Holt Exponential Smoothing

Fallaw Sowell

3/14/2021

Contents

Holt Smoothing – Annual 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 exponential 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 annual 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("Annual_Hot_Water.xlsx")
str( data_from_excel )
```

```
## tibble [14 x 1] (S3: tbl_df/tbl/data.frame)
##  $ PRODA: num [1:14] 38862 40431 35346 34314 30870 ...
```

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.

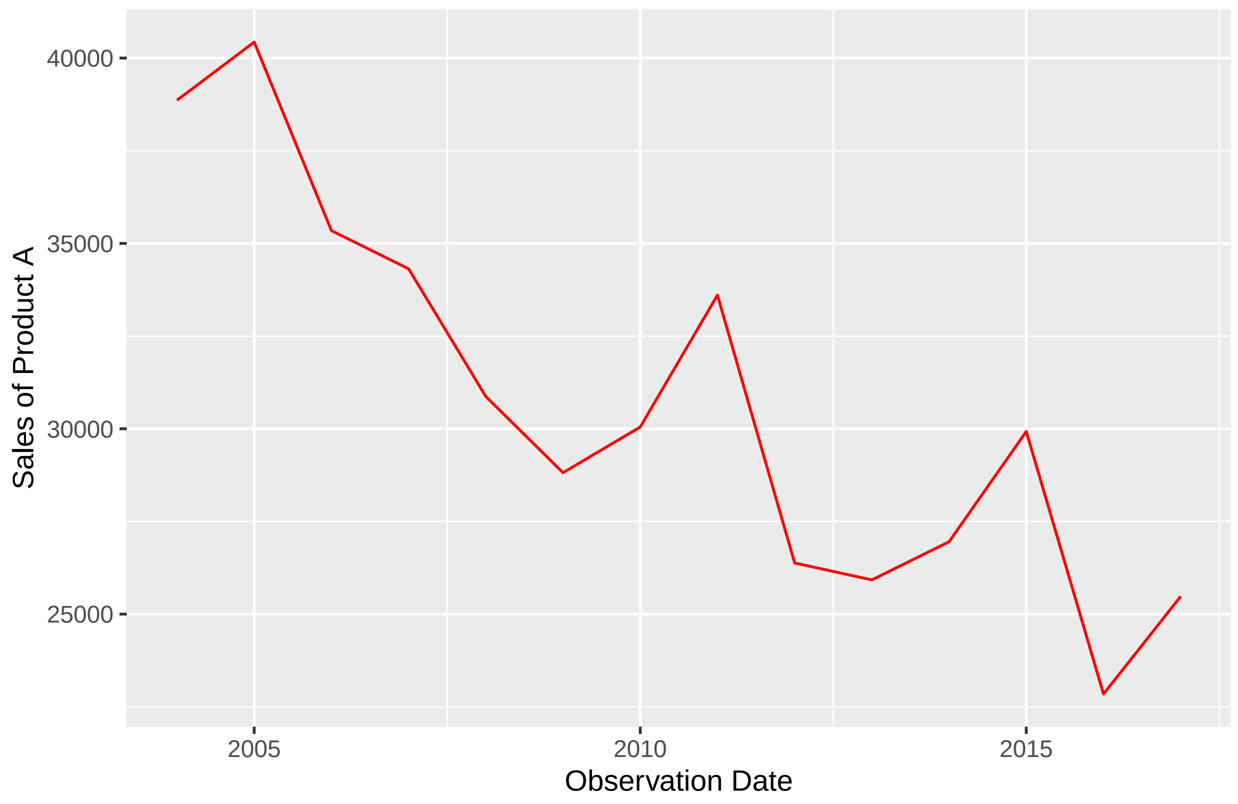
```
PROD_A <- ts(data_from_excel, start=c(2004), frequency = 1, names =c("Sales of Product A") )

str(PROD_A)
```

```
##  Time-Series [1:14, 1] from 2004 to 2017: 38862 40431 35346 34314 30870 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr "Sales of Product A"
```

```
summary(PROD_A)
```

```
##  Sales of Product A
##  Min.   :22844
##  1st Qu.:26522
##  Median :29987
##  Mean   :30700
##  3rd Qu.:34137
##  Max.   :40431
```

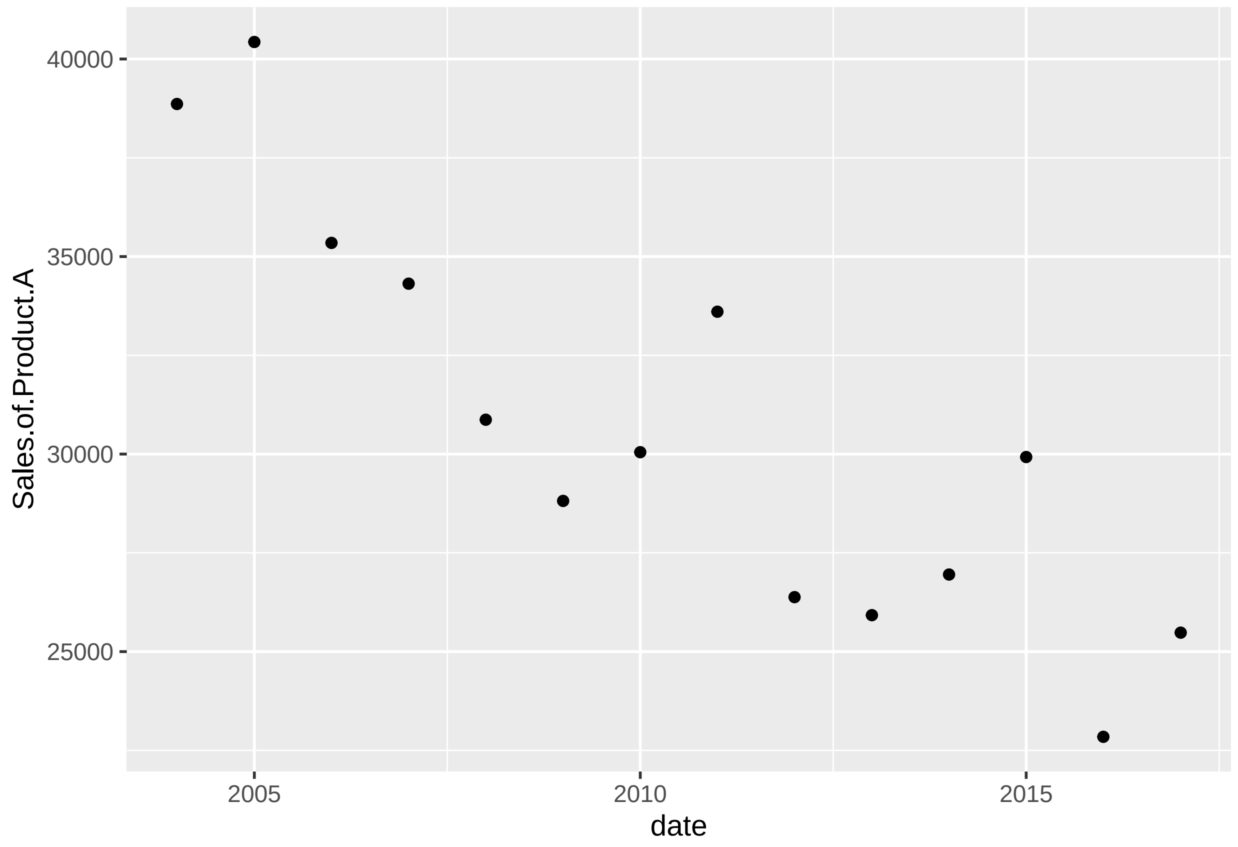
```
autoplot(PROD_A, color = 'red') +
  labs(x = "Observation Date", y = "Sales of Product A")
```



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.yearqtr(time(PROD_A))), Y=as.matrix(PROD_A))
ggplot(data=df_data, mapping=aes(x=date, y=Sales.of.Product.A))+geom_point()
```



Create forecasts using Holt smoothing

The series has a time trend but no seasonal components hence, Holt 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 exponential smoothing with the ets() function with model = "AAN".

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

```
HOLT <- ets(PROD_A, model = "AAN", alpha = .2, beta = .15)
HOLT
```

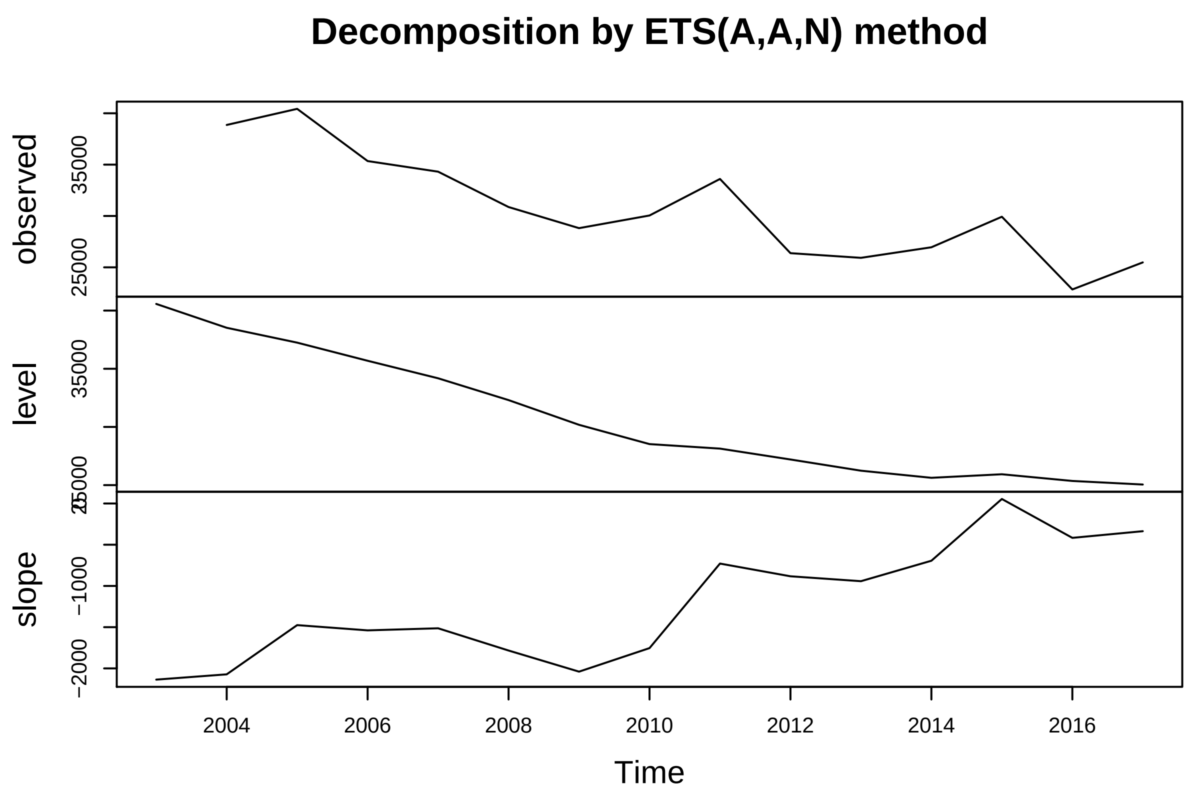
```
## ETS(A,A,N)
```

```
##
## Call:
##  ets(y = PROD_A, model = "AAN", alpha = 0.2, beta = 0.15)
##
##   Smoothing parameters:
##     alpha = 0.2
##     beta  = 0.15
##
##   Initial states:
##     l = 40572.7731
##     b = -2136.0711
##
##   sigma:  3343.141
##
##      AIC     AICc      BIC
## 265.4468 267.8468 267.3640
```

```
str(HOLT)
```

```
## List of 19
##  $ loglik    : num -130
##  $ aic       : num 265
##  $ bic       : num 267
##  $ aicc      : num 268
##  $ mse       : num 7983278
##  $ amse      : num 8719430
##  $ fit       :List of 4
##   ..$ value  : num 259
##   ..$ par    : num [1:2] 40573 -2136
##   ..$ fail   : int 0
##   ..$ fncount: int 49
##  $ residuals : Time-Series [1:14] from 2004 to 2017: 425 3982 -425 167 -1797 ...
##  $ fitted    : Time-Series [1:14, 1] from 2004 to 2017: 38437 36449 35771 34147 32667 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr "y"
##  $ states    : Time-Series [1:15, 1:2] from 2003 to 2017: 40573 38522 37246 35686 34180 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:2] "l" "b"
##  $ par       : Named num [1:4] 40572.77 -2136.07 0.2 0.15
##   ..- attr(*, "names")= chr [1:4] "l" "b" "alpha" "beta"
##  $ m         : num 1
##  $ method    : chr "ETS(A,A,N)"
##  $ series    : chr "PROD_A"
##  $ components: chr [1:4] "A" "A" "N" "FALSE"
##  $ call      : language ets(y = PROD_A, model = "AAN", alpha = 0.2, beta = 0.15)
##  $ initstate : Named num [1:2] 40573 -2136
##   ..- attr(*, "names")= chr [1:2] "l" "b"
##  $ sigma2    : num 11176589
##  $ x         : Time-Series [1:14, 1] from 2004 to 2017: 38862 40431 35346 34314 30870 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr "Sales of Product A"
##  - attr(*, "class")= chr "ets"
```

```
plot(HOLT)
```



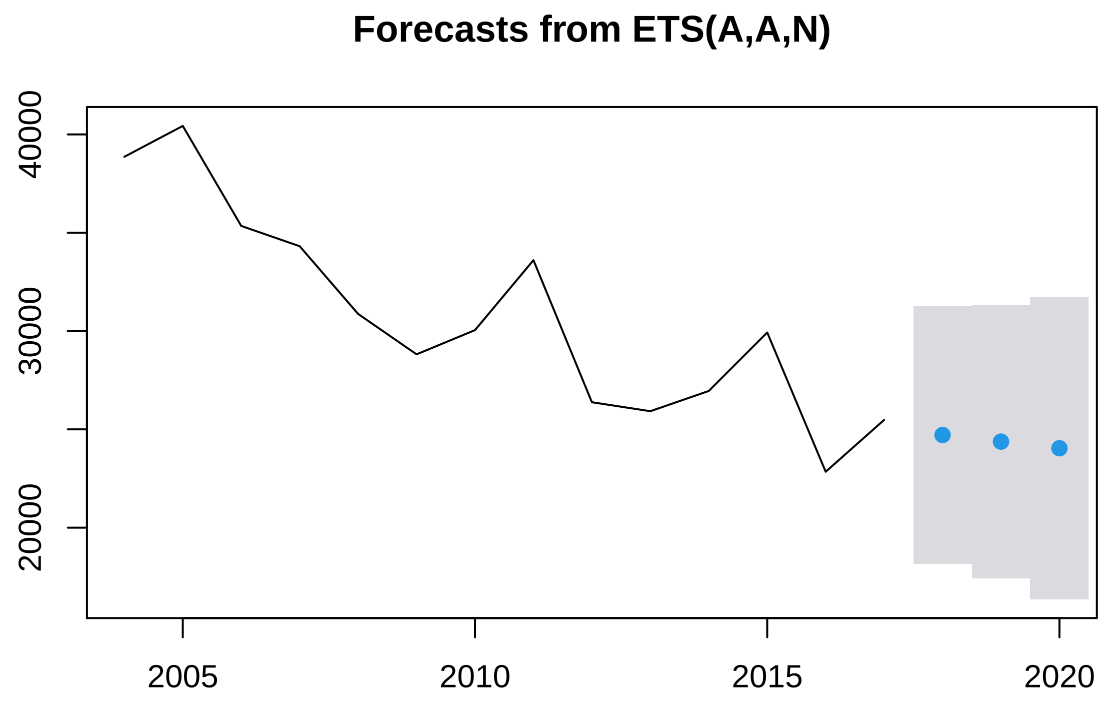
```
HOLT.pred <- forecast(HOLT, h = 3, level = c(95))
HOLT.pred
```

```
##      Point Forecast    Lo 95    Hi 95
## 2018       24712.94 18160.50 31265.37
## 2019       24376.77 17434.59 31318.95
## 2020       24040.60 16364.17 31717.02
```

```
str(HOLT.pred)
```

```
## List of 10
##  $ model     :List of 19
##   ..$ loglik    : num -130
##   ..$ aic       : num 265
##   ..$ bic       : num 267
##   ..$ aicc      : num 268
##   ..$ mse       : num 7983278
##   ..$ amse      : num 8719430
##   ..$ fit       :List of 4
##   .. ..$ value  : num 259
##   .. ..$ par    : num [1:2] 40573 -2136
##   .. ..$ fail   : int 0
##   .. ..$ fncount: int 49
##   ..$ residuals : Time-Series [1:14] from 2004 to 2017: 425 3982 -425 167 -1797 ...
##   ..$ fitted    : Time-Series [1:14, 1] from 2004 to 2017: 38437 36449 35771 34147 32667 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr "y"
##   ..$ states    : Time-Series [1:15, 1:2] from 2003 to 2017: 40573 38522 37246 35686 34180 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:2] "l" "b"
##   ..$ par       : Named num [1:4] 40572.77 -2136.07 0.2 0.15
##   .. ..- attr(*, "names")= chr [1:4] "l" "b" "alpha" "beta"
##   ..$ m         : num 1
##   ..$ method    : chr "ETS(A,A,N)"
##   ..$ series    : chr "PROD_A"
##   ..$ components: chr [1:4] "A" "A" "N" "FALSE"
##   ..$ call      : language ets(y = PROD_A, model = "AAN", alpha = 0.2, beta = 0.15)
##   ..$ initstate : Named num [1:2] 40573 -2136
##   .. ..- attr(*, "names")= chr [1:2] "l" "b"
##   ..$ sigma2    : num 11176589
##   ..$ x         : Time-Series [1:14, 1] from 2004 to 2017: 38862 40431 35346 34314 30870 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr "Sales of Product A"
##   ..- attr(*, "class")= chr "ets"
##  $ mean     : Time-Series [1:3] from 2018 to 2020: 24713 24377 24041
##  $ level    : num 95
##  $ x        : Time-Series [1:14, 1] from 2004 to 2017: 38862 40431 35346 34314 30870 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr "Sales of Product A"
##  $ upper    : Time-Series [1:3, 1] from 2018 to 2020: 31265 31319 31717
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr "95%"
##  $ lower    : Time-Series [1:3, 1] from 2018 to 2020: 18161 17435 16364
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr "95%"
##  $ fitted   : Time-Series [1:14, 1] from 2004 to 2017: 38437 36449 35771 34147 32667 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr "y"
##  $ method   : chr "ETS(A,A,N)"
##  $ series   : chr "PROD_A"
##  $ residuals: Time-Series [1:14] from 2004 to 2017: 425 3982 -425 167 -1797 ...
##  - attr(*, "class")= chr "forecast"
```

```
plot(HOLT.pred)
```



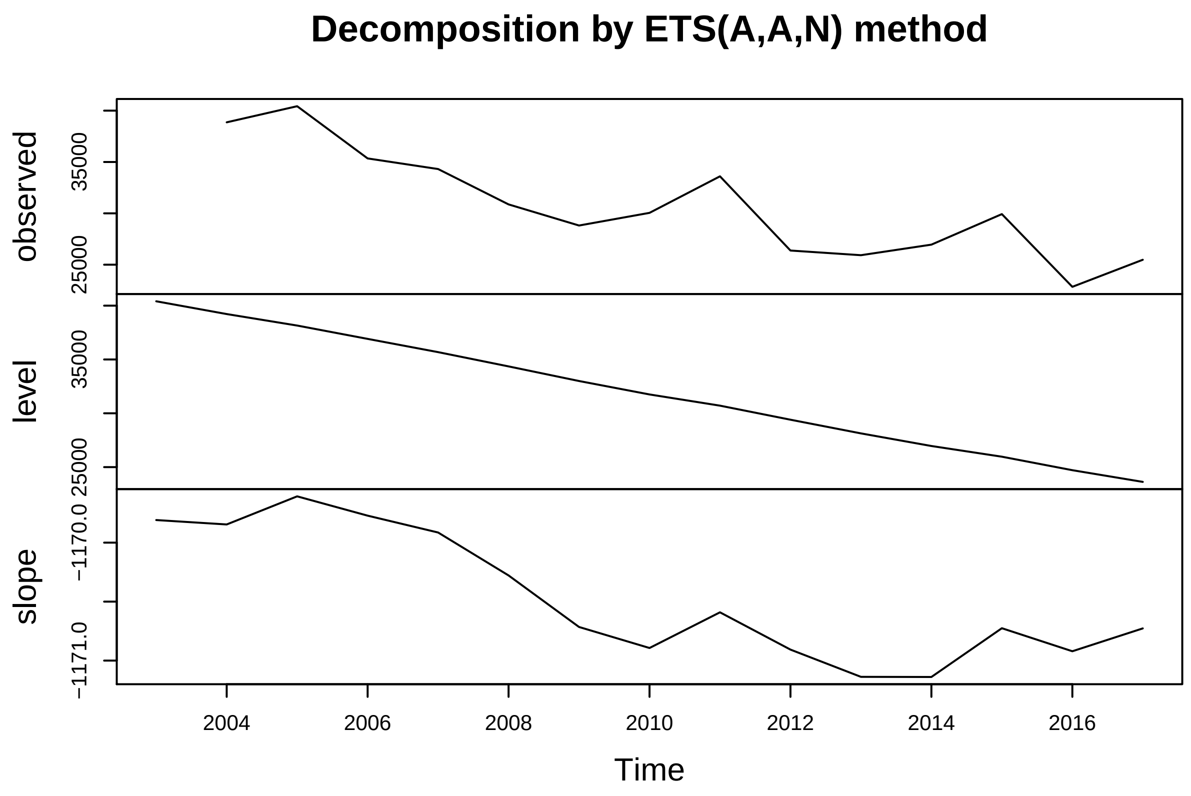
do the same things with the automatic procedure

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

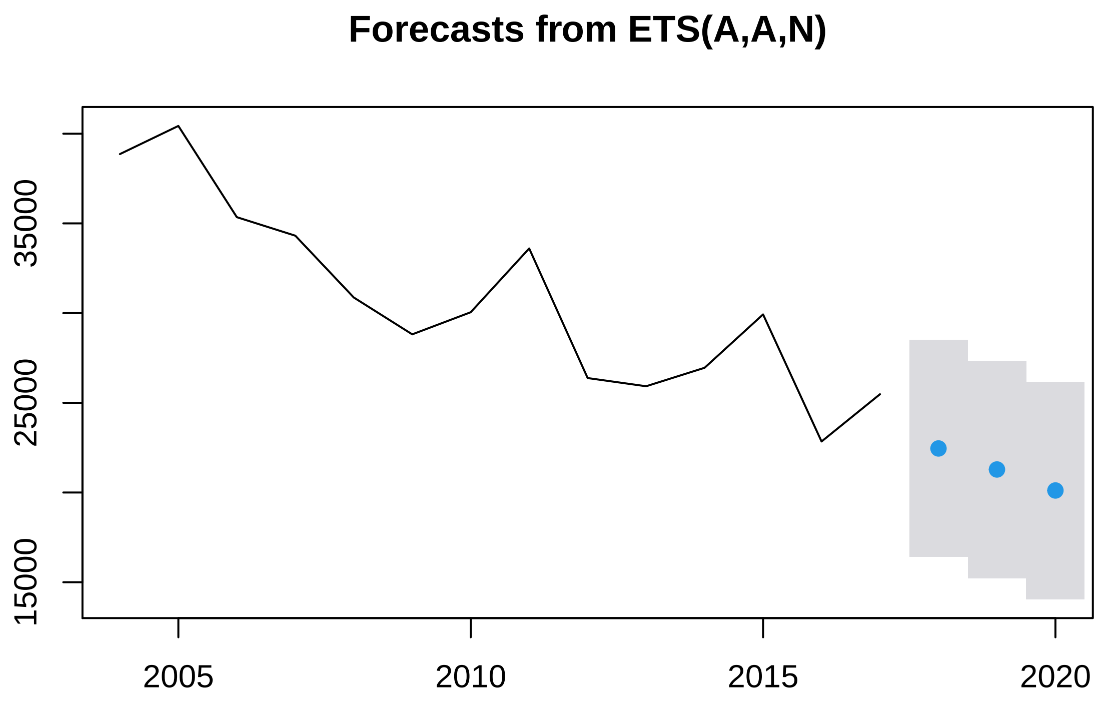
```
HOLT_auto <- ets(PROD_A, model = "AAN")
HOLT_auto
```

```
## ETS(A,A,N)
##
## Call:
##  ets(y = PROD_A, model = "AAN")
##
##   Smoothing parameters:
##     alpha = 0.0428
##     beta  = 1e-04
##
##   Initial states:
##     l = 40405.1948
##     b = -1169.8083
##
##   sigma:  3085.037
##
##      AIC     AICc      BIC
## 267.1971 274.6971 270.3924
```

```
plot(HOLT_auto)
```



```
HOLT_auto.pred <- forecast(HOLT_auto, h = 3, level = c(95))
plot(HOLT_auto.pred)
```



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

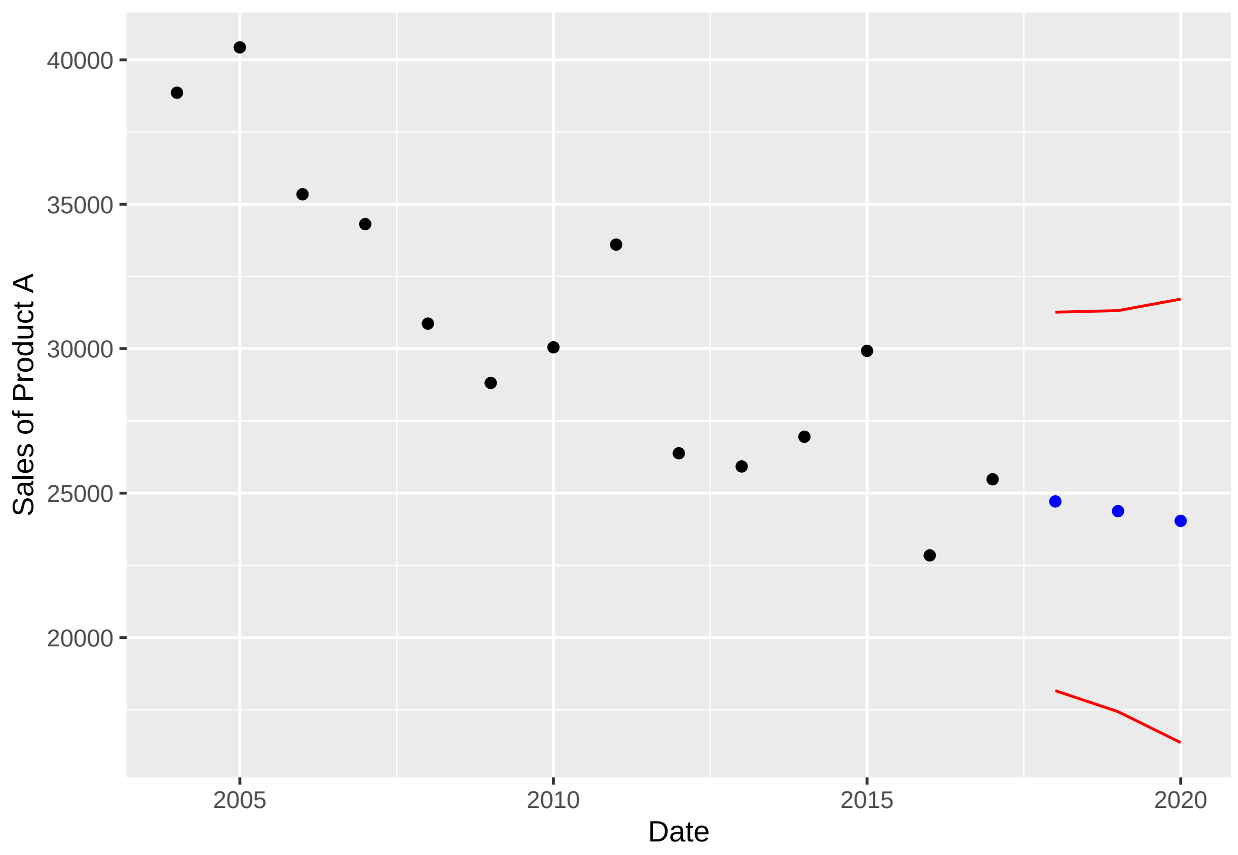
The next chuck of R code saves the forecasts to excel. This will allow you to share your forecasts with others.

```
col_names <- as.yearmon(time(HOLT.pred$mean))
my_df <-  data.frame( as.Date(col_names), HOLT.pred$mean, HOLT.pred$lower, HOLT.pred$upper)
colnames(my_df) <- c("Date", "Forecast", "95% Lower", "95% Upper" )
write_xlsx(my_df, "Annual_Sales_A_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(aes(x = Date, y = `95% Lower`), color="red" ) +
  geom_line(aes(x = Date, y = `95% Upper`), color="red" ) +
  geom_point(data=df_data, mapping=aes(x=date, y=Sales.of.Product.A))+
  labs(x = "Date", y = "Sales of Product A")
```


40000
35000
30000
25000
20000
Sales of Product A
2005
2010
2015
2020
Date