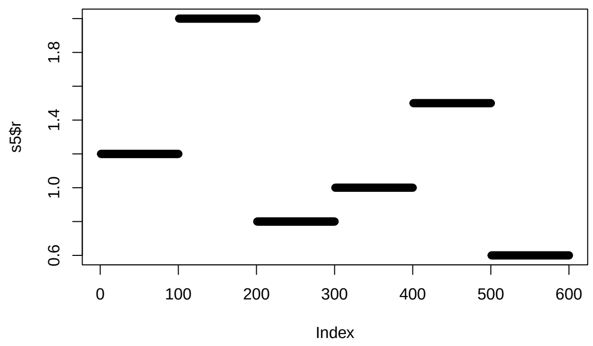
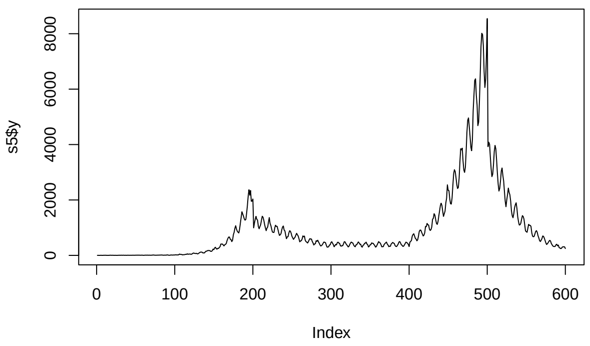
Oct27

2022-10-27

```
### Importing dataset ###
source("../function/get_iwt.R")
source("../function/disc_gamma.R")
source("../constant/constant.R")

s5 <- read.csv("../data/processed/e.csv")
s5_iwt <- get_iwt(s5$y, disc_gamma(1:nrow(s5), sid_ebola_shape, sid_ebola_scale))

plot(s5$r)
plot(s5$y, type = "l")
```





Previous work

Last time, I let the penalty to change as a function of the incidence cases for the following reasons:

1. (Primary) R_t estimation in lower case count areas (beginning and ending) fluctuates a lot. This is expected, but it forces a uniformly large penalty on the whole estimation.
2. For the synthetic dataset, I let the amplitude of the cycle to change as a function of the incidence case counts.

The smoothing function from last time is $F * (DR) \odot (DR)$, where $\odot$ is the hadamard product (element-wise product), where $F = f(W)$. Example used last time has $F = log(w + 1)$

```
r_smooth_c1 <- function(r, iwt, pen_func){
  return(sum((pen_func(iwt[1:length(iwt)-1]+1)*diff(r))^2))
}
```

The ridge objective function is

```
ridge_obj <- function(data, par, loss_func, iwt = iwt, smooth_func, penalties, pen_func = log){

  dat_length = nrow(data)

  loss = loss_func(z=data$y, iwt = iwt, r = par)

  r_pen <- penalties$r* smooth_func(par, iwt, pen_func)

  obj_value = sum(loss+r_pen)

  return(obj_value)
}
```

```
init_r = rep(1, nrow(s5))
```

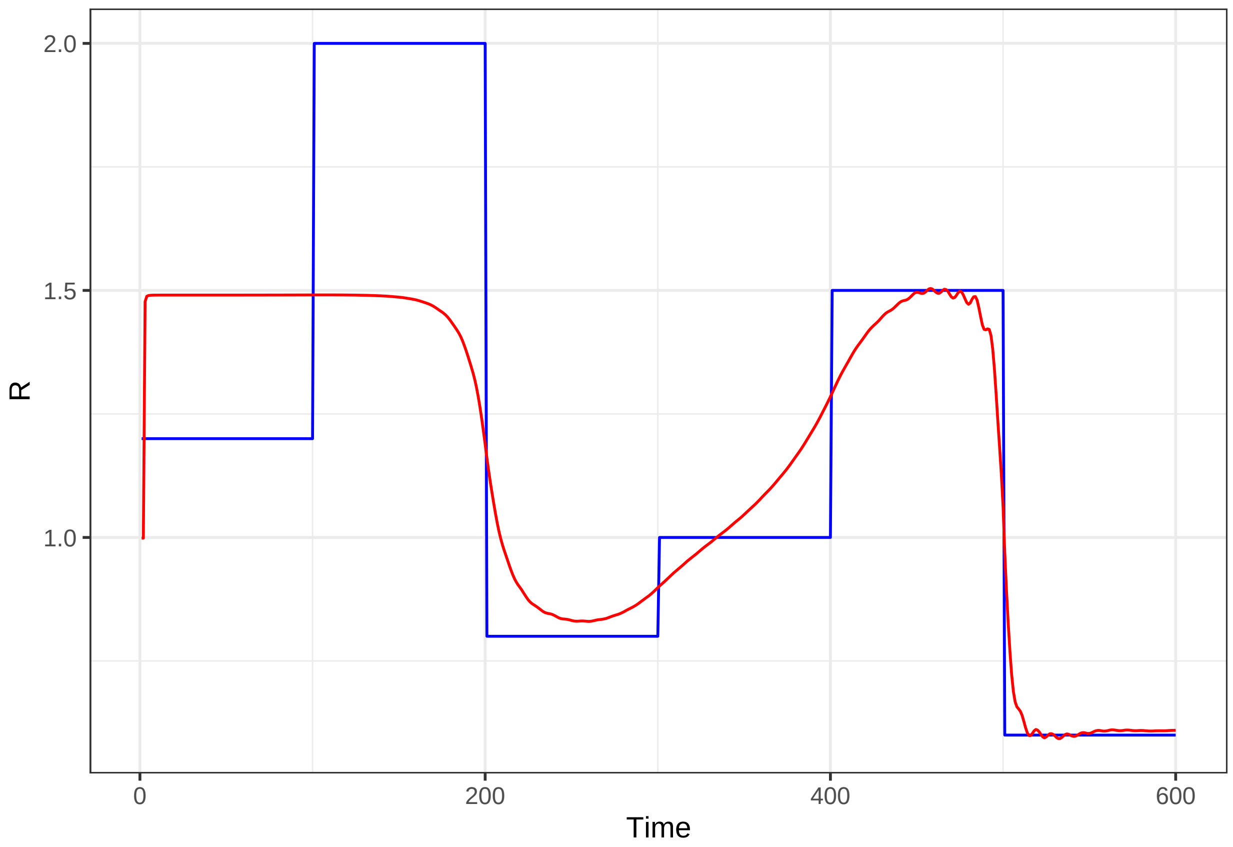
```
source("../model/pls/penalties_smooth.R")

result <- nlm(f=ridge_obj, p = init_r, iterlim =2000, print.level = 0, data=s5, penalties = list("r"=20
```

The result looks like

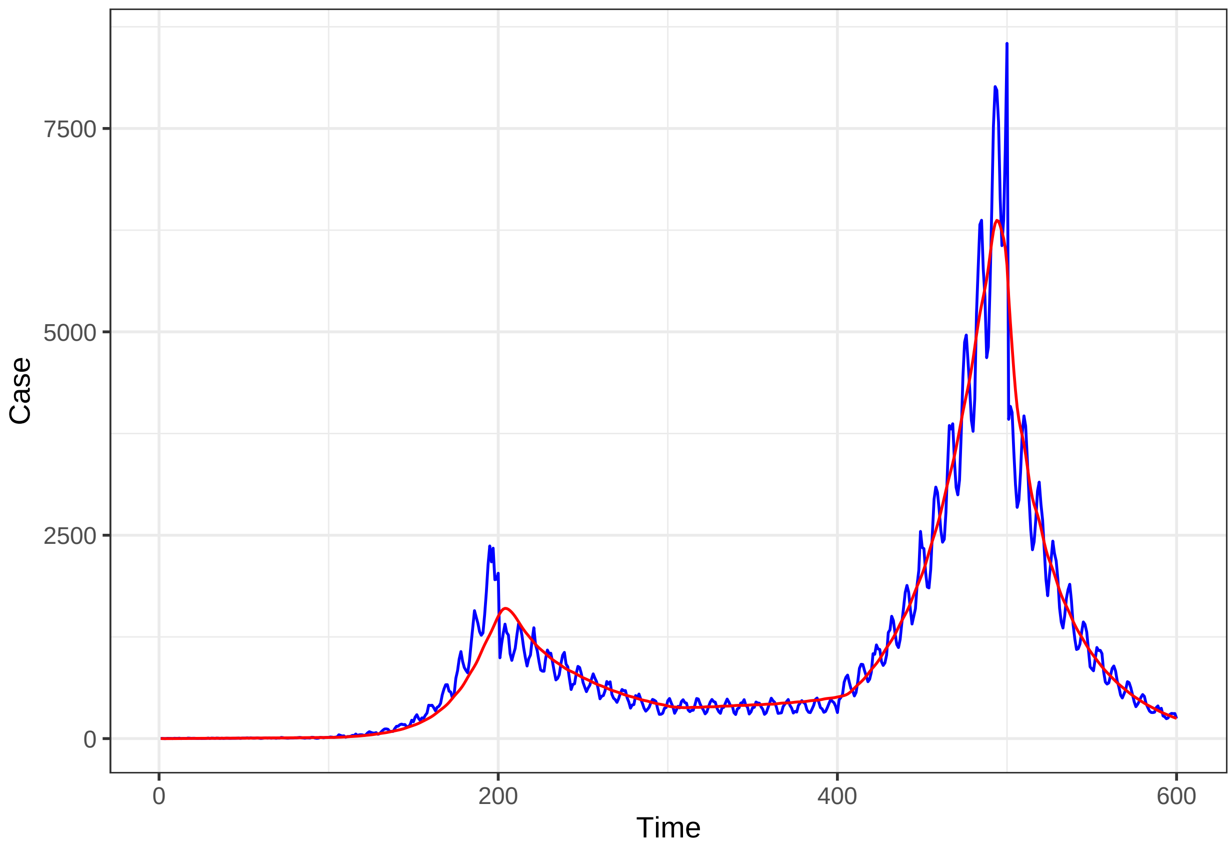
```
source("../function/make_plot.R")
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::rdunif() masks extraDistr::rdunif()
```

```
diags <- diag_plots(s5$r, result$estimate, s5_iwt, s5$y, cap=0)
diags$rt
```



```
diags$oneday
```

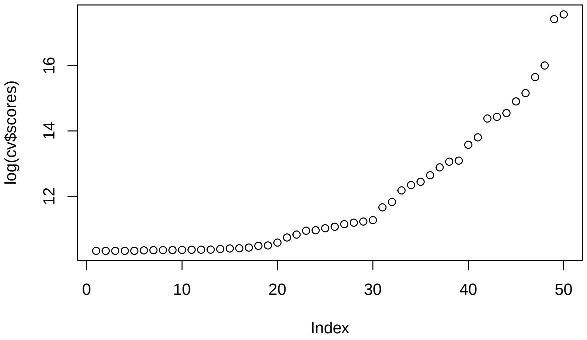


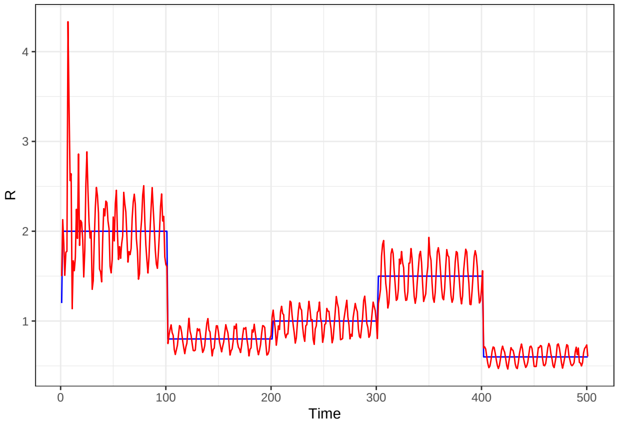
Comparing to the closed form solution:

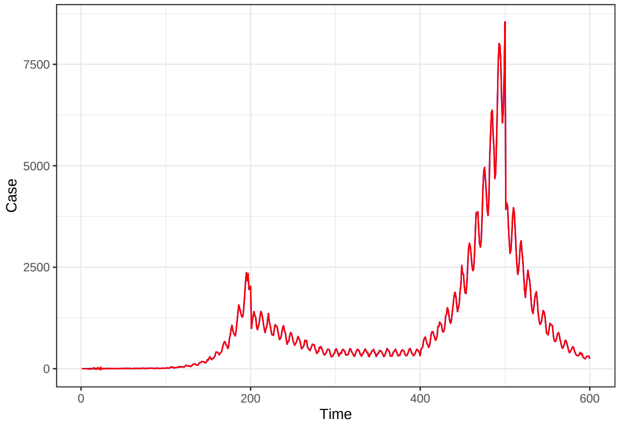
```
source("../model/pls/ridge.R")
source("../function/make_plot.R")
cv <- CV(s5_iwt, s5$y)
plot(log(cv$scores))
ridge_r <- get_r(s5_iwt, s5$y, lambda=cv$lambdas[1])

cv$lambdas[1]
```

```
## [1] 1.105171
```

```
diags_ridge <- diag_plots(s5$r, ridge_r, s5_iwt, s5$y, cap=100)
diags_ridge$rt
diags_ridge$oneday
```


log(cv$scores)
12
14
16
0
10
20
30
40
50
Index


R
1
2
3
4
0
100
200
300
400
500
Time


Case
0
2500
5000
7500
0
200
400
600
Time