Suitability of the SIR Model to the Macedonian COVID-19 Epidemic

Stefan Petrevski

14/04/2020

To date, very few epidemiological models have been applied to the Macedonian epidemic of the novel coronavirus. Through the media, the general public has been informed of two main prediction. The first is due to the team of the Macedonian health minister, Dr. Venko Filipce, who has publicly announced that linear growth (rather than exponential) is to be expected throughout the outbreak, with a peak of 2000-2500 infections sometime in the next few months. The second prediction was made by the Macedonian Academy of Sciences and Arts (MANU), through the paper of Dr. Ljupco Kocarev (http://manu.edu.mk/wp-content/uploads/2020/04/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B3-%D0%B1%D1%80%D0%BE%D1%98-1.pdf). This analysis, conducted with data as of April 8th, 2020, predicted a maximum of 1100 cases somewhere in May. However, by April 17th, 2020, this figure was already attained.

```
library(ggplot2)
library(pomp)
```

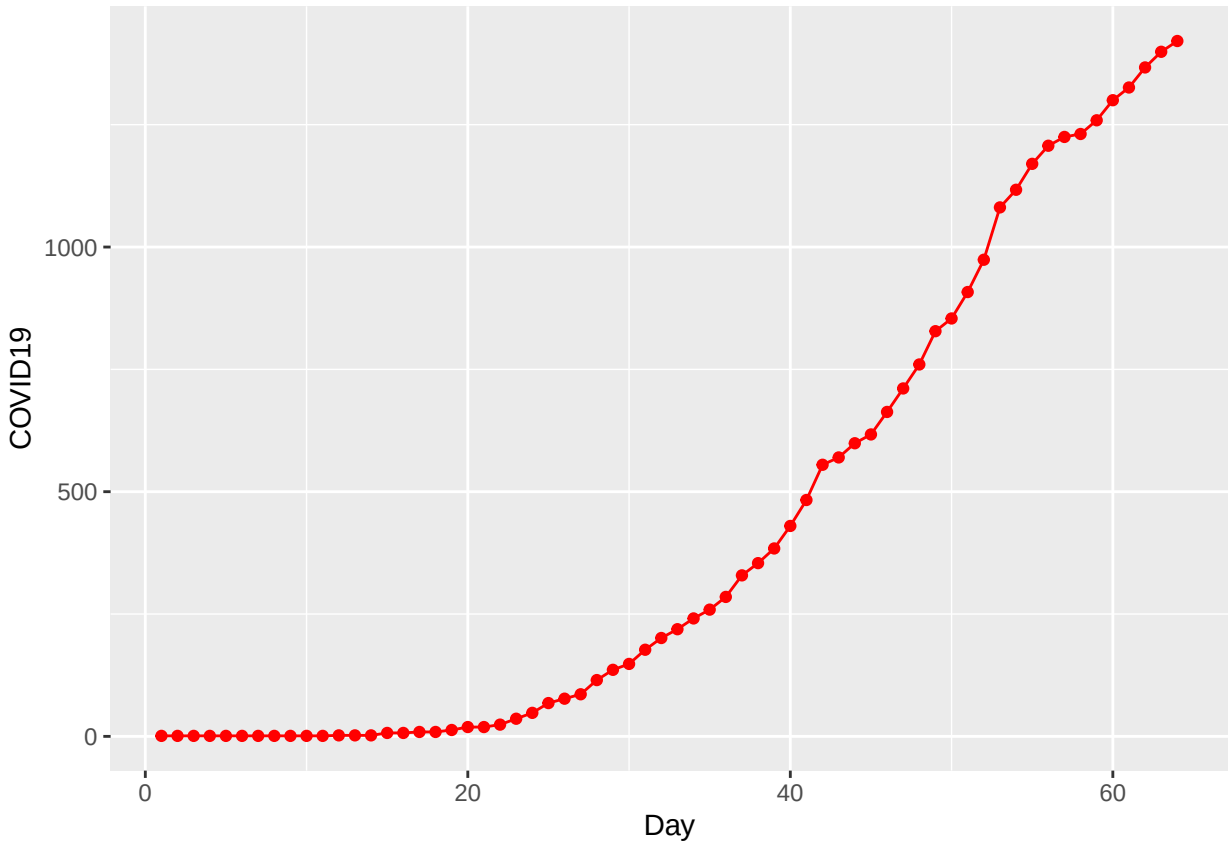
```
## Welcome to pomp!
## As of version 2.7.1.0, important changes have been made to the
## default settings of the particle filtering algorithms in
## 'pfilter', 'mif2', 'pmcmc', 'bsmc2'.
## These changes are not backward compatible.
## See the package NEWS for the details.
##
## For information on upgrading your pomp version < 2 code, see the
## "pomp version 2 upgrade guide" at https://kingaa.github.io/pomp/.
```

```
library(plyr)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:pomp':
##
##     melt
```

```
corona <- read.csv("macedoniadata.csv")
ggplot(corona,mapping=aes(x=Day,y=COVID19))+
  geom_line(color='red')+geom_point(color='red')
```



We will apply the SIR model to the epidemic. The reason for this is that it represents the simplest compartmental model - one which divides the population into separate groups which dynamically interact under a set of assumptions. The basic SIR model features three compartments: susceptible individuals (S), infected individuals (I), and indivuduals who are removed from the epidemic, either by death, immunity, or recovery (R). The usual main assumptions of the SIR model are as follows: 1. The total duration of the epidemic is short enough, so that the total number of individuals is constant. 2. Incubation periods are negligible. 3. Contracting the disease and then recovering results in immunity; it is impossible to contract the same disease twice. 4. The rate of new infections is proportional to the contacts between the infectives and susceptibles. 5. The rate of recovery/death of infectives occurs at a constant rate (i.e. there are no technological/healthcare breakthroughs or evolution of the disease that would increase/decrease this rate, respectively).

We set up the problem via the pomp package.

```
pomp(
  data=corona,
  times="Day",t0=0,
  skeleton=vectorfield(
    Csnippet("
      DS = -Beta*S*I/N;
      DI = Beta*S*I/N-gamma*I;
      DR = gamma*I;")),
  rinit=Csnippet("
      S = S_0;
      I = I_0;
      R = N-S_0-I_0;"),
  statenames=c("S","I","R"),
  paramnames=c("Beta","gamma","N","S_0","I_0")) -> coronaA
```

Next, we define a funcion which will output a square loss for our trajectory, given a some simulation of the model.

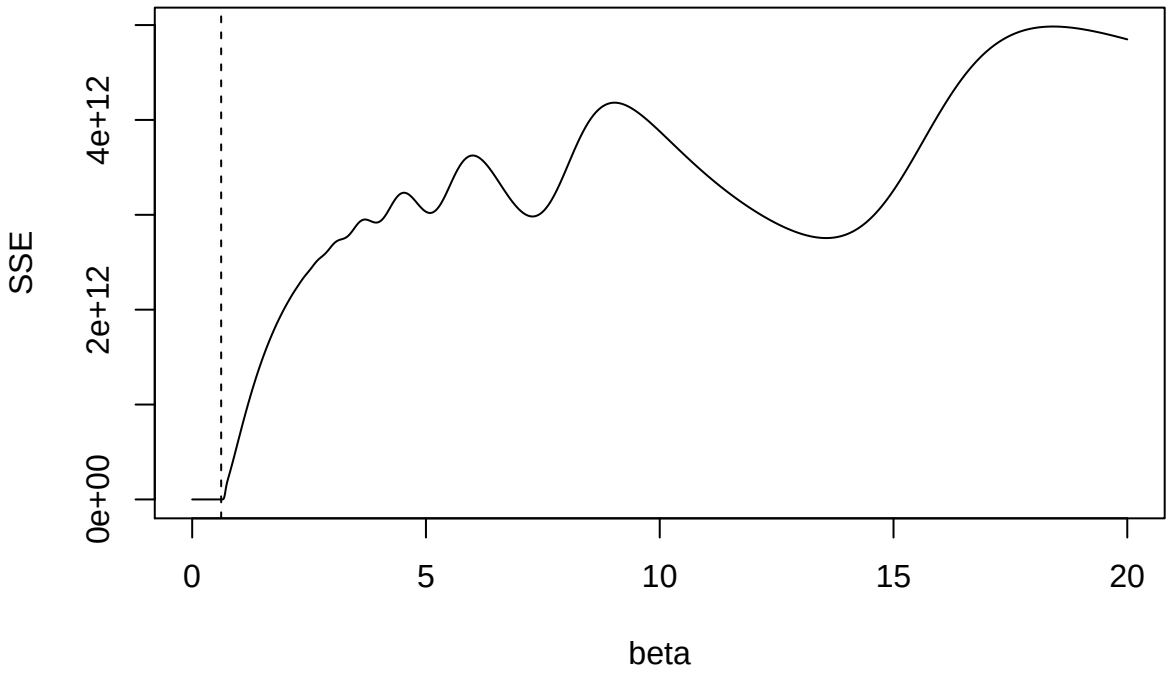
```
sse <- function (params) {
  x <- trajectory(coronaA,params=params)
  discrep <- x["I",,]-obs(coronaA)
  sum(discrep^2)
}
```

Naturally, we want to minimise this across all ranges of beta, assuming we know gamma. We also try to visualise this optimisation problem as a function of beta.

```
f1 <- function (beta) {
  params <- c(Beta=beta,gamma=0.5,N=2000000,S_0=1999999,I_0=1)
  sse(params)
}

beta <- seq(from=0,to=20,by=0.01)
SSE <- sapply(beta,f1)

plot(beta,SSE,type='l')
beta.hat <- beta[which.min(SSE)]
abline(v=beta.hat,lty=2)
```

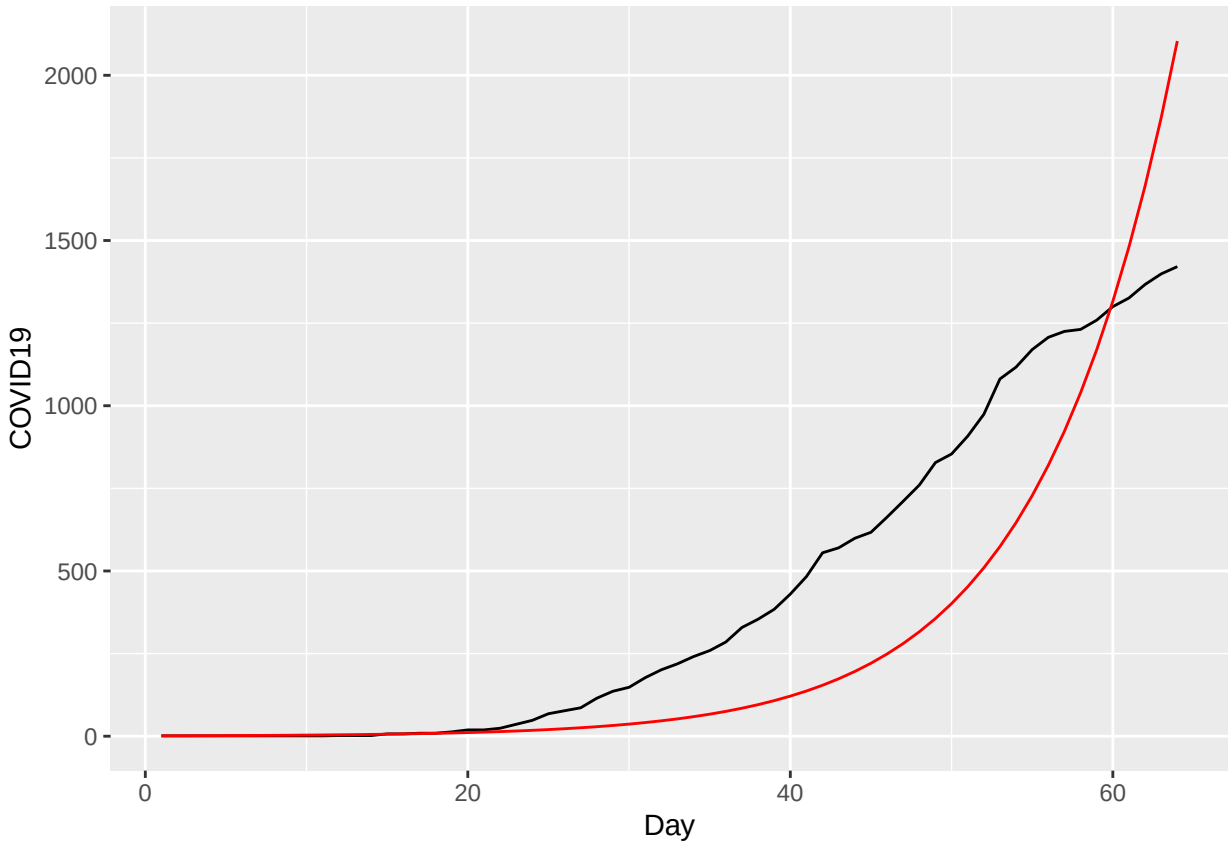


We are now ready to test the fit to our data.

```
beta.hat
```

```
## [1] 0.62
```

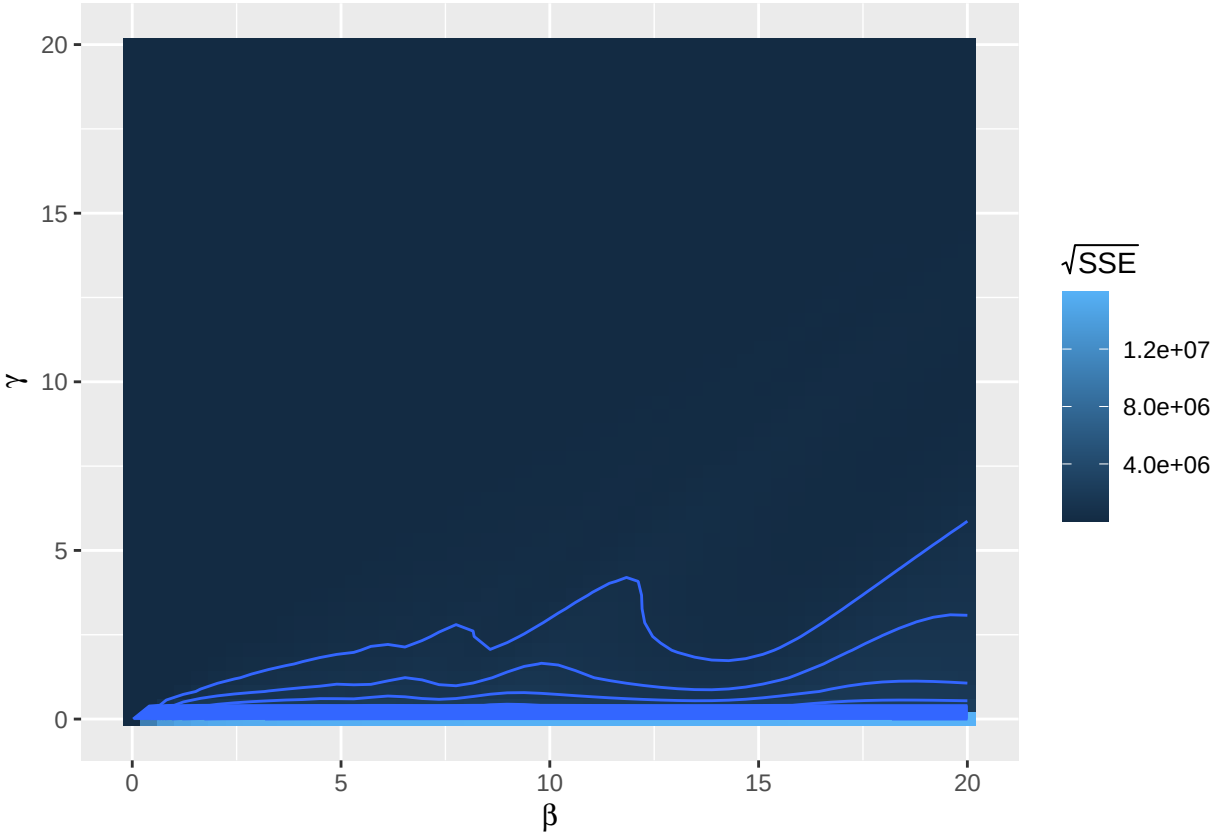
```
coef(coronaA) <- c(Beta=beta.hat,gamma=0.5,N=2000000,S_0=1999999,I_0=1)
x <- trajectory(coronaA,format="data.frame")
ggplot(data=join(as.data.frame(coronaA),x,by='Day'),
       mapping=aes(x=Day))+
  geom_line(aes(y=COVID19),color='black')+
  geom_line(aes(y=I),color='red')
```



As we can see, the fit is pretty disappointing - it systematically under/overestimates certain stages of the epidemic.

```
grid <- expand.grid(Beta=seq(from=0,to=20,length=50),
                    S_0=1999999,
                    N=2000000,gamma=seq(from=0,to=20,length=50),I_0=1)
x <- trajectory(coronaA,params=t(grid),format="data.frame")
library(plyr)
join(x,as.data.frame(coronaA),by="Day") -> x
ddply(x,~.id,summarize,sse=sum((COVID19-I)^2)) -> x
cbind(grid,x) -> grid

library(ggplot2)
ggplot(data=grid,mapping=aes(x=Beta,y=gamma,z=sqrt(sse),fill=sqrt(sse)))+
  geom_tile()+geom_contour(bins=30)+
  labs(fill=expression(sqrt(SSE)),x=expression(beta),y=expression(gamma))
```

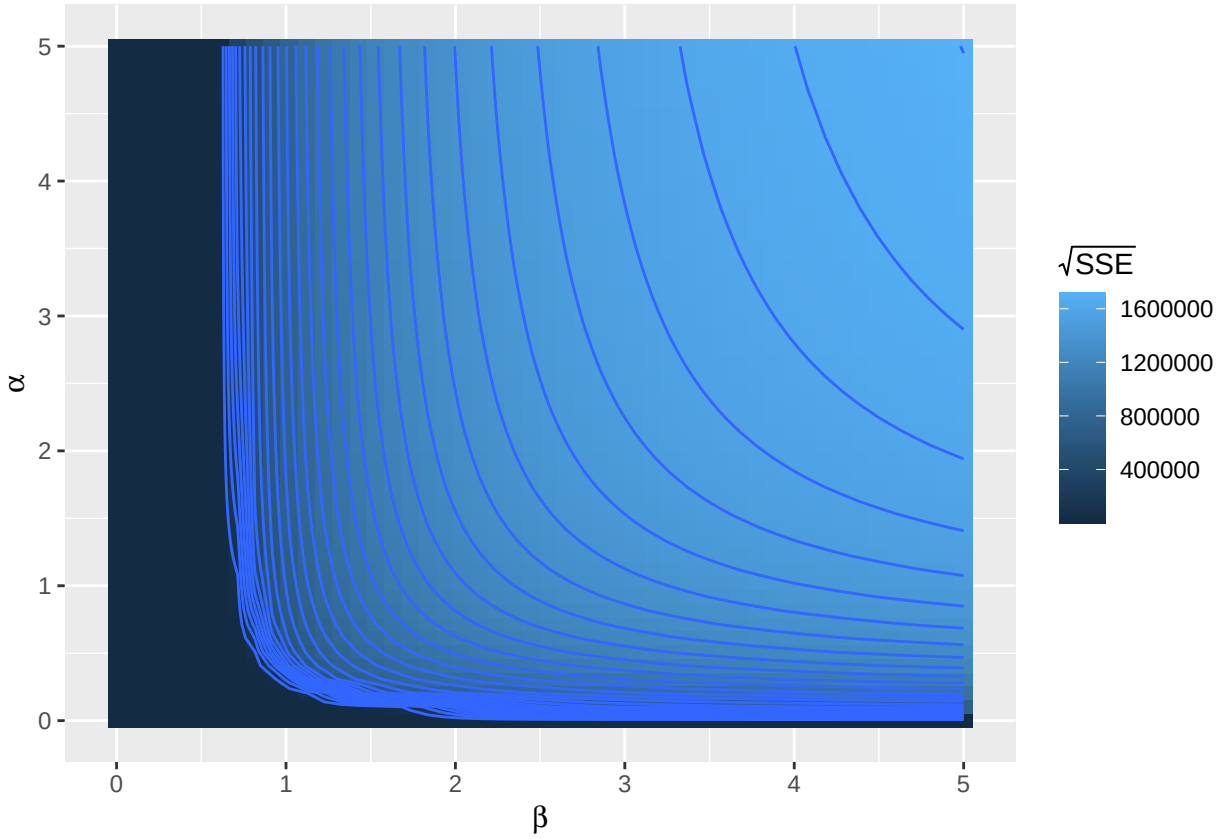


This is not surprising at all. We can see that the optimisation problem is non-convex, and is in fact very sensitive to initial parameter choices. Hence, it is no surprise that the performance of the model is so poor given our data.

```
pomp(
  data=corona,
  times="Day",t0=0,
  skeleton=vectorfield(
    Csnippet("
      DS = -Beta*S*I/N;
      DE = Beta*S*I/N - alpha*E;
      DI = alpha*E-gamma*I;
      DR = gamma*I;")),
  rinit=Csnippet("
      S = S_0;
      E = E_0;
      I = I_0;
      R = N-S_0-E_0-I_0;"),
  statenames=c("S","E","I","R"),
  paramnames=c("Beta","gamma", "alpha", "N","S_0","I_0", "E_0")) -> coronaB
```

```
gridB <- expand.grid(Beta=seq(from=0,to=5,length=50),
                     S_0=1999999, N=2000000, gamma=0.5, alpha=seq(from=0,to=5,length=50), E_0 =10, I_0=1
y <- trajectory(coronaB,params=t(gridB),format="data.frame")
library(plyr)
join(y,as.data.frame(coronaB),by="Day") -> y
ddply(y,~.id,summarize,sse=sum((COVID19-I)^2)) -> y
cbind(gridB,y) -> gridB
```

```
library(ggplot2)
ggplot(data=gridB,mapping=aes(x=Beta,y=alpha,z=sqrt(sse),fill=sqrt(sse)))+
  geom_tile()+geom_contour(bins=30)+
  labs(fill=expression(sqrt(SSE)),x=expression(beta),y=expression(alpha))
```




```
f2 <- function (par) {
  params <- c(Beta=par[1],alpha=par[2],gamma=0.5,N=2000000,S_0=1999999,I_0=1,E_0=10)
  sse(params)
}
optim(fn=f2,par=c(0.5,05)) -> fit2
fit2
```

```
## $par
## [1] 0.6198581 5.8035282
##
## $value
## [1] 4446719
##
## $counts
## function gradient
##       57       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```