Exponential Smoothing

Fallaw Sowell

3/14/2021

Contents

Simple Exponential Smoothing - Price of Natural Gas

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 simple 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.

This data are the monthly price of natural gas from the FRED database at the St. Louis Fed.

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

Get familiar with your data, variables, size, etc.

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

```
str( data_from_excel )
```

```
## tibble [290 x 1] (S3: tbl_df/tbl/data.frame)
##  $ NG: num [1:290] 3.45 2.15 1.89 2.03 2.25 2.2 2.19 2.49 2.88 3.07 ...
```

Make NG a monthly time series

The next chunck of R code, creates that time series data structure for the natural gas prices, 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.

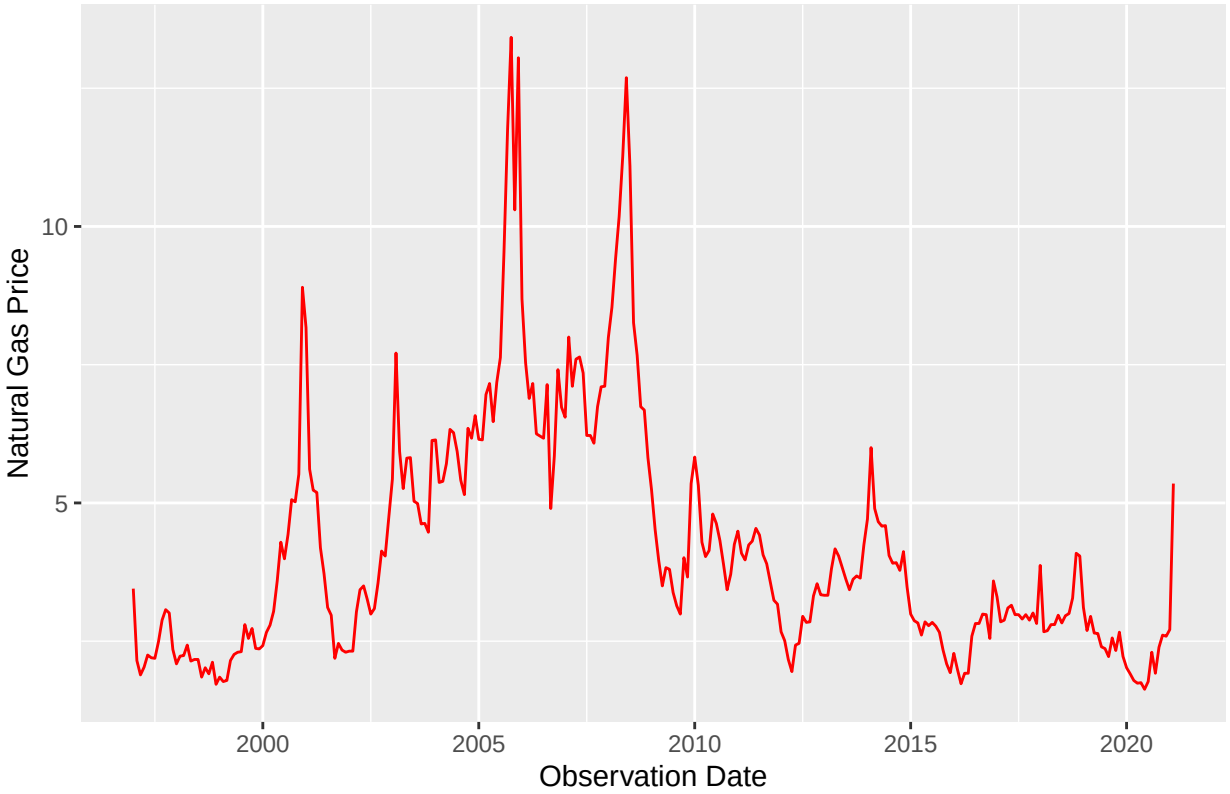
```
NG <- ts(data_from_excel, start=c(1997,1), frequency = 12, names =c("Natural Gas Price") )

str(NG)
```

```
##  Time-Series [1:290, 1] from 1997 to 2021: 3.45 2.15 1.89 2.03 2.25 2.2 2.19 2.49 2.88 3.07 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr "Natural Gas Price"
```

```
summary(NG)
```

```
##  Natural Gas Price
##  Min.   : 1.630
##  1st Qu.: 2.650
##  Median : 3.520
##  Mean   : 4.181
##  3rd Qu.: 5.305
##  Max.   :13.420
```

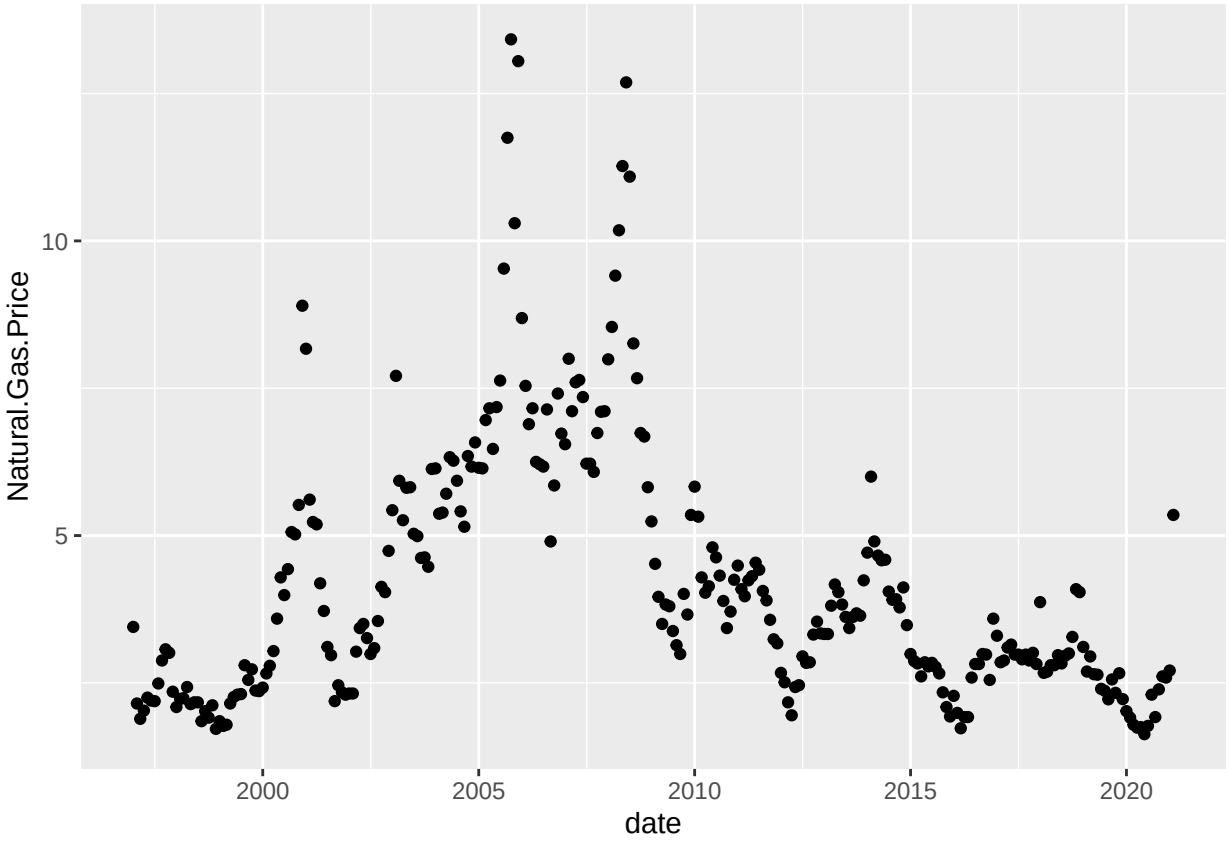
```
autoplot(NG, color = 'red') +
  labs(x = "Observation Date", y = "Natural Gas Price")
```



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(NG))), Y=as.matrix(NG))
ggplot(data=df_data, mapping=aes(x=date, y=Natural.Gas.Price))+geom_point()
```



Create forecasts using simple exponential smoothing

The series has no trend and no seasonal components hence, simple exponential 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

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

```
SES <- ets(NG, model = "ANN", alpha = .2)

SES
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = NG, model = "ANN", alpha = 0.2)
##
##   Smoothing parameters:
##     alpha = 0.2
##
##   Initial states:
```

```
##     l = 2.4677
##
##   sigma:  1.1204
##
##      AIC     AICc      BIC
## 1712.212 1712.253 1719.551
```

Look at the data structure of the output

The following chuck of R code shows the data structure that is saved by ets().

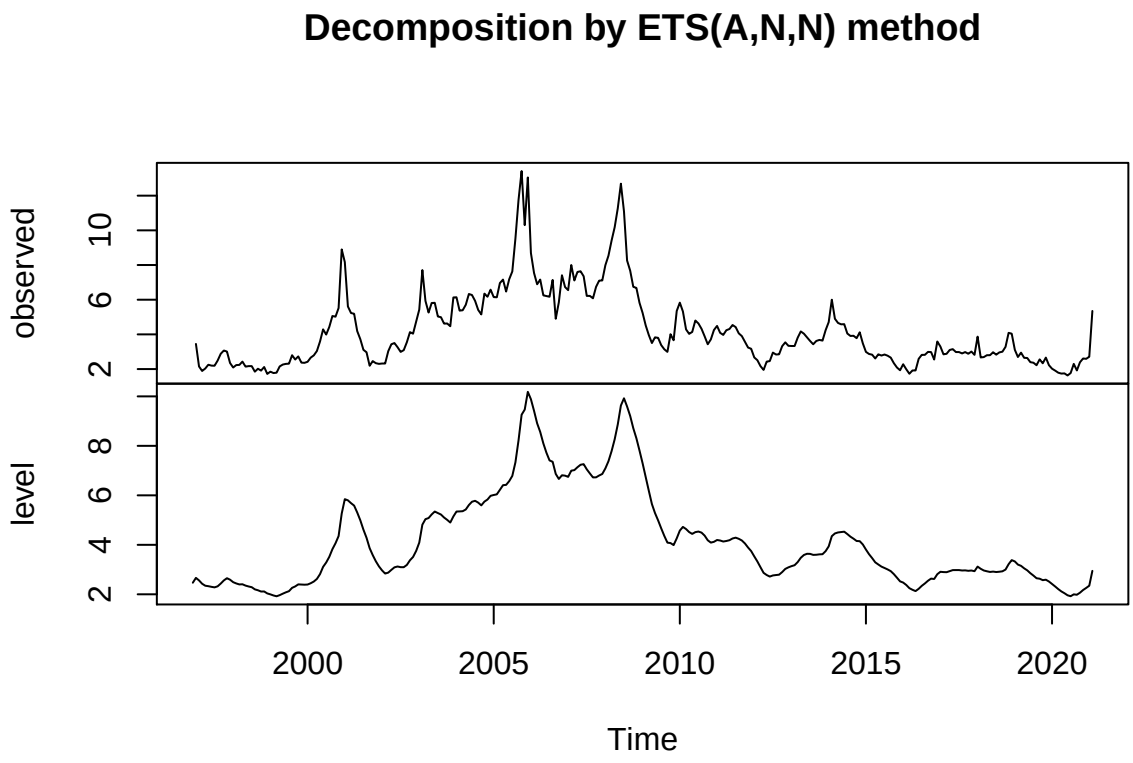
```
str(SES)
```

```
## List of 19
##  $ loglik    : num -854
##  $ aic       : num 1712
##  $ bic       : num 1720
##  $ aicc      : num 1712
##  $ mse       : num 1.25
##  $ amse      : num 1.66
##  $ fit       :List of 4
##   ..$ value  : num 1708
##   ..$ par    : num 2.47
##   ..$ fail   : int 0
##   ..$ fncount: int 12
##  $ residuals : Time-Series [1:290] from 1997 to 2021: 0.9823 -0.5142 -0.6713 -0.3971 -0.0976 ...
##  $ fitted    : Time-Series [1:290, 1] from 1997 to 2021: 2.47 2.66 2.56 2.43 2.35 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr "y"
##  $ states    : Time-Series [1:291, 1] from 1997 to 2021: 2.47 2.66 2.56 2.43 2.35 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr "l"
##  $ par       : Named num [1:2] 2.47 0.2
##   ..- attr(*, "names")= chr [1:2] "l" "alpha"
##  $ m         : num 12
##  $ method    : chr "ETS(A,N,N)"
##  $ series    : chr "NG"
##  $ components: chr [1:4] "A" "N" "N" "FALSE"
##  $ call      : language ets(y = NG, model = "ANN", alpha = 0.2)
##  $ initstate : Named num 2.47
##   ..- attr(*, "names")= chr "l"
##  $ sigma2    : num 1.26
##  $ x         : Time-Series [1:290, 1] from 1997 to 2021: 3.45 2.15 1.89 2.03 2.25 2.2 2.19 2.49 2.88
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr "Natural Gas Price"
##  - attr(*, "class")= chr "ets"
```

plot the output

The next chuck of R code plots the data and the fitted values.

```
plot(SES)
```



Create predictions

The next chuck of R code generates and the displays the forecasts.

```
SES.pred <- forecast(SES, h = 18, level = c(95))


SES.pred
```

```
##          Point Forecast     Lo 95    Hi 95
## Mar 2021       2.947759 0.7517570 5.143760
## Apr 2021       2.947759 0.7082676 5.187250
## May 2021       2.947759 0.6656068 5.229910
## Jun 2021       2.947759 0.6237290 5.271788
## Jul 2021       2.947759 0.5825925 5.312925
## Aug 2021       2.947759 0.5421594 5.353358
## Sep 2021       2.947759 0.5023947 5.393123
## Oct 2021       2.947759 0.4632664 5.432251
## Nov 2021       2.947759 0.4247449 5.470772
## Dec 2021       2.947759 0.3868027 5.508715
## Jan 2022       2.947759 0.3494145 5.546103
## Feb 2022       2.947759 0.3125567 5.582961
```

```
## Mar 2022       2.947759 0.2762074 5.619310
## Apr 2022       2.947759 0.2403460 5.655171
## May 2022       2.947759 0.2049535 5.690564
## Jun 2022       2.947759 0.1700119 5.725505
## Jul 2022       2.947759 0.1355044 5.760013
## Aug 2022       2.947759 0.1014152 5.794102
```

look at the data structure of the predictions

The next chunk of R code displays the structure of the data created by forecast()

```
str(SES.pred)
```

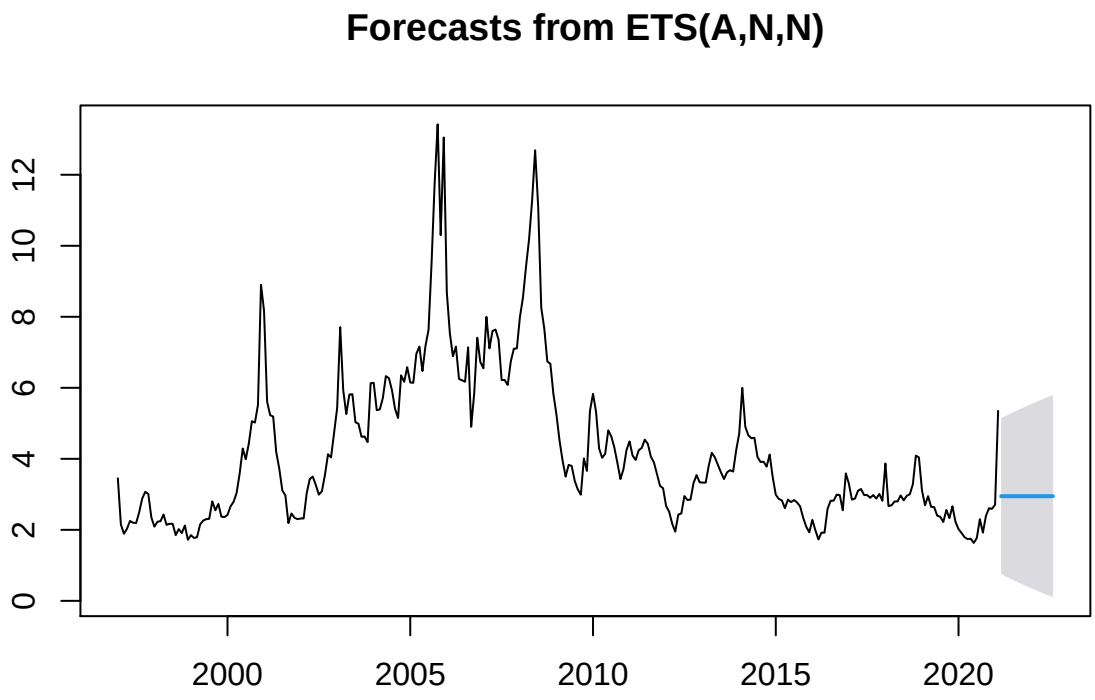
```
## List of 10
##  $ model    :List of 19
##   ..$ loglik    : num -854
##   ..$ aic       : num 1712
##   ..$ bic       : num 1720
##   ..$ aicc      : num 1712
##   ..$ mse       : num 1.25
##   ..$ amse      : num 1.66
##   ..$ fit       :List of 4
##   .. ..$ value  : num 1708
##   .. ..$ par    : num 2.47
##   .. ..$ fail   : int 0
##   .. ..$ fncount: int 12
##   ..$ residuals : Time-Series [1:290] from 1997 to 2021: 0.9823 -0.5142 -0.6713 -0.3971 -0.0976 ...
##   ..$ fitted    : Time-Series [1:290, 1] from 1997 to 2021: 2.47 2.66 2.56 2.43 2.35 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr "y"
##   ..$ states    : Time-Series [1:291, 1] from 1997 to 2021: 2.47 2.66 2.56 2.43 2.35 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr "l"
##   ..$ par       : Named num [1:2] 2.47 0.2
##   .. ..- attr(*, "names")= chr [1:2] "l" "alpha"
##   ..$ m         : num 12
##   ..$ method    : chr "ETS(A,N,N)"
##   ..$ series    : chr "NG"
##   ..$ components: chr [1:4] "A" "N" "N" "FALSE"
##   ..$ call      : language ets(y = NG, model = "ANN", alpha = 0.2)
##   ..$ initstate : Named num 2.47
##   .. ..- attr(*, "names")= chr "l"
##   ..$ sigma2    : num 1.26
##   ..$ x         : Time-Series [1:290, 1] from 1997 to 2021: 3.45 2.15 1.89 2.03 2.25 2.2 2.19 2.49 2
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr "Natural Gas Price"
##   ..- attr(*, "class")= chr "ets"
##  $ mean     : Time-Series [1:18] from 2021 to 2023: 2.95 2.95 2.95 2.95 2.95 ...
##  $ level    : num 95
##  $ x        : Time-Series [1:290, 1] from 1997 to 2021: 3.45 2.15 1.89 2.03 2.25 2.2 2.19 2.49 2.88 3
```

```
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr "Natural Gas Price"
##  $ upper    : Time-Series [1:18, 1] from 2021 to 2023: 5.14 5.19 5.23 5.27 5.31 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr "95%"
##  $ lower    : Time-Series [1:18, 1] from 2021 to 2023: 0.752 0.708 0.666 0.624 0.583 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr "95%"
##  $ fitted   : Time-Series [1:290, 1] from 1997 to 2021: 2.47 2.66 2.56 2.43 2.35 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr "y"
##  $ method   : chr "ETS(A,N,N)"
##  $ series   : chr "NG"
##  $ residuals: Time-Series [1:290] from 1997 to 2021: 0.9823 -0.5142 -0.6713 -0.3971 -0.0976 ...
##  - attr(*, "class")= chr "forecast"
```

Plot the predictions

The next chuck of R code plots the data, the forecasts and the confidence intervals.

```
plot(SES.pred)
```



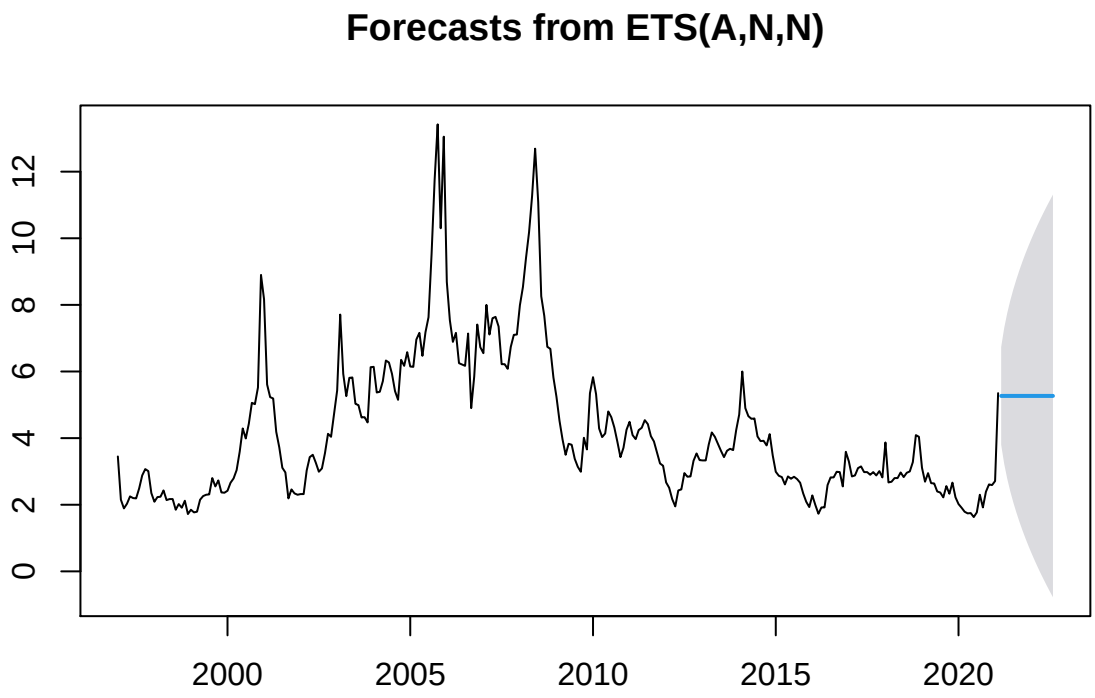
do the same things with the automatic procedure

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

```
SES_auto <- ets(NG, model="ANN")
SES_auto
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = NG, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 0.9685
##
##   Initial states:
##     l = 3.4089
##
##   sigma:  0.7491
##
##      AIC     AICc      BIC
## 1480.714 1480.798 1491.724
```

```
SES_auto.pred <- forecast(SES_auto, h = 18, level = c(95))
plot(SES_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(SES.pred$mean), "%m-%Y")
my_df <-  data.frame( as.Date(col_names), SES.pred$mean, SES.pred$lower, SES.pred$upper)
colnames(my_df) <- c("Date", "Forecast", "95% Lower", "95% Upper")
write_xlsx(my_df, "NG_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=Natural.Gas.Price))+
  labs(x = "Date", y = "Natural Gas Price")
```