BSTS Part 3 Script

Slide 1

Congratulations! You've made it to the moment that you were surely waiting for. Now that you understand the mysteries behind Bayesian structural time series, you will begin to master the practice of Bayseian structural time series.

Here is what you have to look forward to in this part of your adventure (-> Slide 3)

Slide 3

First, we will talk about the SST data that you have seen in previous videos.

(->) Then we will see how to use the R package *bsts*.

(->) We will see how to fit the structural time series models we saw in previous videos. This will be demonstrated for

(->) the local level model.

(->) the local linear trend model.

(->) and the local linear trend with seasonality model.

(->) We will also see how to obtain and plot the posterior distribution,

(->) forecast,

(->) and compare the three models.

We encourage you follow along! To do so, see the links in the description below. (-> Slide 4)

Slide 4

The data we will be using in this video is the sea surface temperature of the first 30 meters around Gibraltar.

(->) The temperature was recorded every day by instruments floating in the ocean. These temperatures were then averaged over 12 day periods.

(->) Here, we consider the sea surface temperature from January 2004 to November 2017.

(->) The data was obtained from the website Argovis. At Argovis.com, you can access data taken from instruments floating in the ocean all over the world. Tyler has even written a Python API so you can obtain data from your favorite ocean region!

(-> Slide 5)

Slide 5

This is what the SST data we will be working with looks like. Most of the temperatures lie between about 17 and 23 degress celcius and there is obviously some seasonality here. Let's see how we can fit some Bayesian structural time series to this data. (-> Slide 6)

Slide 6

Before we start fitting models, we need to do a little setting up. First, we load *readr* to read in the data, and of course the *bsts* package. Note that *bsts* loads several other packages as well.

We read in the data, then convert it to a time series object. We set the frequency to 30 because there are approximately 30 twelve day periods in a year. Then we produce the plot shown on the previous slide. (-> Slide 7)

Slide 7

Remember that the local level model looks like this. y_t is our data, and μ_t is the state component. In this case the level of the series at time t.

To fit models using *bsts*, we first need to construct the state. We start by creating an empty list to hold our state specification. Then we use *AddLocalLevel* to say that we want to add a level component to our state.

To fit the model, we use the function *bsts*. Along with our time series, we pass our state specification and the number of MCMC iterations we want from the posterior distribution. Here we only use 1,000 iterations. But this is just to save time. In practice you would want a much larger posterior sample.

In addition to fitting the model and drawing samples from the posterior, the *bsts* function creates a *bsts* object. A *bsts* object is a list that contains the posterior sample, the state specification, the log likelihood, and many other components we might be interested in. (-> Slide 8)

Slide 8

Once we fit our model and simulate draws from the posterior, we might be interested in plotting the results. *bsts* provides several different plotting methods.

If we use *plot* with a bsts object, what we get is a plot of the posterior mean, conditional on the data. The points on the plot are our original data. (-> Slide 9)

Slide 9

If we specify *components* in the plot function, we get a plot of the posterior of the state components. In this case, the local level. Based on this plot, it seems like the local level is doing a pretty good job of modeling the SST data. (-> Slide 10)

Slide 10

Another option we can give to the *plot* function is *residuals*. This of course plots the residuals. The shading here is the density of the residuals at each time point.(->)

Slide 11

bsts also provides a predict method to compute forecasts. The *horizon* argument specifies how far ahead we want to forecast. In this case we forecast 30 time steps ahead, which corresponds to one year.

To see how our forecasts are doing, we can pass our *bsts prediction* object to the plot function. The *plot.original* argument plots the original series along with the forecasts. The default value is *TRUE* which will plot the entire original series. We can also specify a numerical value as we did here. The value of 90 here means only the last 3 years of the series are plotted since there are about 30 observations per year.

The blue line indicates the median of the predictive distribution and the green lines indicate the prediction interval. The default is a 95% prediction interval. Just like with the residuals plot, the gray shading is the density.

From this plot we can see that the local level model doesn't seem adequate. The forecasts are essentially a flat line which doesn't caputure the seasonality in the series. We also see that the forecast intervals are very wide. We can try another model to see if it can provide better forecasts. (-> Slide 12)

Slide 12

Recall the local linear trend model which we present here. y_t is our data, μ_t is the local level, and ν_t is the local trend.

Fitting this model is just like fitting the local level model. The only difference here is that we use *AddLocalLinearTrend* instead of *AddLocalLevel*.

We could also easily produce similar plots as with the local level model. All we need to do is pass the local linear trend fitted object instead of the local level object. However, we will just look at forecasts for this model. (-> Slide 13)

Slide 13

Here are the forecasts for the local linear trend model. Just like with the local level model, we forecast one year ahead and plot the last three years of the original series.

From this plot, it is obvious that the local linear trend model does a terrible job. (->)

Slide 14

Let's try a local trend with seasonality. Since the SST data has obvious seasonality, we probably should have started with this. But we wanted to illustrate how to fit models starting with the simplest case first.

Remeber what this model looks like. Here we consider μ_t to be a local linear trend and τ_t is our seasonal component.

The local linear trend component is specified just like before. But since we want a seasonal component, we need to add that separately using the *AddSeasonal* function. The argument *nseasons* is the capital s in the equation for τ. For the SST data, this is 30.

Let's see what the posteriors of the 2 state components look like. (-> Slide 15)

Slide 15

Just like before we pass our fitted object to the *plot* function and specify that we want to plot the components. *bsts* will automatically plot all our state components.

From the trend plot, we see a bit of a rise around time 100. This is evident in the plot of the original data corresponding to about 2007. It would be nice if the times on our component plots matched the original series but for some reason *bsts* was not plotting the original time stamps for us.

We also see from the seasonal component plot that this model seems to be capturing the seasonality of the original series. Let's see what our forecasts look like. (-> Slide 16)

Slide 16

This looks a lot better than our previous models. The forecast intervals are much narrower here. They do grow with time but this is to be expected. Remember, the blue line is the median of the predictive distribution. We can see that this model is doing a much better job of capturing the general pattern of our series.