Forecasting Beer Sales

Fred Viole

October 27, 2017

Comparing MAPE of Several Forecasting Techniques

The following blog highlights a time-series forecast using `timetk`. We use the same data with `NNS.ARMA` and find an superior MAPE using `NNS.ARMA`.

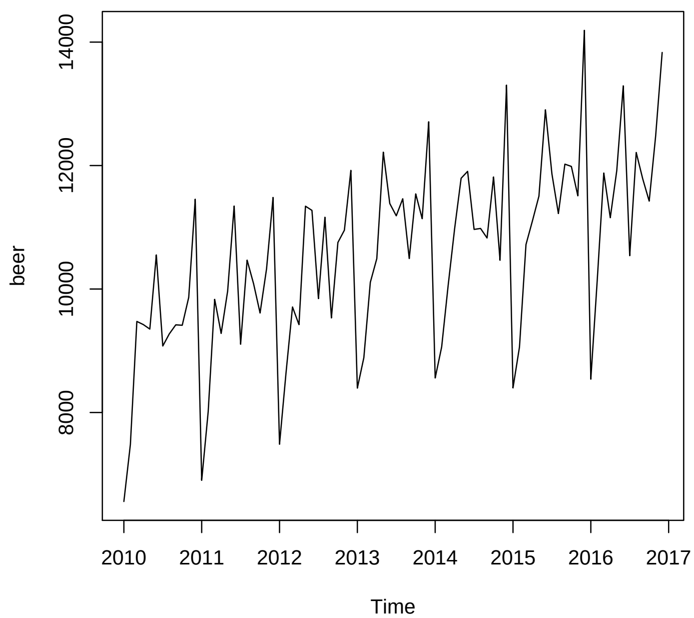
https://www.r-bloggers.com/demo-week-time-series-machine-learning-with-timetk/

We also then try `auto.arima` from the `forecast` package, and `prophet` from Facebook and a combination forecast from `forecastHybrid`. `NNS.ARMA` has the lowest MAPE...

```
> require(Quandl)
> beer = Quandl("FRED/S4248SM144NCEN",type = 'ts',order = "asc",
+         start_date="2010-01-01",end_date="2016-12-31")
> head(beer)
```

```
[1]  6558  7481  9475  9424  9351 10552
```

```
> plot.ts(beer)
```



NNS:

First let's cross-validate our `[seasonal.factor]` parameter:

Step 1: Find our seasonal periods

We find our seasonal periods and store the results in `[seasonal.periods]`.

We create a test set `[test]` and a training set `[train]`.

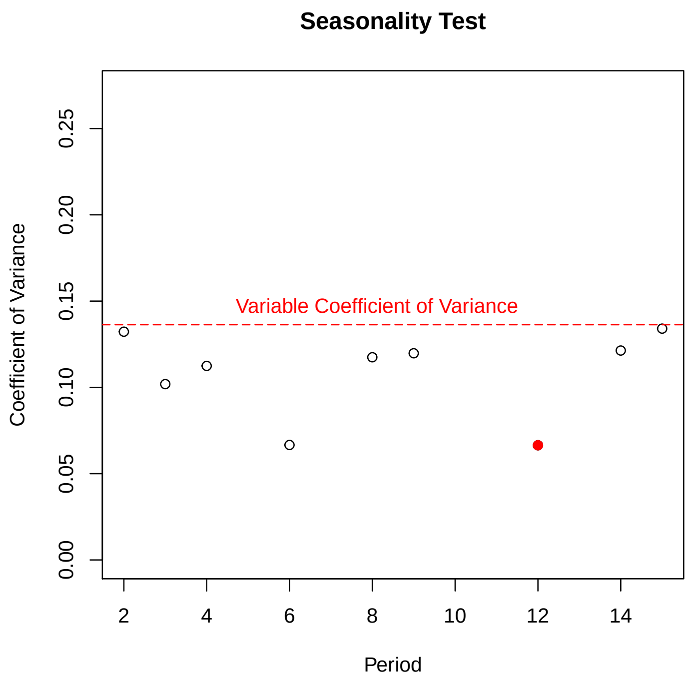
```
> test = tail(beer,8)
> train = head(beer,(length(beer)-8))
```

From our `[train]` training set, we create another training set and test set. We double the original test set number of observations, so our within sample test set has 16 observations to validate our parameters on. This is critical to avoid any leakage of information from our test set into our training set.

```
> in.sample.train = head(train,(length(train)-16))
> in.sample.test = tail(train,16)
```

Now we ascertain the seasonality present in our reduced training set.

```
> require(NNS)
> seasonal.periods = NNS.seas(in.sample.train)
```



Step 2: Test MAPE of all `[seasonal.periods]`

All of the commands are lumped together into 3 lines of code, resulting in an output `[cv.test]` recording all of our MAPE and associated `[seasonal.periods]`.

```
> cv.test = t(sapply(seasonal.periods$all.periods$Period, function(i)
+   c(i,mean(abs(NNS.ARMA(in.sample.train,h=8,
+   seasonal.factor = i,method='both') - in.sample.test)/in.sample.test))))
> colnames(cv.test) = c("Period","MAPE")
> cv.test
```

```
     Period   MAPE
[1,]     12 0.1236
[2,]      6 0.1134
[3,]      3 0.1211
[4,]      4 0.1368
[5,]      8 0.1476
[6,]      9 0.1149
[7,]     14 0.1703
```

```
 [8,]       2 0.1314
 [9,]      15 0.1811
```

NNS ESTIMATES:

Using our lowest MAPE [seasonal.factor = 6], let's see if we benefit from a nonlinear regression...

```
> NNS.estimates = NNS.ARMA(in.sample.train,h=16,
+             seasonal.factor = 6,method='nonlin')
> mean(abs(NNS.estimates - in.sample.test) / in.sample.test)
```

```
[1] 0.141
```

No, we do not. Thus our best estimate is the one from our linear model.

NNS Final Estimate:

Using our linear model on our expanded training set, we can calculate our MAPE:

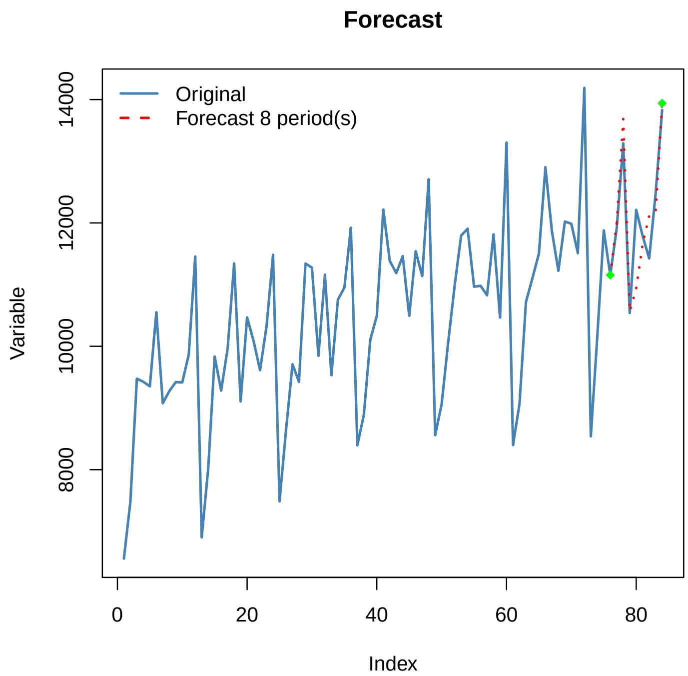
```
> NNS.estimates = NNS.ARMA(train,h=8,
+             seasonal.factor = 6,method='lin')
> final.NNS = mean(abs(NNS.estimates - test) / test)
> round(final.NNS,5)
```

```
[1] 0.03086
```

Yielding the NNS MAPE of 3.086%.

timetk generates a 46.85% worse forecast than NNS.ARMA! Below is a visualization of the NNS.ARMA forecasts (in red) overlayed on the actual observations.

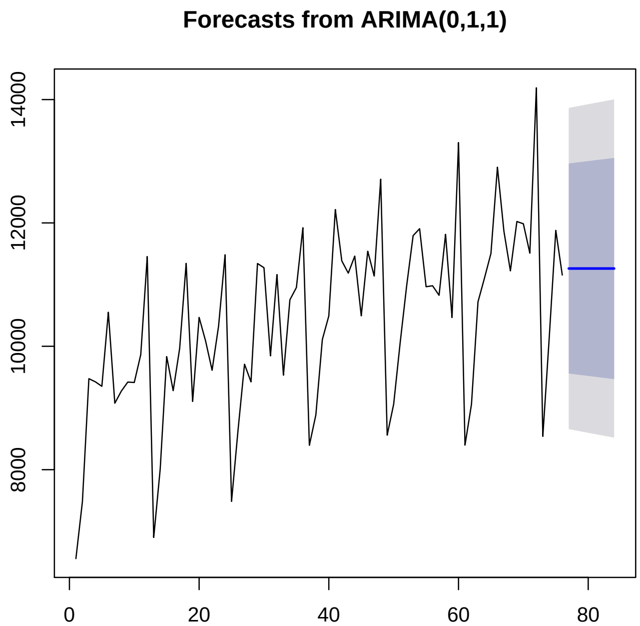
```
> NNS.ARMA(beer,training.set=length(train),h=8,
+             seasonal.factor = 6,method='lin')
```



ARIMA Model:

Using a standard ARIMA model from the `forecast` package, let's see how that fares...

```
> library(forecast)
> fit=auto.arima(train)
> f.cast=forecast(fit,h=8)
> plot(f.cast)
```



ARIMA MAPE:

```
> mean(abs(f.cast$mean - test) / test)
```

```
[1] 0.08706
```

which is considerably worse than `timetk` and `NNS.ARMA`.

`prophet` from Facebook

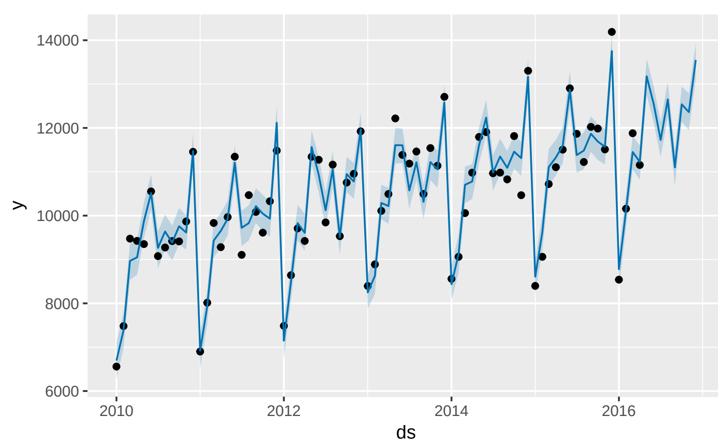
```
> require(prophet)
> m <- prophet(data.frame(ds = seq(as.Date('2010-01-01'),
+       as.Date('2016-04-01'), by = 'm'),y=train))
> future <- make_future_dataframe(m,periods=8,freq='month')
> forecast <- predict(m,future)
```

`prophet` MAPE

```
> mean(abs(test-tail(forecast[,"yhat"],8))/test)
```

```
[1] 0.06206
```

```
> plot(m,forecast)
```



Combination Forecasts:

`forecastHybrid` is another time-series forecasting package that uses a combination of models from the `forecast` package.

```
> require(forecastHybrid)
> fit1 <- hybridModel(train,models = "aefnt",weights = 'insample.errors')
```

```
Fitting the auto.arima model
Fitting the ets model
Fitting the thetam model
Fitting the nnetar model
Fitting the tbats model
```

```
> fc1 <- forecast(fit1, h=8)
> mean(abs(fc1$mean - test) / test)
```

```
[1] 0.04721
```

Combining models clearly results in a better MAPE than the standalone ARIMA model from the `forecast` package.

Can you generate a more accurate forecast with different techniques or parameters?

More NNS:

To learn more about NNS statistics and their theoretical foundations, see

"*Nonlinear Nonparametric Statistics: Using Partial Moments*"

available on Amazon: `http://a.co/5bpHvUg`

Check back to see more NNS examples posted on GitHub:
`https://github.com/OVVO-Financial/NNS/tree/NNS-Beta-Version/examples`