Project 6 Summary

For this project, we explored the state-space SIR model in the GitHub. In the corresponding paper, the author talks about mainly two specific models with time varying transmission rate and quarantine and the MCMC algorithm that can be used to solve the above two ODES. Compared this model with previously models built, it's clear that this model does a better job at predicting the ending time and fitting with the real-life situation because it utilizes the Bayesian Inference when drawing samples that we only need to consider the likelihood function conditional on the unobserved variables, which implies that Bayesian parameter estimation is faster than classical maximum likelihood estimation

The basic SIR model is:

$$\frac{d\theta_t^S}{dt} = -\beta * \theta_t^S * \theta_t^I$$

$$\frac{d\theta_t^I}{dt} = \beta * \theta_t^S * \theta_t^I + \gamma * \theta_t^I$$

$$\frac{d\theta_t^R}{dt} = \gamma * \theta_t^I$$

To solve the parameters beta, gamma and lambda, we can use the Runge-Kutta to get the approximated value as following:

$$f(\boldsymbol{\theta}_{t-1},\beta,\gamma)=\begin{pmatrix}\theta^S_{t-1}+1/6[k^{S_1}_{t-1}+2k^{S_2}_{t-1}+2k^{S_3}_{t-1}+k^{S_4}_{t-1}]\\ \theta^I_{t-1}+1/6[k^{I_1}_{t-1}+2k^{I_2}_{t-1}+2k^{I_3}_{t-1}+k^{I_4}_{t-1}]\\ \theta^R_{t-1}+1/6[k^{R_1}_{t-1}+2k^{R_2}_{t-1}+2k^{R_3}_{t-1}+k^{R_4}_{t-1}]\end{pmatrix}:=\begin{pmatrix}\alpha_{1(t-1)}\\ \alpha_{2(t-1)}\\ \alpha_{3(t-1)}\end{pmatrix},$$

:e

$$\begin{aligned}k^{S_1}_t&=-\beta\theta^S_t\theta^I_t,\\ k^{S_2}_t&=-\beta[\theta^S_t+0.5k^{S_1}_t][\theta^I_t+0.5k^{I_1}_t],\\ k^{S_3}_t&=-\beta[\theta^S_t+0.5k^{S_2}_t][\theta^I_t+0.5k^{I_2}_t],\\ k^{S_4}_t&=-\beta[\theta^S_t+k^{S_3}_t][\theta^I_t+k^{I_3}_t];\end{aligned}$$

$$\begin{aligned}k^{I_1}_t&=\beta\theta^S_t\theta^I_t-\gamma\theta^I_t,\\ k^{I_2}_t&=\beta[\theta^S_t+0.5k^{S_1}_t][\theta^I_t+0.5k^{I_1}_t]-\gamma[\theta^I_t+0.5k^{I_1}_t],\\ k^{I_3}_t&=\beta[\theta^S_t+0.5k^{S_2}_t][\theta^I_t+0.5k^{I_2}_t]-\gamma[\theta^I_t+0.5k^{I_2}_t],\\ k^{I_4}_t&=\beta[\theta^S_t+k^{S_3}_t][\theta^I_t+k^{I_3}_t]-\gamma[\theta^I_t+k^{I_3}_t];\end{aligned}$$

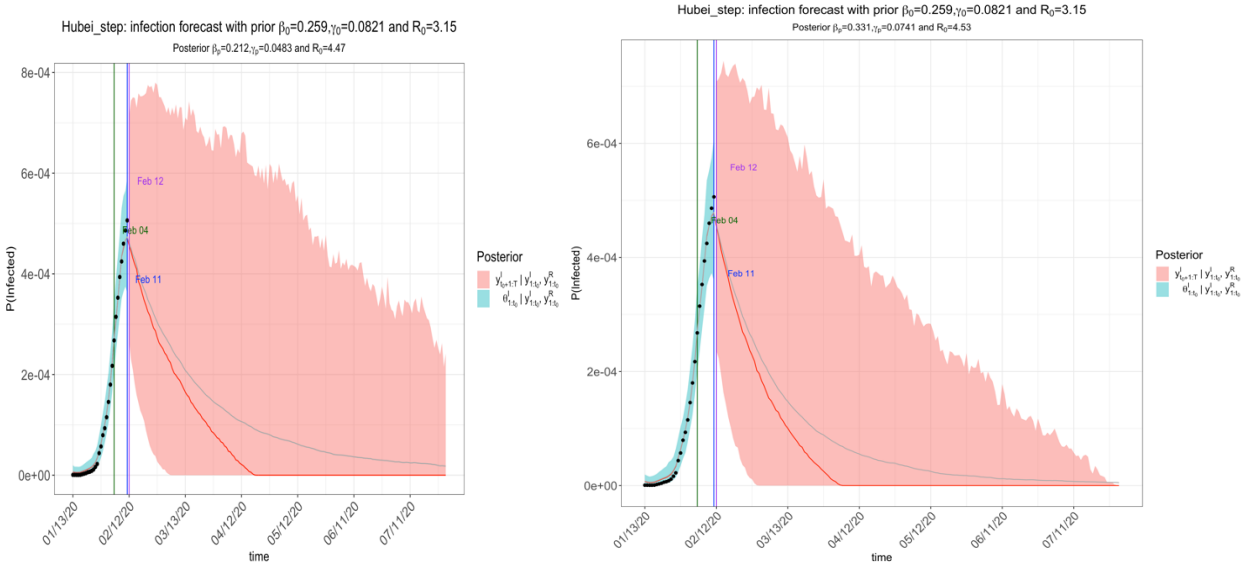
$$\begin{aligned}k^{R_1}_t&=\gamma\theta^I_t,\\ k^{R_2}_t&=\gamma[\theta^I_t+0.5k^{I_1}_t],\\ k^{R_3}_t&=\gamma[\theta^I_t+0.5k^{I_2}_t],\\ k^{R_4}_t&=\gamma[\theta^I_t+k^{I_3}_t].\end{aligned}$$

Coding experience

For the tvt.eSIR model, it does a good job on depicting a series of time-varying changes caused by either external variations like government policies, protective measures and environment changes, or internal variations like mutations and evolutions of the pathogen. However, when compared with the real situation, it's still not as accurate as it's expected to be, and it can only fit well for a short fraction of period. To fix this problem, Implementation of delay effect would be a choice. Here is what I did:

$$\theta^I_t\to\theta^I_{t-5}$$

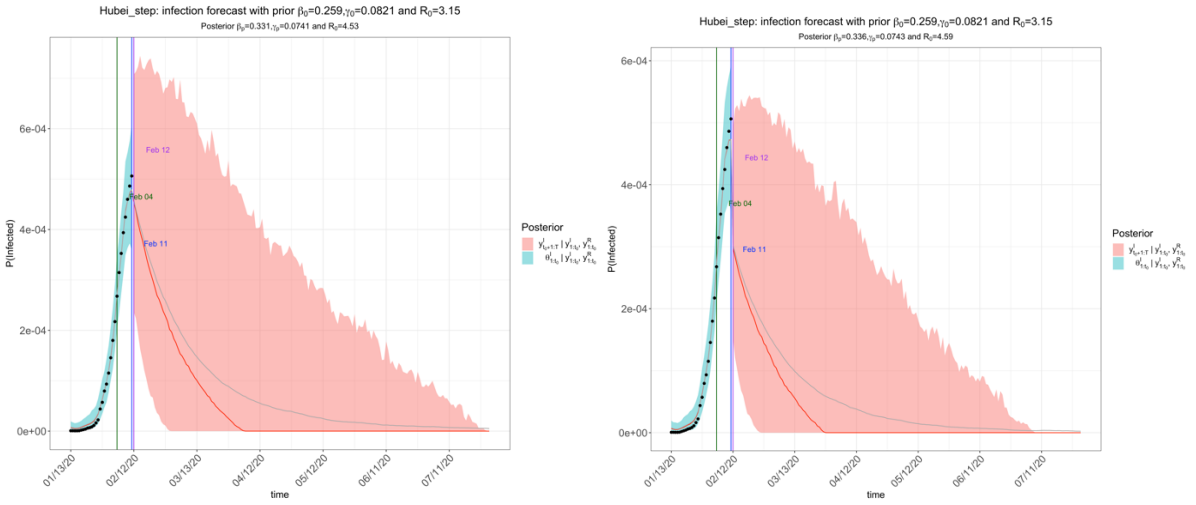
```
################ MCMC ##########
model1.string <- paste0("
model{
   for(t in 2:(T_prime+1)){
     Km[t-1,1] <- -beta*pi[t-1]*theta[t-1,1]*ifelse(t-5<1,theta[1,2],theta[max(t-5,1),2])
     Km[t-1,9] <- gamma*ifelse(t-5<1,theta[1,2],theta[max(t-5,1),2])
     Km[t-1,5] <- -Km[t-1,1]-Km[t-1,9]

     Km[t-1,2] <- -beta*pi[t-1]*(theta[t-1,1]+0.5*Km[t-1,1])*(ifelse(t-5<1,theta[1,2],theta[max(t-5,1),2])+0.5*Km[t-1,5])
     Km[t-1,10] <- gamma*(ifelse(t-5<1,theta[1,2],theta[max(t-5,1),2])+0.5*Km[t-1,5])
     Km[t-1,6] <- -Km[t-1,2]-Km[t-1,10]

     Km[t-1,3] <- -beta*pi[t-1]*(theta[t-1,1]+0.5*Km[t-1,2])*(theta[t-1,2]+0.5*Km[t-1,6])
     Km[t-1,11] <- gamma*(theta[t-1,2]+0.5*Km[t-1,6])
     Km[t-1,7] <- -Km[t-1,3]-Km[t-1,11]

     Km[t-1,4] <- -beta*pi[t-1]*(theta[t-1,1]+Km[t-1,3])*(ifelse(t-5<1,theta[1,2],theta[max(t-5,1),2])+Km[t-1,7])
     Km[t-1,12] <- gamma*(ifelse(t-5<1,theta[1,2],theta[max(t-5,1),2])+Km[t-1,7])
     Km[t-1,8] <- -Km[t-1,4]-Km[t-1,12]
```



Compared the left graph (original one) with the right graph (new one), I found that after implementing the delay effect into the model, it decays to zero faster than before that it goes to

zero before 04/12/2020, which again proves that the delay effect improved the performance of the model. To further implement the delay effect, I also modified the theta2 in forecast part:

```
#### Forecast ####
theta_pp <- array(0,dim=c(len,T_fin-T_prime,3))
Y_pp <- matrix(NA,nrow=len,ncol=T_fin-T_prime)
R_pp <- matrix(NA,nrow=len,ncol=T_fin-T_prime)
for(l in 1:len){
  thetalt1 <- theta_p[l,T_prime+1,1]
  #thetalt2 <- theta_p[l,T_prime+1,2]
  thetalt2 <- ifelse(T_prime<6,theta_p[l,T_prime+1,2],theta_p[l,T_prime-5,2])
```



However, there isn't much change this time. One guess may be that since the model has already used the stochastic simulation, so there will not be much change even if we implement the delay effect in the model. We can conclude that implementing the delay effect does improve the accuracy of this model when fitting the real-life data, but the modified model is still far below the wanting result.

Discussion

The data that the tvt model used is from China and since the data itself doesn't have much oscillation, so no matter how perfect the model is, the result will not show strong oscillation pattern. Based on this idea, the next step to verify the accuracy and to test the performance of this model is to input the data from the US and compare it the real-life scenario. Also, I will try to use the Euler method to solve for the parameters.

Collaboration

This report is based on the research experience with conjoint efforts of Junhao Zhang, Luobin Wang and Dajun Xu.