Holt Winters Exponential Smoothing

Fallaw Sowell

3/14/2021

Contents

Holt Winters Smoothing – Quarterly Hot Water Heater Sales

This file was created with R markdown and Knit. This demonstrates how to read in a data from an excel file, obtain summary statistics, plot the time series, perform Holt-Winters smoothing to get forecasts, plot the forecasts and save the forecasts to an excel file.

Load the needed libraries

```
rm(list=ls()) # clear the work area

if (!require('readxl')) install.packages('readxl')
library('readxl')

if (!require('writexl')) install.packages('writexl')
library('writexl')

if (!require('ggplot2')) install.packages('ggplot2')
library('ggplot2')

if (!require('ggfortify')) install.packages('ggfortify')
library('ggfortify')

if (!require('forecast')) install.packages('forecast')
library('forecast')

if (!require('lubridate')) install.packages('lubridate')
library('lubridate')

if (!require('zoo')) install.packages('zoo')
library('zoo')
```

Load the data into R

First, read the data from the excel file and save as a data.frame. This excel file needs to be in the R working directory. Alternatively, you would need to include the path the excel file.

The data are quarterly hot water heater sales from an consulting project.

It is good practice to review the data by looking at its structure.

```
data_from_excel <- read_excel("WEEKOQUARTERLY.xlsx")

str( data_from_excel )
```

```
## tibble [69 x 1] (S3: tbl_df/tbl/data.frame)
##  $ PRODB: num [1:69] 12910502 16778366 14644695 14822126 13121648 ...
```

Make Hot Water Heater Sales a monthly time series

The next chunck of R code, creates that time series data structure for the sales, displays its structure, reports basic summary values, and plots the time series. It is good practice to label your plots so then you share them everyone knows what is being displayed.

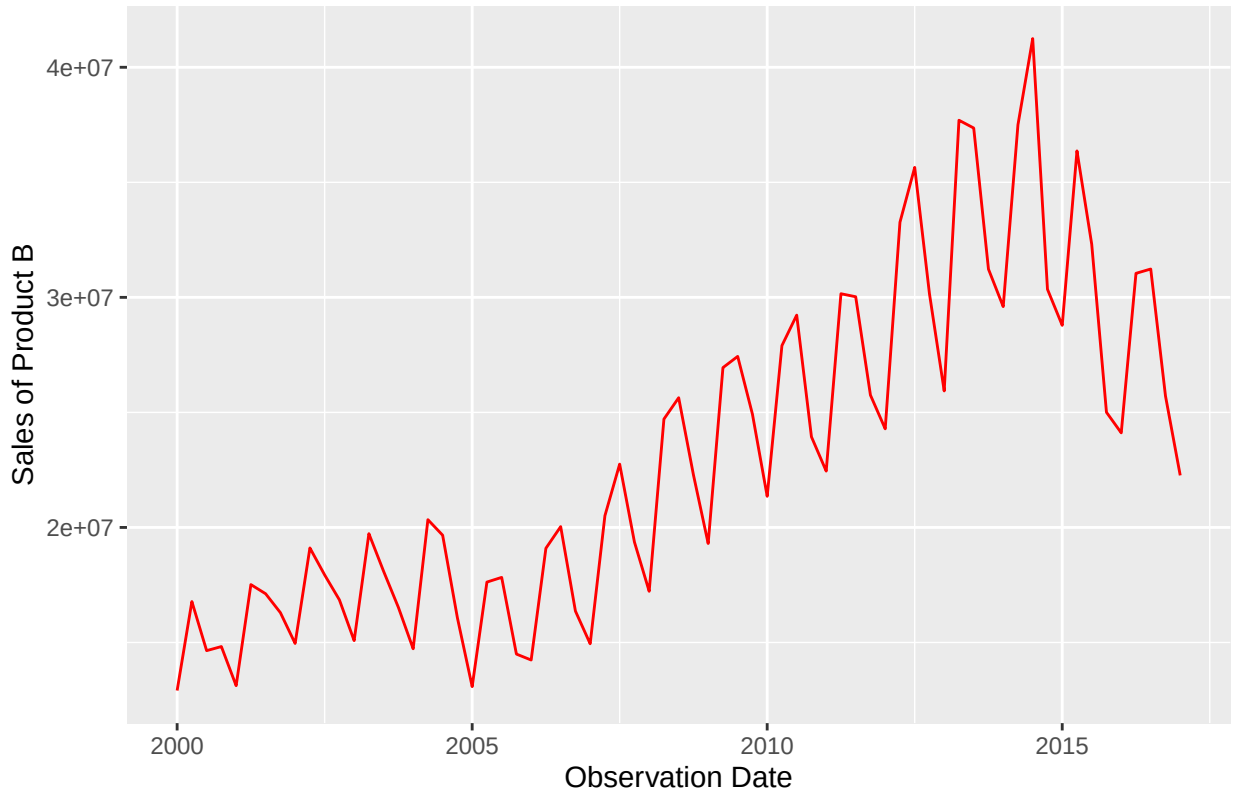
```r
PROD_B <- ts(data_from_excel, start=c(2000,1), frequency = 4, names =c("Sales of Product B") )

str(PROD_B)
```

```
##  Time-Series [1:69, 1] from 2000 to 2017: 12910502 16778366 14644695 14822126 13121648 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr "Sales of Product B"
```

```r
summary(PROD_B)
```

```
##  Sales of Product B
##  Min.   :12910502
##  1st Qu.:17116886
##  Median :22262409
##  Mean   :23086565
##  3rd Qu.:28782195
##  Max.   :41246966
```

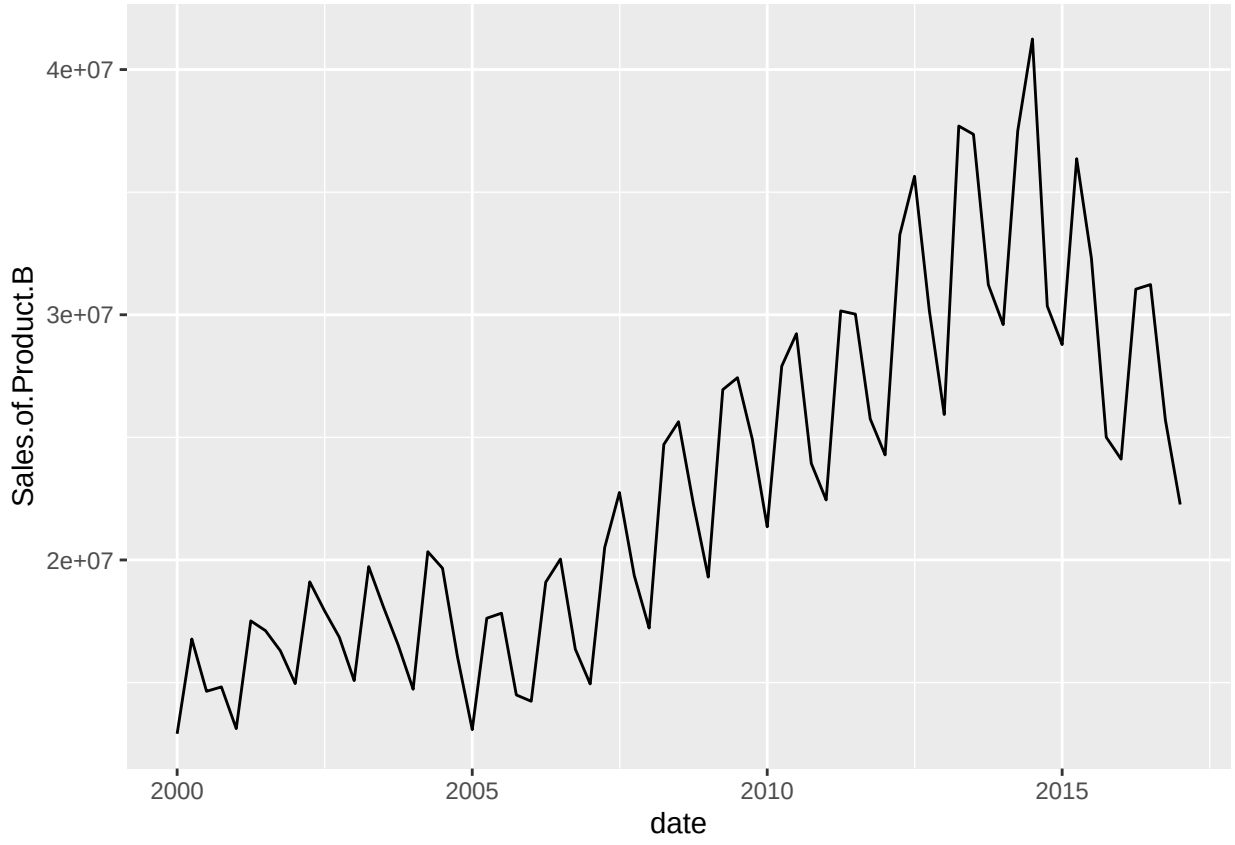
```r
autoplot(PROD_B, colour = 'red') +
  labs(x = "Observation Date", y = "Sales of Product B")
```



Plot with ggplot2

The next chunck of R code uses ggplot to create a plot of the data. The function ggplot only accepts data in the form of a dataframe.

```
df_data <- data.frame(date=as.Date(as.yearmon(time(PROD_B))), Y=as.matrix(PROD_B))
ggplot(data=df_data, mapping=aes(x=date, y=Sales.of.Product.B))+geom_line()
```



Create forecasts using Holt-Winters smoothing

The series has both a time trend and a seasonal components hence, Holt-Winters smoothing would be appropriate.

We will use the function ets(). The first argument is the time series data, the second argument is the model with the three terms "error, trend, seasonal". The options for the terms are

"N" - none "A" - additive "M" - multiplicative "Z" - automatically selected

You perform Holt-Winters exponential smoothing with the ets() function with model = "MAM".

The following chuck of R code perform Holt smoothing and save the generated data. It then displays summary information.

```
Holt_Winters <- ets(PROD_B, model = "MAM", alpha = .4, beta = .15, gamma=.3)
Holt_Winters
```

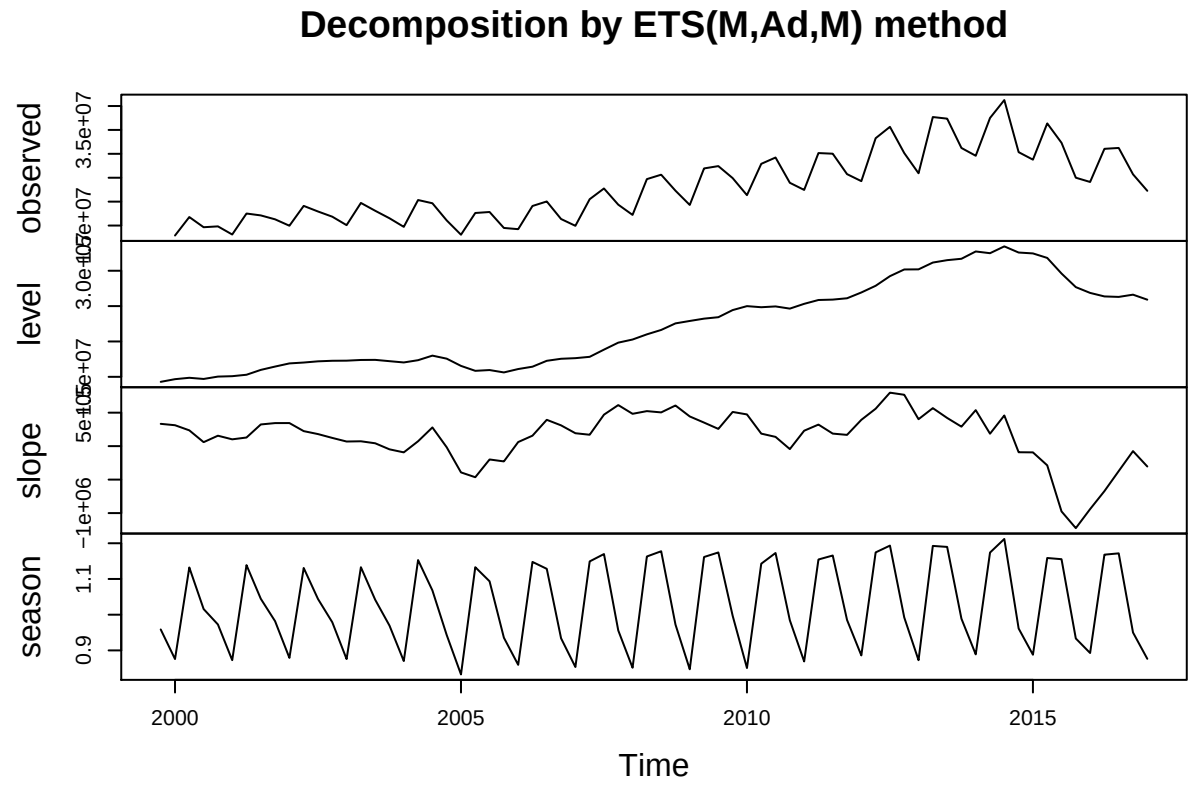
```
## ETS(M,Ad,M)
##
## Call:
##  ets(y = PROD_B, model = "MAM", alpha = 0.4, beta = 0.15, gamma = 0.3)
##
##   Smoothing parameters:
##     alpha = 0.4
##     beta  = 0.15
##     gamma = 0.3
##     phi   = 0.8264
##
##   Initial states:
##     l = 14292337.2808
##     b = 333207.826
##     s = 0.9587 1.0343 1.1356 0.8714
##
##   sigma:  0.0613
##
##      AIC     AICc      BIC
## 2243.607 2245.443 2259.245
```

```
str(Holt_Winters)
```

```
## List of 19
##  $ loglik    : num -1115
##  $ aic       : num 2244
##  $ bic       : num 2259
##  $ aicc      : num 2245
##  $ mse       : num 2.04e+12
##  $ amse      : num 3.03e+12
##  $ fit       :List of 4
##   ..$ value  : num 2230
##   ..$ par    : num [1:6] 8.26e-01 1.43e+07 3.33e+05 9.59e-01 1.03 ...
##   ..$ fail   : int 0
##   ..$ fncount: int 367
##  $ residuals : Time-Series [1:69] from 2000 to 2017: 0.017 -0.0101 -0.0599 0.0483 -0.0119 ...
##  $ fitted    : Time-Series [1:69, 1] from 2000 to 2017: 12694201 16948717 15577430 14139761 13280065
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr "y"
##  $ states    : Time-Series [1:70, 1:6] from 2000 to 2017: 14292337 14667003 14865350 14699559 150335
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:6] "l" "b" "s1" "s2" ...
##  $ par       : Named num [1:9] 8.26e-01 1.43e+07 3.33e+05 9.59e-01 1.03 ...
##   ..- attr(*, "names")= chr [1:9] "phi" "l" "b" "s0" ...
##  $ m         : num 4
##  $ method    : chr "ETS(M,Ad,M)"
##  $ series    : chr "PROD_B"
##  $ components: chr [1:4] "M" "A" "M" "TRUE"
##  $ call      : language ets(y = PROD_B, model = "MAM", alpha = 0.4, beta = 0.15, gamma = 0.3)
##  $ initstate : Named num [1:6] 1.43e+07 3.33e+05 9.59e-01 1.03 1.14 ...
##   ..- attr(*, "names")= chr [1:6] "l" "b" "s1" "s2" ...
##  $ sigma2    : num 0.00376
```

```
##  $ x         : Time-Series [1:69, 1] from 2000 to 2017: 12910502 16778366 14644695 14822126 13121648
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr "Sales of Product B"
##  - attr(*, "class")= chr "ets"
```

```
plot(Holt_Winters)
```




```
Holt_Winters.pred <- forecast(Holt_Winters, h = 6, level = c(95))
Holt_Winters.pred
```

```
##          Point Forecast    Lo 95    Hi 95
## 2017 Q2       29907614 26315622 33499605
## 2017 Q3       29705198 25622290 33788105
## 2017 Q4       23891549 20061335 27721763
## 2018 Q1       21886919 17801710 25972128
## 2018 Q2       29034233 22214354 35854112
## 2018 Q3       28985945 21402067 36569823
```

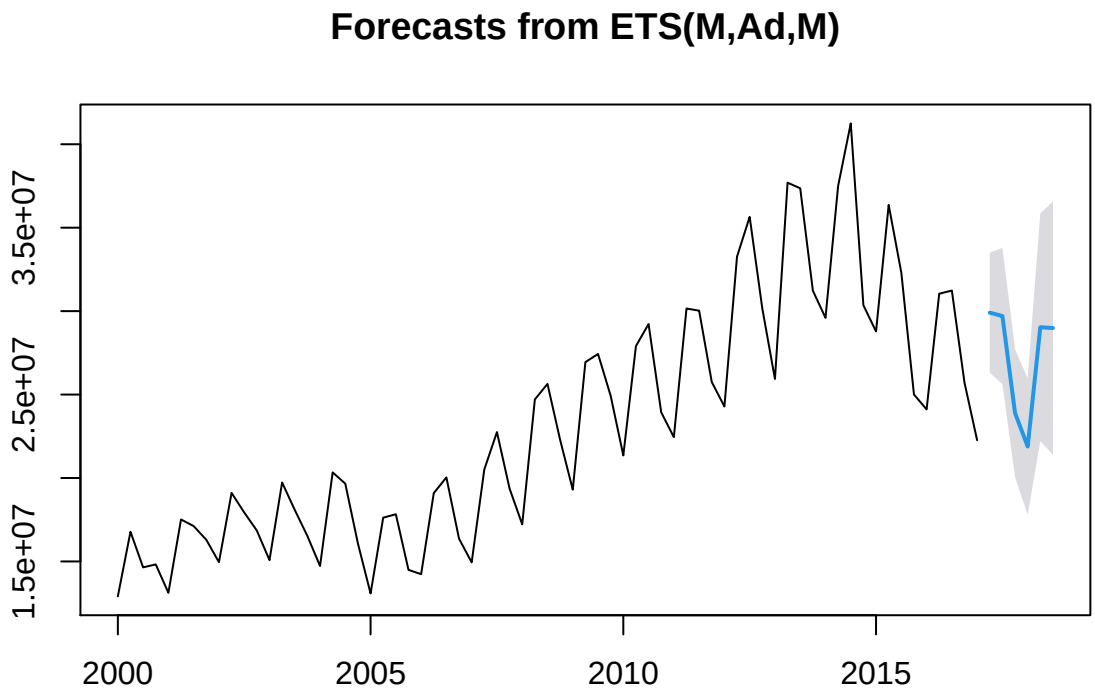
```
str(Holt_Winters.pred)
```

```
## List of 10
##  $ model    :List of 19
##   ..$ loglik    : num -1115
```

```
##   ..$ aic       : num 2244
##   ..$ bic       : num 2259
##   ..$ aicc      : num 2245
##   ..$ mse       : num 2.04e+12
##   ..$ amse      : num 3.03e+12
##   ..$ fit       :List of 4
##   .. ..$ value  : num 2230
##   .. ..$ par    : num [1:6] 8.26e-01 1.43e+07 3.33e+05 9.59e-01 1.03 ...
##   .. ..$ fail   : int 0
##   .. ..$ fncount: int 367
##   ..$ residuals : Time-Series [1:69] from 2000 to 2017: 0.017 -0.0101 -0.0599 0.0483 -0.0119 ...
##   ..$ fitted    : Time-Series [1:69, 1] from 2000 to 2017: 12694201 16948717 15577430 14139761 13280
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr "y"
##   ..$ states    : Time-Series [1:70, 1:6] from 2000 to 2017: 14292337 14667003 14865350 14699559 150
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:6] "l" "b" "s1" "s2" ...
##   ..$ par       : Named num [1:9] 8.26e-01 1.43e+07 3.33e+05 9.59e-01 1.03 ...
##   .. ..- attr(*, "names")= chr [1:9] "phi" "l" "b" "s0" ...
##   ..$ m         : num 4
##   ..$ method    : chr "ETS(M,Ad,M)"
##   ..$ series    : chr "PROD_B"
##   ..$ components: chr [1:4] "M" "A" "M" "TRUE"
##   ..$ call      : language ets(y = PROD_B, model = "MAM", alpha = 0.4, beta = 0.15, gamma = 0.3)
##   ..$ initstate : Named num [1:6] 1.43e+07 3.33e+05 9.59e-01 1.03 1.14 ...
##   .. ..- attr(*, "names")= chr [1:6] "l" "b" "s1" "s2" ...
##   ..$ sigma2    : num 0.00376
##   ..$ x         : Time-Series [1:69, 1] from 2000 to 2017: 12910502 16778366 14644695 14822126 13121
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr "Sales of Product B"
##   ..- attr(*, "class")= chr "ets"
##  $ mean     : Time-Series [1:6] from 2017 to 2018: 29907614 29705198 23891549 21886919 29034233 ...
##  $ level    : num 95
##  $ x        : Time-Series [1:69, 1] from 2000 to 2017: 12910502 16778366 14644695 14822126 13121648
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr "Sales of Product B"
##  $ upper    : Time-Series [1:6, 1] from 2017 to 2018: 33499605 33788105 27721763 25972128 35854112 .
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr "95%"
##  $ lower    : Time-Series [1:6, 1] from 2017 to 2018: 26315622 25622290 20061335 17801710 22214354 .
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr "95%"
##  $ fitted   : Time-Series [1:69, 1] from 2000 to 2017: 12694201 16948717 15577430 14139761 13280065
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr "y"
##  $ method   : chr "ETS(M,Ad,M)"
##  $ series   : chr "PROD_B"
```

```
##  $ residuals: Time-Series [1:69] from 2000 to 2017: 0.017 -0.0101 -0.0599 0.0483 -0.0119 ...
##  - attr(*, "class")= chr "forecast"
```

```
plot(Holt_Winters.pred)
```



Do the same things with the automatic procedure

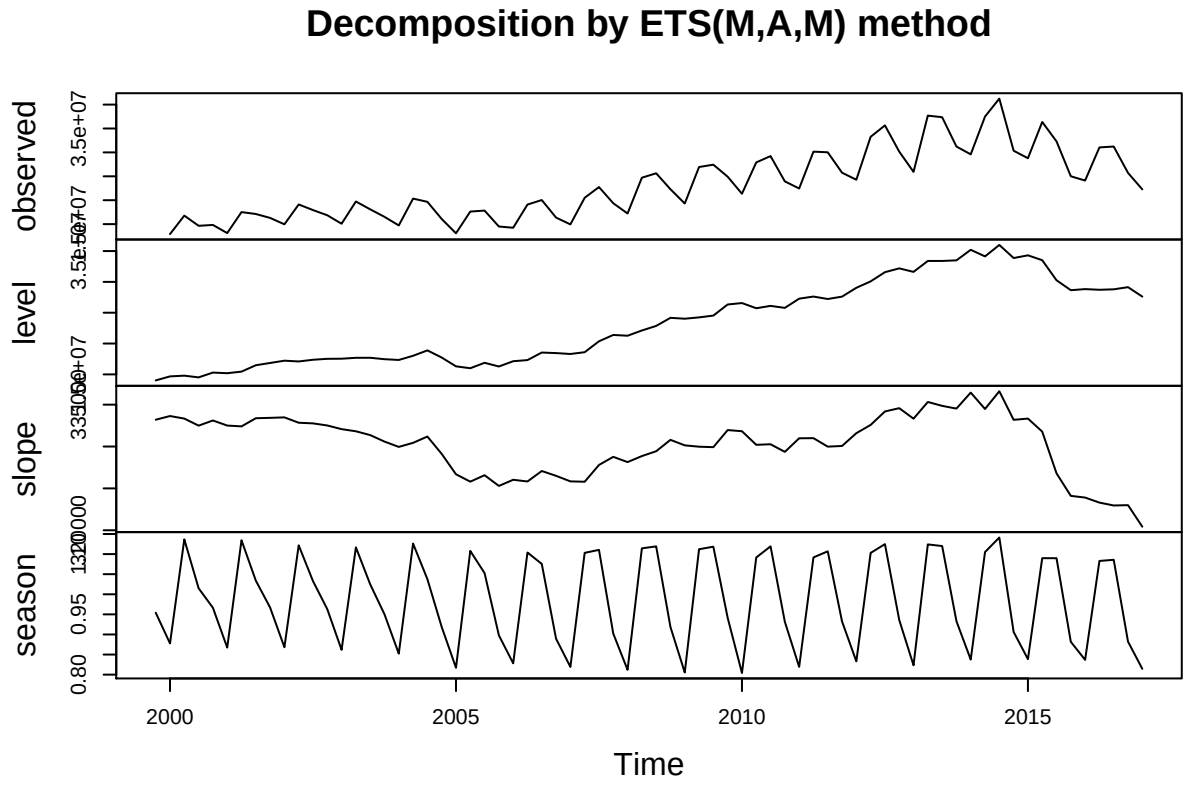
The next chuck of R code automatically generates the default Holt-Winters exponential smoothing forecasts.

```
Holt_Winters_auto <- ets(PROD_B, model = "MAM")
Holt_Winters_auto
```

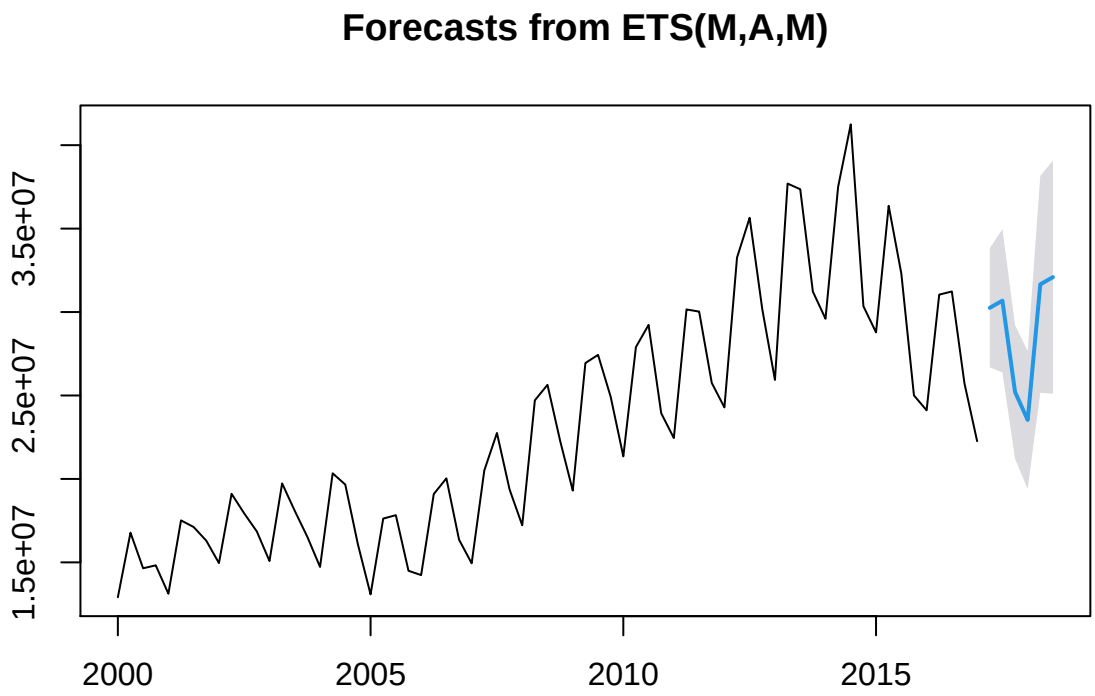
```
## ETS(M,A,M)
##
## Call:
##  ets(y = PROD_B, model = "MAM")
##
##   Smoothing parameters:
##     alpha = 0.6472
##     beta  = 9e-04
##     gamma = 0.2752
##
##   Initial states:
##     l = 14022604.263
```

```
##     b = 333199.8619
##     s = 0.9541 1.0326 1.1437 0.8696
##
##   sigma:  0.0602
##
##      AIC     AICc      BIC
## 2248.284 2251.335 2268.391
```

```
plot(Holt_Winters_auto)
```




```
Holt_Winters_auto.pred <- forecast(Holt_Winters_auto, h = 6, level = c(95))
plot(Holt_Winters_auto.pred)
```



Write the forecasted value and the confidence interval to an excel file

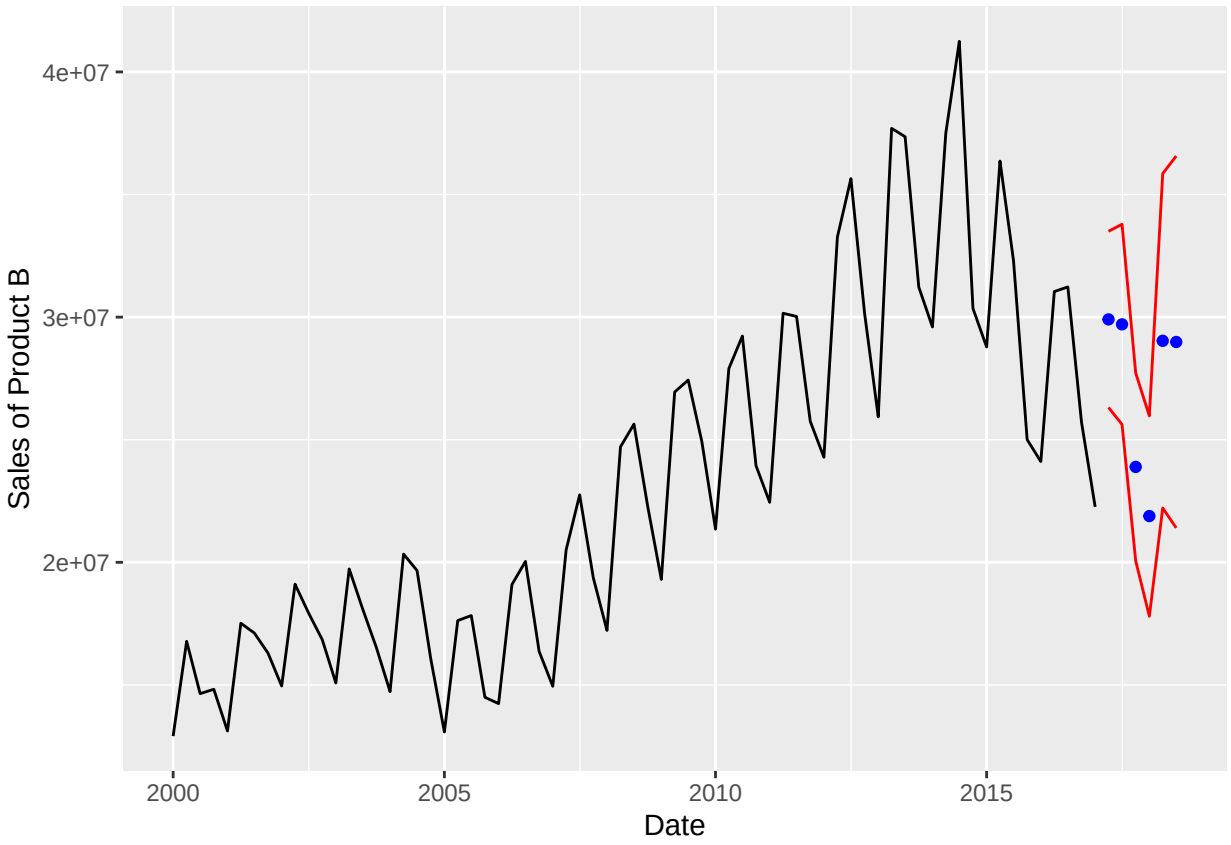
The next chuck of R code saves the forecasts to excel. This will allow you to share your forecasts with others.

```
col_names <- as.yearmon(time(Holt_Winters.pred$mean))
my_df <-  data.frame( as.Date(col_names), Holt_Winters.pred$mean, Holt_Winters.pred$lower, Holt_Winters
colnames(my_df) <- c("Date", "Forecast", "95% Lower", "95% Upper" )
write_xlsx(my_df, "Quarterly_Sales_B_prediction.xlsx")
```

Nice focused plot of the forecasts

The next chunck of R code uses ggplot to generate a plot. The package ggplot allows for control over what is plotted.

```
ggplot(data = my_df, aes(x = Date, y = as.double(Forecast) )) +
  geom_point( color="blue" ) +
  geom_line(data = my_df,  aes(x = Date, y = `95% Lower`), color="red" ) +
  geom_line(data = my_df,  aes(x = Date, y = `95% Upper`), color="red" ) +
  geom_line(data=df_data, mapping=aes(x=date, y=Sales.of.Product.B))+
  labs(x = "Date", y = "Sales of Product B")
```


4e+07
3e+07
2e+07
Sales of Product B
2000
2005
2010
2015
Date