Industrial & Applied Mathematics

4-Vector.org

A Blog with Annotated R-Notebooks: Math, Statistics, and Multiphysics for Learned Peer Discussion on the Science of Data

Simulating Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2) Coupling Constants and Rates with a SEIR Compartmental Model

Michael A. X. Izatt
E | Michael.Izatt@Alum.MIT.Edu (mailto:Michael.Izatt@Alum.MIT.Edu)
E | izatt@UChicago.Edu (mailto:izatt@UChicago.Edu)
W | 4-Vector.org
In | www.linkedin.com/in/max-izatt

May 7, 2020

Version 2020-05-07-1200/MAI

M.A.X. Izatt, “Simulating Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2) Coupling Constants and Rates with a SEIR Compartmental Model”, 4-Vector-org, May 7, 2020, https://4-vector.org/2020/05/12/4vector-org-2020-05-07-1200-mai/ (https://4-vector.org/2020/05/12/4vector-org-2020-05-07-1200-mai/)

Abstract

This paper is a quick communication to demonstrate the stochastic realization of coupled Poisson process, the result of which is family of SEIR trajectories. The coupled differential equations are fitted to South Korean data for Cases and Deaths that are attributed to Coronavirus Disease (COVID-19), which is caused by the Sudden Acute Respiratory Syndrome Coronavirus, SARS-CoV-2 contagion. The Monte Carlo simulation estimates the SEIR Coupling Constants and Rates by Minimum Mean Absolute Percentage Error (MAPE) of Cases and Deaths trajectories vis-à-vis the reported data.

Discussion

This short paper is an extension of a prior illustration of the Susceptible-Exposed-Infectious-Recovered (SEIR) model here:

https://4-vector.org/2020/04/27/4vector-org-2020-04-25-2100-mai/ (https://4-vector.org/2020/04/27/4vector-org-2020-04-25-2100-mai/)

and here

https://4-vector.org/2020/05/04/4vector-org-2020-05-02-0900-mai/ (https://4-vector.org/2020/05/04/4vector-org-2020-05-02-0900-mai/)

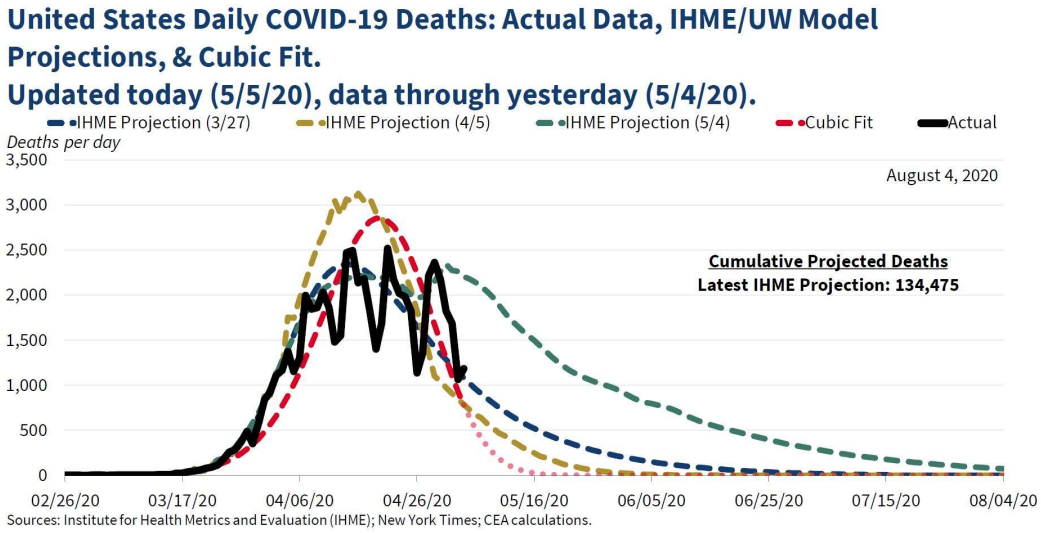
In those writings, we followed Althaus to illustrate the mechanics of fitting the SEIR curve to the data for West-Africa's 2013 Ebolavirus (EVD) epidemic.

https://4vector.files.wordpress.com/2020/04/althaus_west-africa.pdf (https://4vector.files.wordpress.com/2020/04/althaus_west-africa.pdf)

This calculation fits the SEIR model to South-Korean COVID-19 data for the period 12-DEC-2019 to 07-MAY-2020.

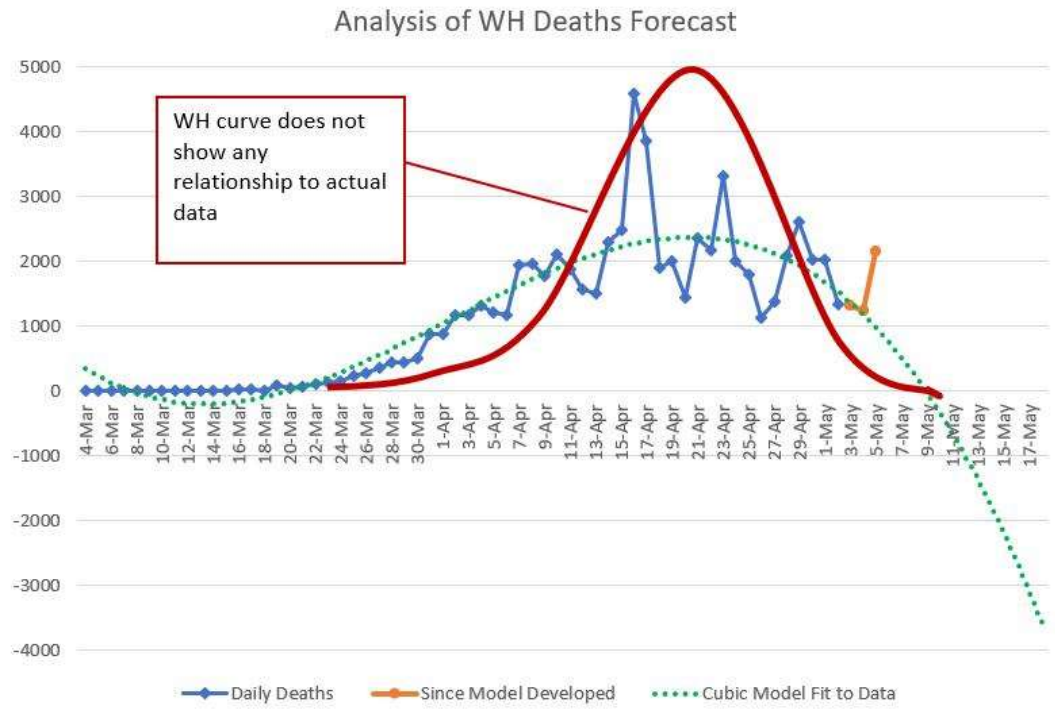
Analysis

A recent chart has become infamous.



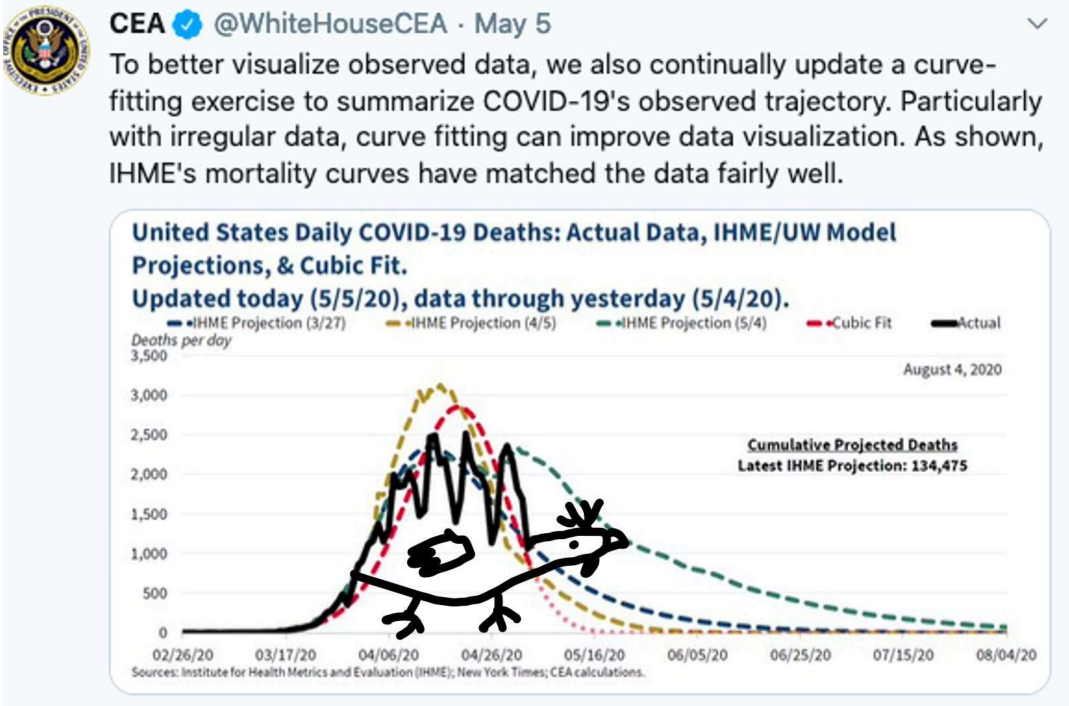
The title of the image makes the unfortunate mistake of attributing the data trend to a "cubic fit," which might be true as far as it goes but does nothing to either describe the data generating function or predict the trajectories of Cases or Deaths due to the contagion.

However, critics also showed a lack of understanding. Someone published the chart with a call out in a box that commented that the "[W]hite [H]ouse curve does not show any relationship to actual data," which is also true as far as it goes, but it hardly goes sufficiently far. Any smooth trend through stochastic data could be equally criticized.



Once it was correctly noticed that the “Cubic Fit” was no better than a freehand trace, many freehand traces emerged, and although IMHE suggested a Cubic Fit, other Data Generating Functions have been suggested, and they do surprisingly well at description, though only time will tell whether they are predictive.

https://twitter.com/PeterGleick/status/1258067329264447488/photo/1 (https://twitter.com/PeterGleick/status/1258067329264447488/photo/1)



Import COVID-19 Data

It turns out that the United States data is a challenge, perhaps owing to reporting, testing, or unrelated Deaths classified as caused by COVID-19; however, the South-Korean data set is very well curated and can serve as a nice benchmark to get the model working. We will analyze 2019-2020 COVID-19 data to calculate South-Korean Cases and Deaths from 31-DEC-2019 through 07-MAY-2020 using the correct data-generating function, which for this problem set, we conjecture to be the SEIR model.

The data can be found here:

https://ourworldindata.org/coronavirus-source-data (https://ourworldindata.org/coronavirus-source-data)

We extracted KOR data and prepared an analytic data set of the following structure.

```
file_ <- '..\\csv\\owid-covid-data-kor.csv'

df_ <- read.csv(file=file_,stringsAsFactors = FALSE,header=TRUE,sep=',')

str(df_)
```

```
## 'data.frame':    129 obs. of  16 variables:
##  $ Day                      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ location                 : chr  "South Korea" "South Korea" "South Korea" "South Korea" ...
##  $ YYYYMMDD                 : chr  "D20191231" "D20200101" "D20200102" "D20200103" ...
##  $ total_cases              : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ new_cases                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ total_deaths             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ new_deaths               : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ total_cases_per_million  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ new_cases_per_million    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ total_deaths_per_million: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ new_deaths_per_million   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ total_tests              : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_tests                : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ total_tests_per_thousand: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_tests_per_thousand   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ tests_units              : logi  NA NA NA NA NA NA ...
```

Subset the data to extract total_cases and total_deaths for South Korea

```
c_ <- c('total_cases','new_cases','total_deaths','new_deaths')

subset_ <- subset(df_, df_$location == 'South Korea')

covid_ <- subset_[,c_]

str(covid_)
```

```
## 'data.frame':    129 obs. of  4 variables:
##  $ total_cases : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ new_cases   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ total_deaths: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ new_deaths  : int  0 0 0 0 0 0 0 0 0 0 ...
```

Replace zeros with NA for plotting purposes, if desired. Let's pass to see how we are going to chart the data when we get there, below.

```
# covid_[,'total_cases']    <- ifelse(covid_[,'total_cases']   == 0,'',covid_[,'total_cases'] )
# covid_[,'new_cases']      <- ifelse(covid_[,'new_cases']  == 0,'',covid_[,'new_cases']  )
# covid_[,'total_deaths']   <- ifelse(covid_[,'total_deaths']  == 0,'',covid_[,'total_deaths']
 )
# covid_[,'new_deaths']     <- ifelse(covid_[,'new_deaths'] == 0,'',covid_[,'new_deaths'] )


cases_g_ <- covid_[,'total_cases']
deaths_g_ <- covid_[,'total_deaths']

which_is_not_na_ <- which(cases_g_ > 0)
```

Ok, we have the data, so we need to calculate the SEIR trajectories. Just pull the code from the paper here:

https://4vector.files.wordpress.com/2020/05/4vector-org-2020-05-02-0800-mai.pdf (https://4vector.files.wordpress.com/2020/05/4vector-org-2020-05-02-0800-mai.pdf)

There are several tasks to get ready for this exercise.

- Set the number of Monte Carlo iterations
- Conjecture a probability distribution for the Rates
- Conjecture a probability distribution for each Coupling Constant

We like triangular distributions because they give you a bounded range and a modal measure of central tendency, but you will see that this paper is published with the Uniform Distributions remaining.

There was a substantial amount of guessing (termed 'tuning parameters'), so there were several dead ends before we got this far. We have left the guess work in the commented code so that you can get a feel for the labor involved in getting a Monte Carlo simulation to converge. As my dissertation advisor commented, "Building a model and getting it to converge are two very different things." Indeed.

In this code block, we define a function for the Transmission Rate, $\beta(t)$, as described in the previous papers.

Also, set the number of Monte Carlo iterations in this block. We like say 1,000 or more, which is sufficient to probe areas of the solution space for parameters that have three significant features of the order of 10^{-3} such as those found in SEIR models.

```
############################################################
nSimulations_ <- 1000 #<-- SET THIS PARAMETER
############################################################



FUNC_B_t_ <- function(B_0_, t_, k_, tau_){

  if(t_ > tau_){

    B_0_ * exp(-k_ * (t_ - tau_))

  } else {

    B_0_

  }

}


right = function (string, char){
  substr(string,nchar(string)-(char-1),nchar(string))
}


left = function (string,char){
  substr(string,1,char)
}
```

In this code block, we do the hard work of defining the Probability Density Functions (PDFs) for the SEIR Coupling Constants and Rates. You will see that I have left the Constants and Rates from the earlier papers for 2013 West-Africa Ebolavirus, but of course, COVID-19 has nothing to do with 2013-EVD. We have simply left those there so that the reader can tie back to the earlier papers. On the one hand, we run the risk of embarrassing ourselves, but on the other hand, it is instructive for the reader who is new to Monte Carlo simulation to see how this work is done.

Monte Carlo simulation is at least as much art as science, so we have left the guess work in the code as superseded comments. There were many, many iterations to converge on a range of Constants and Rates that gave good agreement to the KOR data. Each experiment involves contemplation of the dynamics and Multiphysics that are trying to peek through the simulation. Study the effects of each PDF on the resulting trajectories until you learn the SEIR dynamics, by which time a ‘guess and check’ exercise becomes much more deliberate and purposeful.

Note that there was a hint from the WHO, but only a hint, and in the end, it did not describe the KOR Public-Health response to COVID-19, which is most-likely best-in-class, and for good reason. South Korea was hard-hit by H1N1 in 2009, and they have the federal Public-Health and legal framework to track, isolate, test, arrest, detain, and quarantine that many countries lack. Indeed, in jurisdictions such as the United States, there may well be Constitutional protections **against** such surveillance, which may limit the efficacy of the Public-Health response against such a contagion.

https://www.who.int/bulletin/online_first/20-255695.pdf (https://www.who.int/bulletin/online_first/20-255695.pdf)

```
# Random Number Factory
# Start with these constants:  https://www.who.int/bulletin/online_first/20-255695.pdf


set.seed(137)

#B_0_ <- 0.2700  # PER DAY              #/3/
#B_0_ <- rtriangle(n=nSimulations_,a=0.01,b=2.5,c=1.5)
#B_0_ <- rtriangle(n=nSimulations_,a=0.01,b=2.5,c=1.5)
#B_0_ <- runif(n=nSimulations_,min=1.25,max=2.50)
B_0_ <- runif(n=nSimulations_,min=1.3,max=1.60)



#SIGMA_ <- 1 / 5.3000                  #/5/
#SIGMA_ <- rtriangle(n=nSimulations_,a=0.20, b=0.50, c=0.40)
#SIGMA_ <- rtriangle(n=nSimulations_,a=0.15, b=0.40, c=0.30) #wrong way
#SIGMA_ <- rtriangle(n=nSimulations_,a=0.25, b=0.60, c=0.40)
#SIGMA_ <- runif(n=nSimulations_,min=0.5, max=0.8) good
#SIGMA_ <- runif(n=nSimulations_,min=0.75, max=0.9) good
SIGMA_ <- runif(n=nSimulations_,min=0.60, max=0.81)

#GAMMA_ <- 1 / 5.6100                  #/6/
#GAMMA_ <- rtriangle(n=nSimulations_, a=0.05, b=2.00, c=1.0)
#GAMMA_ <- runif(n=nSimulations_, min=0.75, max=1.25) high
GAMMA_ <- runif(n=nSimulations_, min=0.80, max=1.25)

#K_ <- 0.0023                          #/7/
#K_ <- rtriangle(n=nSimulations_, a=0.0300, b=0.0500, c=0.0394)
#K_ <- rtriangle(n=nSimulations_, a=0.00300, b=0.00500, c=0.00394) run away
#K_ <- rtriangle(n=nSimulations_, a=0.0100, b=0.07500, c=0.035)  kill too early
#K_ <- rtriangle(n=nSimulations_, a=0.00300, b=0.07500, c=0.039) #coverage and go to 50000 itera
tions
K_ <- runif(n=nSimulations_, min=0.035, max=0.045)


# This is the public-health response after the first case is recorded
#TAU_ <- 10                            #/8/
days_ <- 60:80
TAU_ <-  sample(days_,size=nSimulations_, replace=TRUE)

#F_ <- 0.74                            #/15/
# Search shows 2662 deaths per 63840 confirmed cases. Cases to confirmed cases is scale invarian
t. Use 1000 as a factor
# 2662 / 63840 = 0.04107143.
#F_ <- rtriangle(n=nSimulations_,a=0.05, b=0.1, c=0.075)
F_ <- runif(n=nSimulations_, min=0.02, max=0.04)


df_monte_carlo_ <- data.frame(
  Iteration=as.numeric(),
  B0=as.numeric(),
  SIGMA=as.numeric(),
```

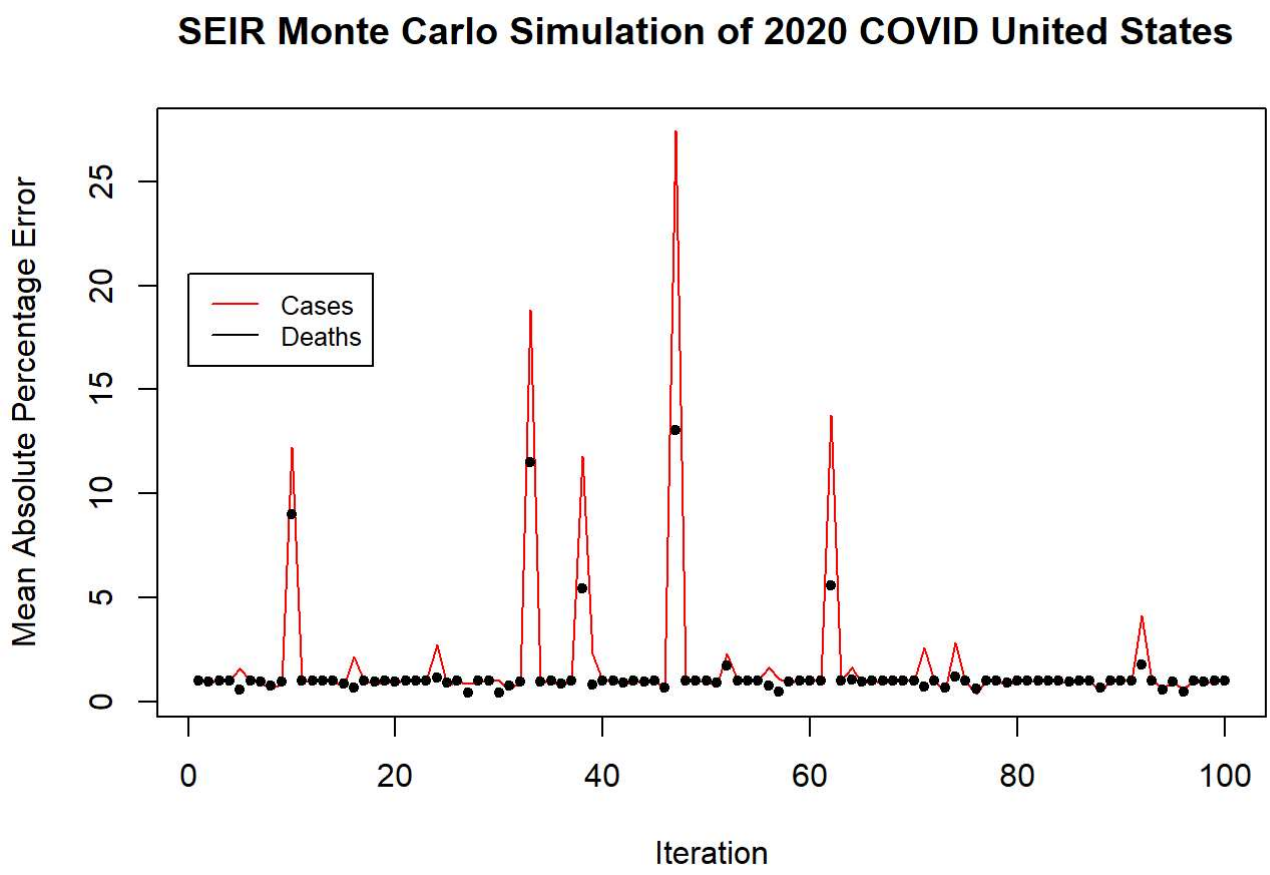
```
  GAMMA=as.numeric(),
  KAPPA=as.numeric(),
  TAU=as.numeric(),
  F=as.numeric(),
  MAPE_C=as.numeric(),
  MAPE_D=as.numeric(),
  stringsAsFactors=FALSE
)
```

This code block performs the iterations, writes the results to the hard disk, and writes a trace of MAPE to the R Notebook for a look/see.

```
for(iSimulation_ in 1:nSimulations_){


  df_monte_carlo_[iSimulation_,'Iteration'] <- iSimulation_
  df_monte_carlo_[iSimulation_,'B0'] <- B_0_[iSimulation_]
  df_monte_carlo_[iSimulation_,'SIGMA'] <- SIGMA_[iSimulation_]
  df_monte_carlo_[iSimulation_,'GAMMA'] <- GAMMA_[iSimulation_]
  df_monte_carlo_[iSimulation_,'KAPPA'] <- K_[iSimulation_]
  df_monte_carlo_[iSimulation_,'TAU'] <- TAU_[iSimulation_]
  df_monte_carlo_[iSimulation_,'F'] <- F_[iSimulation_]


#Create a data frame to store parameters, run-time variable values, and resulting analytic value
s.

df_simulation_ <- data.frame(
  T=as.numeric(),
  B_t=as.numeric(),
  dS=as.numeric(),
  dE=as.numeric(),
  dI=as.numeric(),
  dR=as.numeric(),
  dC=as.numeric(),
  dD=as.numeric(),
  S=as.numeric(),
  E=as.numeric(),
  I=as.numeric(),
  R=as.numeric(),
  C=as.numeric(),
  D=as.numeric(),
  R_e=as.numeric(),
  stringsAsFactors=FALSE
)


# Intitalize the model parameters.

# There is no Multiphysics here. John von Neumann said "With 4 adjustable parameters,
# I can draw an elephant; with five, I can wiggle his trunk."  This is that.

# In other words, there is nothing fundamental about this model.
# It converges at the pleasure of the adjustable parameters.


N_0_ <- 10^7                          #/1/
N_ <- N_0_                            #/2/


B_t_ <- 0                             #/4/

# Initialize the SEIR variables proper
S_ <- N_0_                            #/9/
E_ <- 0                               #/10/
```

```
I_ <- 1                                  #/11/
R_ <- 0                                  #/12/
D_ <- 0                                  #/13/
C_ <- 1                                  #/14/


nT_ <- length(deaths_g_)                                                    #/1/
dT_ <- 1                                                  #/2/

for(iT_ in 1:nT_){

  N_ <- N_0_ - D_                                                           #/3/
  B_t_ <- FUNC_B_t_(B_0_[iSimulation_], iT_, K_[iSimulation_], TAU_[iSimulation_])          #/
4/

  dS_ <- - B_t_ * S_ * I_  * dT_ / N_                                       #/5/
  dE_ <- (B_t_ * S_ * I_ * dT_ / N_) - (SIGMA_[iSimulation_] * E_ * dT_)    #/6/
  dI_ = (SIGMA_[iSimulation_] * E_ * dT_) - (GAMMA_[iSimulation_] * I_ * dT_)  #/7/
  dR_ <- (1 - F_[iSimulation_]) * GAMMA_[iSimulation_] * I_ * dT_           #/8/
  dC_ <- SIGMA_[iSimulation_] * E_ * dT_                                    #/9/
  dD_ <- F_[iSimulation_] * GAMMA_[iSimulation_] * I_ * dT_                 #/10/

  S_ <- S_ + dS_                                          #/11/
  E_ <- E_ + dE_                                          #/12/
  I_ <- I_ + dI_                                          #/13/
  R_ <- R_ + dR_                                          #/14/
  C_ <- C_ + dC_                                          #/15/
  D_ <- D_ + dD_                                          #/16/

  R_e_ <- (B_t_ * S_) / (GAMMA_[iSimulation_] * N_)       #/17/


  # Load the Data Frame for this Time Step
  df_simulation_[iT_,'T']   <- iT_                        #/18/
  df_simulation_[iT_,'B_t'] <- B_t_                       #/19/

  df_simulation_[iT_,'dS']  <- dS_                        #/20/
  df_simulation_[iT_,'dE']  <- dE_                        #/21/
  df_simulation_[iT_,'dI']  <- dI_                        #/22/
  df_simulation_[iT_,'dR']  <- dR_                        #/23/
  df_simulation_[iT_,'dC']  <- dC_                        #/24/
  df_simulation_[iT_,'dD']  <- dD_                        #/25/

  df_simulation_[iT_,'S']   <- S_                         #/26/
  df_simulation_[iT_,'E']   <- E_                         #/27/
  df_simulation_[iT_,'I']   <- I_                         #/28/
  df_simulation_[iT_,'R']   <- R_                         #/29/
  df_simulation_[iT_,'C']   <- C_                         #/30/
  df_simulation_[iT_,'D']   <- D_                         #/31/

  df_simulation_[iT_,'R_e'] <- R_e_                       #/32/


}
```

```
df_monte_carlo_[iSimulation_,'MAPE_C'] <- mean(abs(df_simulation_[which_is_not_na_,'C'] - cases_
g_[which_is_not_na_])/df_simulation_[which_is_not_na_,'C'])
df_monte_carlo_[iSimulation_,'MAPE_D'] <- mean(abs(df_simulation_[which_is_not_na_,'D'] - deaths
_g_[which_is_not_na_])/df_simulation_[which_is_not_na_,'D'])


} #for


# Save the Simulation data frame as a comma-separated file (CSV) to disk for analysis
file_ <- '..\\csv\\df_monte_carlo_covid_out.csv'
write.csv(x=df_monte_carlo_,file=file_,row.names=FALSE)


# For illustrative purposes, show this plot for small N of say 100, but the actual simulation wi
ll be run for say 1,000,000 iterations
if(nSimulations_ <= 100){
  x_ <- seq(1:nSimulations_)
  main_ <- 'SEIR Monte Carlo Simulation of 2020 COVID United States '
  xlab_ <- 'Iteration'
  ylab_ <- 'Mean Absolute Percentage Error'
  ymax_ <- max(df_monte_carlo_[1:100,"MAPE_C"])
  plot(x_, df_monte_carlo_[,"MAPE_C"], type="l", pch=20, col='red', main=main_, xlab=xlab_, ylab
=ylab_)
  points(x_,df_monte_carlo_[,"MAPE_D"], type="p", pch=20,col='black')
  legend(0, 0.75 * ymax_, legend=c("Cases", "Deaths"),col=c("red", "black"), lty=1:1, cex=0.8)
} else {

  x_ <- seq(1:100)
  main_ <- 'SEIR Monte Carlo Simulation of 2020 COVID United States '
  xlab_ <- 'Iteration'
  ylab_ <- 'Mean Absolute Percentage Error'
  ymax_ <- max(df_monte_carlo_[1:100,"MAPE_C"])
  plot(x_, df_monte_carlo_[1:100,"MAPE_C"], type="l", pch=20, col='red', main=main_, xlab=xlab_,
ylab=ylab_)
  points(x_,df_monte_carlo_[1:100,"MAPE_D"], type="p", pch=20,col='black')
  legend(0, 0.75 * ymax_, legend=c("Cases", "Deaths"),col=c("red", "black"), lty=1:1, cex=0.8)

}
```



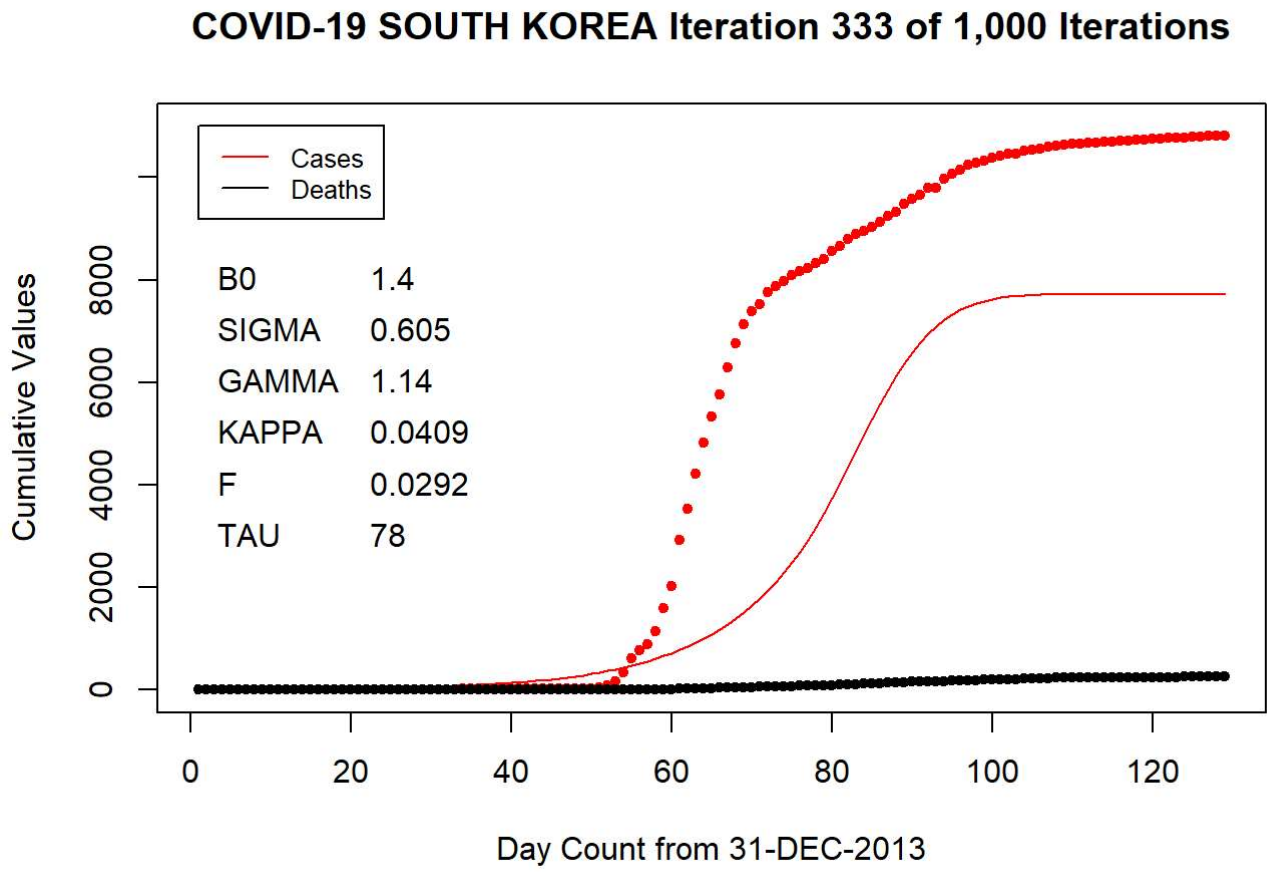
This code block lets us study a specific iteration. We are looking for a few Monte Carlo iterations that have a low Mean Absolute Percentage Error (MAPE) for both Deaths and Cases and a Fatality Rate that is close to the actual South Korean rate of about 250 deaths for 10,000 confirmed cases.

Let's start by looking at the simulation that had the first decile of Mean Absolute Percentage Error (MAPE) for Deaths.

```
# Observe data in [df_monte_carlo_covid_out.csv] file and enter Index here
#############################################################
min_mape_d_ <- min(df_monte_carlo_[,'MAPE_D'])

which_ <- which(df_monte_carlo_[,'MAPE_D']==min_mape_d_)  #<-- ENTER SIMULATION ITERATION INDEX
 HERE
#############################################################


# Intitalize the model parameters.

# There is no Multiphysics here. John von Neumann said "With 4 adjustable parameters,
# I can draw an elephant; with five, I can wiggle his trunk."  This is that.

# In other words, there is nothing fundamental about this model.
# It converges at the pleasure of the adjustable parameters.


N_0_ <- 10^7                          #/1/
N_ <- N_0_                            #/2/

B_0_ <- df_monte_carlo_[which_,"B0"]  # PER DAY             #/3/
B_t_ <- 0                                                  #/4/
SIGMA_ <- df_monte_carlo_[which_,'SIGMA']                  #/5/
GAMMA_ <- df_monte_carlo_[which_,'GAMMA']                  #/6/

K_ <- df_monte_carlo_[which_,"KAPPA"]                      #/7/


TAU_ <- df_monte_carlo_[which_,"TAU"]                      #/8/

# Initialize the SEIR variables proper
S_ <- N_0_                            #/9/
E_ <- 0                               #/10/
I_ <- 1                               #/11/
R_ <- 0                               #/12/
D_ <- 0                               #/13/
C_ <- 1                               #/14/
F_ <- df_monte_carlo_[which_,"F"]                          #/15/


nT_ <- length(deaths_g_)                                   #/1/
dT_ <- 1                                                   #/2/

for(iT_ in 1:nT_){

  N_ <- N_0_ - D_                                          #/3/
  B_t_ <- FUNC_B_t_(B_0_, iT_, K_, TAU_)                   #/4/

  dS_ <- - B_t_ * S_ * I_  * dT_ / N_                      #/5/
  dE_ <- (B_t_ * S_ * I_ * dT_ / N_) - (SIGMA_ * E_ * dT_)   #/6/
  dI_ = (SIGMA_ * E_ * dT_) - (GAMMA_ * I_ * dT_)          #/7/
  dR_ <- (1 - F_) * GAMMA_ * I_ * dT_                      #/8/
```

```
  dC_ <- SIGMA_ * E_ * dT_                                  #/9/
  dD_ <- F_ * GAMMA_ * I_ * dT_                             #/10/

  S_ <- S_ + dS_                                            #/11/
  E_ <- E_ + dE_                                            #/12/
  I_ <- I_ + dI_                                            #/13/
  R_ <- R_ + dR_                                            #/14/
  C_ <- C_ + dC_                                            #/15/
  D_ <- D_ + dD_                                            #/16/

  R_e_ <- (B_t_ * S_) / (GAMMA_ * N_)                       #/17/


  # Load the Data Frame for this Time Step
  df_simulation_[iT_,'T']   <- iT_                          #/18/
  df_simulation_[iT_,'B_t'] <- B_t_                         #/19/

  df_simulation_[iT_,'dS']  <- dS_                          #/20/
  df_simulation_[iT_,'dE']  <- dE_                          #/21/
  df_simulation_[iT_,'dI']  <- dI_                          #/22/
  df_simulation_[iT_,'dR']  <- dR_                          #/23/
  df_simulation_[iT_,'dC']  <- dC_                          #/24/
  df_simulation_[iT_,'dD']  <- dD_                          #/25/

  df_simulation_[iT_,'S']   <- S_                           #/26/
  df_simulation_[iT_,'E']   <- E_                           #/27/
  df_simulation_[iT_,'I']   <- I_                           #/28/
  df_simulation_[iT_,'R']   <- R_                           #/29/
  df_simulation_[iT_,'C']   <- C_                           #/30/
  df_simulation_[iT_,'D']   <- D_                           #/31/

  df_simulation_[iT_,'R_e'] <- R_e_                         #/32/


}

# ```
#
#
#
#
# ```{r}


#cases_g_ <- df_csv_[,'cases_g_']
#deaths_g_ <- df_csv_[,'deaths_g_']
#which_is_not_na_ <- which(!is.na(cases_g_))
x_ <- df_simulation_[,'T']
y1_ <- df_simulation_[,'C']
y2_ <- df_simulation_[,'D']
main_ <- paste(main='COVID-19 SOUTH KOREA Iteration',which_,'of',format(nSimulations_,scientific
=FALSE,big.mark=","),'Iterations', sep=' ')
plot(x_, y1_, ylim=c(0,11000),type='l', xlab='Day Count from 31-DEC-2013',ylab='Cumulative Value
s',col='red',main=main_)
```

```
points(x_, cases_g_, type="p", pch=20, col='red')
lines(x_,y2_,type='l')
points(x_,deaths_g_, type="p", pch=20,col='black')
legend(1, 11000, legend=c("Cases", "Deaths"),col=c("red", "black"), lty=1:1, cex=0.8)

text(1,8000,labels=c('B0'),pos=4)
text(20,8000,labels=c(format(B_0_,digits=3)),pos=4)

text(1,7000,labels=c('SIGMA'),pos=4)
text(20,7000,labels=c(format(SIGMA_,digits=3)),pos=4)

text(1,6000,labels=c('GAMMA'),pos=4)
text(20,6000,labels=c(format(GAMMA_,digits=3)),pos=4)

text(1,5000,labels=c('KAPPA'),pos=4)
text(20,5000,labels=c(format(K_,digits=3)),pos=4)

text(1,4000,labels=c('F'),pos=4)
text(20,4000,labels=c(format(F_,digits=3)),pos=4)

text(1,3000,labels=c('TAU'),pos=4)
text(20,3000,labels=c(format(TAU_,digits=3)),pos=4)
```



Note that other researchers have published R_0 values for COVID SARS-CoV-2 of ~ 2.65. Since $R_0 = \frac{\beta_0}{\gamma}$, our simulated R_0 does not agree. That work was done for China, and our data is from South Korea, which has superlative Public Health and legal infrastructure to combat contagion, so their measured R_0 is expected to be

less than that of China.

https://www.cebm.net/covid-19/when-will-it-be-over-an-introduction-to-viral-reproduction-numbers-r0-and-re/ (https://www.cebm.net/covid-19/when-will-it-be-over-an-introduction-to-viral-reproduction-numbers-r0-and-re/)

Now let's build a loop that interrogates the data frame for the minimum MAPE and prints the best say 15 charts. Once we have those, we will observe the Coupling Constants and Rates for consistency and determine whether they are grouping in neighborhoods that demonstrate some central tendency.

So we are simply going to take our Viewer in the code block immediately above and wrap it in a loop to make our print engine. Let's sort the data frame and write out the 10 best iterations. They are all finding about the right Fatality Rate and they all have rough agreement on Transmission Rates, Durations of Incubation and Infectiousness, Exponential Decay Constant, and the Time of Implementation of Public Health Control Intervention Measures….

```
df_monte_carlo_['MAPE_Total'] <- df_monte_carlo_['MAPE_C'] + df_monte_carlo_['MAPE_D']

df_sorted_ <- df_monte_carlo_[order(df_monte_carlo_$MAPE_Total),]

df_sorted_[1:10,]
```

	Iteration	B0	SIGMA	GAMMA	KAPPA	…	F	MAPE_C	MAPE_D
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
986	986	1.421367	0.7789994	1.128329	0.03925102	61	0.02442217	0.3740355	0.5591786
76	76	1.399648	0.7512566	1.108434	0.03706553	63	0.02509040	0.3648516	0.5951642
993	993	1.463110	0.7730707	1.207139	0.04037772	73	0.02310537	0.4978494	0.4757829
316	316	1.508274	0.6090840	1.204442	0.04368297	73	0.02310198	0.5124200	0.4683216
760	760	1.459989	0.7951735	1.164834	0.03866363	63	0.02080713	0.4160562	0.5692104
988	988	1.475398	0.6393929	1.152995	0.04109571	64	0.02996488	0.3941897	0.6112000
395	395	1.423345	0.6555930	1.104265	0.03852747	62	0.03055804	0.3717298	0.6356818
379	379	1.437145	0.6899519	1.142576	0.04074373	67	0.02912761	0.3905595	0.6222761
507	507	1.353401	0.7354515	1.079222	0.04307072	67	0.03035627	0.3910801	0.6314123
732	732	1.340612	0.7700454	1.105657	0.04002165	76	0.02333188	0.5886335	0.4387834

1-10 of 10 rows | 1-10 of 11 columns

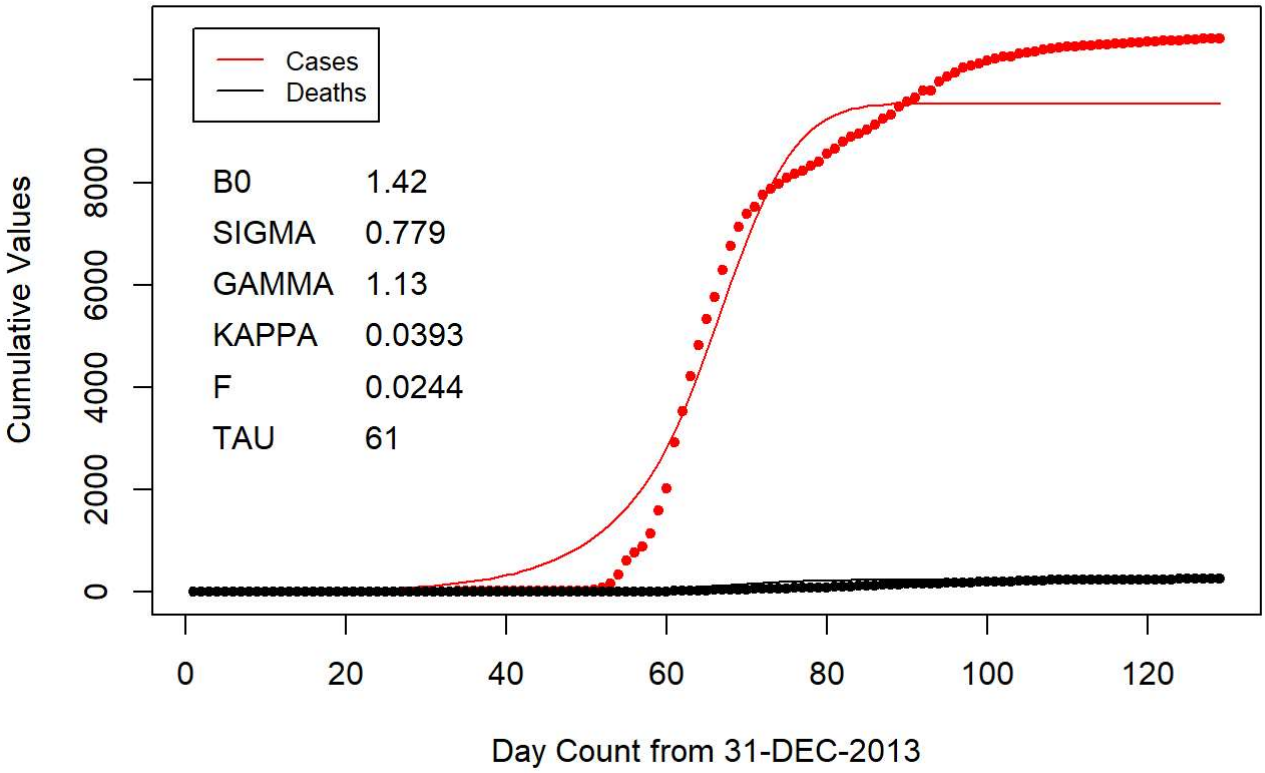
… and now let's view the plots. As always, we want to fit Deaths perfectly, then do the best we can with Cases. A Death is always well-documented, so we anchor to that trajectory, then we let the coupled differential equations of the SEIR model do the best it can with fitting the Cases, which are much less well defined from a Public Health and epidemiology perspective.

```
for(iWhich_ in 1:10){

# Observe data in [df_monte_carlo_covid_out.csv] file and enter Index here
#############################################################
which_ <- df_sorted_[iWhich_,'Iteration']  #<-- ENTER SIMULATION ITERATION INDEX HERE
#############################################################


# Intitalize the model parameters.

# There is no Multiphysics here. John von Neumann said "With 4 adjustable parameters,
# I can draw an elephant; with five, I can wiggle his trunk."  This is that.

# In other words, there is nothing fundamental about this model.
# It converges at the pleasure of the adjustable parameters.


N_0_ <- 10^7                          #/1/
N_ <- N_0_                            #/2/

B_0_ <- df_monte_carlo_[which_,"B0"]  # PER DAY             #/3/
B_t_ <- 0                                                  #/4/
SIGMA_ <- df_monte_carlo_[which_,'SIGMA']                  #/5/
GAMMA_ <- df_monte_carlo_[which_,'GAMMA']                  #/6/

K_ <- df_monte_carlo_[which_,"KAPPA"]                      #/7/


TAU_ <- df_monte_carlo_[which_,"TAU"]                      #/8/

# Initialize the SEIR variables proper
S_ <- N_0_                            #/9/
E_ <- 0                               #/10/
I_ <- 1                               #/11/
R_ <- 0                               #/12/
D_ <- 0                               #/13/
C_ <- 1                               #/14/
F_ <- df_monte_carlo_[which_,"F"]                          #/15/


nT_ <- length(deaths_g_)                                   #/1/
dT_ <- 1                                                   #/2/

for(iT_ in 1:nT_){

  N_ <- N_0_ - D_                                          #/3/
  B_t_ <- FUNC_B_t_(B_0_, iT_, K_, TAU_)                   #/4/

  dS_ <- - B_t_ * S_ * I_  * dT_ / N_                      #/5/
  dE_ <- (B_t_ * S_ * I_ * dT_ / N_) - (SIGMA_ * E_ * dT_)   #/6/
  dI_ = (SIGMA_ * E_ * dT_) - (GAMMA_ * I_ * dT_)          #/7/
  dR_ <- (1 - F_) * GAMMA_ * I_ * dT_                      #/8/
  dC_ <- SIGMA_ * E_ * dT_                                 #/9/
```

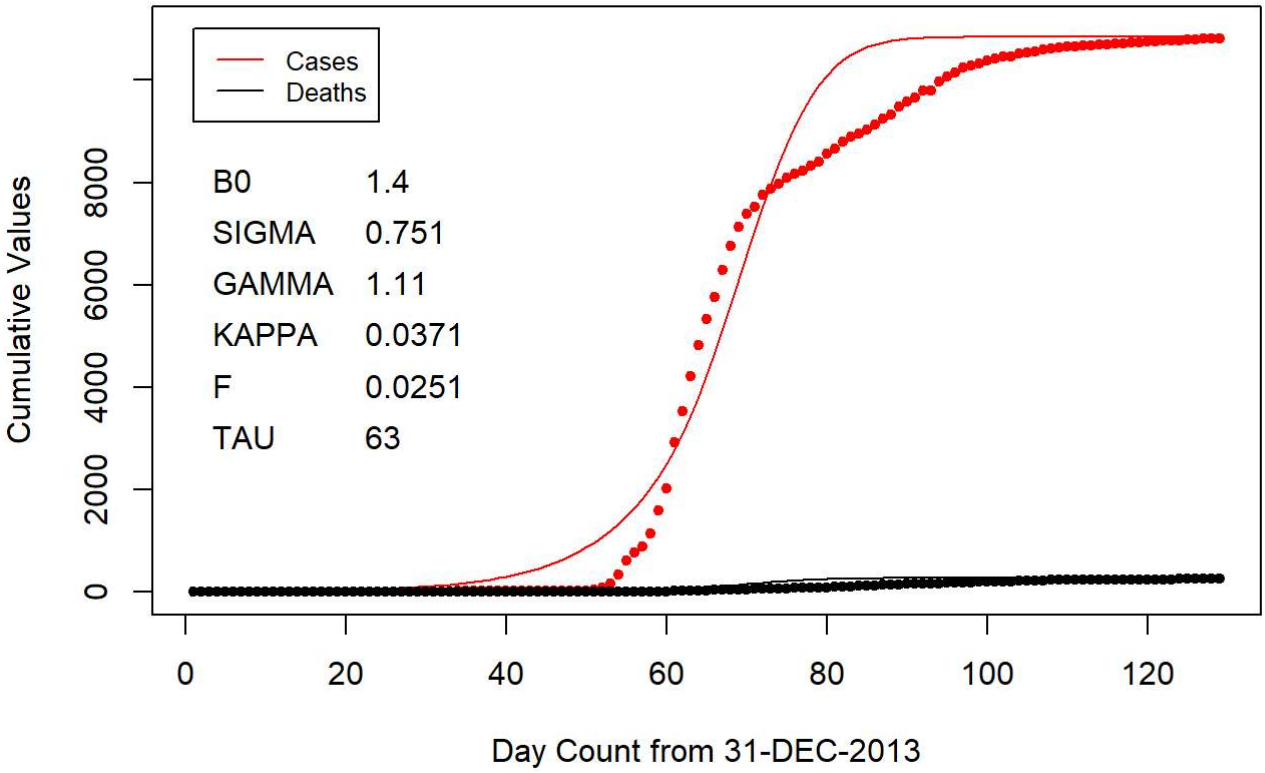
```
  dD_ <- F_ * GAMMA_ * I_ * dT_                                  #/10/

  S_ <- S_ + dS_                                                 #/11/
  E_ <- E_ + dE_                                                 #/12/
  I_ <- I_ + dI_                                                 #/13/
  R_ <- R_ + dR_                                                 #/14/
  C_ <- C_ + dC_                                                 #/15/
  D_ <- D_ + dD_                                                 #/16/

  R_e_ <- (B_t_ * S_) / (GAMMA_ * N_)                            #/17/


  # Load the Data Frame for this Time Step
  df_simulation_[iT_,'T']   <- iT_                               #/18/
  df_simulation_[iT_,'B_t'] <- B_t_                              #/19/

  df_simulation_[iT_,'dS']  <- dS_                               #/20/
  df_simulation_[iT_,'dE']  <- dE_                               #/21/
  df_simulation_[iT_,'dI']  <- dI_                               #/22/
  df_simulation_[iT_,'dR']  <- dR_                               #/23/
  df_simulation_[iT_,'dC']  <- dC_                               #/24/
  df_simulation_[iT_,'dD']  <- dD_                               #/25/

  df_simulation_[iT_,'S']   <- S_                                #/26/
  df_simulation_[iT_,'E']   <- E_                                #/27/
  df_simulation_[iT_,'I']   <- I_                                #/28/
  df_simulation_[iT_,'R']   <- R_                                #/29/
  df_simulation_[iT_,'C']   <- C_                                #/30/
  df_simulation_[iT_,'D']   <- D_                                #/31/

  df_simulation_[iT_,'R_e'] <- R_e_                              #/32/


}

# ```
#
#
#
#
# ```{r}


#cases_g_ <- df_csv_[,'cases_g_']
#deaths_g_ <- df_csv_[,'deaths_g_']
#which_is_not_na_ <- which(!is.na(cases_g_))
x_ <- df_simulation_[,'T']
y1_ <- df_simulation_[,'C']
y2_ <- df_simulation_[,'D']
main_ <- paste(main='COVID-19 SOUTH KOREA Iteration',which_,'of',format(nSimulations_,scientific
=FALSE,big.mark=","),'Iterations', sep=' ')
plot(x_, y1_, ylim=c(0,11000),type='l', xlab='Day Count from 31-DEC-2013',ylab='Cumulative Value
s',col='red',main=main_)
points(x_, cases_g_, type="p", pch=20, col='red')
```

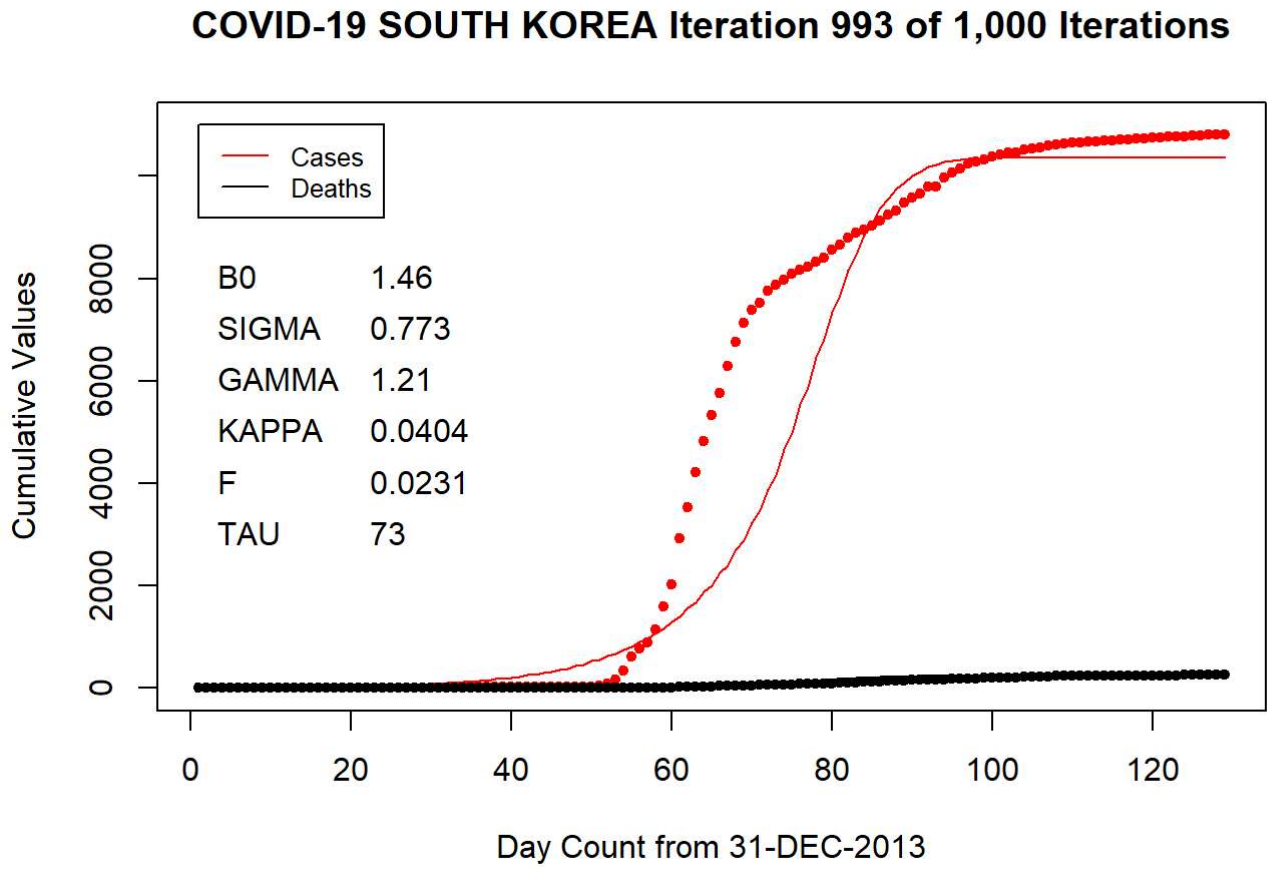
```
lines(x_,y2_,type='l')
points(x_,deaths_g_, type="p", pch=20,col='black')
legend(1, 11000, legend=c("Cases", "Deaths"),col=c("red", "black"), lty=1:1, cex=0.8)

text(1,8000,labels=c('B0'),pos=4)
text(20,8000,labels=c(format(B_0_,digits=3)),pos=4)

text(1,7000,labels=c('SIGMA'),pos=4)
text(20,7000,labels=c(format(SIGMA_,digits=3)),pos=4)

text(1,6000,labels=c('GAMMA'),pos=4)
text(20,6000,labels=c(format(GAMMA_,digits=3)),pos=4)

text(1,5000,labels=c('KAPPA'),pos=4)
text(20,5000,labels=c(format(K_,digits=3)),pos=4)

text(1,4000,labels=c('F'),pos=4)
text(20,4000,labels=c(format(F_,digits=3)),pos=4)

text(1,3000,labels=c('TAU'),pos=4)
text(20,3000,labels=c(format(TAU_,digits=3)),pos=4)

} #for
```

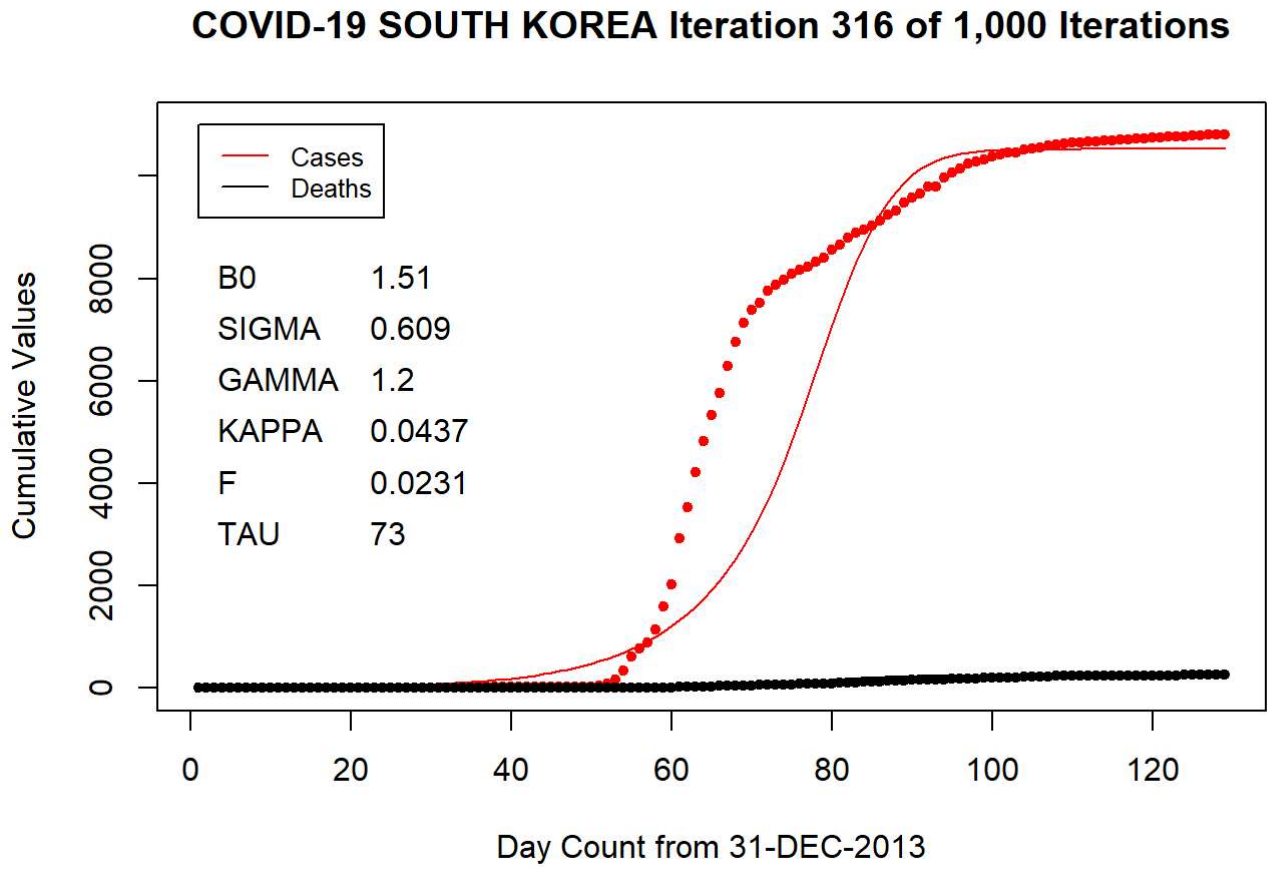
COVID-19 SOUTH KOREA Iteration 986 of 1,000 Iterations

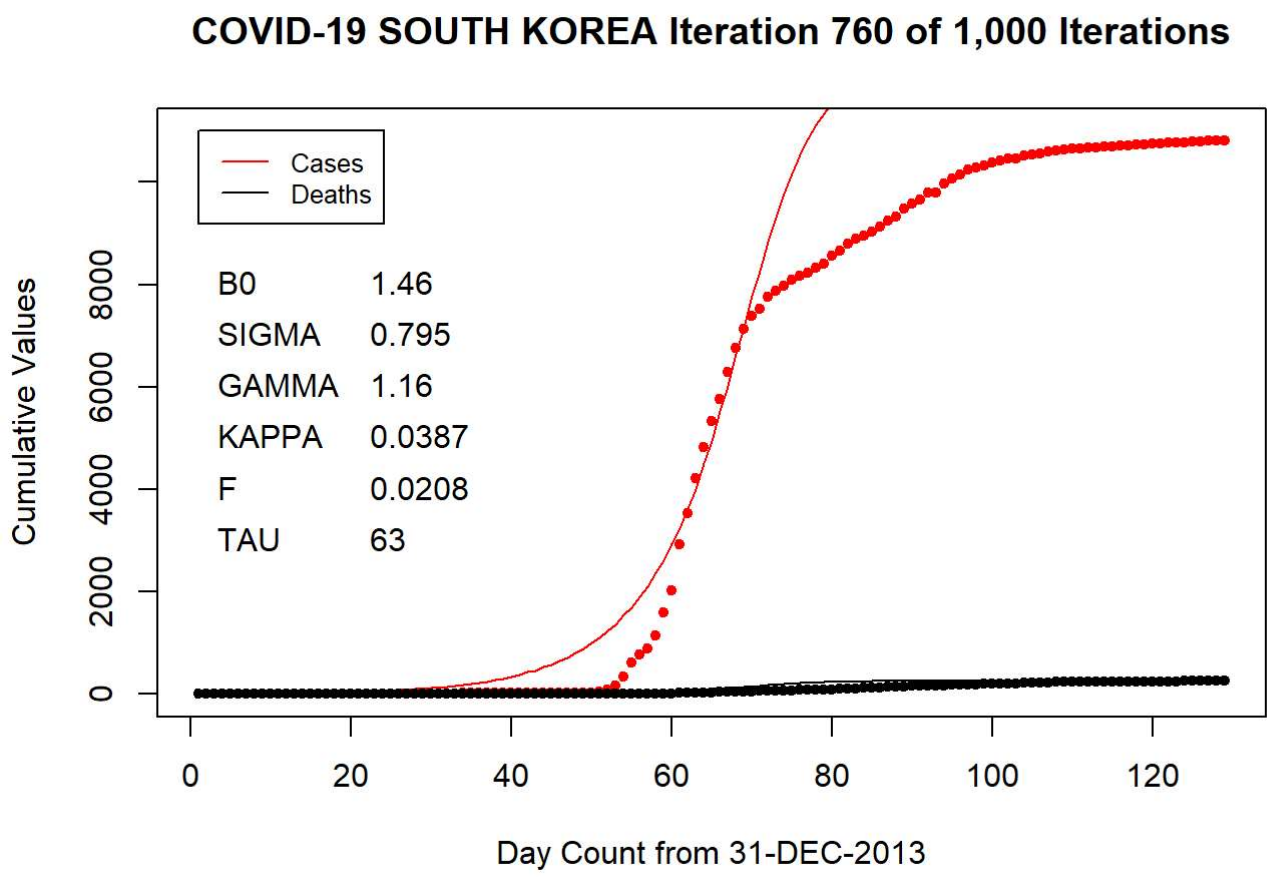


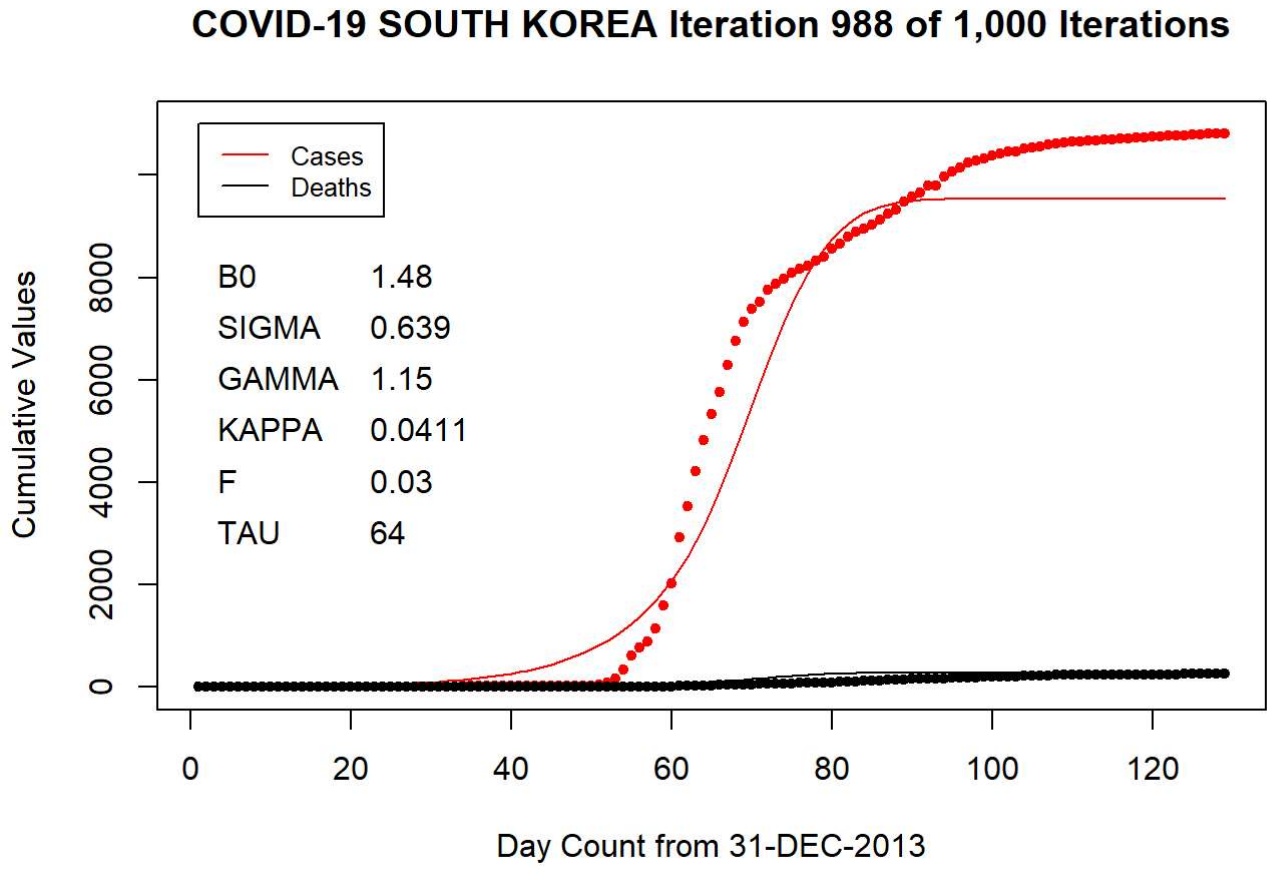
COVID-19 SOUTH KOREA Iteration 76 of 1,000 Iterations



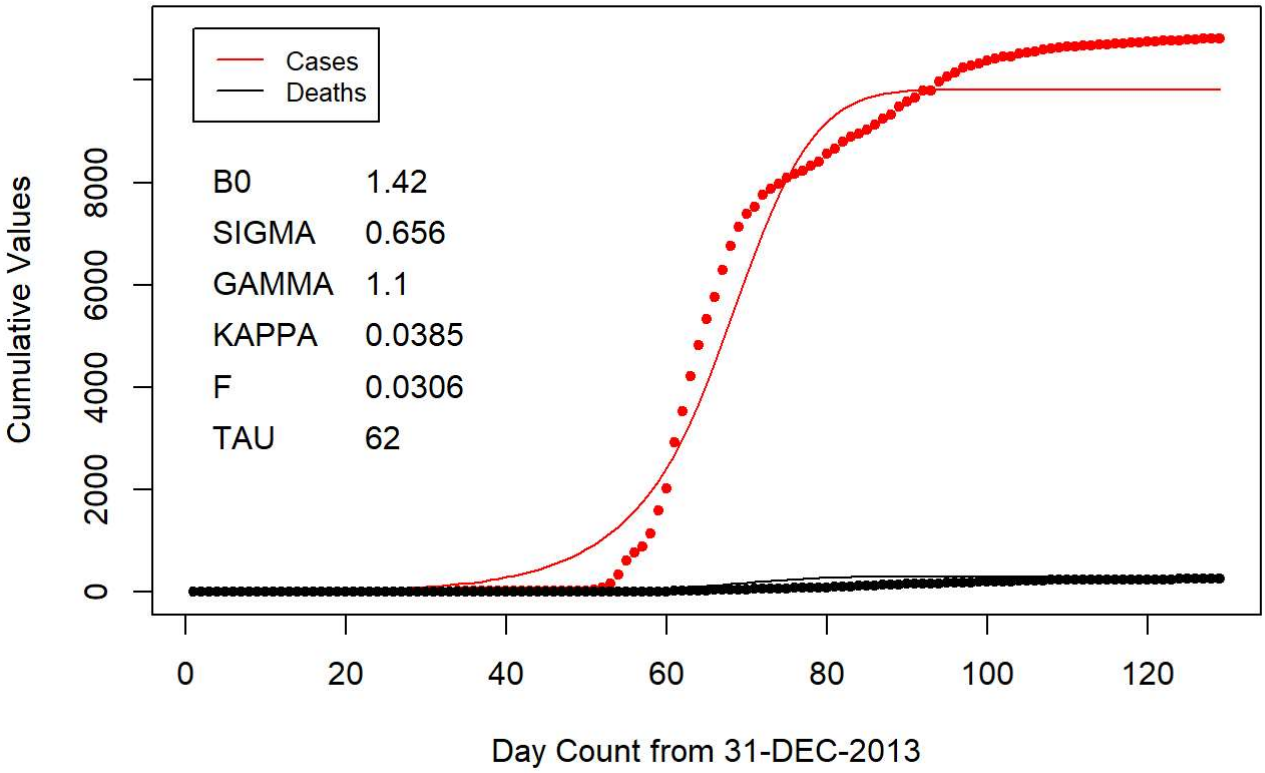

COVID-19 SOUTH KOREA Iteration 993 of 1,000 Iterations
Cases
Deaths
B0 1.46
SIGMA 0.773
GAMMA 1.21
KAPPA 0.0404
F 0.0231
TAU 73
Cumulative Values
0
2000
4000
6000
8000
0
20
40
60
80
100
120
Day Count from 31-DEC-2013


COVID-19 SOUTH KOREA Iteration 316 of 1,000 Iterations
Cases
Deaths
B0 1.51
SIGMA 0.609
GAMMA 1.2
KAPPA 0.0437
F 0.0231
TAU 73
Cumulative Values
0
2000
4000
6000
8000
0
20
40
60
80
100
120
Day Count from 31-DEC-2013

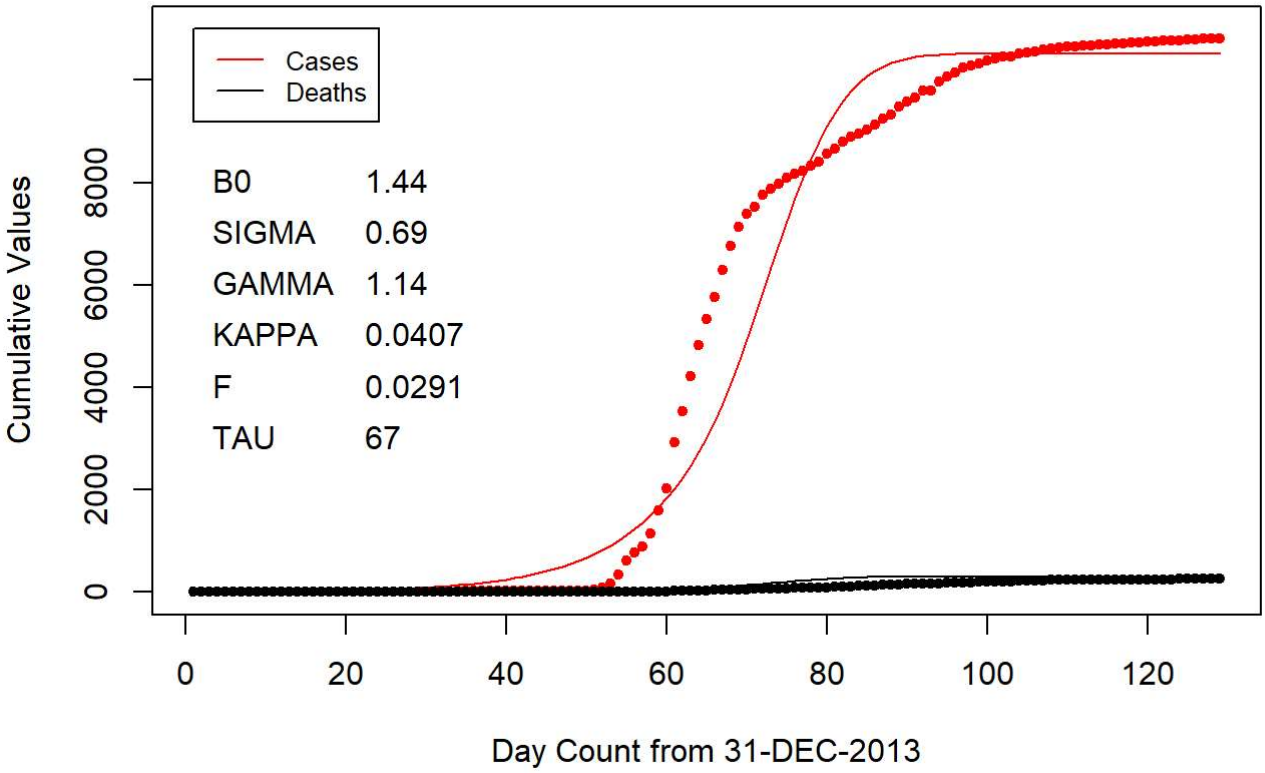

COVID-19 SOUTH KOREA Iteration 760 of 1,000 Iterations
Cases
Deaths
B0 1.46
SIGMA 0.795
GAMMA 1.16
KAPPA 0.0387
F 0.0208
TAU 63
Cumulative Values
0
2000
4000
6000
8000
0
20
40
60
80
100
120
Day Count from 31-DEC-2013

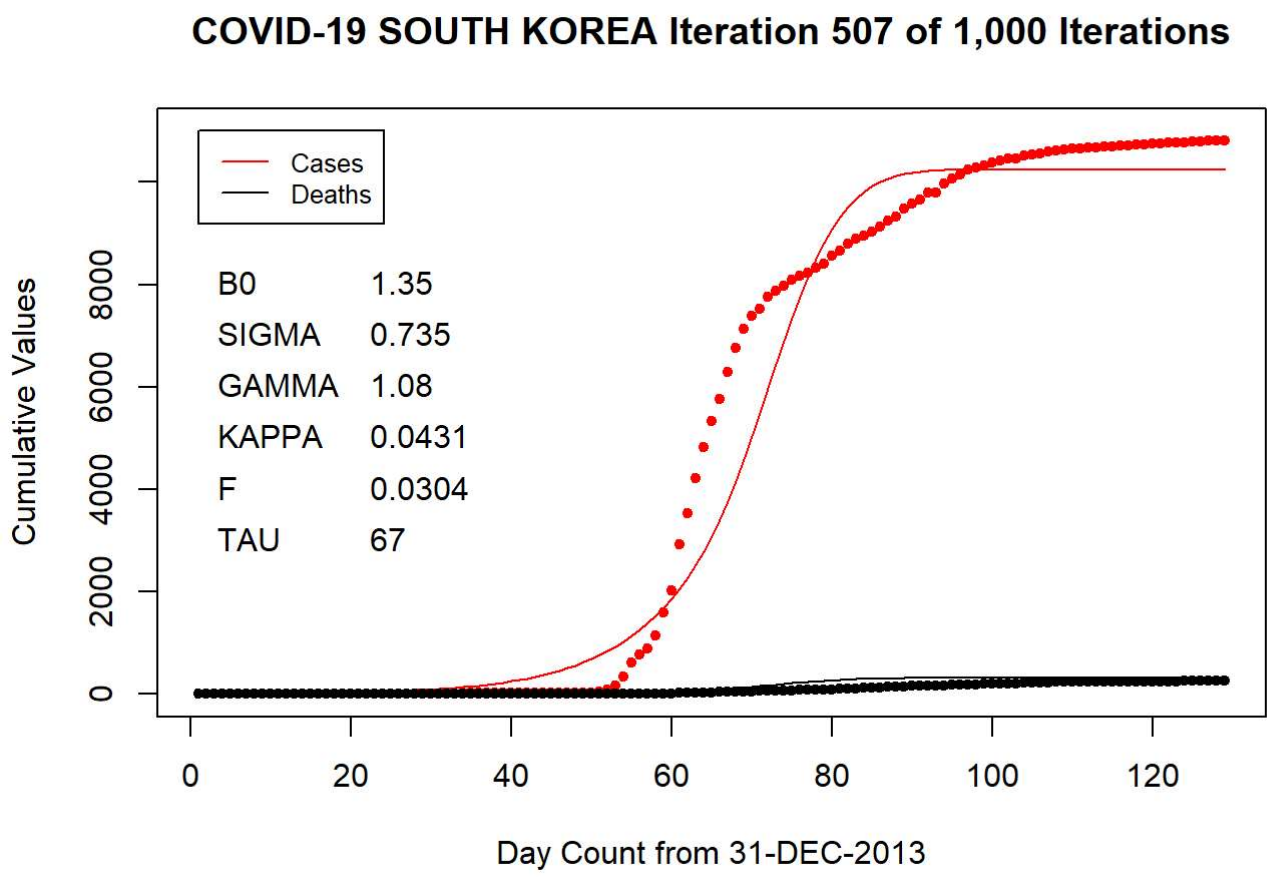

COVID-19 SOUTH KOREA Iteration 988 of 1,000 Iterations
Cases
Deaths
B0 1.48
SIGMA 0.639
GAMMA 1.15
KAPPA 0.0411
F 0.03
TAU 64
Cumulative Values
0
2000
4000
6000
8000
0
20
40
60
80
100
120
Day Count from 31-DEC-2013

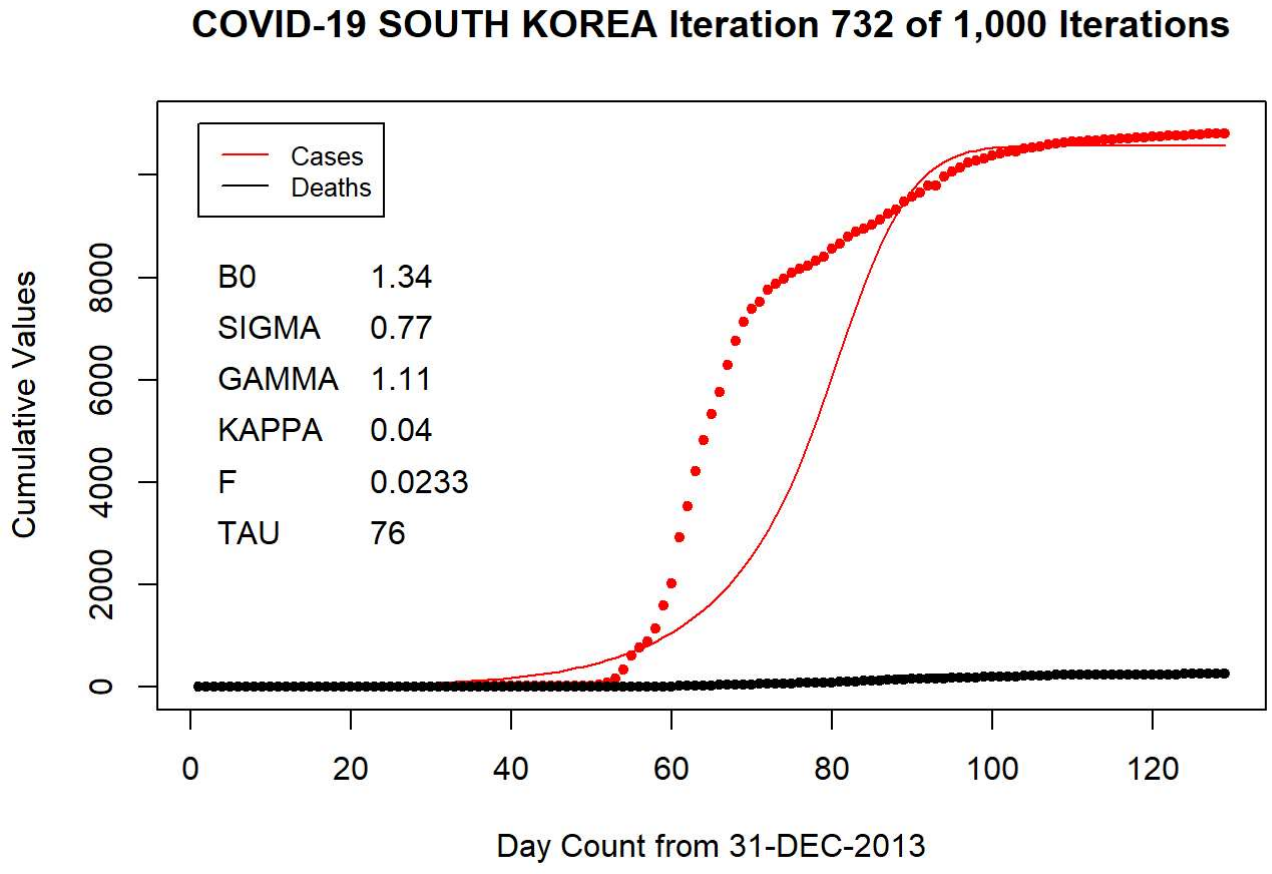
COVID-19 SOUTH KOREA Iteration 395 of 1,000 Iterations



COVID-19 SOUTH KOREA Iteration 379 of 1,000 Iterations




COVID-19 SOUTH KOREA Iteration 507 of 1,000 Iterations
Cases
Deaths
B0 1.35
SIGMA 0.735
GAMMA 1.08
KAPPA 0.0431
F 0.0304
TAU 67
Cumulative Values
0
2000
4000
6000
8000
0
20
40
60
80
100
120
Day Count from 31-DEC-2013


COVID-19 SOUTH KOREA Iteration 732 of 1,000 Iterations
Cases
Deaths
B0 1.34
SIGMA 0.77
GAMMA 1.11
KAPPA 0.04
F 0.0233
TAU 76
Cumulative Values
0
2000
4000
6000
8000
0
20
40
60
80
100
120
Day Count from 31-DEC-2013

Conclusions and Afterword

It is an open question as to whether infected subjects develop immunity to COVID-19. If they do, then the SEIR model should give us a good fit. The act of fitting SEIR involves bounding the possible values of the Coupling Constants and Rates, then choosing wisely so that the trajectories for Cases and Deaths match the Public Health records. A predictive-quality score, such as minimum Mean Absolute Percentage Error (MAPE), maximum likelihood, or minimum Mean Square Error (MSE) guides the way. A model with modest MAPE and a good fit to the historical Fatality Rate is a good candidate for inferring Coupling Constants and Rates, and a good number of such models should group nicely to give a small neighborhood around each parameter, which motivates further study.

If previously infected subjects do **not** develop Immunity, then the SEIR model must be discounted and the SIS model examined more fully. The procedure is the same.

This paper finds that the South Korean data can be modeled with modest success by a SEIR model.

In any case, this simple exercise in integrating coupled differential equations and fitting to Public Health data is only a modest start to establishing the Coupling Constants and Rates of the COVID-19 contagion in South Korea.

Note that these constants are a function of the Public Health Response to COVID-19, and would change from jurisdiction-to-jurisdiction.