cases. Therefore, I believe the posterior predictive distribution does a good job of providing accuracy and precision.

part d

```
data = read.csv("A3EpidemicSheep.csv")
head(data)
```

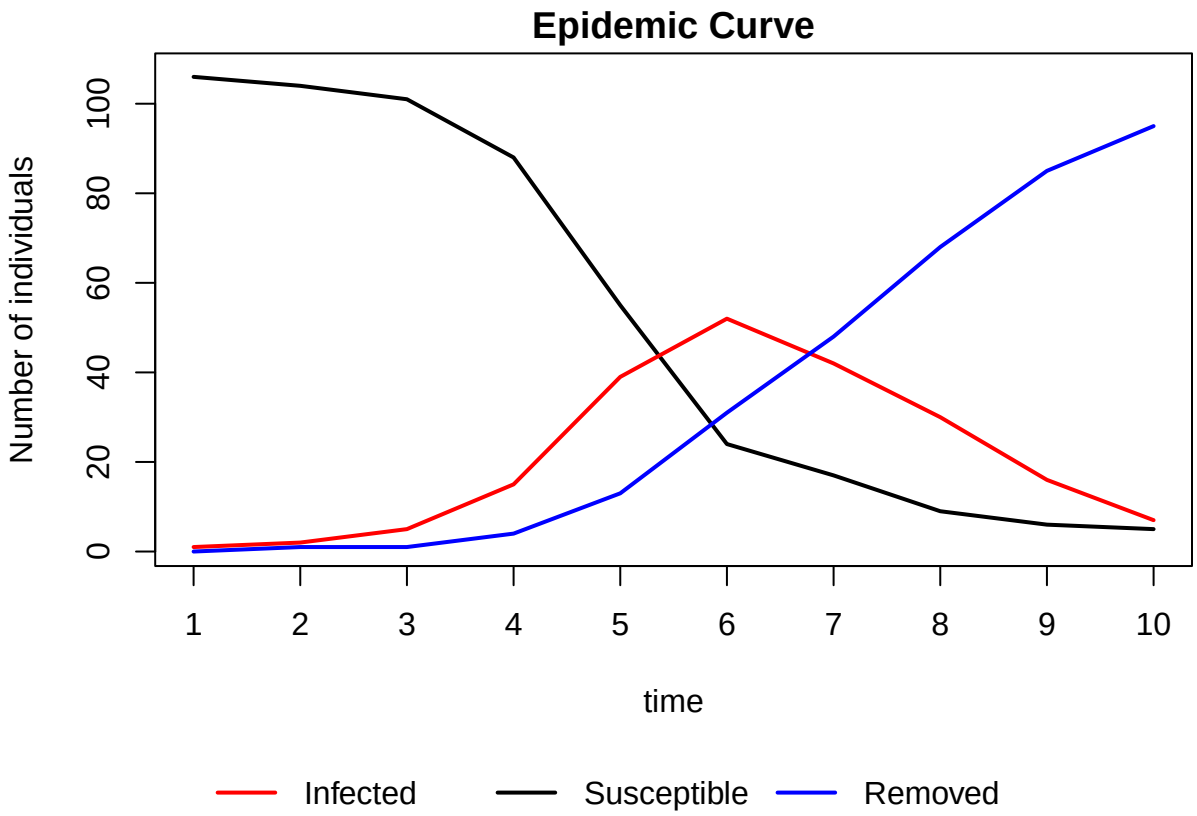
```
##   X    V1    V2 V3 V4 V5 V6         V7
## 1 1  93.7 109.1  6  8  1  1 0.93280870
## 2 2 101.8 103.8  5  7  1  1 0.20198302
## 3 3 121.6 126.8  6  9  1  1 0.79237876
## 4 4 116.0  93.6  7  9  0  1 0.22463051
## 5 5 103.3  95.4  4  5  0  0 0.03075657
## 6 6 121.8 124.3  6  8  0  1 0.86203404
```

```
library(EpiILM)
```

The final chosen spatial ILM from part c) is the model with covaraite V6. The contact matrix is euclidean distance between each pair of points. I also apply the logistic transformation to it and round to 2 decimal places because I want to make sure the transformed range is (0,1). Although it could be any positive number, I feel that this transformation is needed for making the computation easy

```
set.seed(2023)
# for a population of 107 individuals
n = 107
# Generate a random matrix with dimensions 107x107:
# put the xy coordinates in a matrix called "data_mat"
data_mat = matrix(epidat$XYcoordinates, ncol = 2, nrow = n)
contact_mat <- matrix(0, ncol = n, nrow = n)  # initialize contact matrix to 0

# Compute Euclidean distance between each pair of points
for (i in 1:n) {
  for (j in 1:n) {
    if (i == j) {
      contact_mat[i, j] <- 0  # diagonal elements are 0
    } else {
      dist <- sqrt((data_mat[i, 1]-data_mat[j, 1])^2 + (data_mat[i, 2]-data_mat[j, 2])^2)  # Euclidean
      dist_mod <- round(dist/10, 2)  # modify distance by dividing by 10 and keeping 2 decimal places
      contact_mat[i, j] <- dist_mod  # set value in contact matrix
    }
  }
}

# Apply the logistic transformation to contact_mat and round to 2 decimal places because
# I want to make sure the transformed range is (0,1). Although it could be any positive
# number, I feel that this transformation is needed for making the computation easy
logistic_mat <- round(plogis(contact_mat), 2)
```

```
#epidat2 = as.epidata(type = "SIR", n =107, inftime = data$V3,
#                     infperiod = data$V4-data$V3, contact = contact_mat)
epidat2 = as.epidata(type = "SIR", n =107, contact = logistic_mat,
                  inftime = data$V3, infperiod = (data$V4-data$V3))

plot(epidat2, plottype = "curve", curvetype = "complete")
```



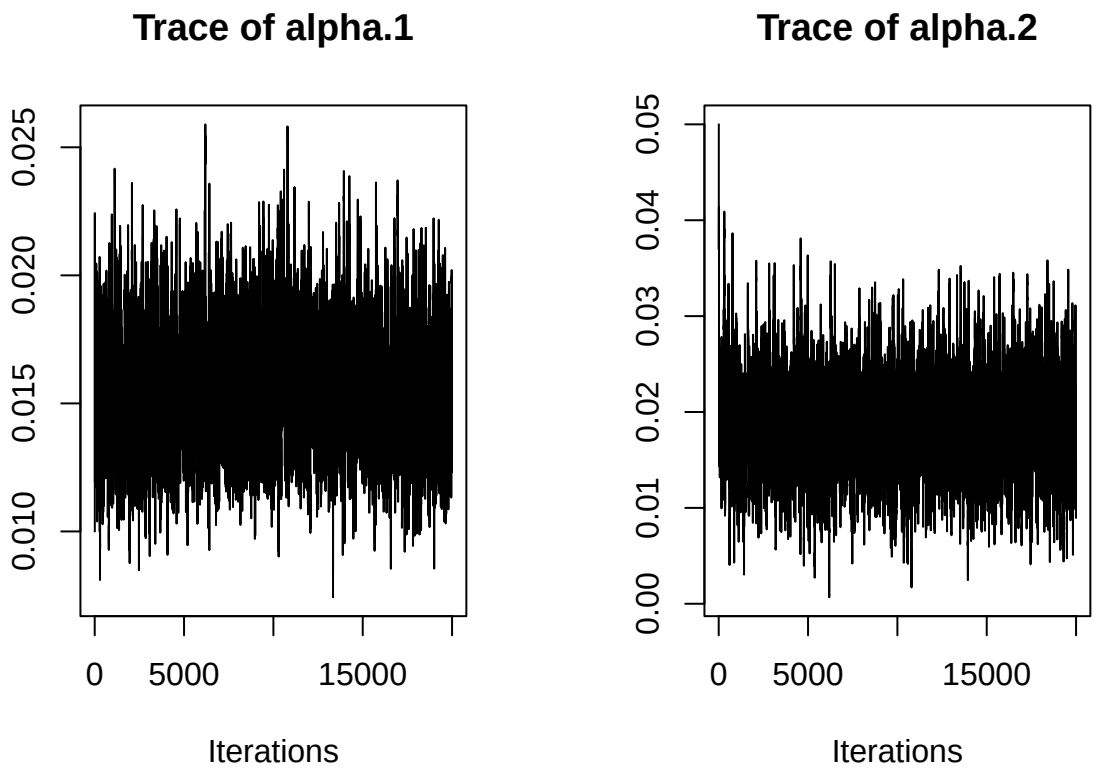
Run MCMC for 20000 iterations

```
mcmcout_partd <- epimcmc(epidat2, tmax = max(epidat2$inftime), niter=20000, sus.par.ini= c(0.01,0.05),
                         adapt=TRUE, Sformula = ~ data$V6, pro.sus.var = c(0.001,0.001),
                         prior.sus.dist=c("uniform", "uniform"),
                         prior.sus.par=matrix(c(0,0,10000,10000), ncol = 2, nrow = 2),
                         acc.rate = 0.5)
```

```
##   generate 20000 samples
```

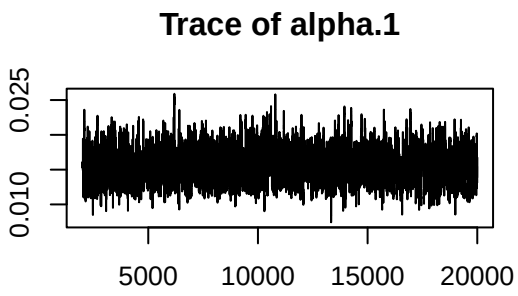
Trace Plots including burn-in

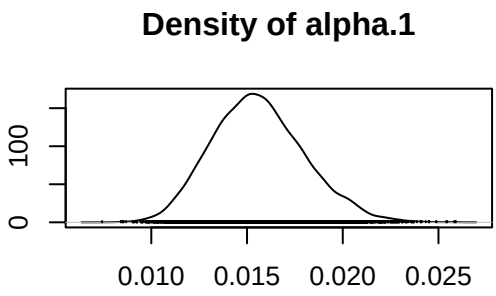
```
plot(mcmcout_partd, partype = "parameter", start = 1, density = FALSE)
```

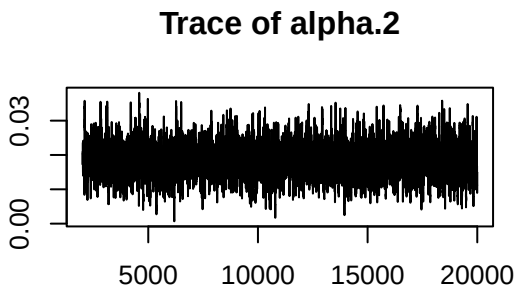


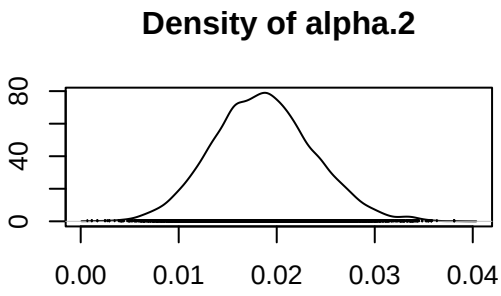
Trace Plots with Posterior Density (removed burn-in, say 2000)

```
plot(mcmcout_partd, partype = "parameter", start = 2000, density = TRUE)
```










```
# DIC computation
sus.parameters = c(mean(unlist(mcmcout_partd$Estimates[2000:20000,1])),
                   mean(unlist(mcmcout_partd$Estimates[2000:20000,2])))
# LLchain: Loglikelihood values from the MCMC output
# LLpostmean: Loglikelihood value of the model with posterior mean of estimates
loglike <- epilike(epidat2, tmax = max(epidat2$inftime), Sformula = ~data$V6,
                   sus.par = sus.parameters)
dic_partd <- epidic(burnin = 2000, niter = 20000, LLchain = mcmcout_partd$Loglikelihood,
                    LLpostmean = loglike)
dic_partd
```

```
## [1] 485.0105
```

Summary of Posterior Statistics

```
summary(mcmcout_partd, start = 2000)
```

```
## Model: SIR network-based discrete-time ILM
## Method: Markov chain Monte Carlo (MCMC)

##
## Iterations = 2000:20000
## Thinning interval = 1
## Number of chains = 1
```

```
## Sample size per chain = 18001
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##            Mean       SD  Naive SE Time-series SE
## alpha.1 0.01560 0.002405 1.793e-05      5.793e-05
## alpha.2 0.01863 0.005106 3.805e-05      1.312e-04
##
## 2. Quantiles for each variable:
##
##             2.5%     25%     50%     75%   97.5%
## alpha.1 0.011305 0.01391 0.01547 0.01716 0.02064
## alpha.2 0.009078 0.01519 0.01853 0.02192 0.02908
```

Use Posterior Predictive Approach to Judge ...

- We will employ a posterior predictive approach to make predictions based on data up to
  - time 3 (the number of newly infected individuals skyrocketed after this time point)
  - time 5 (peak)
  - time 7 (after a sudden drop in the number of newly infected individuals, the number of new infections is expected to remain stable).

This approach utilizes the available data up to these time points to estimate the posterior distribution of parameters and subsequently generate predictions for future outcomes.

```
# Posterior predictive forecasting
# Posterior prediction starting from time point 5
pred.SIR.net.point.3 <- pred.epi(epidat2, xx = mcmcout_partd, tmin = 3,
    Sformula = ~data$V6, Tformula = NULL,
    burnin = 2000, criterion = "newly infectious",
    n.samples = 500)
```

```
##   generate 500 epidemics
```

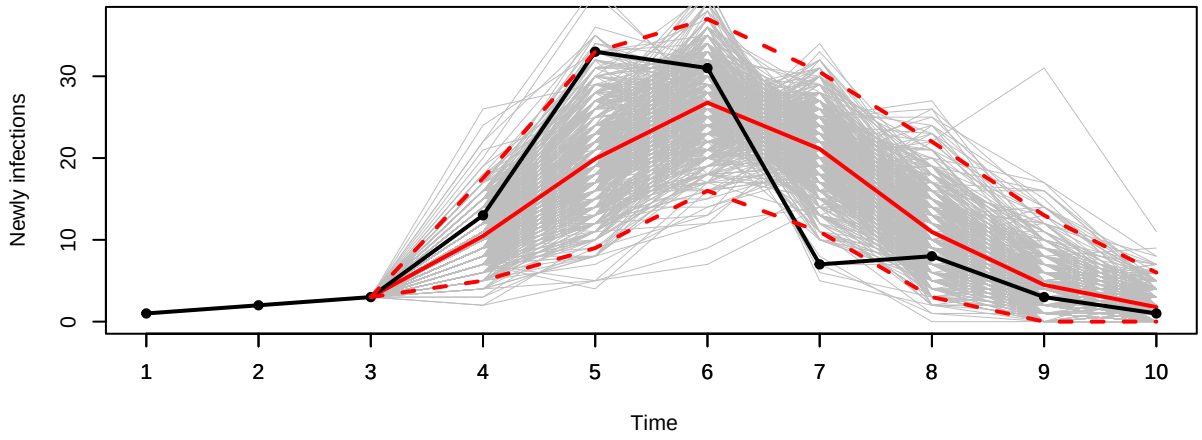
```
pred.SIR.net.point.5 <- pred.epi(epidat2, xx = mcmcout_partd, tmin = 5,
    Sformula = ~data$V6, Tformula = NULL,
    burnin = 2000, criterion = "newly infectious",
    n.samples = 500)
```

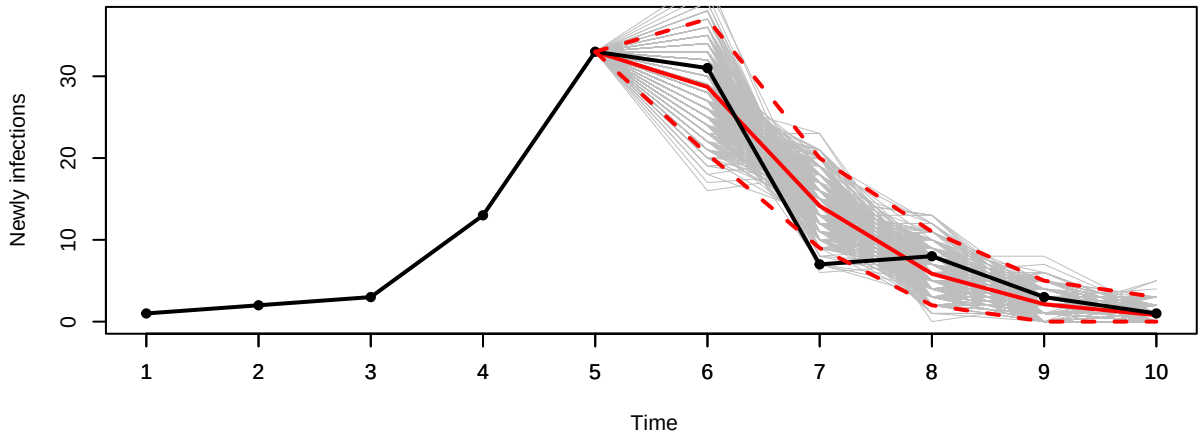
```
##   generate 500 epidemics
```

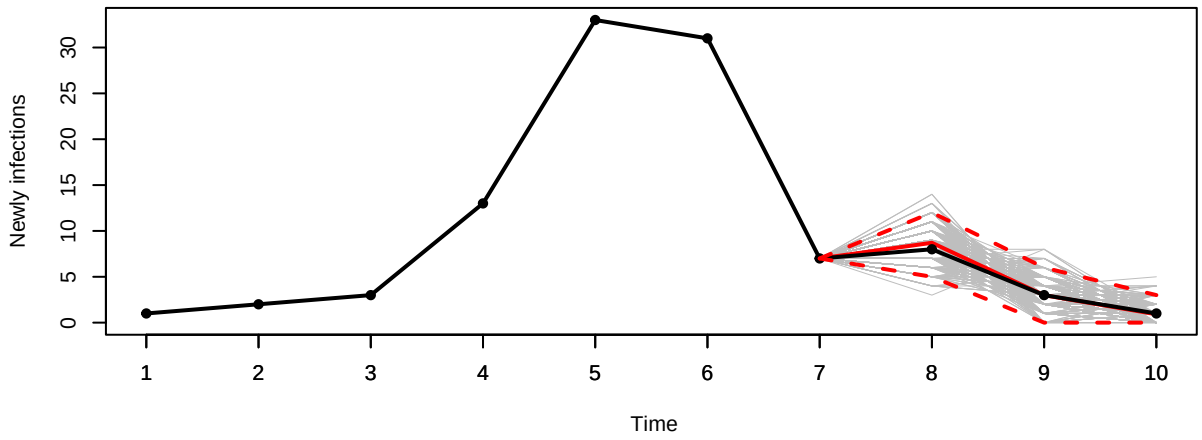
```
pred.SIR.net.point.7 <- pred.epi(epidat2, xx = mcmcout_partd, tmin = 7,
    Sformula = ~data$V6, Tformula = NULL,
    burnin = 2000, criterion = "newly infectious",
    n.samples = 500)
```

```
##   generate 500 epidemics
```

```r
# plot predictions:
par(mfrow = c(3,1))
plot(pred.SIR.net.point.3, col = "red", lwd = 2, pch = 19)
plot(pred.SIR.net.point.5, col = "red", lwd = 2, pch = 19)
plot(pred.SIR.net.point.7, col = "red", lwd = 2, pch = 19)
```

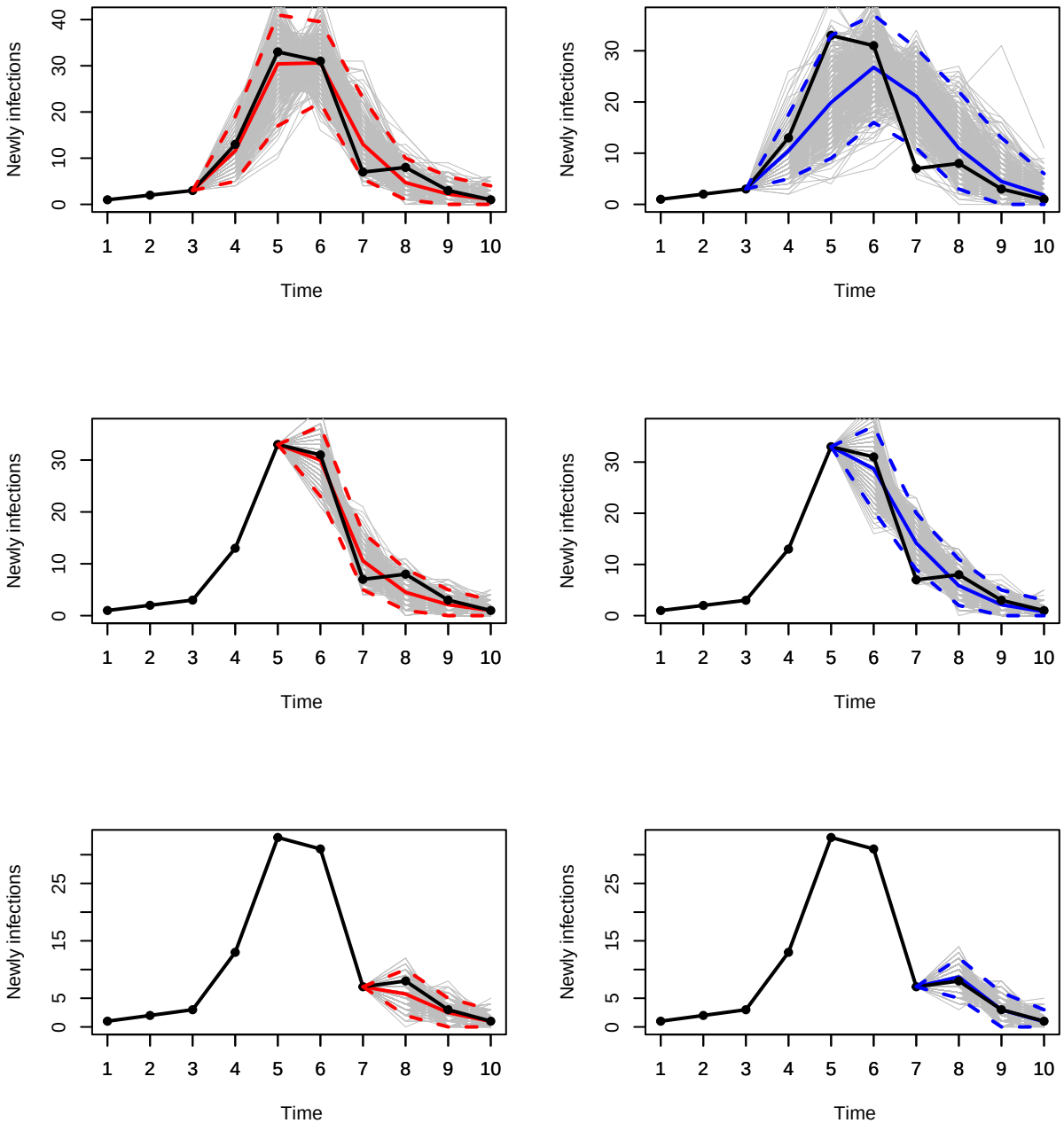






Red is spatial model from part c and Blue is network model from part d

```r
# plot predictions:
par(mfrow = c(3,2))
plot(pred.SIR.spatial.point.3, col = "red", lwd = 2, pch = 19)
plot(pred.SIR.net.point.3, col = "blue", lwd = 2, pch = 19)
plot(pred.SIR.spatial.point.5, col = "red", lwd = 2, pch = 19)
plot(pred.SIR.net.point.5, col = "blue", lwd = 2, pch = 19)
plot(pred.SIR.spatial.point.7, col = "red", lwd = 2, pch = 19)
plot(pred.SIR.net.point.7, col = "blue", lwd = 2, pch = 19)
```



The DIC value obtained in part c) is around 441, while the DIC value obtained in part d) is around 485. Therefore, I would say that the spatial model in part c) performs better than the network model in part d). We can actually draw the same conclusion from the six plots illustrated above. The first column of three plots shows the simulation based on a spatial model, and the second column shows the simulation based on a network model. In my opinion, the network model has greater variance than the spatial model, and the credible interval generated from a network model may not contain all the real data values. Based on the DIC and the visualization of the posterior predictive distribution, I prefer the spatial model over the network model.

part e

A certain proportion (e.g., 25%, 50%, 75%) of infectious periods are set equal to zero, so those individuals will enter the removed state as soon as they are infected.

- I code this culling simulation into three parts:
  - Remove individuals who are in the removed state before time 4 because they are not considered to be susceptible individuals
    - inftime + infperiod < 4
  - Randomly sample/select 25% or 50% or 75% individuals and set their infperiod values to 0
    - V4-V3 $= 0$
  - use as.epidata() function to create epidemic

```
# Posterior predictive forecasting
# Posterior prediction starting from time point 4
pred.SIR <- pred.epi(epidat, xx = mcmcout_v6, tmin = 4,
                     Sformula = ~data$V6, Tformula = NULL, burnin = 2000,
                     criterion = "newly infectious", n.samples = 500)
```

```
## generate 500 epidemics
```

```
# Compute the mean value of the simulated newly infectious at time 5,6,..,10
# create a data frame with the means and column names
means <- c(round(mean(pred.SIR$crit.sim[,5])),
           round(mean(pred.SIR$crit.sim[,6])),
           round(mean(pred.SIR$crit.sim[,7])),
           round(mean(pred.SIR$crit.sim[,8])),
           round(mean(pred.SIR$crit.sim[,9])),
           round(mean(pred.SIR$crit.sim[,10])))
table_data <- data.frame(Time = c("time 5", "time 6", "time 7", "time 8", "time 9", "time 10"),
                         Mean = means)
# print the table using knitr::kable()
knitr::kable(table_data, row.names = FALSE,
             caption = "Mean value of the simulated newly infectious")
```

Table 2: Mean value of the simulated newly infectious

Time	Mean
time 5	34
time 6	28
time 7	11
time 8	5
time 9	2
time 10	1

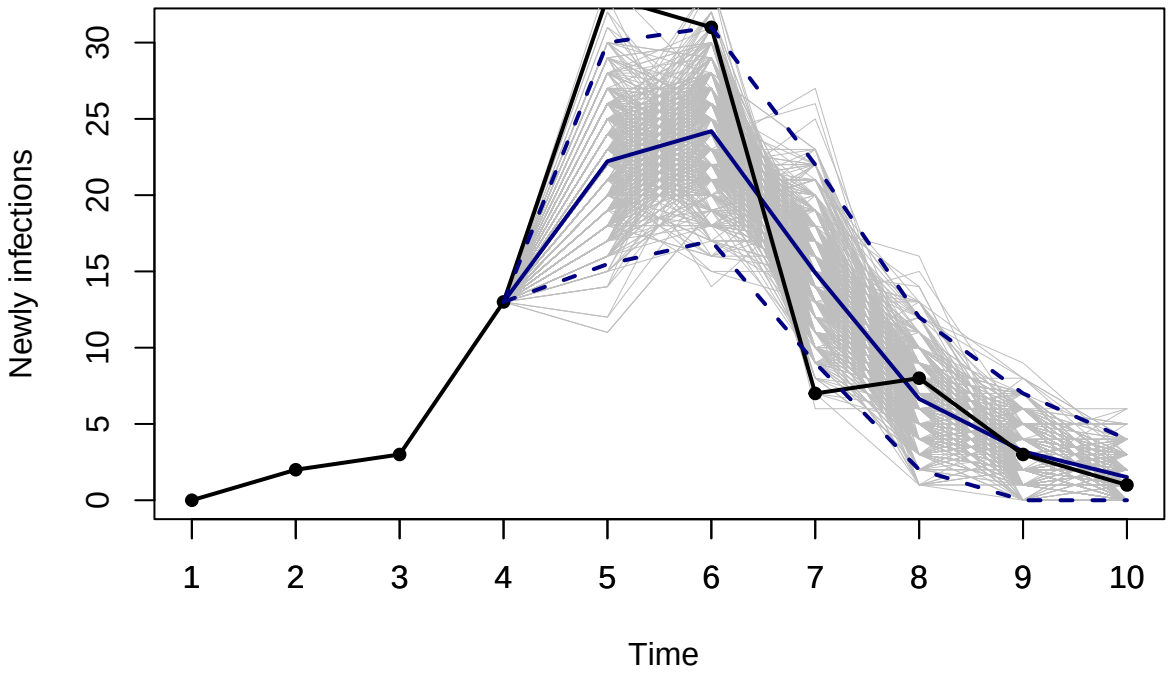
```
epidemic_simulation <- function(quarantine_level) {

  # create newdata dataset with removal time >= 4
  data = read.csv("A3EpidemicSheep.csv")
```

```r
newdata <- data[data$V4 >= 4, ]

# randomly select individuals for quarantine
if (quarantine_level == 1) {
  quarantine_prop <- 0.25
} else if (quarantine_level == 2) {
  quarantine_prop <- 0.5
} else if (quarantine_level == 3) {
  quarantine_prop <- 0.75
} else {
  stop("Invalid quarantine level. Please enter 1, 2, or 3.")
}

set.seed(123) # for reproducibility

# create subgroups
subgroup1 <- newdata[newdata$V3 < 4, ]
subgroup2 <- newdata[newdata$V3 >= 4, ]
print(subgroup1)

# set quarantine_idx
n_quarantine <- round(quarantine_prop * nrow(newdata))

# set infperiod based on subgroup
subgroup1$infperiod <- c(2,2,1,1,1)

# Calculate the number of individuals to set to 0
n_zero_infperiod <- round(quarantine_prop * nrow(newdata))-2

# Randomly select n_zero_infperiod individuals
zero_infperiod_index <- sample(1:nrow(subgroup2), n_zero_infperiod, replace = FALSE)

# Set their infperiod values to 0
subgroup2$infperiod <- subgroup2$V4-subgroup2$V3
subgroup2$infperiod[zero_infperiod_index] <- 0

# combine subgroups
newdata <- rbind(subgroup1, subgroup2)
head(newdata,4)

# create epidat object
epidat <- as.epidata(type = "SIR", n = nrow(newdata), x = newdata$V1, y = newdata$V2,
                     inftime = newdata$V3, infperiod = newdata$infperiod)

# run MCMC simulation (same as what we did before)
mcmcout <- epimcmc(epidat, tmax = max(epidat$inftime), niter = 20000, sus.par.ini = c(0.01, 0.05),
                   beta.ini = 0.01, adapt = TRUE, Sformula = ~data$V6, pro.sus.var = c(0.1, 0.1),
                   pro.beta.var = 0.1, prior.sus.dist = c("uniform", "uniform"),
                   prior.sus.par = matrix(c(0, 0, 10000, 10000), ncol = 2, nrow = 2),
                   prior.beta.dist = "uniform", prior.beta.par = c(0, 10000), acc.rate = 0.5)

# posterior predictive forecasting
pred.SIR <- pred.epi(epidat, xx = mcmcout, tmin = 4, Sformula = ~data$V6, Tformula = NULL,
```

```
                        burnin = 2000, criterion = "newly infectious", n.samples = 500)

  plot(pred.SIR, col="navy", lwd=2, pch=19)

  # compute mean values of simulated newly infectious at time 5-10
  time_points <- 5:10
  mean_vals <- sapply(time_points, function(t) round(mean(pred.SIR$crit.sim[,t])))

  # print results
  cat("Mean values of newly infectious individuals at time points 5-10:\n")
  cat(paste("Time point", time_points, ": ", mean_vals, "\n"))
}
```

```
epidemic_simulation(1)
```

```
##     X    V1    V2 V3 V4 V5 V6         V7
## 35 35 116.2 121.8  2  4  0  0 0.36229828
## 38 38  99.4  88.8  2  4  1  1 0.27667649
## 46 46  92.9  96.8  3  5  0  1 0.42609810
## 49 49  98.9  89.2  3  5  1  1 0.07943969
## 61 61 124.0  85.8  3  4  1  0 0.75088219
##   generate 20000 samples
## generate 500 epidemics
```

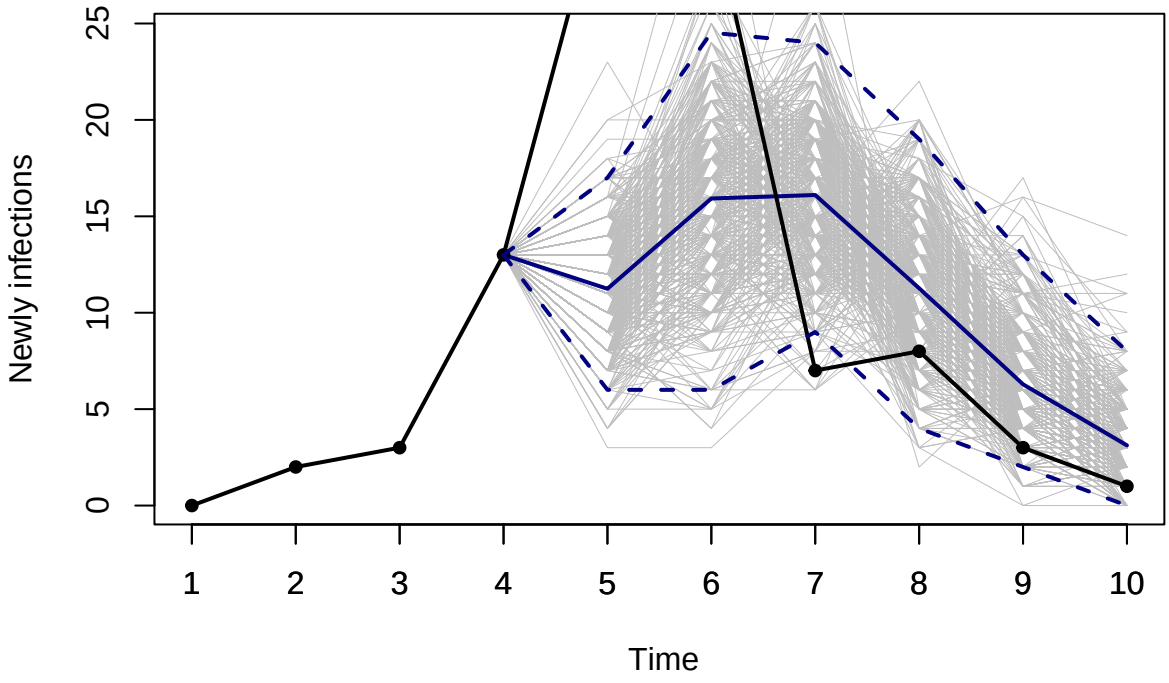



```
## Mean values of newly infectious individuals at time points 5-10:
```

```
## Time point 5 :  22
##  Time point 6 :  24
##  Time point 7 :  15
##  Time point 8 :  7
##  Time point 9 :  3
##  Time point 10 :  2
```

```
epidemic_simulation(2)
```

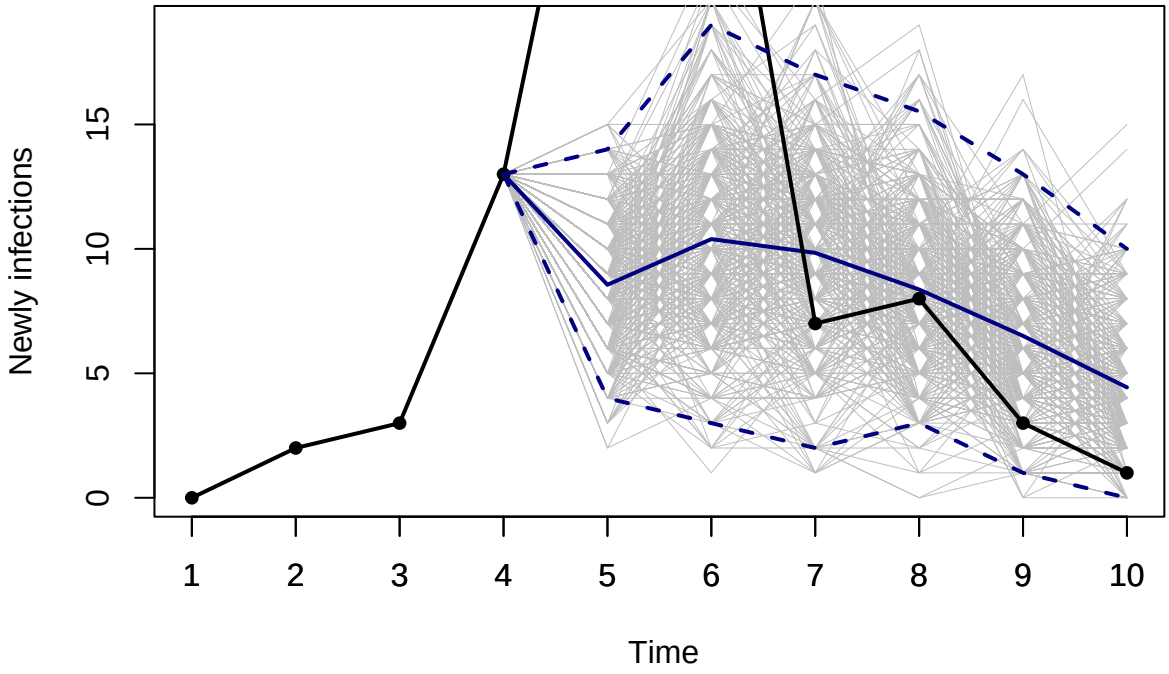
```
##     X    V1    V2 V3 V4 V5 V6         V7
## 35 35 116.2 121.8  2  4  0  0 0.36229828
## 38 38  99.4  88.8  2  4  1  1 0.27667649
## 46 46  92.9  96.8  3  5  0  1 0.42609810
## 49 49  98.9  89.2  3  5  1  1 0.07943969
## 61 61 124.0  85.8  3  4  1  0 0.75088219
##   generate 20000 samples
## generate 500 epidemics
```




```
## Mean values of newly infectious individuals at time points 5-10:
## Time point 5 :  11
##  Time point 6 :  16
##  Time point 7 :  16
##  Time point 8 :  11
##  Time point 9 :  6
##  Time point 10 :  3
```

```
epidemic_simulation(3)
```

```
##      X    V1    V2 V3 V4 V5 V6         V7
## 35 35 116.2 121.8  2  4  0  0 0.36229828
## 38 38  99.4  88.8  2  4  1  1 0.27667649
## 46 46  92.9  96.8  3  5  0  1 0.42609810
## 49 49  98.9  89.2  3  5  1  1 0.07943969
## 61 61 124.0  85.8  3  4  1  0 0.75088219
##   generate 20000 samples
## generate 500 epidemics
```




```
## Mean values of newly infectious individuals at time points 5-10:
## Time point 5 :  9
##  Time point 6 :  10
##  Time point 7 :  10
##  Time point 8 :  8
##  Time point 9 :  7
##  Time point 10 :  4
```

I modify the function epidemic_simulation() to take one additional arguments: **alert_distance**

alert_distance: if the Euclidean distance between two farms is less than or equal to the alert distance, quarantine measures will be applied. If the Euclidean distance is greater than the alert distance, quarantine measures will not be applied. Note that the Euclidean distance is based on the XY-coordinates of the farms. There are four levels defined: *none*, *very close*, *close*, *far* and *very far*

For each combination of quarantine level and alert distance, we can create subgroups, we can run the simulation and calculate the total number of farms that were quarantined and infected. It's important to note that if a farm is beyond the alert distance, it will not be selected for any level of quarantine measure. For instance, if we set the quarantine level to 1 (which means 25%) and the alert distance to "close", only 25% of individuals located "close" to the infectious farm before time 4 (i.e., at a distance less than 17.51) will be randomly selected for quarantine. Farms located beyond 17.51 will not be selected for quarantine."

```
# information about the distance
data3 = data.frame(x=data$V1, y=data$V2)
summary(dist(data3))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3606 10.5665 17.5111 22.7741 33.1931 76.3551
```

```
# alert_distance
# "none": randomly select individuals for quarantine, distance is not specified
# "very close": distance is less than 10.57 (1st quartile)
# "close": distance is less than 17.51 (median)
# "far": distance is less than 33.19 (3rd quartile)
```

```
epidemic_simulation <- function(quarantine_level, alert_distance) {

  # create newdata dataset with removal time >= 4
  data = read.csv("A3EpidemicSheep.csv")
  newdata <- data[data$V4 >= 4, ]

  # randomly select individuals for quarantine
  if (quarantine_level == 1) {
    quarantine_prop <- 0.25
  } else if (quarantine_level == 2) {
    quarantine_prop <- 0.5
  } else if (quarantine_level == 3) {
    quarantine_prop <- 0.75
  } else {
    stop("Invalid quarantine level. Please enter 1, 2, or 3.")
  }

  set.seed(123) # for reproducibility

  # create subgroups
  # subgroup1 is individuals whose infectious time begins before the culling time (time 4)
  # subgroup2 is indivuduals whose infectious time begins after the culling time (time 4)
  subgroup1 <- newdata[newdata$V3 < 4, ]
  subgroup2 <- newdata[newdata$V3 >= 4, ]

  # create new column "very_close" in subgroup1
  # set the value to 0 because those 5 individuals in subgroup1 are going for quarantine as default
```

```r
subgroup1$very_close <- 0
subgroup1$close <- 0
subgroup1$far <- 0

# Create new columns "very_close", "close", and "far" in subgroup2.
# For example, if one farm is "very close" to one of the infectious individuals in subgroup1,
# then its "very_close" column will be labeled as 1; otherwise, it will be labeled as 0. However,
# it's important to note that being labeled as "very_close" doesn't necessarily mean that this
# farm will be chosen for quarantine, since quarantine measures only apply to those who are selected.

subgroup2$very_close <- apply(subgroup2[,1:2], 1, function(x) {
  min_dist <- min(sqrt(rowSums((subgroup1[,1:2] - x)^2)))
  if (min_dist < 10.57) {
    return(1)
  } else {
    return(0)
  }
})

subgroup2$close <- apply(subgroup2[,1:2], 1, function(x) {
  min_dist <- min(sqrt(rowSums((subgroup1[,1:2] - x)^2)))
  if (min_dist < 17.51) {
    return(1)
  } else {
    return(0)
  }
})

subgroup2$far <- apply(subgroup2[,1:2], 1, function(x) {
  min_dist <- min(sqrt(rowSums((subgroup1[,1:2] - x)^2)))
  if (min_dist < 33.19) {
    return(1)
  } else {
    return(0)
  }
})

# none means that every individual is possible to be selected for quarantine
subgroup1$none <- 1
subgroup2$none <- 1

# count number of each alert_distance
n_very_close <- sum(subgroup2$very_close==1)
n_close <- sum(subgroup2$close==1)
n_far <- sum(subgroup2$far==1)
n_none <- sum(subgroup2$none==1)

############################################################
######################### none #############################
if (alert_distance == "none"){
# set quarantine_idx
n_quarantine <- round(quarantine_prop * nrow(newdata))
```

```r
# set infperiod based on subgroup
subgroup1$infperiod <- c(2,2,1,1,1)

# Calculate the number of individuals to set to 0
# minus 2 because record number 46 and 47 in subgroup1 have V4>4, and they are, by default,
# selected for quarantine, so the original remtime will change from 5 to 4
n_zero_infperiod <- round(quarantine_prop * nrow(newdata))-2

# Randomly select n_zero_infperiod individuals
zero_infperiod_index <- sample(1:nrow(subgroup2), n_zero_infperiod, replace = FALSE)

# Set their infperiod values to 0
subgroup2$infperiod <- subgroup2$V4-subgroup2$V3
subgroup2$infperiod[zero_infperiod_index] <- 0

# combine subgroups
newdata <- rbind(subgroup1, subgroup2)
head(newdata)

# create epidat object
epidat <- as.epidata(type = "SIR", n = nrow(newdata), x = newdata$V1, y = newdata$V2,
                     inftime = newdata$V3, infperiod = newdata$infperiod)

# run MCMC simulation (same as what we did before)
mcmcout <- epimcmc(epidat, tmax = max(epidat$inftime), niter = 20000, sus.par.ini = c(0.01, 0.05),
                   beta.ini = 0.01, adapt = TRUE, Sformula = ~data$V6, pro.sus.var = c(0.1, 0.1),
                   pro.beta.var = 0.1, prior.sus.dist = c("uniform", "uniform"),
                   prior.sus.par = matrix(c(0, 0, 10000, 10000), ncol = 2, nrow = 2),
                   prior.beta.dist = "uniform", prior.beta.par = c(0, 10000), acc.rate = 0.5)

# posterior predictive forecasting
pred.SIR <- pred.epi(epidat, xx = mcmcout, tmin = 4, Sformula = ~data$V6, Tformula = NULL,
                     burnin = 2000, criterion = "newly infectious", n.samples = 500)

plot(pred.SIR, col="red", lwd=2, pch=19)

# compute mean values of simulated newly infectious at time 5-10
time_points <- 5:10
mean_vals <- sapply(time_points, function(t) round(mean(pred.SIR$crit.sim[,t])))

# print results
cat("Number of quarantine farms:", n_quarantine)
cat("Mean values of newly infectious individuals at time points 5-10:\n")
cat(paste("Time point", time_points, ": ", mean_vals, "\n"))
}
##########################################################
########################## very close ####################
if (alert_distance == "very close"){
# set quarantine_idx
# minus 2 because record number 46 and 47 in subgroup1 have V4>4, and they are, by default,
# selected for quarantine, so the original remtime will change from 5 to 4
n_quarantine <- round(quarantine_prop * n_very_close)-2
```

```r
# set infperiod based on subgroup
subgroup1$infperiod <- c(2,2,1,1,1)

# Randomly select n_zero_infperiod individuals
zero_infperiod_index <- sample(1:n_very_close, n_quarantine, replace = FALSE)

# Set their infperiod values to 0
subgroup2$infperiod <- subgroup2$V4 - subgroup2$V3

# find records with very_close = 1 in subgroup2
very_close_records <- subgroup2[subgroup2$very_close == 1, ]

# set infperiod to zero for the very_close records
very_close_records[zero_infperiod_index, "infperiod"] <- 0

# remove the rows in very_close_records with infperiod = 0 from subgroup2
subgroup2 <- setdiff(subgroup2, very_close_records[zero_infperiod_index, ])

#print(subgroup1)
#print(subgroup2)
#print(very_close_records[zero_infperiod_index,])

# combine subgroups
newdata <- rbind(subgroup1, subgroup2, very_close_records[zero_infperiod_index,])
head(newdata)

# create epidat object
epidat <- as.epidata(type = "SIR", n = nrow(newdata), x = newdata$V1, y = newdata$V2,
                     inftime = newdata$V3, infperiod = newdata$infperiod)

# run MCMC simulation (same as what we did before)
mcmcout <- epimcmc(epidat, tmax = max(epidat$inftime), niter = 20000, sus.par.ini = c(0.01, 0.05),
                   beta.ini = 0.01, adapt = TRUE, Sformula = ~data$V6, pro.sus.var = c(0.1, 0.1),
                   pro.beta.var = 0.1, prior.sus.dist = c("uniform", "uniform"),
                   prior.sus.par = matrix(c(0, 0, 10000, 10000), ncol = 2, nrow = 2),
                   prior.beta.dist = "uniform", prior.beta.par = c(0, 10000), acc.rate = 0.5)

# posterior predictive forecasting
pred.SIR <- pred.epi(epidat, xx = mcmcout, tmin = 4, Sformula = ~data$V6, Tformula = NULL,
                     burnin = 2000, criterion = "newly infectious", n.samples = 500)

plot(pred.SIR, col="red", lwd=2, pch=19)

# compute mean values of simulated newly infectious at time 5-10
time_points <- 5:10
mean_vals <- sapply(time_points, function(t) round(mean(pred.SIR$crit.sim[,t])))

# print results
cat("Number of quarantine farms:", n_quarantine)
cat("Mean values of newly infectious individuals at time points 5-10:\n")
cat(paste("Time point", time_points, ": ", mean_vals, "\n"))
}
##########################################################
```

```r
######################### close ###################
if (alert_distance == "close"){
# set quarantine_idx
# minus 2 because record number 46 and 47 in subgroup1 have V4>4, and they are, by default,
# selected for quarantine, so the original remtime will change from 5 to 4
n_quarantine <- round(quarantine_prop * n_close)-2

# set infperiod based on subgroup
subgroup1$infperiod <- c(2,2,1,1,1)

# Randomly select n_zero_infperiod individuals
zero_infperiod_index <- sample(1:n_close, n_quarantine, replace = FALSE)

# Set their infperiod values to 0
subgroup2$infperiod <- subgroup2$V4 - subgroup2$V3

# find records with close = 1 in subgroup2
close_records <- subgroup2[subgroup2$close == 1, ]

# set infperiod to zero for the close records
close_records[zero_infperiod_index, "infperiod"] <- 0

# remove the rows in close_records with infperiod = 0 from subgroup2
subgroup2 <- setdiff(subgroup2, close_records[zero_infperiod_index, ])

# combine subgroups
newdata <- rbind(subgroup1, subgroup2, close_records[zero_infperiod_index,])
head(newdata)

# create epidat object
epidat <- as.epidata(type = "SIR", n = nrow(newdata), x = newdata$V1, y = newdata$V2,
                     inftime = newdata$V3, infperiod = newdata$infperiod)

# run MCMC simulation (same as what we did before)
mcmcout <- epimcmc(epidat, tmax = max(epidat$inftime), niter = 20000, sus.par.ini = c(0.01, 0.05),
                   beta.ini = 0.01, adapt = TRUE, Sformula = ~data$V6, pro.sus.var = c(0.1, 0.1),
                   pro.beta.var = 0.1, prior.sus.dist = c("uniform", "uniform"),
                   prior.sus.par = matrix(c(0, 0, 10000, 10000), ncol = 2, nrow = 2),
                   prior.beta.dist = "uniform", prior.beta.par = c(0, 10000), acc.rate = 0.5)

# posterior predictive forecasting
pred.SIR <- pred.epi(epidat, xx = mcmcout, tmin = 4, Sformula = ~data$V6, Tformula = NULL,
                     burnin = 2000, criterion = "newly infectious", n.samples = 500)

plot(pred.SIR, col="red", lwd=2, pch=19)

# compute mean values of simulated newly infectious at time 5-10
time_points <- 5:10
mean_vals <- sapply(time_points, function(t) round(mean(pred.SIR$crit.sim[,t])))

# print results
cat("Number of quarantine farms:", n_quarantine)
cat("Mean values of newly infectious individuals at time points 5-10:\n")
```

```r
cat(paste("Time point", time_points, ": ", mean_vals, "\n"))
}
############################################################
######################### far ##########################
if (alert_distance == "far"){
# set quarantine_idx
# minus 2 because record number 46 and 47 in subgroup1 have V4>4, and they are, by default,
# selected for quarantine, so the original remtime will change from 5 to 4
n_quarantine <- round(quarantine_prop * n_far)-2

# set infperiod based on subgroup
subgroup1$infperiod <- c(2,2,1,1,1)

# Randomly select n_zero_infperiod individuals
zero_infperiod_index <- sample(1:n_far, n_quarantine, replace = FALSE)

# Set their infperiod values to 0
subgroup2$infperiod <- subgroup2$V4 - subgroup2$V3

# find records with far = 1 in subgroup2
far_records <- subgroup2[subgroup2$far == 1, ]

# set infperiod to zero for the far records
far_records[zero_infperiod_index, "infperiod"] <- 0

# remove the rows in far_records with infperiod = 0 from subgroup2
subgroup2 <- setdiff(subgroup2, far_records[zero_infperiod_index, ])

# combine subgroups
newdata <- rbind(subgroup1, subgroup2, far_records[zero_infperiod_index,])
head(newdata)

# create epidat object
epidat <- as.epidata(type = "SIR", n = nrow(newdata), x = newdata$V1, y = newdata$V2,
                     inftime = newdata$V3, infperiod = newdata$infperiod)

# run MCMC simulation (same as what we did before)
mcmcout <- epimcmc(epidat, tmax = max(epidat$inftime), niter = 20000, sus.par.ini = c(0.01, 0.05),
                   beta.ini = 0.01, adapt = TRUE, Sformula = ~data$V6, pro.sus.var = c(0.1, 0.1),
                   pro.beta.var = 0.1, prior.sus.dist = c("uniform", "uniform"),
                   prior.sus.par = matrix(c(0, 0, 10000, 10000), ncol = 2, nrow = 2),
                   prior.beta.dist = "uniform", prior.beta.par = c(0, 10000), acc.rate = 0.5)

# posterior predictive forecasting
pred.SIR <- pred.epi(epidat, xx = mcmcout, tmin = 4, Sformula = ~data$V6, Tformula = NULL,
                     burnin = 2000, criterion = "newly infectious", n.samples = 500)

plot(pred.SIR, col="red", lwd=2, pch=19)

# compute mean values of simulated newly infectious at time 5-10
time_points <- 5:10
mean_vals <- sapply(time_points, function(t) round(mean(pred.SIR$crit.sim[,t])))
```

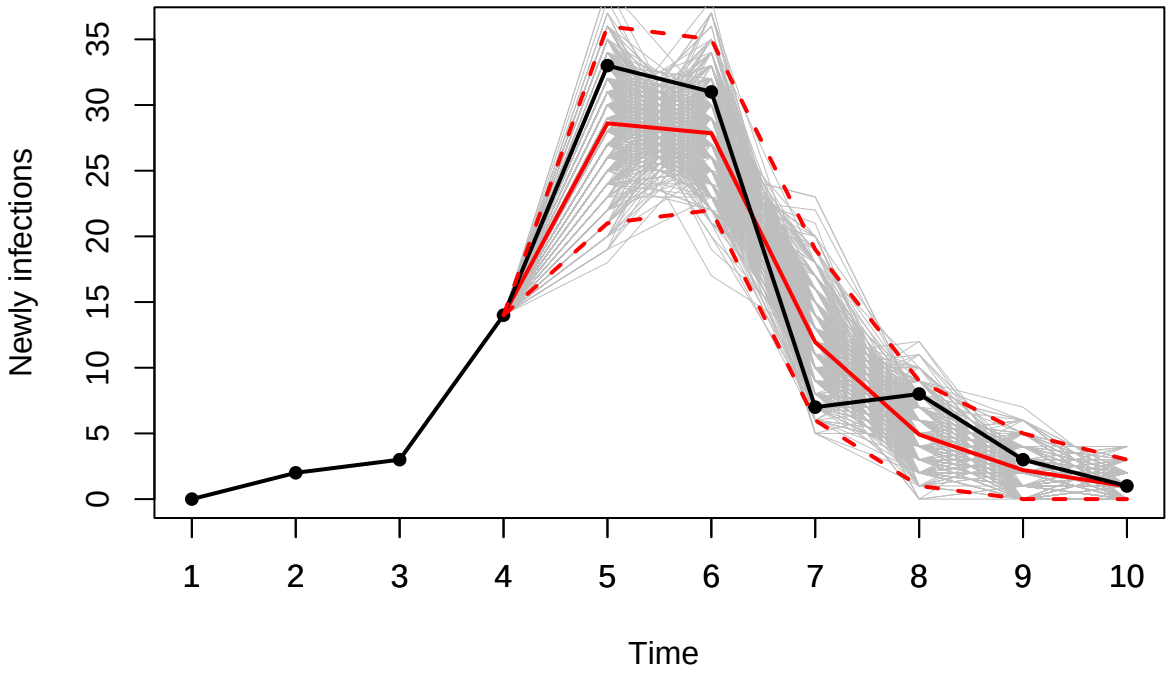
```
  # print results
  cat("Number of quarantine farms:", n_quarantine)
  cat("Mean values of newly infectious individuals at time points 5-10:\n")
  cat(paste("Time point", time_points, ": ", mean_vals, "\n"))
  }
}
```

Results and Simulations

To explore various scenarios with different levels of alert distance under a 25% quarantine measurement, we only consider the farms that is close to the infectious farm before time 4 (i.e., at a distance less than the alert distance) will be quarantined. Specifically, only 25% of such farms will be selected for quarantine

```
library(EpiILM)
epidemic_simulation(1, "very close")
```

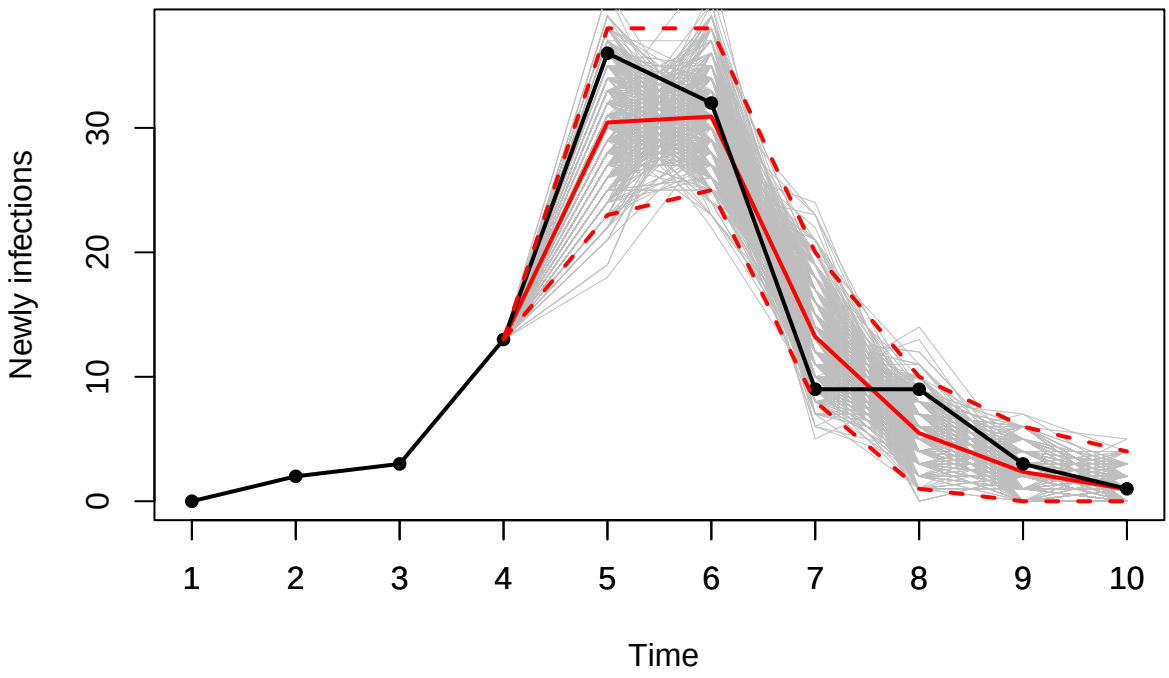
```
##   generate 20000 samples
## generate 500 epidemics
```




```
## Number of quarantine farms: 1Mean values of newly infectious individuals at time points 5-10:
## Time point 5 :  29
##  Time point 6 :  28
##  Time point 7 :  12
##  Time point 8 :  5
##  Time point 9 :  2
##  Time point 10 :  1
```

```
epidemic_simulation(1, "close")
```

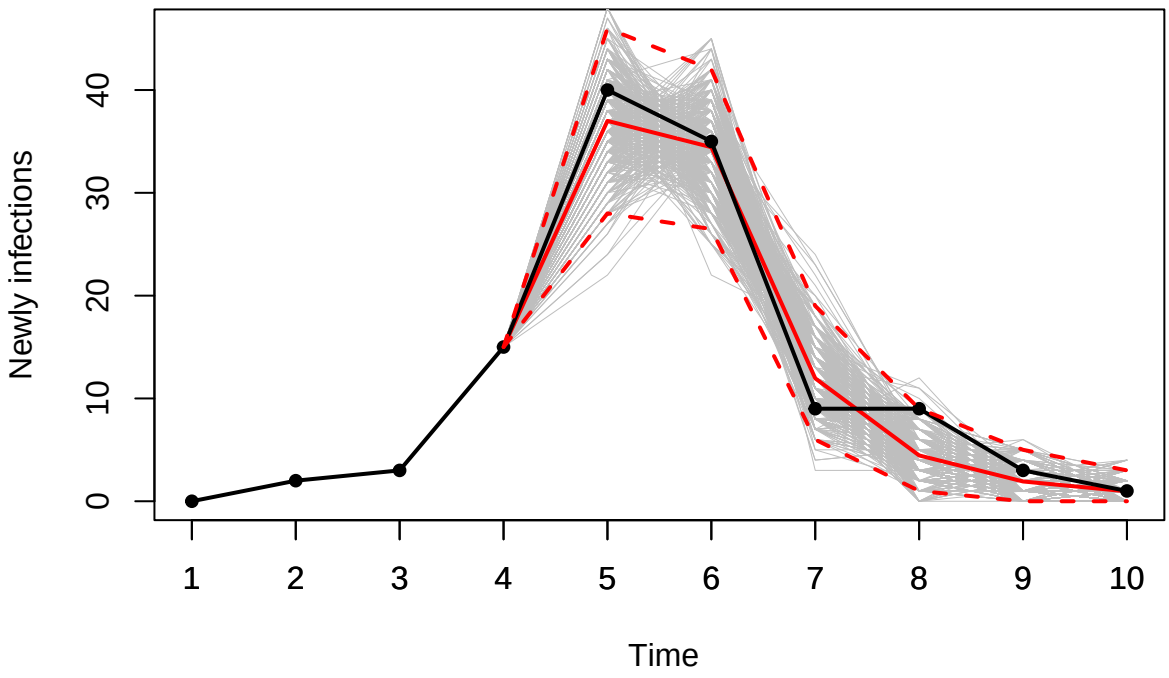
```
##   generate 20000 samples
## generate 500 epidemics
```




```
## Number of quarantine farms: 7Mean values of newly infectious individuals at time points 5-10:
## Time point 5 :  30
##  Time point 6 :  31
##  Time point 7 :  13
##  Time point 8 :  5
##  Time point 9 :  2
##  Time point 10 :  1
```

```
epidemic_simulation(1, "far")
```

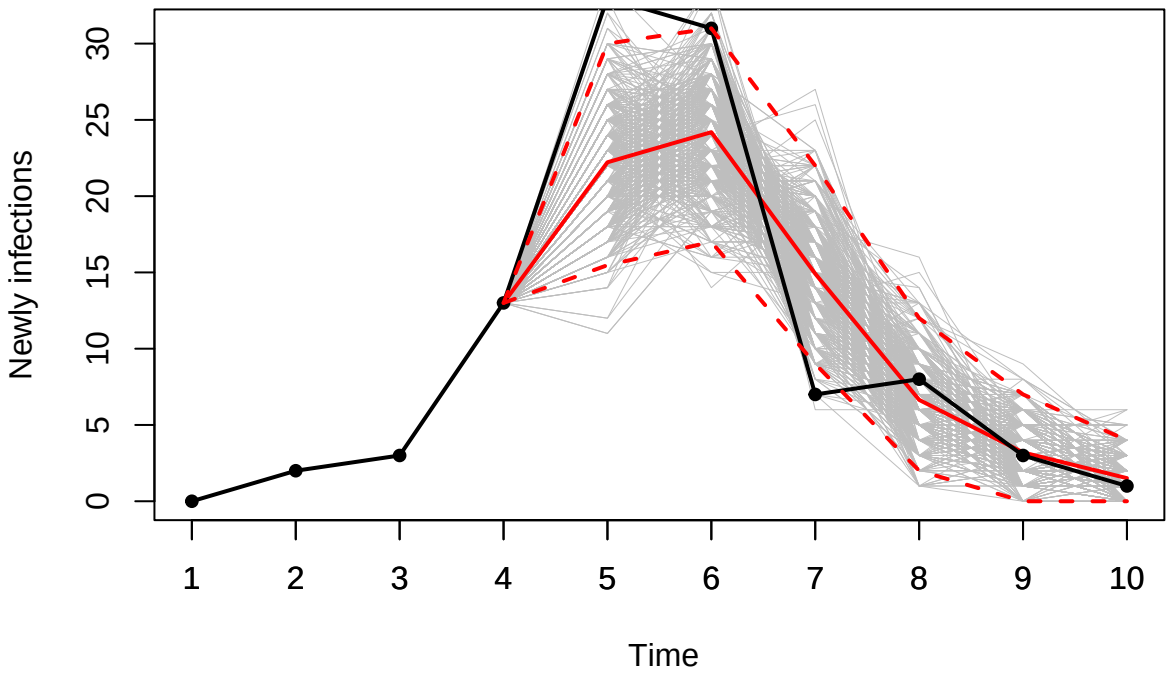
```
##   generate 20000 samples
## generate 500 epidemics
```




```
## Number of quarantine farms: 16Mean values of newly infectious individuals at time points 5-10:
## Time point 5 :  37
##  Time point 6 :  34
##  Time point 7 :  12
##  Time point 8 :  4
##  Time point 9 :  2
##  Time point 10 :  1
```

```
epidemic_simulation(1, "none")
```

```
##   generate 20000 samples
## generate 500 epidemics
```

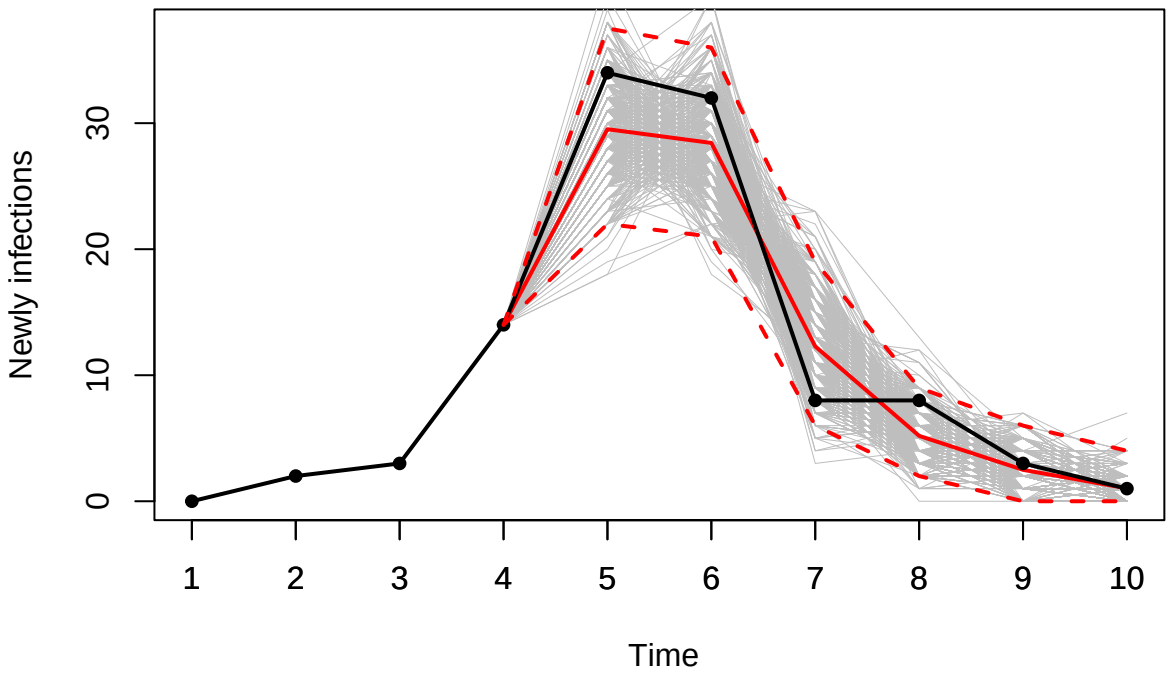



```
## Number of quarantine farms: 25Mean values of newly infectious individuals at time points 5-10:
## Time point 5 :  22
##  Time point 6 :  24
##  Time point 7 :  15
##  Time point 8 :  7
##  Time point 9 :  3
##  Time point 10 :  2
```

To explore various scenarios with different levels of alert distance under a 50% quarantine measurement, we only consider the farms that is close to the infectious farm before time 4 (i.e., at a distance less than the alert distance) will be quarantined. Specifically, only 50% of such farms will be selected for quarantine

```
epidemic_simulation(2, "very close")
```

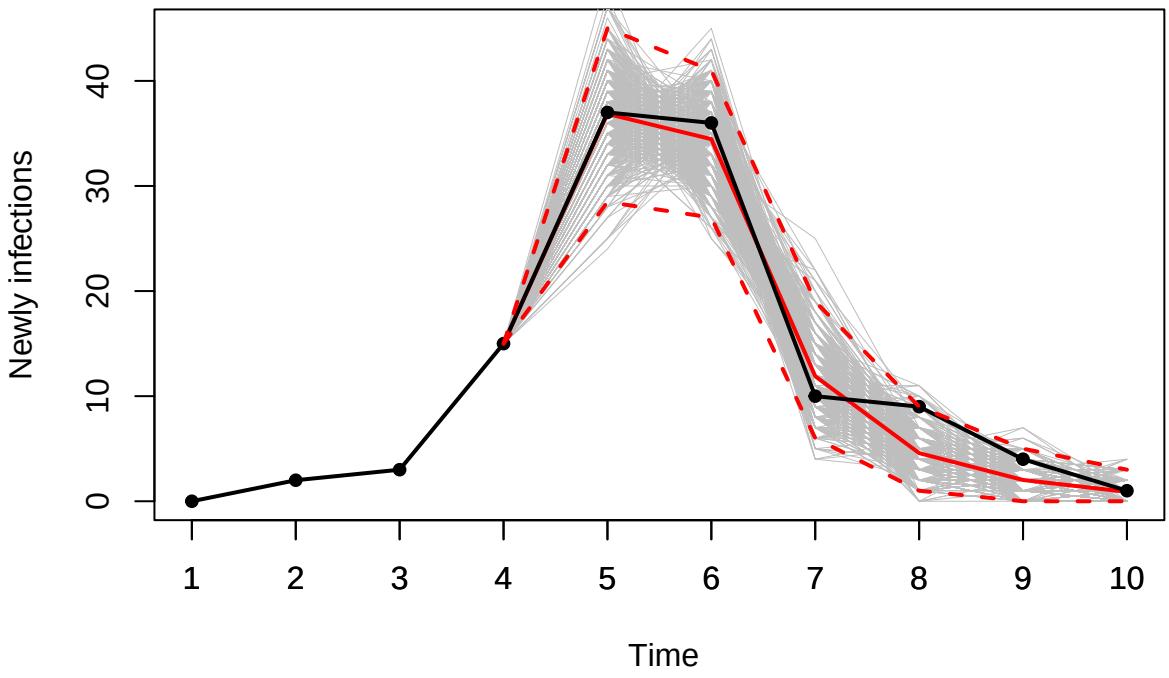
```
##   generate 20000 samples
## generate 500 epidemics
```




```
## Number of quarantine farms: 4Mean values of newly infectious individuals at time points 5-10:
## Time point 5 :  30
##  Time point 6 :  28
##  Time point 7 :  12
##  Time point 8 :  5
##  Time point 9 :  2
##  Time point 10 :  1
```

```
epidemic_simulation(2, "close")
```

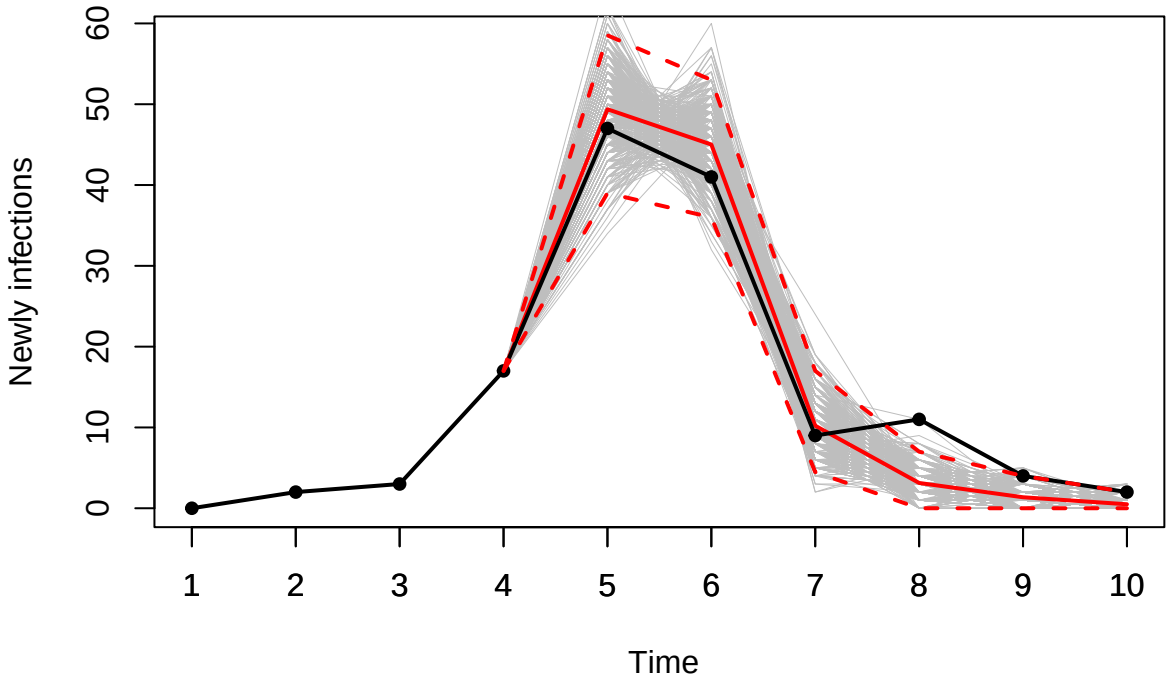
```
##   generate 20000 samples
## generate 500 epidemics
```




```
## Number of quarantine farms: 16Mean values of newly infectious individuals at time points 5-10:
## Time point 5 :  37
##  Time point 6 :  34
##  Time point 7 :  12
##  Time point 8 :  5
##  Time point 9 :  2
##  Time point 10 :  1
```

```
epidemic_simulation(2, "far")
```

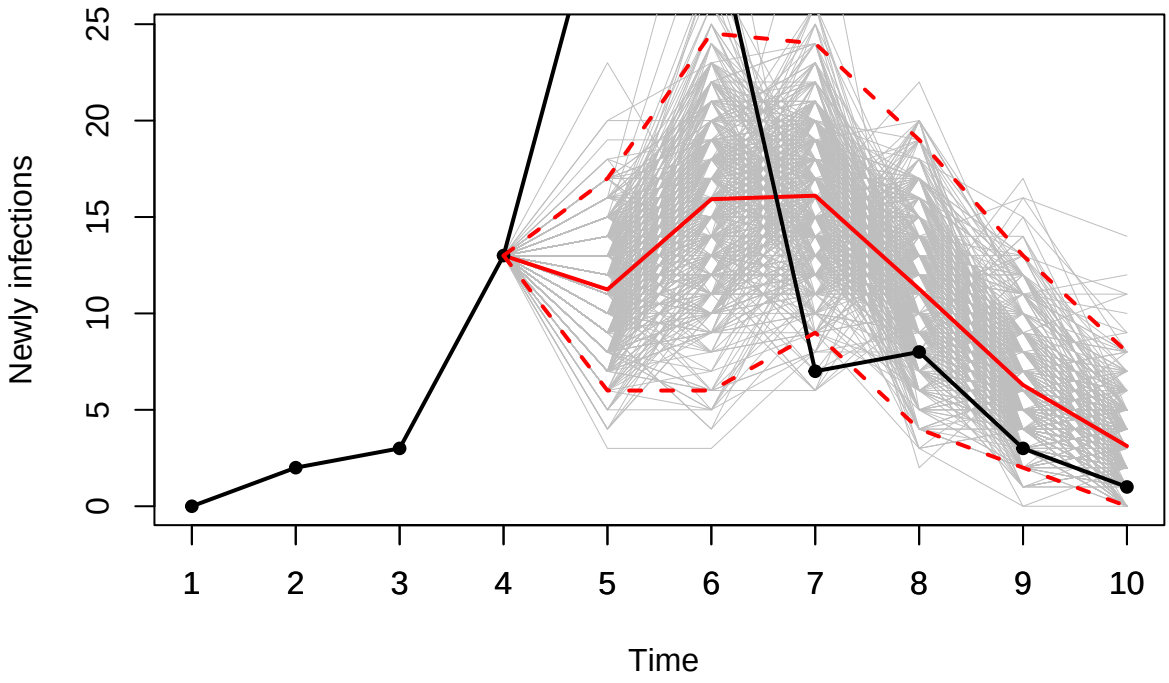
```
##   generate 20000 samples
## generate 500 epidemics
```




```
## Number of quarantine farms: 35Mean values of newly infectious individuals at time points 5-10:
## Time point 5 :  49
##  Time point 6 :  45
##  Time point 7 :  10
##  Time point 8 :  3
##  Time point 9 :  1
##  Time point 10 :  0
```

```
epidemic_simulation(2, "none")
```

```
##   generate 20000 samples
## generate 500 epidemics
```

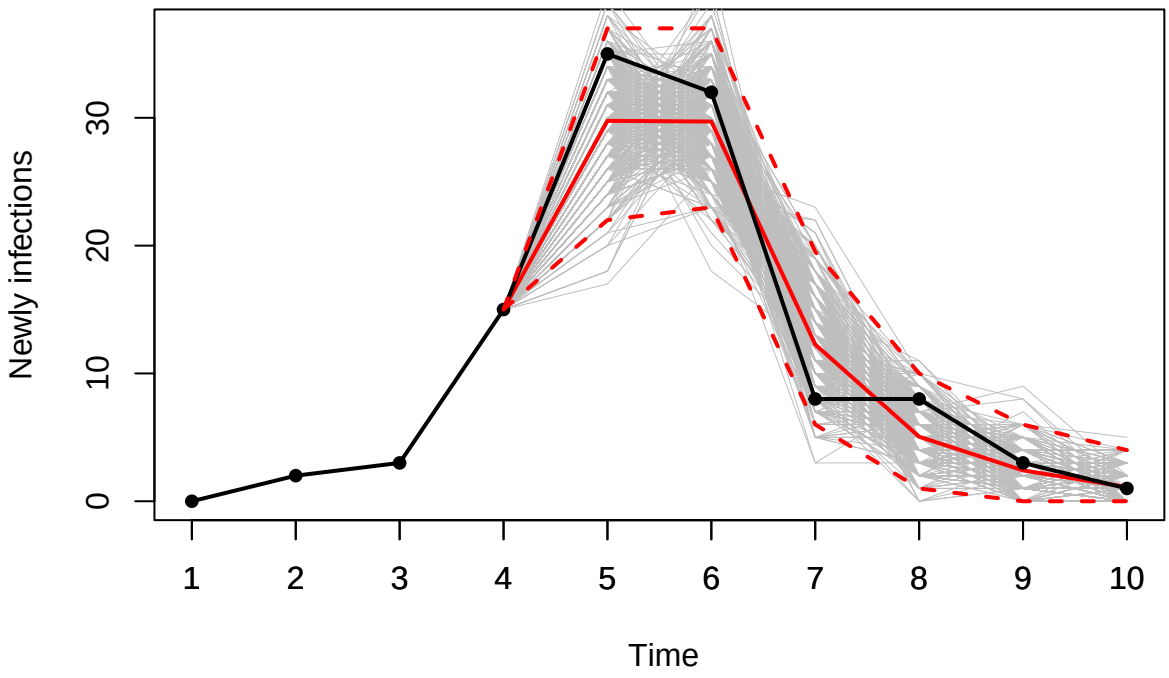



```
## Number of quarantine farms: 50Mean values of newly infectious individuals at time points 5-10:
## Time point 5 :  11
##  Time point 6 :  16
##  Time point 7 :  16
##  Time point 8 :  11
##  Time point 9 :  6
##  Time point 10 :  3
```

To explore various scenarios with different levels of alert distance under a 75% quarantine measurement, we only consider the farms that is close to the infectious farm before time 4 (i.e., at a distance less than the alert distance) will be quarantined. Specifically, only 75% of such farms will be selected for quarantine

```
epidemic_simulation(3, "very close")
```

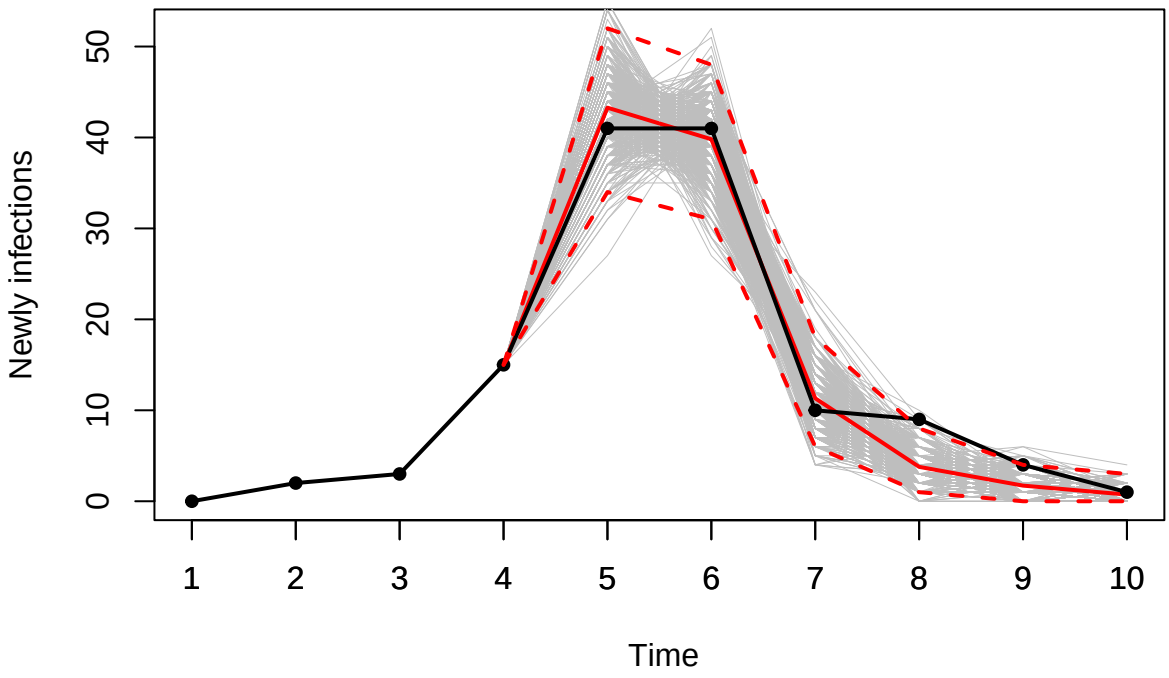
```
##   generate 20000 samples
## generate 500 epidemics
```




```
## Number of quarantine farms: 6Mean values of newly infectious individuals at time points 5-10:
## Time point 5 :  30
##  Time point 6 :  30
##  Time point 7 :  12
##  Time point 8 :  5
##  Time point 9 :  2
##  Time point 10 :  1
```

```
epidemic_simulation(3, "close")
```

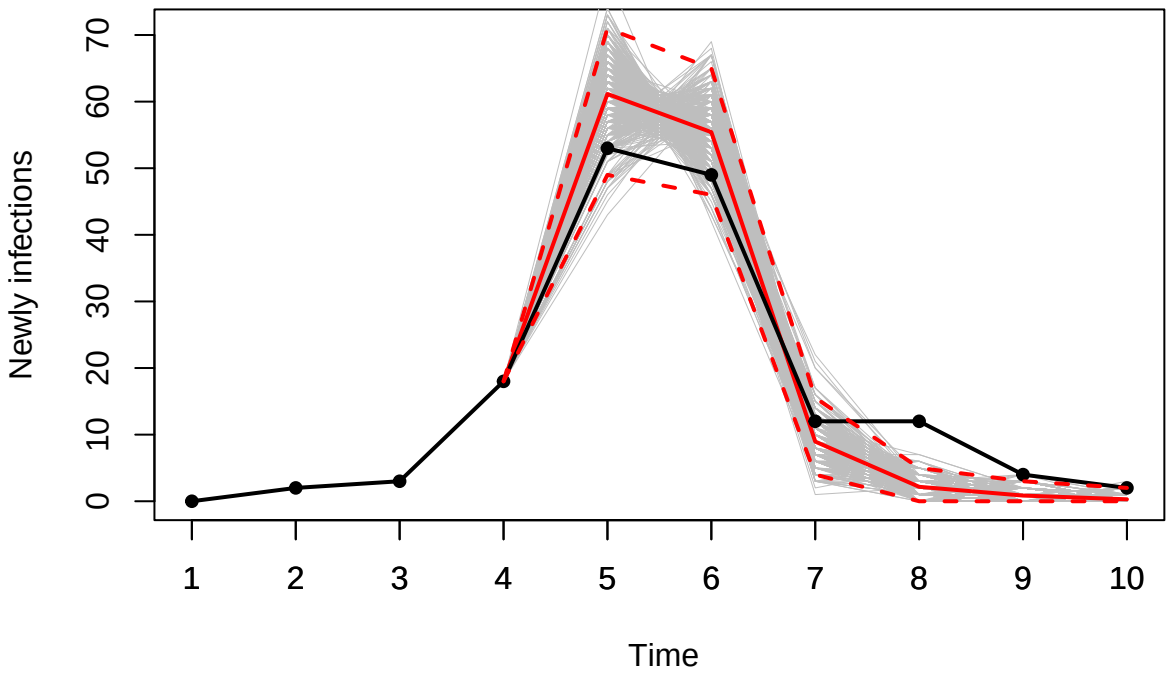
```
##   generate 20000 samples
## generate 500 epidemics
```




```
## Number of quarantine farms: 25Mean values of newly infectious individuals at time points 5-10:
## Time point 5 :  43
##  Time point 6 :  40
##  Time point 7 :  11
##  Time point 8 :  4
##  Time point 9 :  2
##  Time point 10 :  1
```

```
epidemic_simulation(3, "far")
```

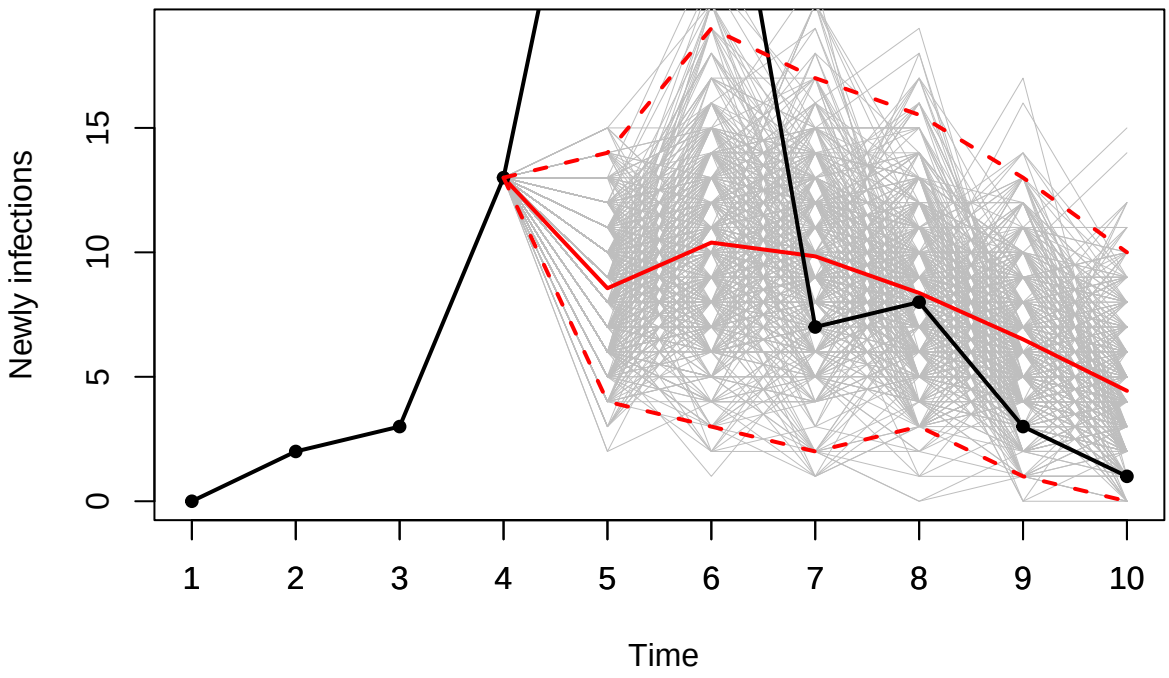
```
##   generate 20000 samples
## generate 500 epidemics
```




```
## Number of quarantine farms: 54Mean values of newly infectious individuals at time points 5-10:
## Time point 5 :  61
##  Time point 6 :  55
##  Time point 7 :  9
##  Time point 8 :  2
##  Time point 9 :  1
##  Time point 10 :  0
```

```
epidemic_simulation(3, "none")
```

```
##   generate 20000 samples
## generate 500 epidemics
```




```
## Number of quarantine farms: 76Mean values of newly infectious individuals at time points 5-10:
## Time point 5 :  9
##  Time point 6 :  10
##  Time point 7 :  10
##  Time point 8 :  8
##  Time point 9 :  7
##  Time point 10 :  4
```

Summary Table

```
alert = c("very close", "close", "far", "none")
levelone = c(3,9,18,25)
leveltwo = c(6,18,37,50)
levelthree = c(8,27,56,76)
quarantine <- data.frame(alert, levelone, leveltwo, levelthree)
knitr::kable(quarantine, "pipe", col.names = c("Alert Distance", "Quarantine Level: 25%",
                                               "Quarantine Level: 50%", "Quarantine Level: 75%"),
             align = c("c", "c", "c", "c"), caption = "Summary Table 1: Number of Quarantine Farms")
```

Table 3: Summary Table 1: Number of Quarantine Farms

Alert Distance	Quarantine Level: 25%	Quarantine Level: 50%	Quarantine Level: 75%
very close	3	6	8
close	9	18	27
far	18	37	56
none	25	50	76

```
alert = c("very close", "close", "far", "none")
levelone = c("29+28+12+5+2+1=77", "30+31+13+5+2+1=82",
             "37+34+12+4+2+1=90", "22+24+15+7+3+2=73")
leveltwo = c("30+28+12+5+2+1=78", "37+34+12+5+2+1=91",
             "49+45+10+3+1+0=108", "11+16+16+11+6+3=63")
levelthree = c("30+30+12+5+2+1=80", "43+40+11+4+2+1=101",
               "61+55+9+2+1+0=128", "9+10+10+8+7+4=48")
quarantine <- data.frame(alert, levelone, leveltwo, levelthree)
knitr::kable(quarantine, "pipe", col.names = c("Alert Distance", "Quarantine Level: 25%",
                                               "Quarantine Level: 50%", "Quarantine Level: 75%"),
             align = c("c", "c", "c", "c"), caption = "Summary Table 2: Number of infectious Farms")
```

Table 4: Summary Table 2: Number of infectious Farms

Alert Distance	Quarantine Level: 25%	Quarantine Level: 50%	Quarantine Level: 75%
very close	29+28+12+5+2+1=77	30+28+12+5+2+1=78	30+30+12+5+2+1=80
close	30+31+13+5+2+1=82	37+34+12+5+2+1=91	43+40+11+4+2+1=101
far	37+34+12+4+2+1=90	49+45+10+3+1+0=108	61+55+9+2+1+0=128
none	22+24+15+7+3+2=73	11+16+16+11+6+3=63	9+10+10+8+7+4=48

note: $x_5 + x_6 + ... + x_{10} = x$ actually represents the number of newly infectious farms at that time point

```
alert = c("very close", "close", "far", "none")
levelone = c(80, 91, 108, 98)
leveltwo = c(84, 109, 145, 113)
levelthree = c(88, 128, 184, 124)
quarantine <- data.frame(alert, levelone, leveltwo, levelthree)
knitr::kable(quarantine, "pipe", col.names = c("Alert Distance", "Quarantine Level: 25%",
                                               "Quarantine Level: 50%", "Quarantine Level: 75%"),
             align = c("c", "c", "c", "c"),
             caption = "Summary Table 3: Total Number of Culling (pre-emptive and result of infection)")
```

Table 5: Summary Table 3: Total Number of Culling (pre-emptive and result of infection)

Alert Distance	Quarantine Level: 25%	Quarantine Level: 50%	Quarantine Level: 75%
very close	80	84	88
close	91	109	128
far	108	145	184
none	98	113	124

Conclusion

Based on the findings presented in Summary Table One, we can observe that smaller alert distances and lower quarantine proportions are associated with fewer quarantined farms.

Summary Table Two highlights the surprising result that no alert distance measurement leads to a lower number of infectious farms, which becomes more significant as the quarantine proportion increases. This effect can be attributed to two reasons: first, random segregation can break the chain of infection irregularly and prevent rapid disease spread; second, scientific research on alert distances may lead to more effective disease control than using quartile values that we proposed in this study.

From Summary Table Three, we can observe that setting the quarantine proportion to 25% and the alert distance to "very close" results in the minimum number of culled farms, indicating that even small-scale quarantine measures at the onset of a virus outbreak can significantly reduce the number of affected farms.

However, implementing isolation measures is a complex issue that extends beyond the realm of science. The simulation and posterior prediction of the model may be impacted by factors that are not included in the model. When dealing with large populations and limited medical resources per capita, implementing isolation measures poses significant challenges.