Tourism Data Analysis

31 July 2019

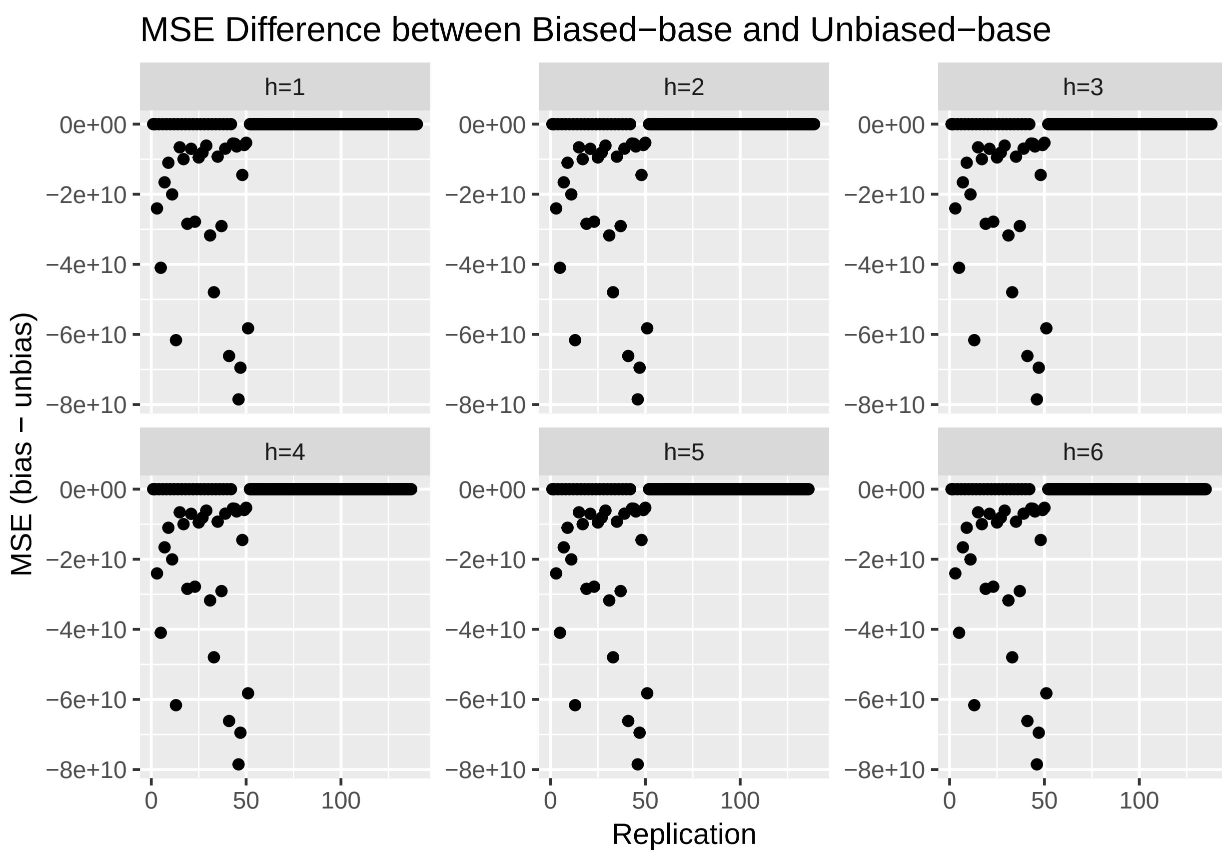
Bias adjusted forecasts are given as follows:

$$y_t = \begin{cases} \exp x_t[1 + \frac{\sigma_h^2}{2}] & \text{if } \lambda = 0 \\ \\ (\lambda x_t + 1)^{1/\lambda}[1 + \frac{\sigma_h^2(1-\lambda)}{2(\lambda x_t + 1)^2}] & \text{Otherwise} \end{cases}$$

For BoxCox transformed data (Using the BoxCox function in the forecast package)

In this exercise I have used the `BoxCox.lambda()` function in forecast package to automatically find a proper transformation for each series.

	h=1		h=2		h=3		h=4	
R-method	Biased	Unbiased	Biased	Unbiased	Biased	Unbiased	Biased	Unbia
Base	12212.05	5.133715e+09	13171.69	5.170730e+09	13853.12	5.208223e+09	15940.78	5.246292e-
Bottom-up	17868.80	2.053495e+10	17861.05	2.068446e+10	18480.28	2.083190e+10	20500.91	2.098719e-
MinT(Shrink)	10196.57	9.981284e+03	11507.51	1.080718e+04	12402.17	1.179984e+04	15529.77	1.401968e-
OLS	11765.69	3.766111e+09	12833.82	3.793271e+09	13516.04	3.820789e+09	15618.33	3.848752e-
WLS	15637.99	1.398369e+04	15930.56	1.370782e+04	16638.94	1.450090e+04	18910.95	1.625339e-

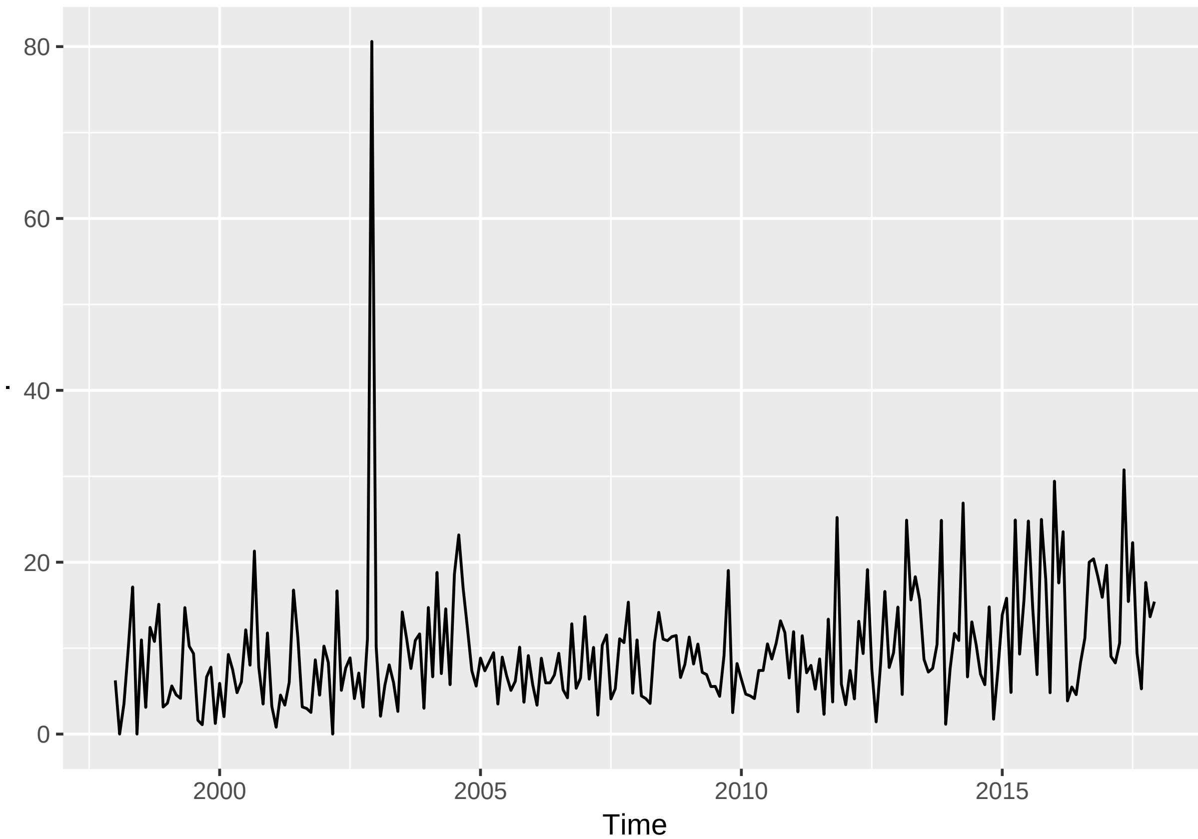


The above graph shows the difference between the MSE of `Base-biased forecasts` and `Base-unbiased forecasts`. As we can see, this difference is very large for some of the replications which courses the large average MSE in unbiased-base forecasts shown in the table 1.

However an important thing to notice here is that although the unbiased base forecasts are terrible, MinT has adjusted these through the reconciliation. In fact the MinT reconciled unbiased forecasts are even better than the MinT reconciled biased forecasts.

We have also noticed that the based-bias adjusted forecasts are going worst for only one variable which is `Adelaide Hills`. If we visualise this time series we get,

```
AllTS %>%
  dplyr::select(`Adelaide Hills`) %>%
  ts(start = c(1998, 1), frequency = 12) %>%
  autoplot()
```



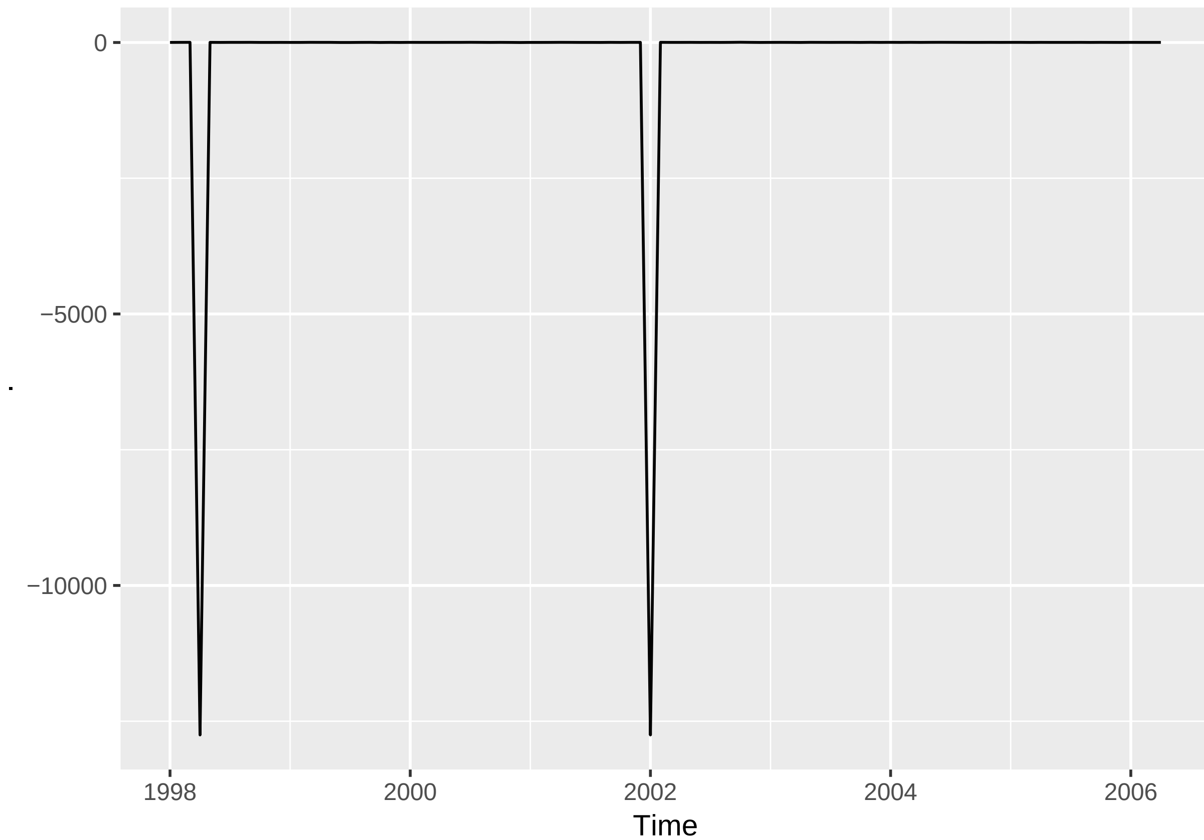
We can see this variable has an anomalous observation on Dec-2002 which might be an outlier.

```
AllTS %>%
  dplyr::select(`Adelaide Hills`) %>%
  slice(3:102) %>%
  as.ts() %>%
  BoxCox.lambda() -> lambda
```

If we do Box-Cox transformation for this series we get $\lambda = 7.8411995 \times 10^{-5}$. The transformed series with this λ looks like

```
AllTS %>%
  dplyr::select(`Adelaide Hills`) %>%
```

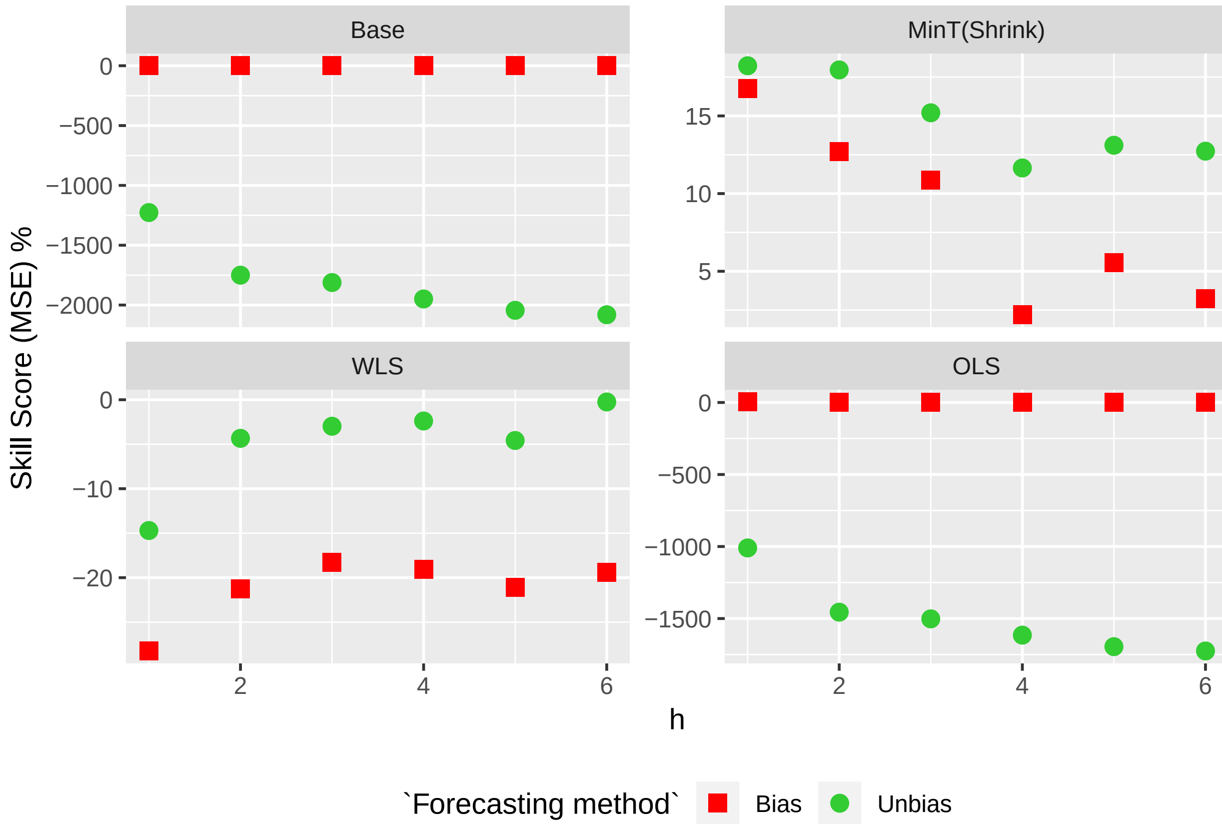
```
slice(3:102) %>%
ts(start = c(1998, 1), frequency = 12) %>%
BoxCox(lambda = lambda) %>%
autoplot()
```



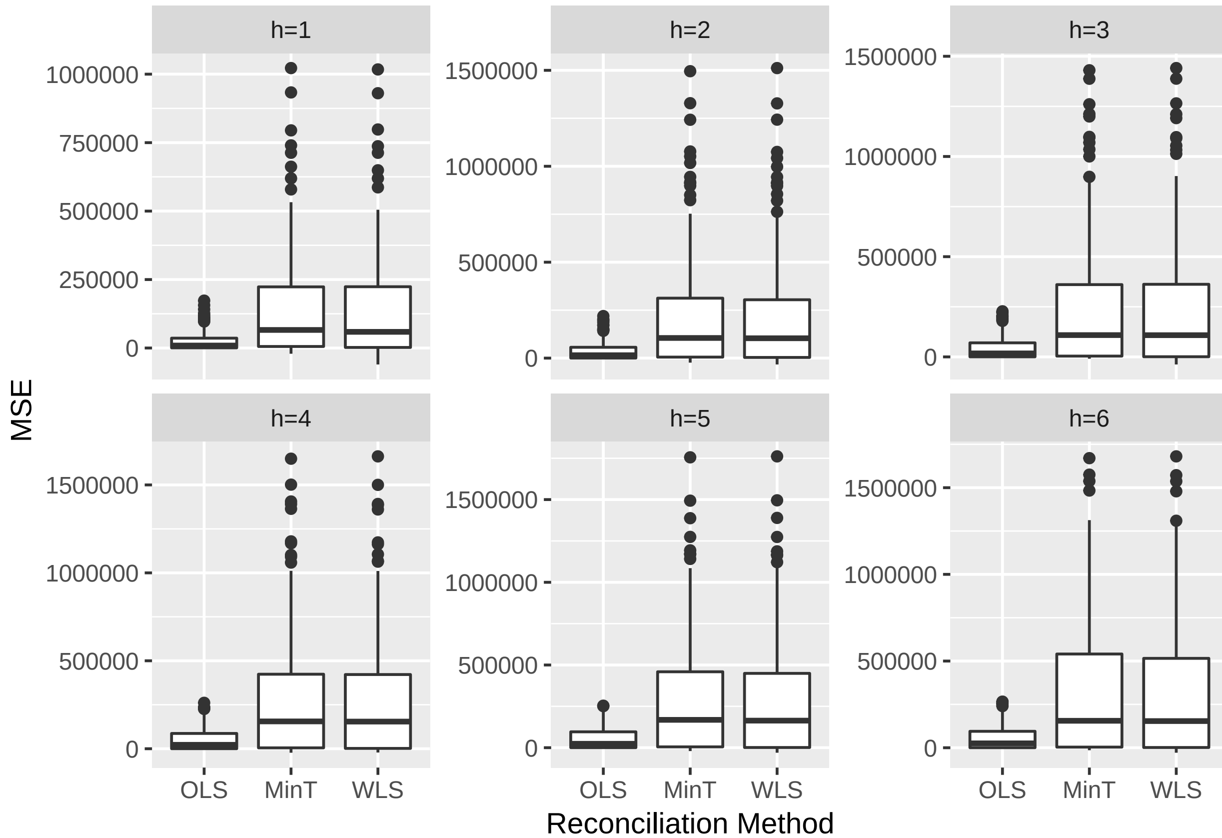
I have replaced the anomalous observation in this series with the average of December for predecessive and successive years of 2002. Then I have refitted the models for automatically chosen Box-Cox transformed data. Back-transformed and back-transformed bias-adjusted forecasts were then obtained. The results are as follows.

	h=1		h=2		h=3		h=4		h=5	
R-method	Biased	Unbiased	Biased	Unbiased	Biased	Unbiased	Biased	Unbiased	Biased	Unbiased
Base	12.18	161.63	13.14	243.13	14.08	269.20	15.88	325.41	15.49	332.11
Bottom-up	17.86	15.55	17.86	14.90	18.47	15.61	20.50	17.21	20.34	17.04
MinT(Shrink)	10.14	9.96	11.47	10.78	12.55	11.94	15.53	14.03	14.63	13.46
OLS	11.73	135.17	12.80	204.42	13.73	225.59	15.56	272.41	15.17	278.08
WLS	15.62	13.97	15.93	13.71	16.65	14.50	18.91	16.26	18.76	16.20

These results also shows that, even though the bias-adjusted based forecasts are worst than that of biased base forecasts, the MinT reconciliation is outperforming.




```
## Adding missing grouping variables: `F-method`
```



For Log transformed data

In this exercise I have used only the log transformation. As we can see from the results, the bias adjusted forecasts are outperforming biased forecasts as we would expect.

	h=1		h=2		h=3		h=4		h=
R-method	Biased	Unbiased	Biased	Unbiased	Biased	Unbiased	Biased	Unbiased	Biased
Base	12.32137	12.132443	12.92670	12.70852	14.15600	13.95030	15.39609	15.06325	15.49290
Bottom-up	17.14512	15.283081	17.27624	15.04939	17.95151	15.51418	19.77243	17.14844	20.00422
MinT(Shrink)	9.39431	9.309393	11.29788	11.22862	12.45607	12.28674	13.42079	13.22514	13.66527
OLS	11.84118	11.670953	12.57479	12.37798	13.79253	13.62081	15.02111	14.71486	15.13972
WLS	14.85844	13.725030	15.15962	13.83053	15.98118	14.50529	17.95012	16.25810	18.17351