Instruction and an Example of COVID-19 Data Analysis for US States and Counties

Here is an example of COVID-19 data analysis for US states in the manuscript. We demonstrate the process of model 1, model 2.3, and model 3. For additional results for model 2.1, 2.2 and 2.4, see "Covid-19-state.R" for details. For county-level results, see "Covid-19-county.R" for details.

Setting Up

Before implementing the algorithm, we need the following set up:

- Install required packages and load packages.
- Call functions from .R file and .cpp files
  - "functions_BFL.R"
  - "functions_BFL.cpp"

```
install.packages("mvtnorm")
install.packages("lattice")
install.packages("RColorBrewer")
install.packages("factoextra")
install.packages("usmap")
install.packages("maps")
install.packages("zoo")
install.packages("vars")
install.packages("Rcpp")
install.packages("RcppArmadillo")
```

```
library("mvtnorm")
library("lattice")
library("RColorBrewer")
library("factoextra")
library("usmap")
library("maps")
library("zoo")
library("vars")
library("Rcpp")
library("RcppArmadillo")
```

```
source("functions_BFL.R")
sourceCpp("functions_BFL.cpp")
```

Loading Datasets

Here, we obtain the COVID-19 dataset from the New York Times Github repository. The population data is extracted from National Bureau of Economic Research.

```
#########  Loading Datasets ########################
## data extracted from New York Times state-level data
#obtained from following Github repository: https://github.com/nytimes/covid-19-data
data.states <- read.csv("states-08-18.csv", header = TRUE)
data.states$date <-  as.Date(data.states$date)

# population data extracted from NATIONAL BUREAU OF ECONOMIC RESEARCH
states.population <- read.csv("co-est2019-alldata.csv", header = TRUE)
states.population <- states.population[as.character(states.population$STNAME)
                                       == as.character(states.population$CTYNAME),
                                       c("STNAME", "CTYNAME", "POPESTIMATE2019")]
```

Data Preprocessing

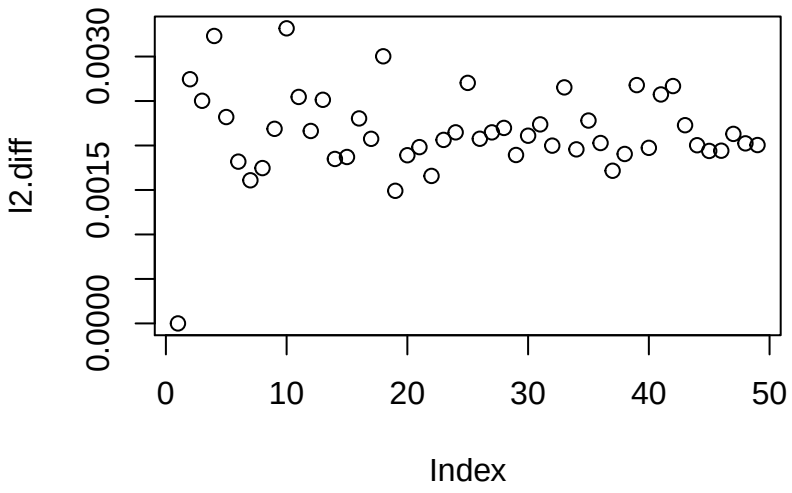
After loading the datasets above, we can now clean the data and construct the variables/attributes. Here, we choose the New York state as an example. One can change the state name and corresponding lockdown date and reopening date to test other states.

```
#########Change/choose the state name, lockdown date and reopen date below
state.name <- "New York"
Date.1 <- '2020-03-22'
Date.2 <- '2020-05-15'
```

```
# get all the states fips in the United States
neighbor.fips <- c()
# region.fips <- fips(state = state.name)
state.fip <- as.numeric(fips(state.name))
for(i in 1: length(state.fips$fips)){
  if(state.fips$fips[i]!= state.fip ){
    neighbor.fips <- c(neighbor.fips, state.fips$fips[i])

  }
}
neighbor.fips <- unique(neighbor.fips)
# fips_info(neighbor.fips)
state.names <- fips_info(neighbor.fips)$full
state.names <- c(state.name, state.names)
state.lowernames <- tolower(state.names)

n.all <- rep(0,length(state.names) )
for(i in 1:length(state.names)){
  n.all[i] <- unique(states.population[
    (states.population$CTYNAME  == state.names[i]) &
      states.population$STNAME== state.names[i] , "POPESTIMATE2019"])
}

state.names.all <- state.names
state.lowernames.all <- state.lowernames

dataList <- list()
for( i in c(1:length(state.names))){
  data.subset <- data.states[data.states$state==state.names[i], c("date", "cases", "deaths")]
  assign(paste("data", state.lowernames[i] , sep="."), data.subset)
```

```
  dataList <-append(dataList, list(data.subset))
}

multi_full <- Reduce(
  function(x, y){merge(x, y, all = TRUE, by ="date")},
  dataList
)
for(i in 1:length(state.names)){
  names(multi_full)[2*i] <- paste0('cases.', state.lowernames[i])
  names(multi_full)[2*i+1] <- paste0('deaths.', state.lowernames[i])
}
#replace the na value with 0
n.domains <- length(state.names)
for(i in 1:(n.domains*2) ){
  multi_full[is.na(multi_full[,i+1]),i+1]=0
}
# multi_full$date <- as.Date(droplevels(multi_full$date))
multi_full$date <- as.Date(multi_full$date)
#choose the first day when the region has one confrimed case as the start date
multi_full <- multi_full[multi_full[,2]>0,]


cases.all <- multi_full[,seq(2, ncol(multi_full), 2)]
deaths.all<- multi_full[,seq(3, ncol(multi_full), 2)]

#T: number of time points
T <- nrow(cases.all);
#R: the number of people who have recovered
#using the nationwide recovered and death number to predict the recovered number
R.all <- floor((1 + (5.5))*deaths.all);
#I: the number of people infected at time t
I.all <- cases.all - R.all;

#S: the number of susceptible people
n.all.matrix <- matrix(rep(n.all,T),ncol = n.domains,byrow = TRUE)
S.all <- n.all.matrix - I.all - R.all;

#the fraction of S, I and R
S.rate.all <- sapply(1:n.domains, function(jjj) S.all[,jjj]/n.all[jjj])
I.rate.all <- sapply(1:n.domains, function(jjj) I.all[,jjj]/n.all[jjj])
R.rate.all <- sapply(1:n.domains, function(jjj) R.all[,jjj]/n.all[jjj])

n.regions <- ncol(I.rate.all)
y.list <- vector("list", T-1);
for(i in 2:T){
  y.list[[i-1]] <- matrix(c(R.rate.all[i, ] - R.rate.all[i-1, ],
                            I.rate.all[i, ] - I.rate.all[i-1, ]), 2,
                          ncol = n.regions, byrow = TRUE);
}

Y.all <- y.list[[1]];
for(i in 2:(T-1)){
  Y.all <- rbind(Y.all, y.list[[i]])
```

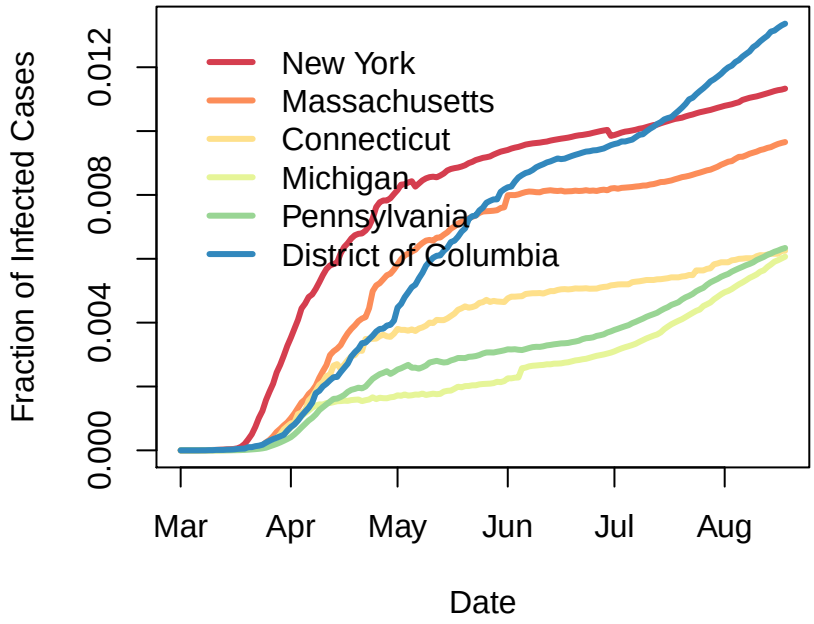
```
}

# max number of neighboring regions
max.neighbor <- 5 + 1
l2.diff <- c()
for(i in 1:n.regions){
  l2.diff <- c(l2.diff, sqrt(sum( (Y.all[ , i] - Y.all[ , 1] )^2)))
}
plot(l2.diff)
```



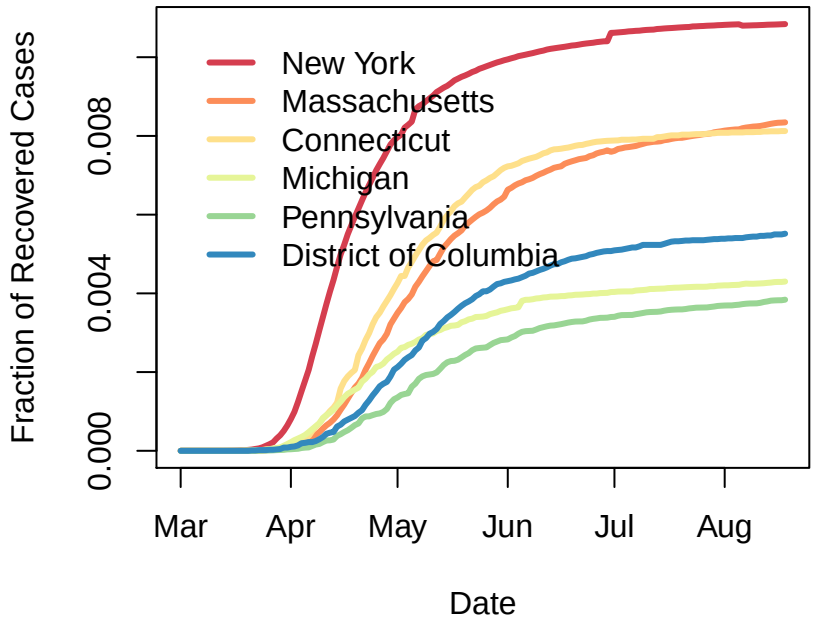

```
state.index <- order(l2.diff)[1: max.neighbor]
state.names <- state.names.all[state.index]
state.lowernames <- state.lowernames.all[state.index]
state.names
```

```
## [1] "New York"             "Massachusetts"        "Connecticut"
## [4] "Michigan"             "Pennsylvania"         "District of Columbia"
```

```
n.domains <- length(state.names)
n.all <- n.all[state.index]
S.rate.all <- S.rate.all[, state.index]
I.rate.all <- I.rate.all[, state.index]
R.rate.all <- R.rate.all[, state.index]

date.region <- multi_full$date
I <- I.all[,1]
R <- R.all[,1]
cols <- brewer.pal(n.domains, "Spectral")

par(mar = c(4., 4.5, 1.5, 1))
plot(multi_full$date, I.rate.all[,1], col=cols[1], lty=1,type="l", lwd = 3,
     ylim = c(0,max(I.rate.all)),
     ylab ='Fraction of Infected Cases', xlab= 'Date', cex.lab=1 , cex.axis=1)
for(i in 2:length(state.names)){
  lines(multi_full$date, I.rate.all[,i], col=cols[i], lty=1, type="l", lwd = 3)
}
legend(multi_full$date[1], max(I.rate.all) , legend=c(state.names),
       col=cols,bg="transparent",bty = "n",
       lwd = 3, cex=1, pt.cex = 1, seg.len=1.5, y.intersp=1, x.intersp=1)
```




```
par(mar = c(4., 4.5, 1.5, 1))
plot(multi_full$date, R.rate.all[,1] , col=cols[1], lty=1,type="l", lwd = 3,
     ylim = c(0,max(R.rate.all)),
     ylab ='Fraction of Recovered Cases', xlab= 'Date',cex.lab=1, cex.axis=1)
for(i in 2:length(state.names)){
  lines(multi_full$date, R.rate.all[,i] ,col=cols[i],lty=1,type="l", lwd = 3)
}
legend(multi_full$date[1], max(R.rate.all), legend=c(state.names),
       col=cols,bg="transparent",bty = "n",
       lwd = 3, cex=1, pt.cex = 1, seg.len=1.5, y.intersp=1 , x.intersp=1)
```




```
##########################################
######## construct varibles ###############
##########################################
y.list <- vector("list",T-1);
```

```
x.list <- vector("list",T-1);
for(i in 2:T){
  y.list[[i-1]] <- matrix(c(R.rate.all[i, 1]-R.rate.all[i-1, 1],
                            I.rate.all[i, 1]-I.rate.all[i-1, 1]), 2, 1);
  x.temp <- matrix(0,2,2);
  x.temp[1,2] <- I.rate.all[i-1, 1];
  x.temp[2,1] <-  I.rate.all[i-1, 1];
  x.temp[2,2] <- -I.rate.all[i-1, 1];
  x.list[[i-1]] <- x.temp;
}

Y <- y.list[[1]];
for(i in 2:(T-1)){
  Y <- rbind(Y, y.list[[i]])
}
X <- x.list[[1]];
for(i in 2:(T-1)){
  X <- rbind(X, x.list[[i]])
}

beta_t <- sapply(1:length(y.list),
                 function(jjj) (y.list[[jjj]][1]+y.list[[jjj]][2])/I.rate.all[jjj,1])
gamma_t <- sapply(1:length(y.list),
                  function(jjj)  y.list[[jjj]][1]/I.rate.all[jjj,1])
```

Model 1: Piecewise Constant SIR Model

Change Point Detection in Model 1

After data preprocessing, we now perform the block fused lasso algorithm to detect the structural breaks in the piecewise constant SIR models.

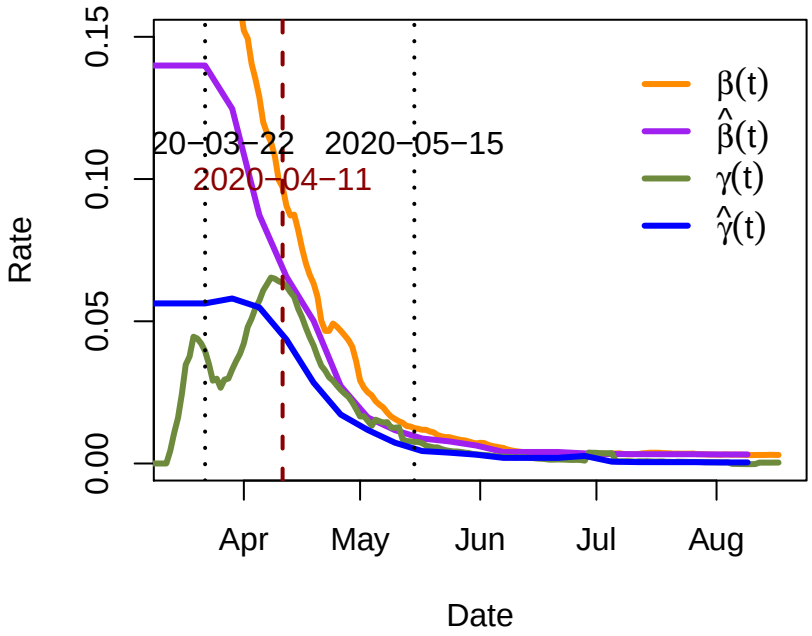
The proposed algorithm in the manuscript is implemented by the function `tbfl()`. In addition to arguments `data_y`, `data_x`, there are other optional arguments for the function `tbfl()` which are listed below:

- `method`: "MLR" for multiple linear regression, and "VAR" for VAR model.
- `lambda.1.cv`: the tuning parmaeter λ_1 for fused lasso. By default, `lambda.1.cv` is a vector with decreasing values constructed based on the time series data, the number of time series components `p`, and the block size `block.size`.
- `lambda.2.cv`: the tuning parmaeter λ_2 for fused lasso. By default, `lambda.2.cv` is a vector with decreasing values constructed based on the number of time points `T` and the number of time series components `p`.
- `q`: the AR order for VAR model. By default, `q` $= 1$.
- `max.iteration`: the max number of iteration for the fused lasso. By default, `max.iteration` $= 100$.
- `tol`: tolerance for the fused lasso. By default, `tol` $= 10^{-2}$.
- `block.size`: the block size b_n. By default, $b_n = \lfloor \sqrt{T} \rfloor$.
- `blocks`: the blocks (sequence). By default, `blocks = seq(0,T,block.size)`. One could also use `blocks` to handle varying-size blocks. For example, `blocks= c(seq(0,5000,100), seq(5200,10000,200))`.
- `HBIC`: if `TRUE`, use high-dimensional BIC criterion (HBIC); if `FALSE`, use traditional BIC criterion (BIC).
- `gamma.val`: the tuning parameter for HBIC.

```
index.date <- seq(1, length(date.region), b_t)
if(index.date[length(index.date)] < length(date.region)){
  index.date <- c(index.date[-length(index.date)], length(date.region))
```

```
}

cols <- c("dark orange", "purple", "darkolivegreen4", "blue" )
beta_t_est <- unlist(beta.est)[seq(1, length(unlist(beta.est)), 2)]
gamma_t_est <- unlist(beta.est)[seq(2, length(unlist(beta.est)), 2)]
ylim_max <- 0.15

beta_t_smooth <- beta_t
beta_t_smooth[beta_t == Inf] = 0
beta_t_smooth[-c(1:6)] <- rollmean(beta_t_smooth, k = 7, align = 'right')
beta_t_smooth[ c(1:6)] <- mean(beta_t_smooth[ c(1:6)])

gamma_t_smooth <- gamma_t
gamma_t_smooth[-c(1:6)] <- rollmean(gamma_t, k = 7, align = 'right')
gamma_t_smooth[ c(1:6)] <- mean(gamma_t[ c(1:6)])

par(mar = c(4., 4.5, 1.5, 1))
plot(date.region[-length(date.region)], beta_t_smooth, type = 'l', col = cols[1],
     lty = 1, lwd = 3, ylab = 'Rate', xlab = 'Date', cex.lab = 1, cex.axis = 1,
     ylim = c(0, ylim_max),
     xlim = c(as.Date('2020-03-15'), date.region[length(date.region)]))
lines(date.region[index.date[-length(index.date)]], beta_t_est, col = cols[2],
      lty = 1, type = "l", lwd = 3)
lines(as.Date(date.region[-length(date.region)]), gamma_t_smooth,col = cols[3],
      lty=1, type = "l", lwd = 3)
lines(date.region[index.date[-length(index.date)]], gamma_t_est, col = cols[4],
      lty = 1, type = "l", lwd = 3)
legend(as.Date(date.region[length(date.region)*3/4]), ylim_max,
       legend = c(expression(beta(t)),  expression(hat(beta)(t)),
                  expression(gamma(t)),  expression(hat(gamma)(t))),
       col = cols, bty = "n", lwd = 3, cex=1, pt.cex = 1, bg = "transparent",
       seg.len=1.5, y.intersp=1 , x.intersp=1)
abline(v = as.Date(cp.date), col = "dark red", cex = 1 ,lwd = 2, lty = 2)
if(length(cp.date) > 2){
  text(x= as.Date(cp.date[2]), y = 2/4*ylim_max, col = "dark red",
       labels = as.character(cp.date[2]), cex = 1.00)
  text(x = as.Date(cp.date[-2]), y = 2/3*ylim_max, col = "dark red",
       labels = as.character(cp.date[-2]), cex = 1.00)
}else if(length(cp.date) == 2){
  if(cp.date[2] - cp.date[1] < 30){
    text(x= as.Date(cp.date[2]), y = 2/4*ylim_max, col = "dark red",
         labels = as.character(cp.date[2]), cex = 1.00)
    text(x = as.Date(cp.date[1]), y = 2/3*ylim_max, col = "dark red",
         labels = as.character(cp.date[1]), cex = 1.00)
  }else{
    text(x = as.Date(cp.date), y = 2/3*ylim_max, col = "dark red",
         labels = as.character(cp.date), cex = 1.00)
  }
}else{
  text(x = as.Date(cp.date), y = 2/3*ylim_max, col = "dark red",
       labels = as.character(cp.date), cex = 1.00)
}
abline(v = as.Date(Date.1), col = 1, cex = 1, lwd = 2, lty = 3)
```

```
abline(v = as.Date(Date.2), col = 1, cex = 1, lwd = 2, lty = 3)
text(x = as.Date(Date.1), y = 3/4*ylim_max, col ="black",
     labels = as.character(as.Date(Date.1)), cex = 1.00)
text(x = as.Date(Date.2), y = 3/4*ylim_max, col = "black",
     labels = as.character(as.Date(Date.2)), cex = 1.00)
```



After detecting the change points, we refit the model. The in-sample MRPE and fitted numbers of infected and recovered cases are provided.

```
##################refit the model
cp <- c(1, temp.1$cp.final, n+1)
m <- length(cp) - 1
Y.hat.1 <- matrix(0, n)
for(i in 1:m){
  Y.temp.1 <- Y[cp[i]: (cp[i+1]-1), ]
  X.temp.1 <- X[cp[i]: (cp[i+1]-1), ]
  est.temp.1 <- lm(Y.temp.1 ~ X.temp.1  - 1)
  Y.hat.1[cp[i]: (cp[i+1]-1), ] <- est.temp.1$fitted.values
}

R.hat.1 <- rep(0,T)
R.hat.1[1] <- R.rate.all[1,1]
for(i in 2:T){
  R.hat.1[i] <- R.rate.all[i-1,1] + Y.hat.1[(i-2)*2+1]
}
R.hat.1 <- R.hat.1*n.all[1]

I.hat.1 <- rep(0,T)
I.hat.1[1] <- I.rate.all[1,1]
for(i in 2:T){
  I.hat.1[i] <-  I.rate.all[i-1,1]+Y.hat.1[(i-2)*2+2]
}
I.hat.1 <- I.hat.1*n.all[1]
```

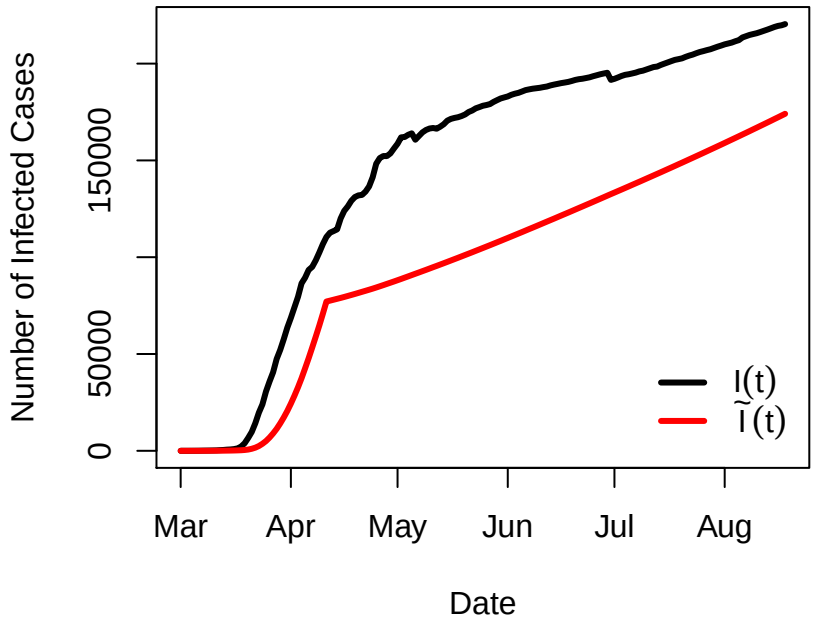
```
MRPE_1_I <- mean(  abs ( (     c(I.hat.1[-1]) - c(I[-1])     )  /c(I[-1])  )
                   [intersect( which(c(I[-1]) > 0  ), 2:(T-1) )  ]  )
print(round(MRPE_1_I, 4))
```

```
## [1] 0.0471
```

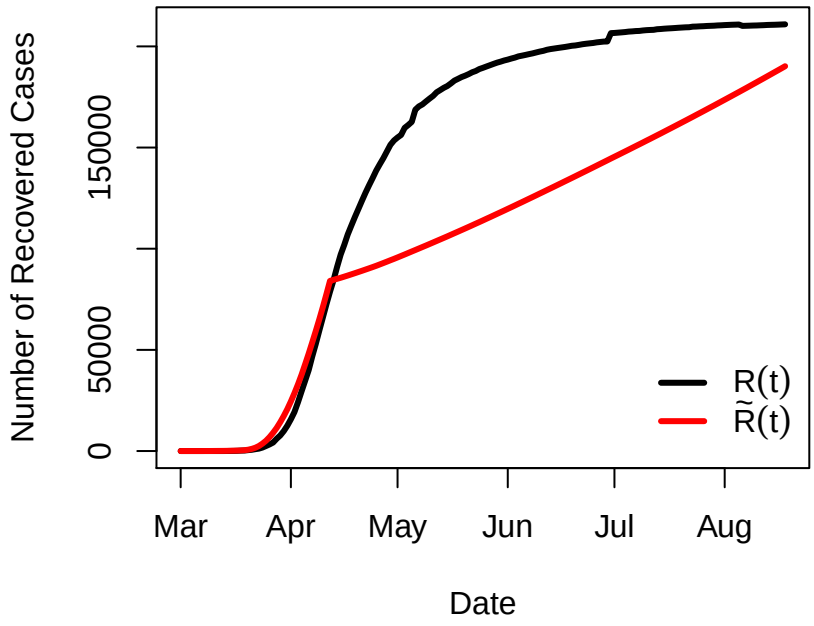
```
MRPE_1_R <- mean(  abs ( (     c(R.hat.1[-1]) - c(R[-1])     )  /c(R[-1])  )
                   [intersect( which(c(R[-1]) > 0  ), 2:(T-1) )  ]  )
print(round(MRPE_1_R, 4))
```

```
## [1] 0.0369
```

```
R.tilde.1 <- rep(0,T)
R.tilde.1[1] <- R.rate.all[1,1]
for(i in 2:T){
  R.tilde.1[i] <- R.tilde.1[i-1] + Y.hat.1[(i-2)*2+1]
}
R.tilde.1 <- R.tilde.1*n.all[1]
I.tilde.1 <- rep(0,T)
I.tilde.1[1] <- I.rate.all[1,1]
for(i in 2:T){
  I.tilde.1[i] <- I.tilde.1[i-1] + Y.hat.1[(i-2)*2+2]
}
I.tilde.1 <- I.tilde.1*n.all[1]

par(mar = c(4., 4.5, 1.5, 1))
plot(date.region, I , col='1', ylim=c(0,max(I, I.tilde.1)), lty=1, type="l", lwd = 3,
     ylab ='Number of Infected Cases', xlab= 'Date', cex.lab=1 , cex.axis=1)
lines(date.region, I.tilde.1, col='2',lty=1,type="l", lwd = 3)
legend(as.Date(date.region[length(date.region)*3/4]), 1/4*max(I,I.tilde.1),
       legend=c(expression(I(t)), expression(tilde(I)(t))),
       col=c( 1,2), bg="transparent", bty = "n",
       lwd = 3, cex=1, pt.cex = 1, seg.len=1.5, y.intersp=1 , x.intersp=1)
```



```r
par(mar = c(4., 4.5, 1.5, 1))
plot(date.region, R, type='l', col='1', ylim=c(0,max(R, R.tilde.1)),lty=1,lwd = 3,
     ylab ='Number of Recovered Cases', xlab= 'Date', cex.lab=1 , cex.axis=1)
lines(date.region, R.tilde.1, col='2', lty=1, type="l", lwd = 3)
legend(as.Date(date.region[length(date.region)*3/4]), 1/4*max(R, R.tilde.1),
       legend=c(expression(R(t)), expression(tilde(R)(t))),
       col=c( 1,2), bg="transparent", bty = "n",
       lwd = 3, cex=1, pt.cex = 1, seg.len=1.5, y.intersp=1 , x.intersp=1)
```



Model 2: Piecewise Stationary SIR Model with Spatial Heterogeneity

Model 2.3: Similarity-based Weight

```r
#####################################################
###### Model 2.3 similarity based weight
###### spatial components are chosen by similarity
#####################################################

distance_3 <- rep(0, n.domains)
#Power Distance Weights.
for(i in 2:n.domains){
  distance.temp <- sort(12.diff)[i]
  distance_3[i] <- 1/(distance.temp)^1
}
Omega_3 <- distance_3/sum(distance_3)

rows.combined <- 2*(T-1)
cols.combined <- n.domains
matrix.combined <- matrix(0, nrow=rows.combined, ncol=cols.combined)
matrix.combined[ seq(1, rows.combined, 2),] <-
  as.matrix(unname(R.rate.all))[-1,]-as.matrix(unname(R.rate.all))[-T,]
matrix.combined[ seq(2, rows.combined, 2),] <-
```

```
  as.matrix(unname(I.rate.all))[-1,]-as.matrix(unname(I.rate.all))[-T,]
neighbor.matrix <- matrix.combined
neighbor.weighted_3 <- neighbor.matrix %*% (Omega_3)

#remove the last two elements  at time point t=T
#and add two elements for time point t= 0
neighbor.weighted_3 <- as.matrix(neighbor.weighted_3[-c(rows.combined-1,rows.combined),])
neighbor.weighted_3 <- as.matrix(c(neighbor.weighted_3[c(1,2),], neighbor.weighted_3))

p.x <- ncol(X)
p.y <-ncol(Y)
n <- nrow(X)

cp <- c(1, temp.1$cp.final, n+1)
m <- length(cp) - 1
X.new.3 <- matrix(0, nrow = n, ncol = m*p.x + 1)
for(i in 1:m){
  X.new.3[cp[i]: (cp[i+1]-1), (p.x*(i-1)+1) : (p.x*i) ] <- X[cp[i]: (cp[i+1]-1), ]
}
X.new.3[, m*p.x + 1] <- neighbor.weighted_3

est.2.3 <- lm(Y[-c(1:2), ] ~ X.new.3[-c(1:2), ]  - 1)
Y.hat.2.3 <- c(0, 0, est.2.3 $fitted.values)
summary(est.2.3 )
```

```
##
## Call:
## lm(formula = Y[-c(1:2), ] ~ X.new.3[-c(1:2), ] - 1)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -2.379e-04 -1.799e-05 -3.820e-07  1.465e-05  2.645e-04
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## X.new.3[-c(1:2), ]1  0.0713047  0.0062544  11.401   <2e-16 ***
## X.new.3[-c(1:2), ]2  0.0468878  0.0037011  12.669   <2e-16 ***
## X.new.3[-c(1:2), ]3 -0.0025890  0.0011524  -2.247   0.0253 *
## X.new.3[-c(1:2), ]4 -0.0004384  0.0006775  -0.647   0.5180
## X.new.3[-c(1:2), ]5  1.2613744  0.0921640  13.686   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.147e-05 on 333 degrees of freedom
## Multiple R-squared:  0.7101, Adjusted R-squared:  0.7057
## F-statistic: 163.1 on 5 and 333 DF,  p-value: < 2.2e-16
```

```
R.hat.2.3 <- rep(0, T)
R.hat.2.3[1] <- R.rate.all[1, 1]
for(i in 2:T){
  R.hat.2.3[i] <- R.rate.all[i-1, 1] + Y.hat.2.3[(i-2)*2+1]
}
R.hat.2.3 <- R.hat.2.3*n.all[1]
```

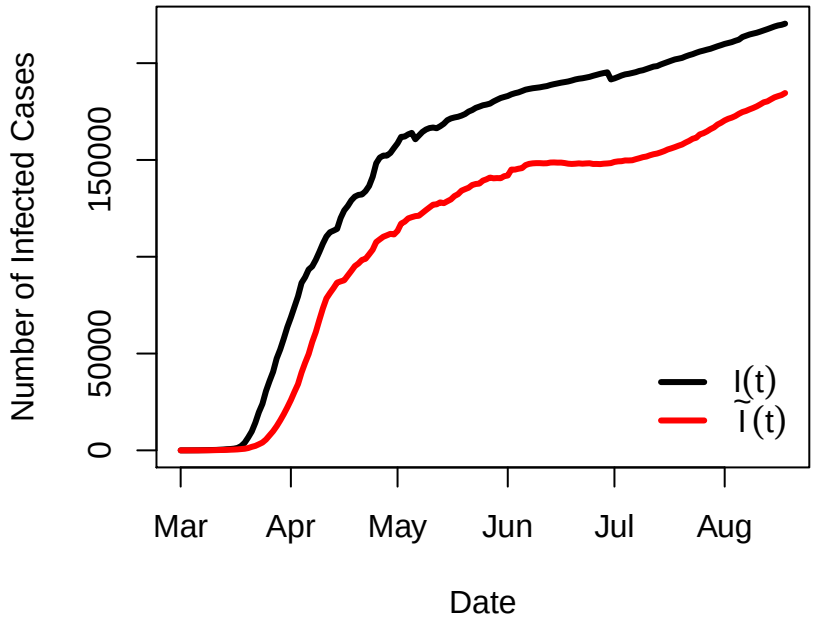
```
I.hat.2.3 <- rep(0,T)
I.hat.2.3[1] <- I.rate.all[1,1]
for(i in 2:T){
  I.hat.2.3[i] <-  I.rate.all[i-1,1] + Y.hat.2.3[(i-2)*2+2]
}
I.hat.2.3 <- I.hat.2.3*n.all[1]

MRPE_2.3_I <- mean(  abs ( (     c(I.hat.2.3[-1]) - c(I[-1])     )  /c(I[-1])  )[intersect( which(c(I[-
print(round(MRPE_2.3_I,4))
```

```
## [1] 0.0397
```

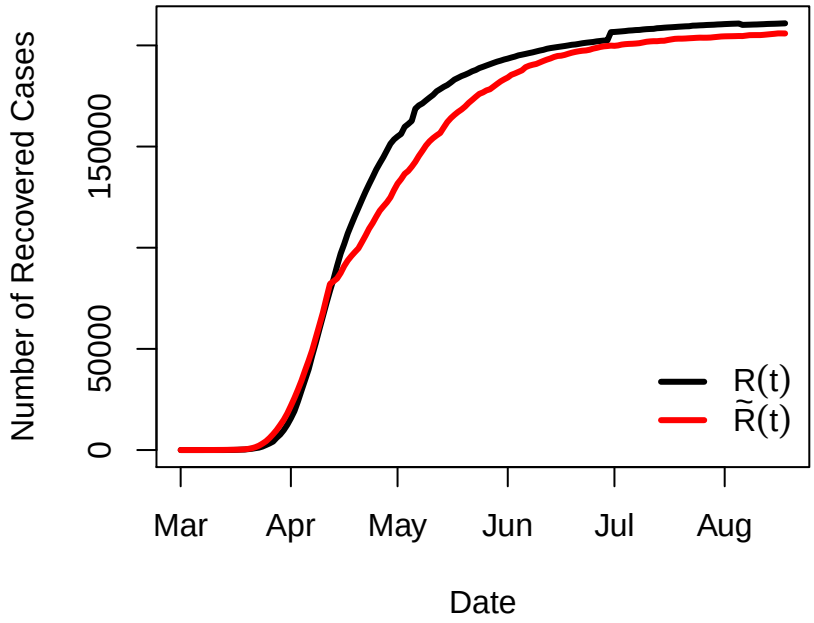
```
MRPE_2.3_R <- mean(  abs ( (     c(R.hat.2.3[-1]) - c(R[-1])     )  /c(R[-1])  )[intersect( which(c(R[-
print(round(MRPE_2.3_R,4))
```

```
## [1] 0.0253
```

```
R.tilde.2.3 <- rep(0,T)
R.tilde.2.3[1] <- R.rate.all[1,1]
for(i in 2:T){
  R.tilde.2.3[i] <- R.tilde.2.3[i-1] + Y.hat.2.3[(i-2)*2+1]
}
R.tilde.2.3 <- R.tilde.2.3*n.all[1]

I.tilde.2.3 <- rep(0,T)
I.tilde.2.3[1] <- I.rate.all[1,1]
for(i in 2:T){
  I.tilde.2.3[i] <- I.tilde.2.3[i-1] + Y.hat.2.3[(i-2)*2+2]
}
I.tilde.2.3 <- I.tilde.2.3*n.all[1]

par(mar = c(4., 4.5, 1.5, 1))
plot(date.region, I, col='1', ylim=c(0,max(I, I.tilde.2.3)), lty=1, type="l", lwd = 3,
     ylab ='Number of Infected Cases', xlab= 'Date',cex.lab=1, cex.axis=1)
lines(date.region, I.tilde.2.3, col='2', lty=1, type="l", lwd = 3)
legend(as.Date(date.region[length(date.region)*3/4]), 1/4*max(I, I.tilde.2.3),
       legend=c(expression(I(t)), expression(tilde(I)(t))),
       col=c( 1,2), bg="transparent", bty = "n",
       lwd = 3, cex=1, pt.cex = 1, seg.len=1.5, y.intersp=1 , x.intersp=1)
```

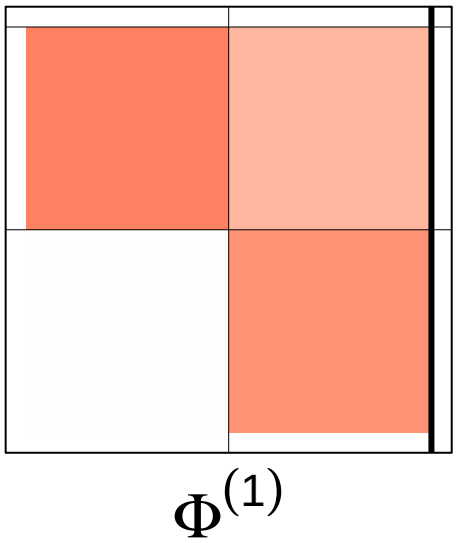



```
par(mar = c(4., 4.5, 1.5, 1))
plot(date.region, R, type='l',col='1', ylim=c(0,max(R, R.tilde.2.3)), lty=1, lwd = 3,
     ylab ='Number of Recovered Cases', xlab= 'Date', cex.lab=1 , cex.axis=1)
lines(date.region, R.tilde.2.3, col='2',lty=1,type="l",lwd = 3)
legend(as.Date(date.region[length(date.region)*3/4]), 1/4*max(R, R.tilde.2.3),
       legend=c(expression(R(t)), expression(tilde(R)(t))),
       col=c( 1,2), bg="transparent", bty = "n",
       lwd = 3, cex=1, pt.cex = 1, seg.len=1.5, y.intersp=1, x.intersp=1)
```

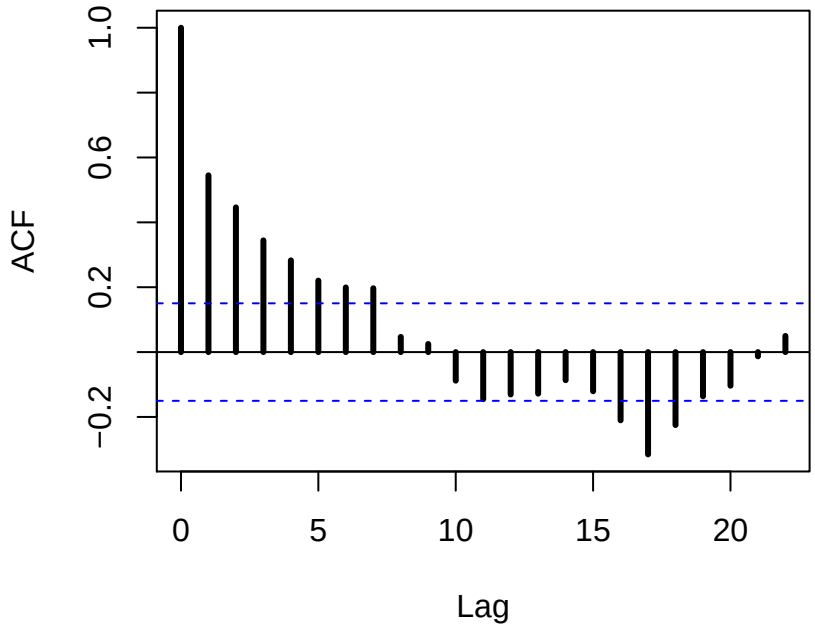


Model 3: VAR(p) Model

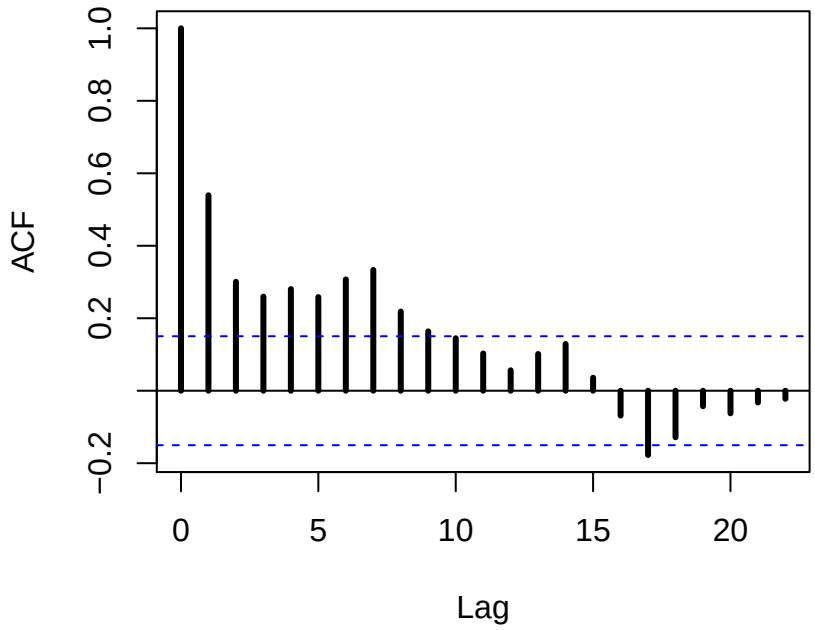
```
par(mar = c(4., 4.5, 1.5, 1))
print(plot.ar.matrix(coef.matrix, p = p.est))
```



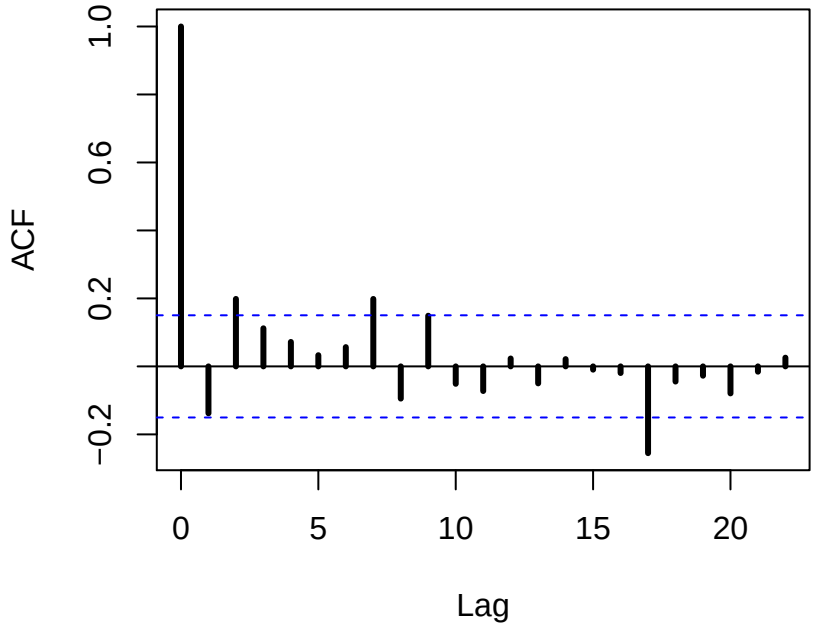

```
par(mar = c(4., 4.5, 1.5, 1))
acf(residual_Delta.I, cex.lab=1 , cex.axis=1  , lwd = 3)
```



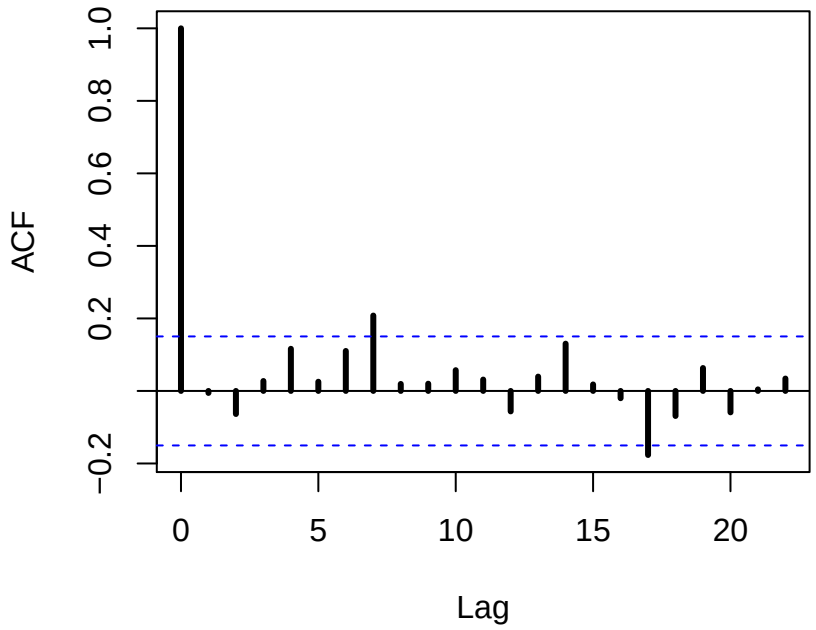

```
par(mar = c(4., 4.5, 1.5, 1))
acf(residual_Delta.R, cex.lab=1 , cex.axis=1  , lwd = 3)
```




```
par(mar = c(4., 4.5, 1.5, 1))
acf(residual.tilde[, 1], cex.lab=1 , cex.axis=1, lwd = 3)
```




```
par(mar = c(4., 4.5, 1.5, 1))
acf(residual.tilde[, 2], cex.lab=1 , cex.axis=1, lwd = 3)
```



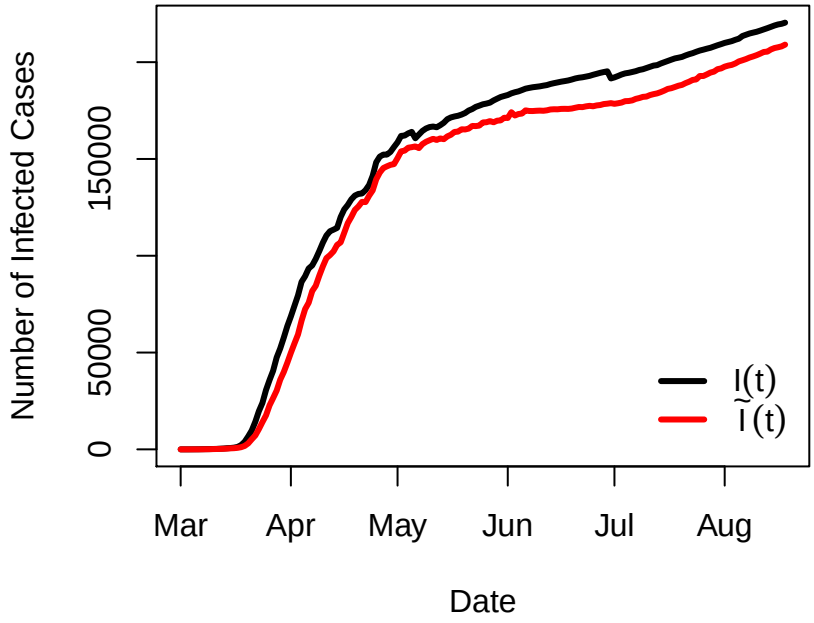

```
#vectorize the fitted residuals
residual.hat.vec <- rep(0, nrow(residual.hat)*2)
#fitted residual of Delta R
residual.hat.vec[seq(1, nrow(residual.hat)*2, 2)] <- residual.hat[, 2]
#fitted residual of Delta I
residual.hat.vec[seq(2, nrow(residual.hat)*2, 2)] <- residual.hat[, 1]

Y.hat.3 <- Y.hat.2.3 + residual.hat.vec/n.all[1]

R.hat.3 <- rep(0, T)
R.hat.3[1] <- R.rate.all[1, 1]
for(i in 2:T){
  R.hat.3[i] <- R.rate.all[i-1, 1] + Y.hat.3[(i-2)*2+1]
}
R.hat.3 <- R.hat.3*n.all[1]

I.hat.3 <- rep(0,T)
I.hat.3[1] <- I.rate.all[1,1]
for(i in 2:T){
  I.hat.3[i] <-  I.rate.all[i-1,1] + Y.hat.3[(i-2)*2+2]
}
I.hat.3 <- I.hat.3*n.all[1]

MRPE_3_I <- mean(  abs ( (     c(I.hat.3[-1]) - c(I[-1])     )  /c(I[-1])  )
                   [intersect( which(c(I[-1]) > 0  ), 2:(T-1) ) ]  )
print(round(MRPE_3_I,4))
```

```
## [1] 0.0301
```

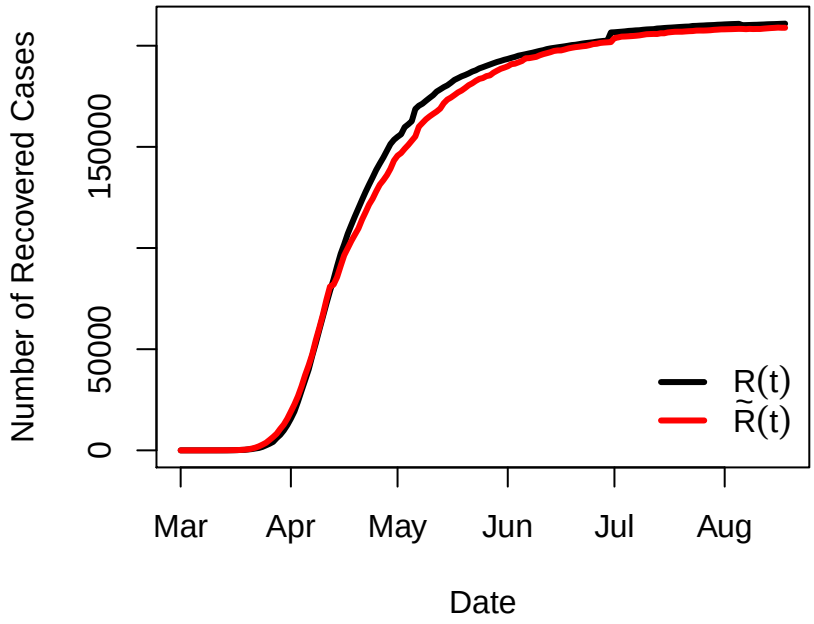
```
MRPE_3_R <- mean(  abs ( (     c(R.hat.3[-1]) - c(R[-1])     )  /c(R[-1])  )
                   [intersect( which(c(R[-1]) > 0  ), 2:(T-1) ) ]  )
print(round(MRPE_3_R,4))
```

```
## [1] 0.0175
```

```
R.tilde.3 <- rep(0,T)
R.tilde.3[1] <- R.rate.all[1,1]
for(i in 2:T){
  R.tilde.3[i] <- R.tilde.3[i-1] + Y.hat.3[(i-2)*2+1]
}
R.tilde.3 <- R.tilde.3*n.all[1]

I.tilde.3 <- rep(0,T)
I.tilde.3[1] <- I.rate.all[1,1]
for(i in 2:T){
  I.tilde.3[i] <- I.tilde.3[i-1] + Y.hat.3[(i-2)*2+2]
}
I.tilde.3 <- I.tilde.3*n.all[1]

par(mar = c(4., 4.5, 1.5, 1))
plot(date.region, I, col='1', ylim=c(0,max(I, I.tilde.3)), lty=1, type="l", lwd = 3,
     ylab ='Number of Infected Cases', xlab= 'Date',cex.lab=1, cex.axis=1)
lines(date.region, I.tilde.3, col='2', lty=1, type="l", lwd = 3)
legend(as.Date(date.region[length(date.region)*3/4]), 1/4*max(I, I.tilde.3),
       legend=c(expression(I(t)), expression(tilde(I)(t))),
       col=c( 1,2), bg="transparent", bty = "n",
       lwd = 3, cex=1, pt.cex = 1, seg.len=1.5, y.intersp=1 , x.intersp=1)
```




```
par(mar = c(4., 4.5, 1.5, 1))
plot(date.region, R, type='l',col='1', ylim=c(0,max(R, R.tilde.3)), lty=1, lwd = 3,
     ylab ='Number of Recovered Cases', xlab= 'Date', cex.lab=1 , cex.axis=1)
lines(date.region, R.tilde.3, col='2',lty=1,type="l",lwd = 3)
legend(as.Date(date.region[length(date.region)*3/4]), 1/4*max(R, R.tilde.3),
       legend=c(expression(R(t)), expression(tilde(R)(t))),
       col=c( 1,2), bg="transparent", bty = "n",
       lwd = 3, cex=1, pt.cex = 1, seg.len=1.5, y.intersp=1, x.intersp=1)
```


Number of Recovered Cases
0
50000
150000
Mar
Apr
May
Jun
Jul
Aug
Date
R(t)
R̃(t)