Stochastic SIR Forecasting Showdown Part 2: Producing Forecasts

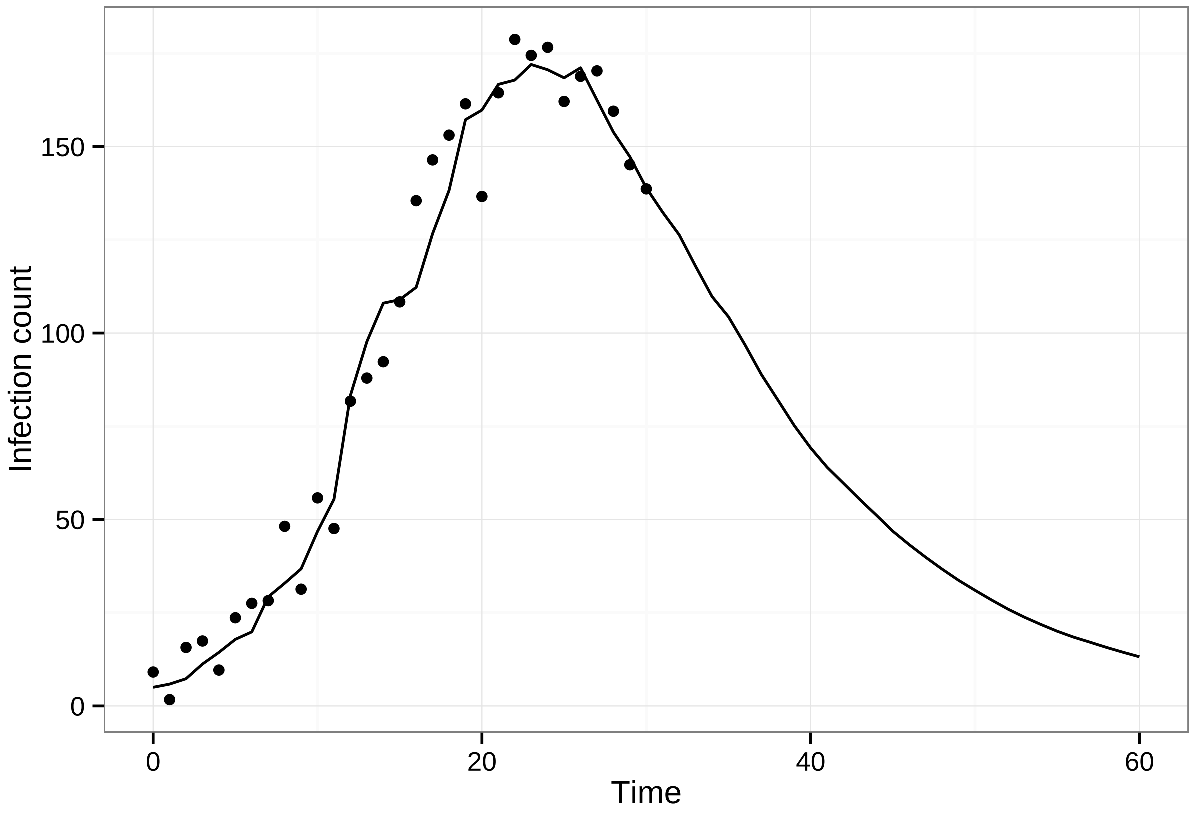
Dexter Barrows

15:31 15 December 2015

Forecasting Setup

This section will focus on taking the stochastic SIR model from the previous section, truncating the synthetic data output from realizations of that model, and seeing how well IF2 and HMC can reconstruct out-of-sample forecasts.

Below is an example of a simulated system with truncated data



As before, the solid line shows the true system states, while the dots show the data. In essence we want to be able to give either IF2 of HMC only the data points and have it reconstruct the entirety of the true system states.

IF2

For IF2, we will take advantage of the fact that the particle filter will produce state estimates for every datum in the time series given to it, as well as producing parameter MLE densities. Both of these sources of information will be used to produce forecasts by bootstrapping using the final system state estimates from the particle swarm after the last IF2 pass. We will take the swarm states as a starting point for the trajectories and simply extending their simulation forward to whatever point we wish to produce forecasts.

We will truncate the data at half the original time series length (to $T = 30$), and fit the model as previously described.

The total runtime for the fitting was determined using the R `system.time()` function, and yielded

```
##    user  system elapsed
##  15.570   0.096  15.744
```

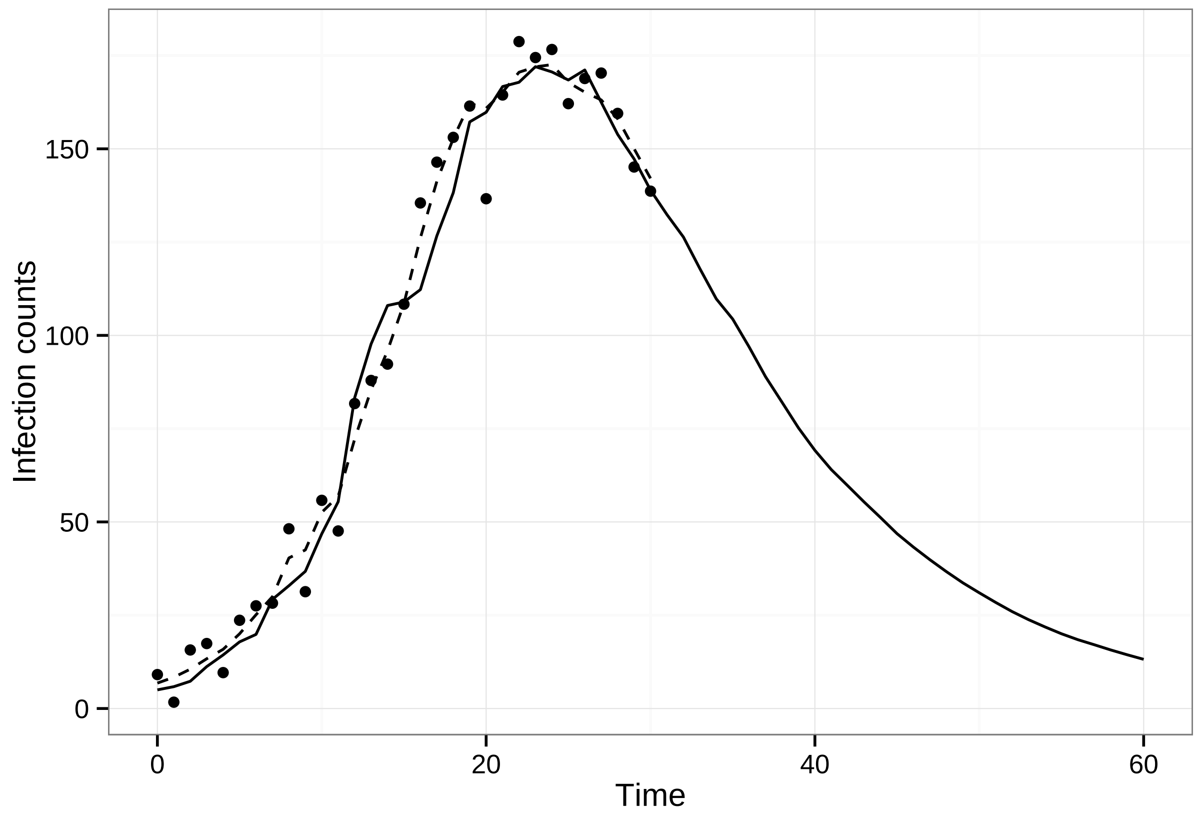
The MLE parameter estimates, taken to be the mean of the particle swarm values after the final pass was

```
##         R0          r         I0      sigma        eta       berr
## 3.30736456 0.09631091 6.68825260 8.58636107 0.69961379 0.12697226
```

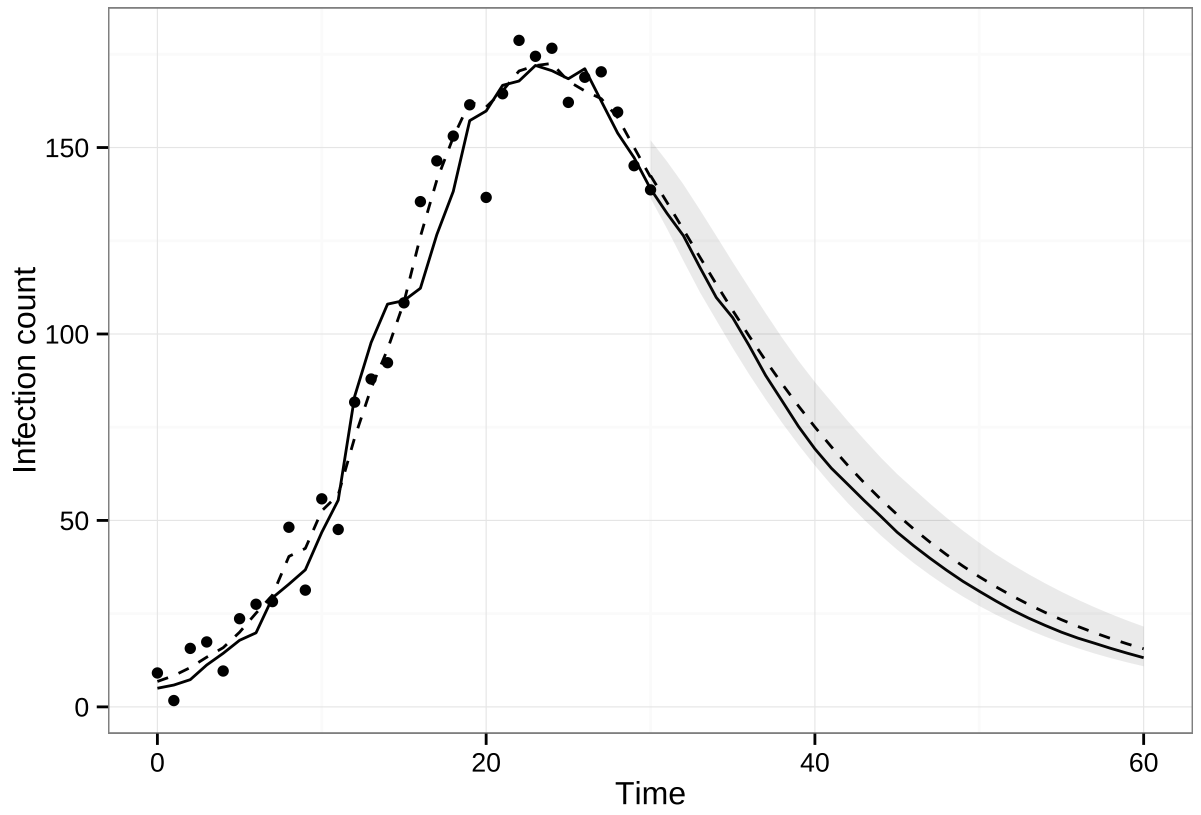
giving a relative error of

```
##          R0           r          I0       sigma         eta        berr
##  0.10245485 -0.03689088  0.33765052 -0.14136389  0.39922757 -0.74605548
```

From last IF2 particle filtering iteration, the mean state values from the particle swarm at each time step are shown with the true underlying state and data in the plot below.



Now by bootstrapping from 100 randomly drawn final particle swarm states, we obtain the following plot.



As before, the solid line shows the true system states,the dots show the data, and the dashed line before the data cut-off shows the particle filtering mean state estimates from the last IF2 pass. In addition the dashed line after the data cut-off shows the mean bootstrap values, and the grey ribbon shows the centre 95th quantile of the bootstrap trajectories.

We can define a metric to gauge forecast effectiveness by calculating the SSE. Here the value is

```
## [1] 170.8835
```

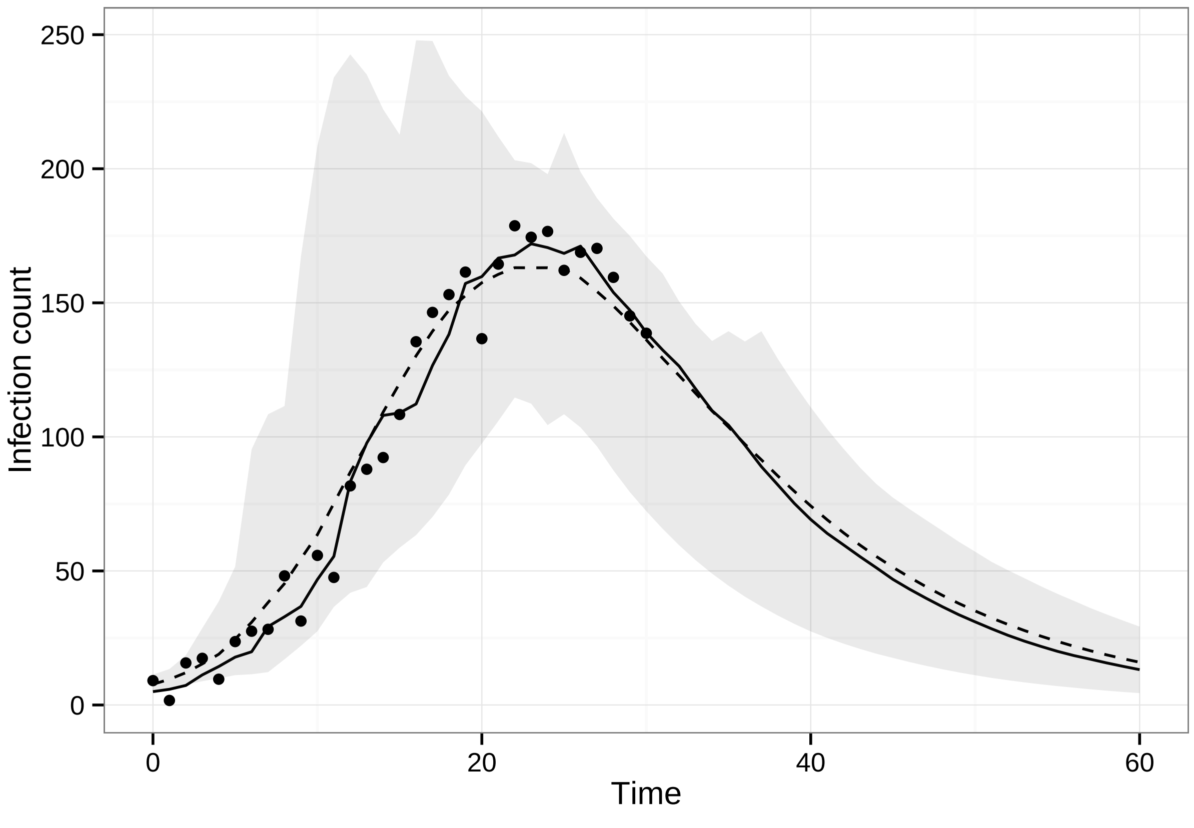
HMC

For HMC we cannot use final state estimates as in IF2 as we only have the parameter density estimates to work with. Instead we have to rely on a pure bootstrapping approach. As before we fit the stochastic SIR mode to the partial data.

The runtime retrieved again using R's `system.time()` shows

```
##    user  system elapsed
##  88.106   2.832  94.061
```

Now taking 100 samples from the HMC chain, we bootstrap them forward to obtain the following plot.



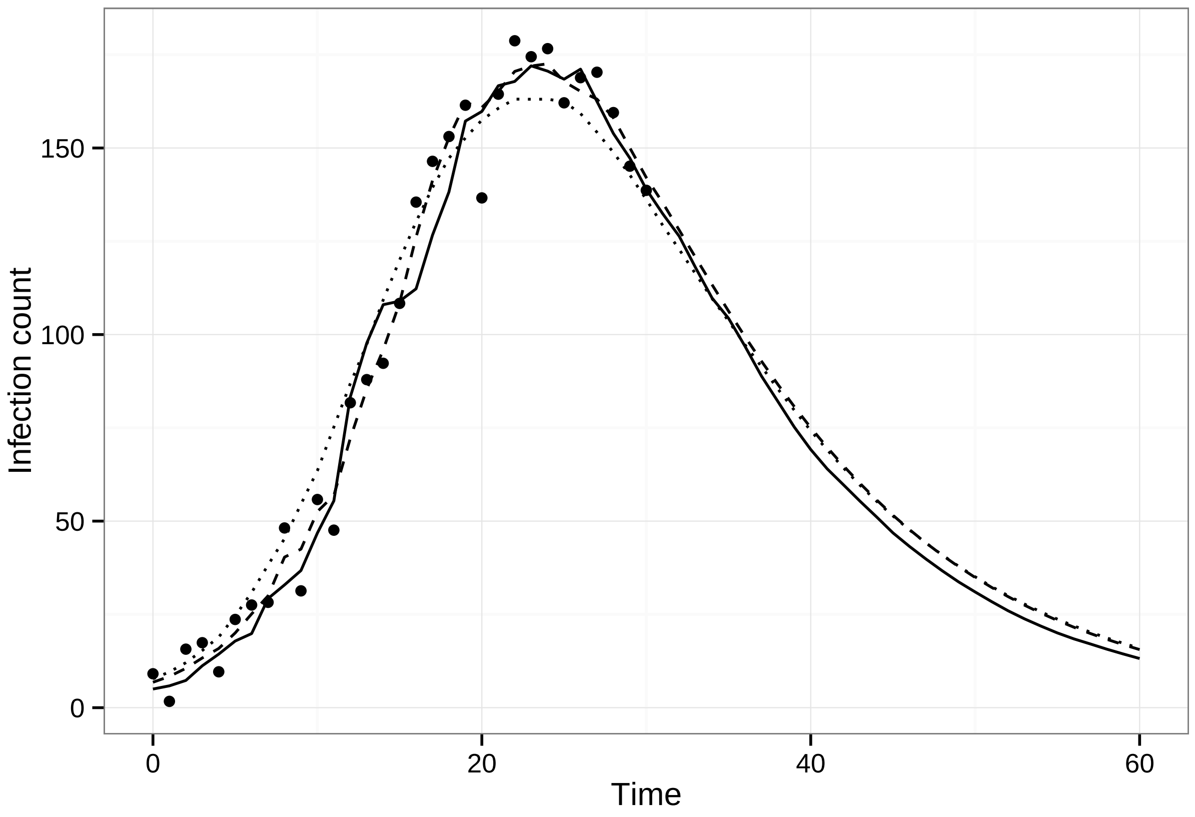
The solid line shows the true states, the points are the data, the dashed line shows the bootstrap mean, and the grey ribbon shows the centre 95th quantile of the bootstrap.

As before we can evaluate the SSE of the forecast, giving

```
## [1] 405.7022
```

Comparison

This plot showing the forecasted states for each method against the true states and the data

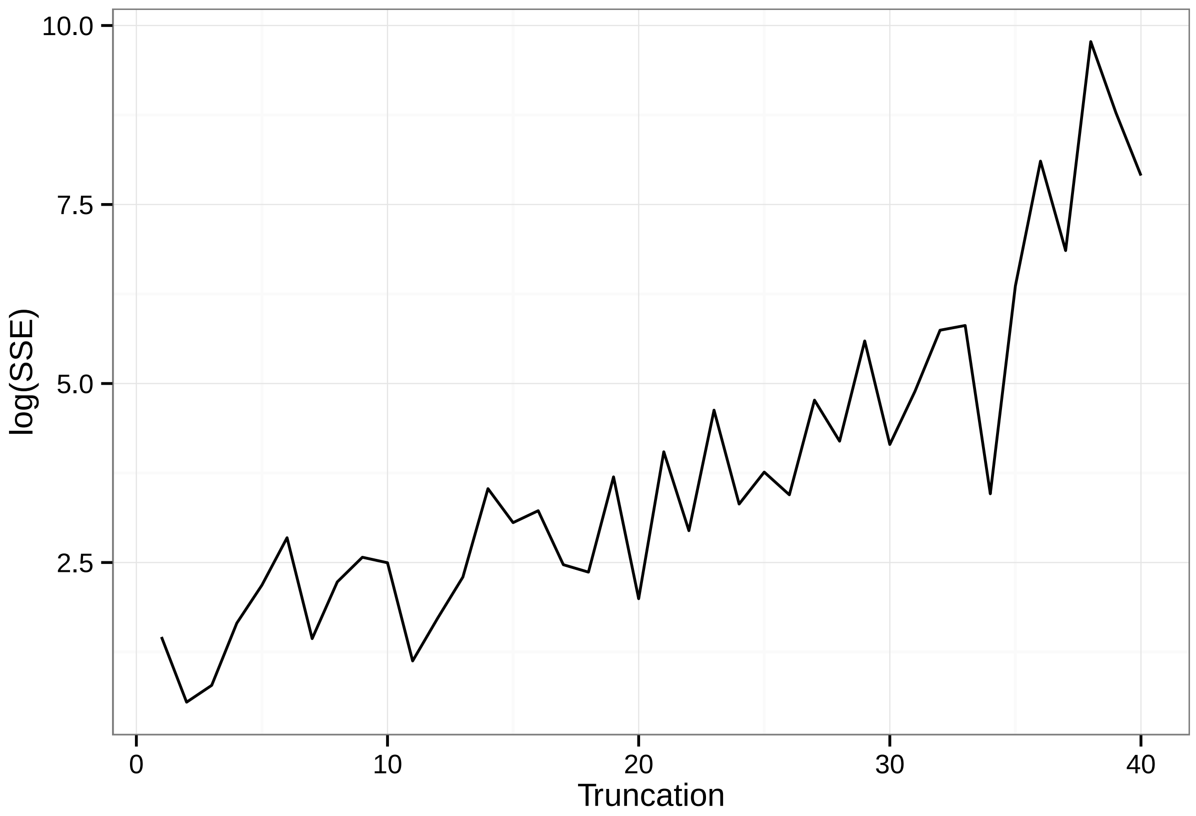


The solid line shows the true states, the dots are the data, the dashed line shows the IF2 estimates states, and the dotted line shows the HMC estimated true states

Truncation vs. Error

Of course the above mini-comparison only shows one truncation value for one trajectory. Really, we need to know how each method performs on average given different trajectories and truncation amounts.

We can fit the model to successively smaller (or larger) time series to see the effect of truncation on forecast SSE. This was done below, with several *(2 right now, I'll run this for a larger number of trials in the future)* new trajectories drawn for each of the desired lengths.



This clearly shows a drastic increase in forecast error the more the data is truncated.

(I'll do this for HMC too, but it will take a while to run.)