Module 7 - Nonlinear and Heteroscedastic Innovations State Space Models

MATH1307 Forecasting

RMIT University, School of Science, Mathematical Sciences

All the contents in this presentation are mainly based on the textbook of MATH1307 Forecasting course 'Hyndman et al., Forecasting with Exponential Smoothing: The State Space Approach. Springer, 2008.' Other resources than the textbook are cited accordingly.

Introduction

In this module, we will study how to deal with heteroscedasticity using nonlinear innovations state-space models, which is a broader class of innovations state-space models.

This class of models enables us to examine multiplicative structures for any or all of the trend, the seasonal pattern and the innovations process.

We will focus on

- the general innovations form of the state space model,
- various special cases,
- seasonal models, and
- several variations on the core models.

Innovations Form of the General State Space Model

In the innovations state-space models, we select the elements of the state vector to describe the trend and seasonal elements of the series, using these terms as building blocks to enable us to formulate a model that captures the key components of the data generating process.

The general model involves a state vector

$$\boldsymbol{X}_t = (\ell_t, b_t, s_t, s_{t-1}, \ldots, s_{t-m+1})' \tag{1}$$

and state-space equations of the form

$$\begin{aligned} Y_t &= w(\boldsymbol{X}_{t-1}) + r(\boldsymbol{X}_{t-1})\epsilon_t, \\ \boldsymbol{X}_t &= \boldsymbol{f}(\boldsymbol{X}_{t-1}) + \boldsymbol{g}(\boldsymbol{X}_{t-1})\epsilon_t, \end{aligned} \tag{2}$$

where $\omega(\cdot)$ an $r(\cdot)$ are scalar functions and $\boldsymbol{f}(\cdot)$ and $\boldsymbol{g}(\cdot)$ are vector functions, and ϵ_t is a white noise process with variance $\boldsymbol{\sigma}^2$.

In the previous module, the functions r and $\boldsymbol{g}$ were constants, whereas ω and $\boldsymbol{f}$ were linear in the state vector.

To get the simplest nonlinear form of the model, we set $w(\boldsymbol{X}_{t-1}) = r(\boldsymbol{X}_{t-1}) = f(\boldsymbol{X}_{t-1}) = \ell_{t-1}$ and $g(\boldsymbol{X}_{t-1}) = \alpha\ell_{t-1}$ and obtain the following:

$$\begin{aligned} Y_t &= \ell_{t-1}(1+\epsilon_t), \\ \ell_t &= \ell_{t-1}(1+\alpha\epsilon_t). \end{aligned} \tag{3}$$

This model corresponds to ETS(M,N,N) or the simple exponential smoothing model with multiplicative errors.

In this model, we can eliminate ϵ_t and find the following recursive relationship:

$$\ell_t = \alpha Y_t + (1-\alpha)\ell_{t-1}. \tag{4}$$

The recursive relationship for ETS(M,N,N) is thus seen to be identical to that for ETS(A,N,N).

However, the reduced form equations are different, showing that the predictive distributions (and hence the prediction intervals) will differ.

Basic Special Cases

The models in the ETS(M,$*$,$*$) class produce the same point forecasts as those in the ETS(A,$*$,$*$) class because we are deploying the same recursive relationships.

The stochastic elements of the process determine whether to use the *additive* or *multiplicative* version.

- If the error process is homoscedastic, the constant variance assumptions are appropriate, and the prediction intervals for h-steps ahead have constant widths regardless of the current level of the process. So, we can use the approach of the additive error.
- If the process is heteroscedastic, and in particular, the error variance is proportional to the current level of the process, the nonlinear schemes introduced are appropriate. So, we can use the approach of the multiplicative error.

Local Level Model: ETS(M,N,N)

This model is described in (2). From Eq. (2), we observe that the state equation reveals that the quantity $\alpha\ell_{t-1}\epsilon_t$ has a persistent effect, feeding into the expected level for the next time period.

We have the following extreme cases for this model:

- If $\alpha = 0$, multiplicative and additive models become identical.
- When $\alpha = 1$, the model reverts to a form of random walk with the reduced form $Y_t = Y_{t-1}(1 + \epsilon_t)$; hence, the complete effect of the random error is passed on to the next period.

The one-step-ahead predictions are given by

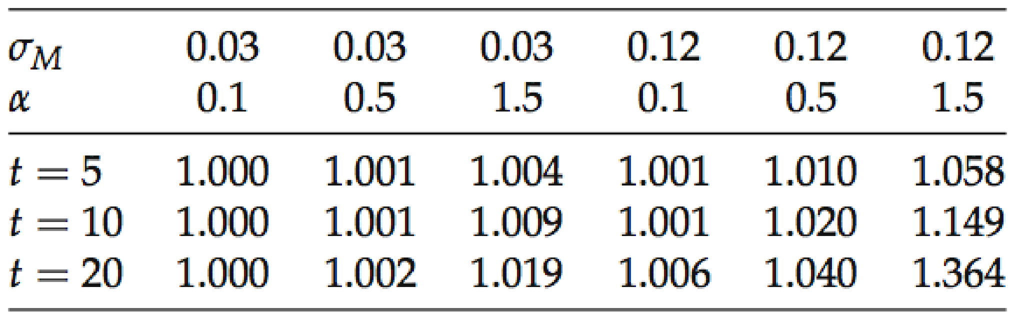
$$\hat{Y}_{t+1|t} = (1-\alpha)^t \ell_0 + \alpha \sum_{j=0}^{t-1} (1-\alpha)^j Y_{t-j}. \tag{5}$$

When $0, \alpha < 2$, the stability condition is satisfied.

To illustrate the difference between NN model with additive and multiplicative errors, we consider their conditional variances.

$$\begin{aligned} V_A(Y_t|X_0) &= \sigma_A^2[1 + (t-1)\alpha^2], \\ V_M(Y_t|X_0) &= X_0^2[(1+\sigma_M^2)(1+\alpha^2\sigma_M^2)^{t-1} - 1]. \end{aligned} \tag{6}$$

If we set $\sigma_A = X_0\sigma_M$ and compute the ratio of variances V_M/V_A, we get the following table for various values of t and α.

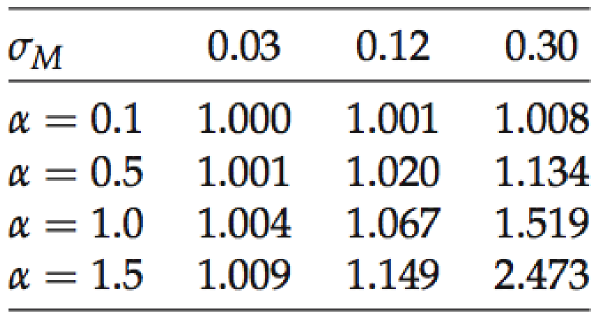


σ_M	0.03	0.03	0.03	0.12	0.12	0.12
α	0.1	0.5	1.5	0.1	0.5	1.5
$t = 5$	1.000	1.001	1.004	1.001	1.010	1.058
$t = 10$	1.000	1.001	1.009	1.001	1.020	1.149
$t = 20$	1.000	1.002	1.019	1.006	1.040	1.364

The difference between fitting with additive or multiplicative errors become more apparent as t, α and σ_M increase.

When $\sigma_M = 0.30$, the multiplicative error has a mean that is about three times its standard deviation, and the differences become noticeable quite quickly.

See the results for $t = 10$ in the following table:



σ_M	0.03	0.12	0.30
$\alpha = 0.1$	1.000	1.001	1.008
$\alpha = 0.5$	1.001	1.020	1.134
$\alpha = 1.0$	1.004	1.067	1.519
$\alpha = 1.5$	1.009	1.149	2.473

So, in case of a heteroskedasticity in the series use of multiplicative errors approach makes sense in terms of the quality of forecasts.

For example, consider stock price volatility. Based on the efficient market hypothesis, we would expect that $\alpha = 1$.

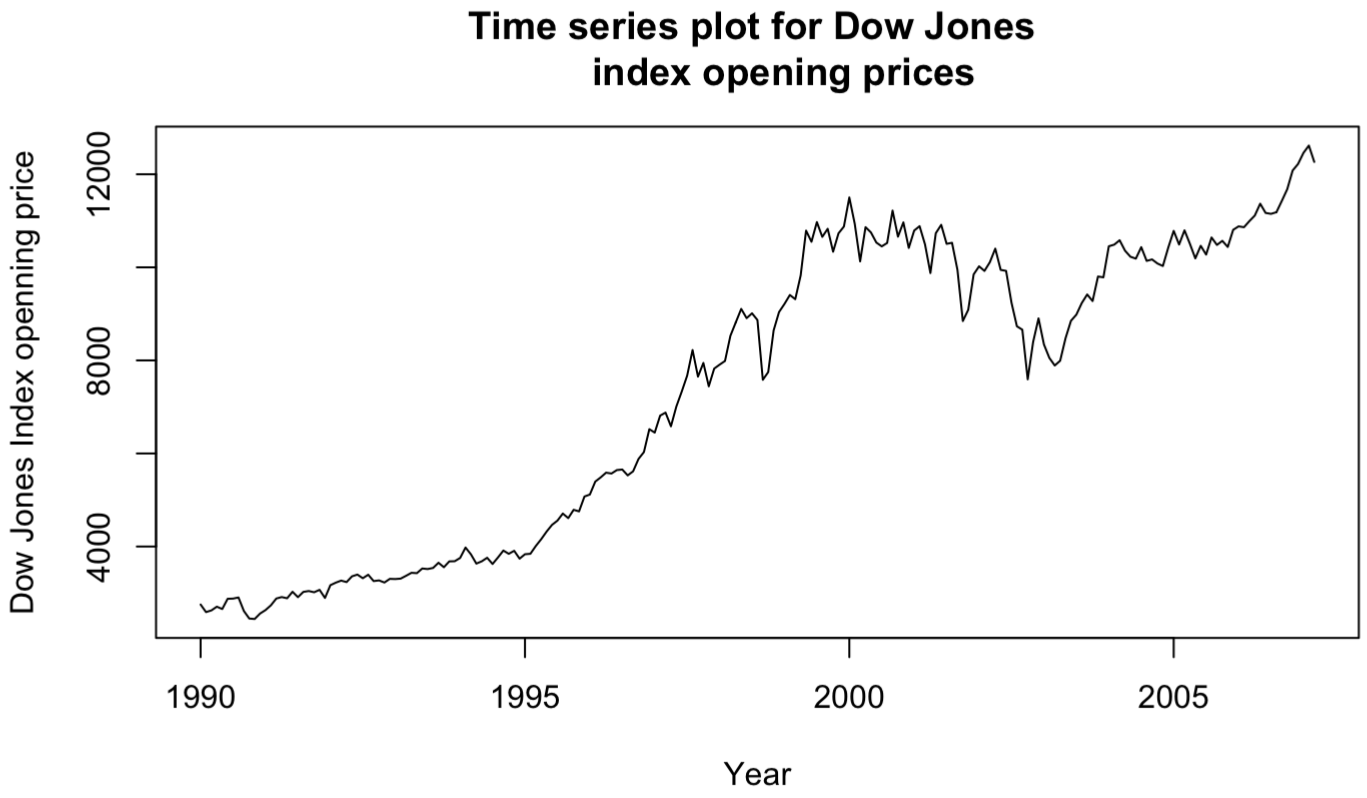
The process might be observed at hourly or even minute intervals, yet the purpose behind the modelling would be to evaluate the volatility (essentially as measured by the variance) over much longer periods.

So, both α and t will have large values as in the example tables above.

In this case, use of the additive approach instead of the multiplicative one could lead to considerable underestimation of the risks involved.

Let’s analyse monthly Dow Jones index opening series using MNN model. The following display is the time series plot of monthly Dow Jones index opening series.

```
data("dji")
dji.open = dji[,"Open"]
plot(dji.open,ylab = "Dow Jones Index openning price", xlab="Year", main="Time series plot
     index openning prices" )
```


Time series plot for Dow Jones
index opening prices
Dow Jones Index openning price
4000
8000
12000
1990
1995
2000
2005
Year

It’s clear from this time series plot that opening index series has multiple trends and changing variance.

Seasonality is not obvious from the time series plot. We conclude that there is a volatility clustering exists in this series.

So we expect to get better performance from the MNN model than the ANN model. We fit both models with the following code chunk.

```
fit.dji.MNN = ets(dji.open, model="MNN")
summary(fit.dji.MNN)
fit.dji.ANN = ets(dji.open, model="ANN")
summary(fit.dji.ANN)
```

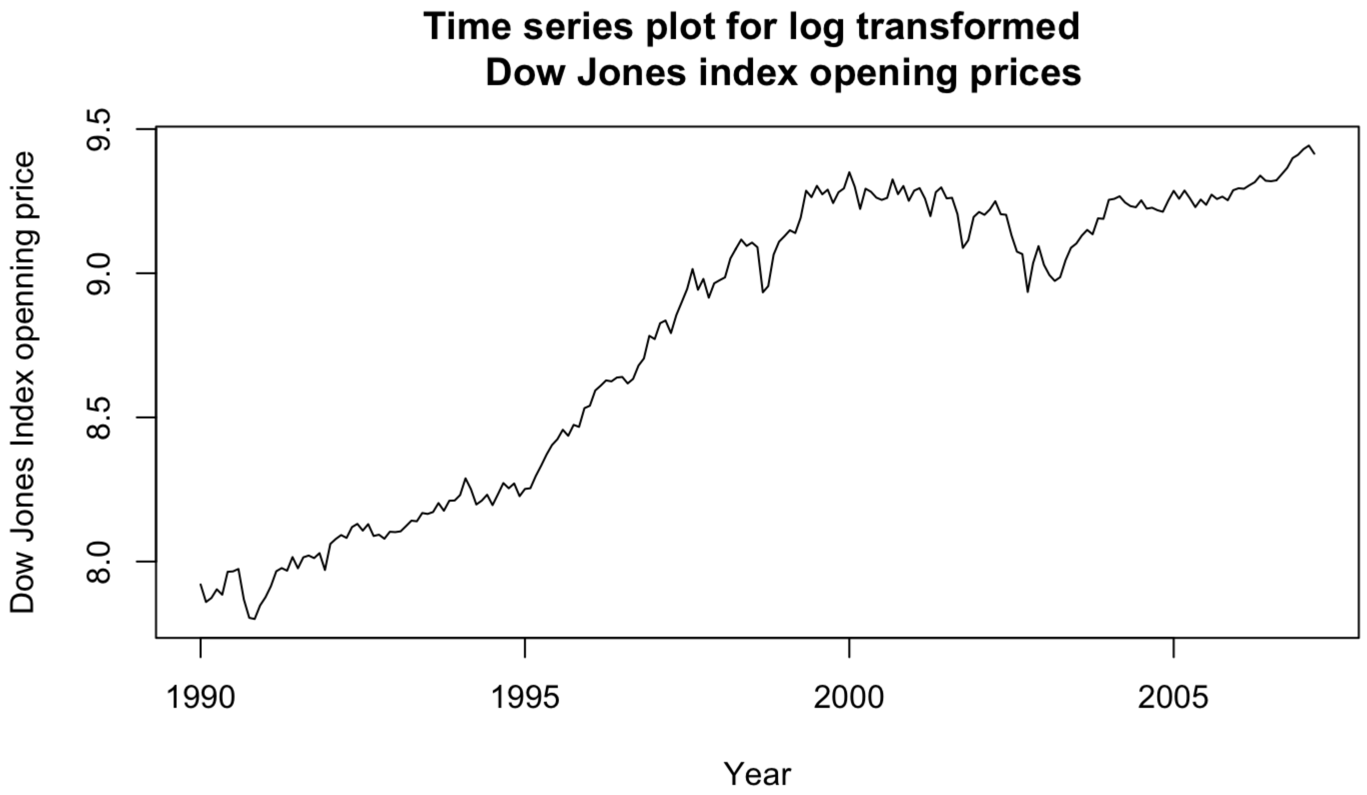
```
## ETS(M,N,N)
##
## Call:
##  ets(y = dji.open, model = "MNN")
##
##   Smoothing parameters:
##     alpha = 0.9601
##
##   Initial states:
##     l = 2742.1249
##
##   sigma:  0.0411
##
##      AIC     AICc      BIC
## 3419.885 3420.003 3429.883
##
## Training set error measures:
##                    ME     RMSE     MAE       MPE     MAPE      MASE
## Training set 47.98735 327.2698 234.174 0.6668563 3.159344 0.2666724
##                     ACF1
## Training set -0.02755172
## ETS(A,N,N)
##
## Call:
##  ets(y = dji.open, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 0.9536
##
##   Initial states:
##     l = 2737.5879
##
##   sigma:  328.8567
##
##      AIC     AICc      BIC
## 3507.251 3507.369 3517.249
##
## Training set error measures:
##                    ME     RMSE      MAE       MPE    MAPE      MASE
## Training set 48.35058 327.2642 234.1805 0.6723869 3.16028 0.2666797
##                     ACF1
## Training set -0.02148202
```

As expected, we got better model performance for MNN model in terms of AIC, BIC, RMSE, and MASE.

To deal with changing variance, we could apply logarithmic transformation and see if it helps.

The following display shows the time series plot of log-transformed opening series.

```
dji.open.log = log(dji.open)
plot(dji.open.log,ylab = "Dow Jones Index openning price", xlab="Year", main="Time series
     Dow Jones index openning prices" )
```


Time series plot for log transformed
Dow Jones index opening prices
Dow Jones Index openning price
9.5
9.0
8.5
8.0
1990
1995
2000
2005
Year

It seems that the log transformation helps to stabilize the variance.

To apply log transformation before model fitting, we can use the argument `lambda = 0` in `ets()` function.

However, this is limited to only models with additive errors.

For multiplicative errors approach, we can apply log transformation before feeding data into the model.

The following code chunk demonstrates both approaches.

```r
fit.dji.MNN.log = ets(dji.open.log, model="MNN")
# Log transformed data is used as the model is multiplicative
summary(fit.dji.MNN.log)
fit.dji.ANN.log = ets(dji.open, model="ANN",lambda=0)
# Let the ets() apply log transformation for us
summary(fit.dji.ANN.log)
```

```
## ETS(M,N,N)
##
## Call:
##  ets(y = dji.open.log, model = "MNN")
##
##   Smoothing parameters:
##     alpha = 0.9792
##
##   Initial states:
##     l = 7.9191
##
##   sigma:  0.0047
##
##       AIC      AICc       BIC
## -214.0210 -213.9028 -204.0229
##
## Training set error measures:
##                       ME       RMSE        MAE        MPE      MAPE
## Training set 0.007380713 0.04097398 0.03166486 0.08425936 0.3608663
##                   MASE       ACF1
## Training set 0.2530142 -0.0350858
## ETS(A,N,N)
##
## Call:
##  ets(y = dji.open.log, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 0.9789
##
##   Initial states:
##     l = 7.9192
##
##   sigma:  0.0412
##
##       AIC      AICc       BIC
## -212.7820 -212.6637 -202.7838
##
## Training set error measures:
##                       ME       RMSE        MAE        MPE      MAPE
## Training set 0.007382591 0.04097397 0.03166415 0.08427989 0.3608577
##                   MASE        ACF1
## Training set 0.2530086 -0.03472173
```

Even with a more stable variance, we got better to fit from the MNN model than the ANN model.

Forecasts for MNN model are shown in the following display.

```
frc.dji.MNN = forecast(fit.dji.MNN , h=1)
print(frc.dji.MNN)
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Apr 2007       12279.36 11631.82 12926.89 11289.04 13269.67
```

```r
frc.dji.MNN.log = forecast(fit.dji.MNN.log , h=1)
# To take the log transformation back, we will apply the back
# transformation to all elements
frc.dji.MNN.log$mean<-exp(frc.dji.MNN.log$mean)
frc.dji.MNN.log$upper<-exp(frc.dji.MNN.log$upper)
frc.dji.MNN.log$lower<-exp(frc.dji.MNN.log$lower)
frc.dji.MNN.log$x<-exp(frc.dji.MNN.log$x)

print(frc.dji.MNN.log)
```

```
##          Point Forecast    Lo 80   Hi 80    Lo 95    Hi 95
## Apr 2007       12272.74 11598.03 12986.7 11256.01 13381.31
```

Notice also that because the mean absolute scaled error (MASE) is independent of the scale of the data, we can compare two MNN models with and without log-transformed data using MASE.

So, MASE is 0.266 for the log-transformed version while it is 0.361 for the non-transformed version.

We can conclude that the MNN model with log-transformed data is the best choice for the opening series within the set of considered models.

Local Trend Model: ETS(M,A,N)

This model corresponds to Holt's linear model with multiplicative errors.

We upgrade the local level model by adding an growth rate b_t as follows:

$$\begin{aligned} Y_t &= (\ell_{t-1} + b_{t-1})(1 + \epsilon_t), \\ \ell_t &= (\ell_{t-1} + b_{t-1})(1 + \alpha\epsilon_t), \\ b_t &= b_{t-1} + \beta(\ell_{t-1} + b_{t-1})\epsilon_t. \end{aligned} \tag{7}$$

The state space structure of this model is

$$\begin{aligned}\boldsymbol{X}_t &= \begin{bmatrix}\ell_t & b_t\end{bmatrix}', \qquad (8)\\ \omega(\boldsymbol{X}_{t-1}) = r(\boldsymbol{X}_{t-1}) &= \ell_{t-1} + b_{t-1},\\ \boldsymbol{f}(\boldsymbol{X}_{t-1}) &= \begin{bmatrix}\ell_{t-1} + b_{t-1} & b_{t-1}\end{bmatrix}' \text{ and}\\ \boldsymbol{g} &= \begin{bmatrix}\alpha(\ell_{t-1} + b_{t-1}) & \beta(\ell_{t-1} + b_{t-1})\end{bmatrix}'.\end{aligned}$$

We have the following extreme cases for two smoothing parameters α and β:

- If $\beta = 0$, the model becomes a global trend model.
- If $\beta = 0$ and $\alpha = 1$, the model produces a random walk with a constant trend element, often known as the random walk with drift.
- If $\beta = 0$ and $\alpha = 0$, the model has a fixed level and trend, so reduces to a classical or global linear trend model.

We have the following recursive relationship for this model:

$$\begin{aligned} \ell_t &= \alpha Y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}), \\ b_t &= \beta(Y_t - \ell_{t-1}) + (1-\beta) b_{t-1}. \end{aligned} \tag{9}$$

The stability conditions for this model are $\alpha > 0, \beta > 0$, and $2\alpha + \beta < 4$.

We will fit this model to the Dow Jones openings series as well.

In this series, we have trend and volatility clustering at the same time.

Therefore, we expect to improve the model fitting of the MNN model.

```r
fit.dji.MAN = ets(dji.open, model="MAN")
summary(fit.dji.MAN)
```

```
## ETS(M,A,N)
##
## Call:
##  ets(y = dji.open, model = "MAN")
##
##   Smoothing parameters:
##     alpha = 0.9247
##     beta  = 1e-04
##
##   Initial states:
##     l = 2699.4178
##     b = 39.4208
##
##   sigma:  0.0402
##
##      AIC     AICc      BIC
## 3414.939 3415.237 3431.602
##
## Training set error measures:
##                    ME     RMSE      MAE         MPE     MAPE      MASE
## Training set 7.432461 323.6509 226.5569 -0.05079229 3.038635 0.2579982
##                     ACF1
## Training set 0.005154334
```

```r
fit.dji.AAN = ets(dji.open, model="AAN")
summary(fit.dji.AAN)
```

```
## ETS(A,A,N)
##
## Call:
##  ets(y = dji.open, model = "AAN")
##
##   Smoothing parameters:
##     alpha = 0.9321
##     beta  = 0.0049
##
##   Initial states:
##     l = 2719.6659
##     b = 10.9827
##
##   sigma:  328.3857
##
##      AIC     AICc      BIC
## 3508.628 3508.926 3525.291
##
## Training set error measures:
##                    ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 23.66266 325.1974 229.5621 0.3280304 3.089348 0.2614204
##                      ACF1
## Training set -0.001879956
```

The local trend model with multiplicative errors gives a better fit than the one with additive errors in terms of AIC and BIC.

Their MASE values (forecasting performances) are very close to each other.

Due to the existence of changing variance, we can again use the log transformation.

```r
fit.dji.MAN.log = ets(dji.open.log, model="MNN")
summary(fit.dji.MAN.log)
```

```
## ETS(M,N,N)
##
## Call:
##  ets(y = dji.open.log, model = "MNN")
##
##   Smoothing parameters:
##     alpha = 0.9792
##
##   Initial states:
##     l = 7.9191
##
##   sigma:  0.0047
##
##       AIC      AICc       BIC
## -214.0210 -213.9028 -204.0229
##
## Training set error measures:
##                       ME       RMSE        MAE        MPE      MAPE
## Training set 0.007380713 0.04097398 0.03166486 0.08425936 0.3608663
##                   MASE       ACF1
## Training set 0.2530142 -0.0350858
```

```r
fit.dji.AAN.log = ets(dji.open, model="ANN",lambda=0)
summary(fit.dji.AAN.log)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = dji.open, model = "ANN", lambda = 0)
##
##   Box-Cox transformation: lambda= 0
##
##   Smoothing parameters:
##     alpha = 0.9789
##
##   Initial states:
##     l = 7.9192
##
##   sigma:  0.0412
##
##       AIC      AICc       BIC
## -212.7820 -212.6637 -202.7838
##
## Training set error measures:
##                    ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 47.13142 327.4027 234.292 0.6541379 3.158866 0.2668067
##                     ACF1
## Training set -0.04499207
```

For the log-transformed opening series, MAN model performs better than AAN in terms of AIC, BIC, and MASE.

Also, according to the comparison of MASE values, the best performing model is the MAN model with log-transformed series.

When we compare MAN and MNN approach for this series, the smallest MASE value is obtained with MAN model applied over the log-transformed series.

Local Multiplicative Trend, Additive Error Model: ETS(A,M,N)

In this model, we have an additive error structure while the trend is multiplicative and there is no seasonality considered.

So, this model corresponds to exponential trend method with additive errors and it is represented as ETS(A,M,M) with the following formulation:

$$\begin{aligned} Y_t &= \ell_{t-1} b_{t-1} + \epsilon_t, \\ \ell_t &= \ell_{t-1} b_{t-1} + \alpha\epsilon_t), \\ b_t &= b_{t-1} + \beta\epsilon_t/\ell_{t-1}. \end{aligned} \tag{10}$$

We have the following recursive relationships for this model:

$$\begin{aligned}
\ell_t &= (1-\alpha)\ell_{t-1}b_{t-1} + \alpha Y_t, \\
b_t &= (1-\beta)b_{t-1} + \beta Y_t/\ell_{t-1} \\
&= (1-\beta^*)b_{t-1} + \beta^* \ell_t/\ell_{t-1}.
\end{aligned} \tag{11}$$

It is no longer possible to derive simple expressions to ensure that the stability conditions are satisfied. As for the implementation of the model, the function `ets()` is not capable of applying this model due to the stability issues.

Local Multiplicative Trend, Multiplicative Error Model: ETS(M,M,N)

When we change error structure to multiplicative from additive, we can use the function `ets()` to implement multiplicative trend models.

The model formulation for the multiplicative trend model with multiplicative errors is

$$\begin{aligned} Y_t &= \ell_{t-1} b_{t-1}(1+\epsilon_t), \\ \ell_t &= \ell_{t-1} b_{t-1}(1+\alpha\epsilon_t), \\ b_t &= b_{t-1}(1+\beta\epsilon_t). \end{aligned} \tag{12}$$

We have the same recursive relationships for this model as the ETS(A,M,N) model given in (10).

We use this model for the series composed of strictly positive values using a set of conditions such as: $0 < \alpha < 1$, $0 < \beta < 1$, and $1 + \epsilon_t > 0$.

When we set $\beta = 0$, we have a constant trend term corresponding to a fixed growth rate, b.

If we had $b < 1$ this would correspond to a form of damping, whereas $b > 1$ allows perpetual growth and could not satisfy the stability condition.

Let’s focus on the series composed of monthly observations of the number of short term overseas visitors to Australia. Recall that this series includes changing variance along with seasonality.

However, the MMN model does not cater for seasonality but it accounts for changing variance.

We will apply the exponential trend model and ETS(M,M,N) to this data for comparison purposes.

```r
fit5 <- holt(visitors, exponential=TRUE, h=5)
summary(fit5)
fit.elec.AMN = ets(visitors, model="MMN")
summary(fit.elec.AMN)
```

```
##
## Forecast method: Holt's method with exponential trend
##
## Model Information:
## Holt's method with exponential trend
##
## Call:
##  holt(y = visitors, h = 5, exponential = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.4654
##     beta  = 1e-04
##
##   Initial states:
##     l = 51.266
##     b = 1.0022
##
##   sigma:  0.1584
##
##      AIC     AICc      BIC
## 3106.415 3106.671 3123.818
##
## Error measures:
##                    ME     RMSE      MAE        MPE     MAPE     MASE
## Training set 2.034866 45.10976 34.50668 -0.3121464 12.17561 1.274299
##                    ACF1
## Training set 0.05609321
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## May 2005       460.8019 365.9490 554.4323 316.7174 603.2470
## Jun 2005       462.0554 359.4273 568.5351 312.4547 627.4844
## Jul 2005       463.3122 354.9019 580.3019 301.7411 650.3121
## Aug 2005       464.5724 348.3020 591.5222 292.2906 660.5641
## Sep 2005       465.8361 344.0872 595.6701 289.3304 686.3222
## ETS(M,M,N)
##
## Call:
##  ets(y = visitors, model = "MMN")
##
##   Smoothing parameters:
##     alpha = 0.6753
##     beta  = 1e-04
##
##   Initial states:
##     l = 71.2792
##     b = 1.0126
##
##   sigma:  0.1489
##
##      AIC     AICc      BIC
## 3085.882 3086.138 3103.285
##
## Training set error measures:
##                     ME     RMSE      MAE       MPE    MAPE     MASE
## Training set -3.086056 46.24005 35.13962 -2.199294 12.2778 1.297673
##                    ACF1
## Training set -0.1090781
```

Exponential trend model performs better in this case.

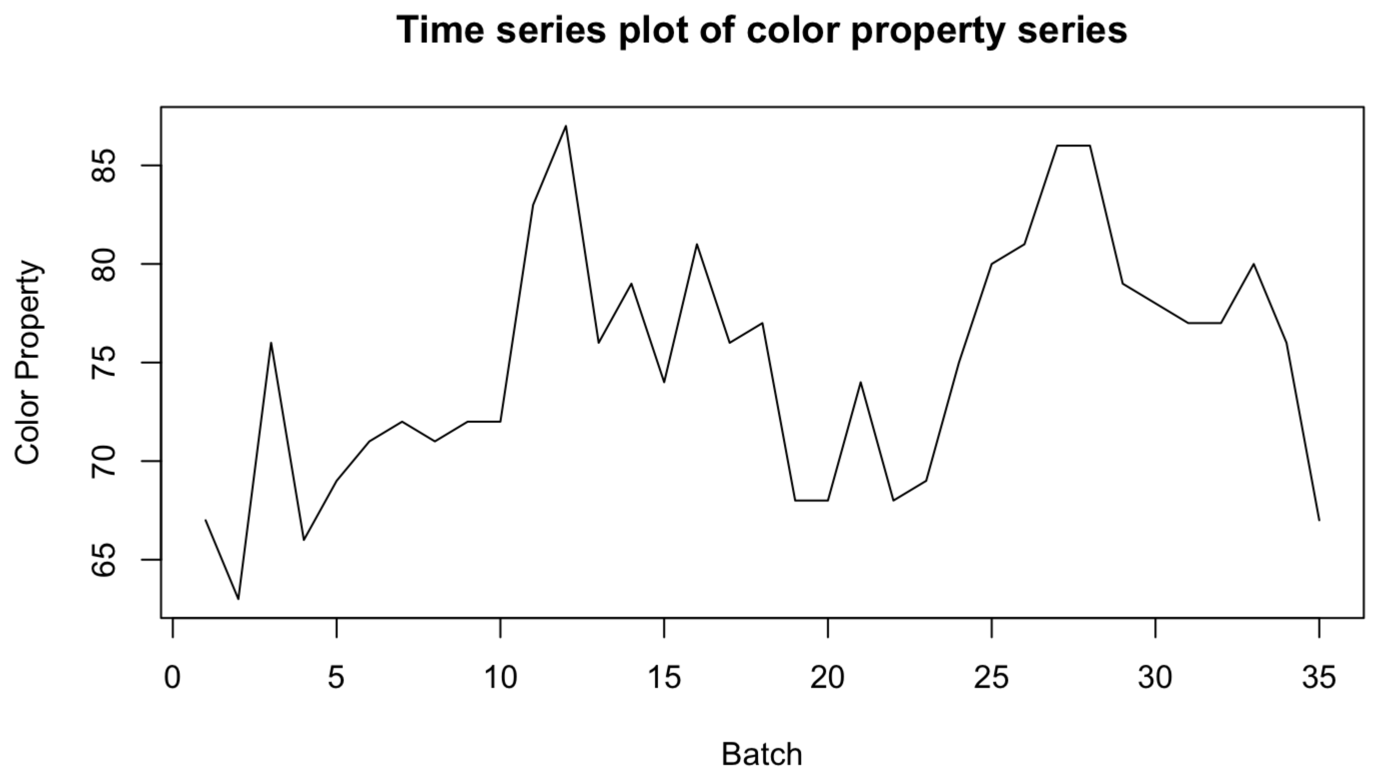
So, the inclusion of multiplicative error approach did not work when we captured trend without considering seasonality.

Now, let’s consider a time series from an industrial chemical process.

A colour property from consecutive batches in the process is measured and given by dataset `color` in the `TSA` package.

The following shows the time series plot for this data.

```
library(TSA)
data("color")
plot(color,ylab='Color Property',xlab='Batch', main="Time series plot of color property se
```


Time series plot of color property series
Color Property
65
70
75
80
85
0
5
10
15
20
25
30
35
Batch

Trend and changing variance exists in this series. We will fit the exponential trend model and ETS(M,M,N) to this data.

```r
fit5 <- holt(color, initial="simple", exponential=TRUE, h=5)
summary(fit5)
```

```
##
## Forecast method: Holt's method with exponential trend
##
## Model Information:
## Holt's method with exponential trend
##
## Call:
##  holt(y = color, h = 5, initial = "simple", exponential = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.7898
##     beta  = 0.1431
##
##   Initial states:
##     l = 67
##     b = 0.9403
##
##   sigma:  0.0824
## Error measures:
##                     ME     RMSE      MAE       MPE    MAPE    MASE
## Training set 0.5461306 5.85692 4.488297 0.5542036 5.99135 1.04522
##                     ACF1
## Training set -0.02253264
##
## Forecasts:
##    Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 36       67.92926 60.82317 75.14022 57.38487 78.94542
## 37       66.85051 57.78721 76.47629 52.89861 81.97568
## 38       65.78889 54.37406 78.26990 48.92292 86.04699
## 39       64.74414 51.18086 80.42976 45.49216 90.03406
## 40       63.71597 48.10739 82.04273 41.60006 94.61273
```

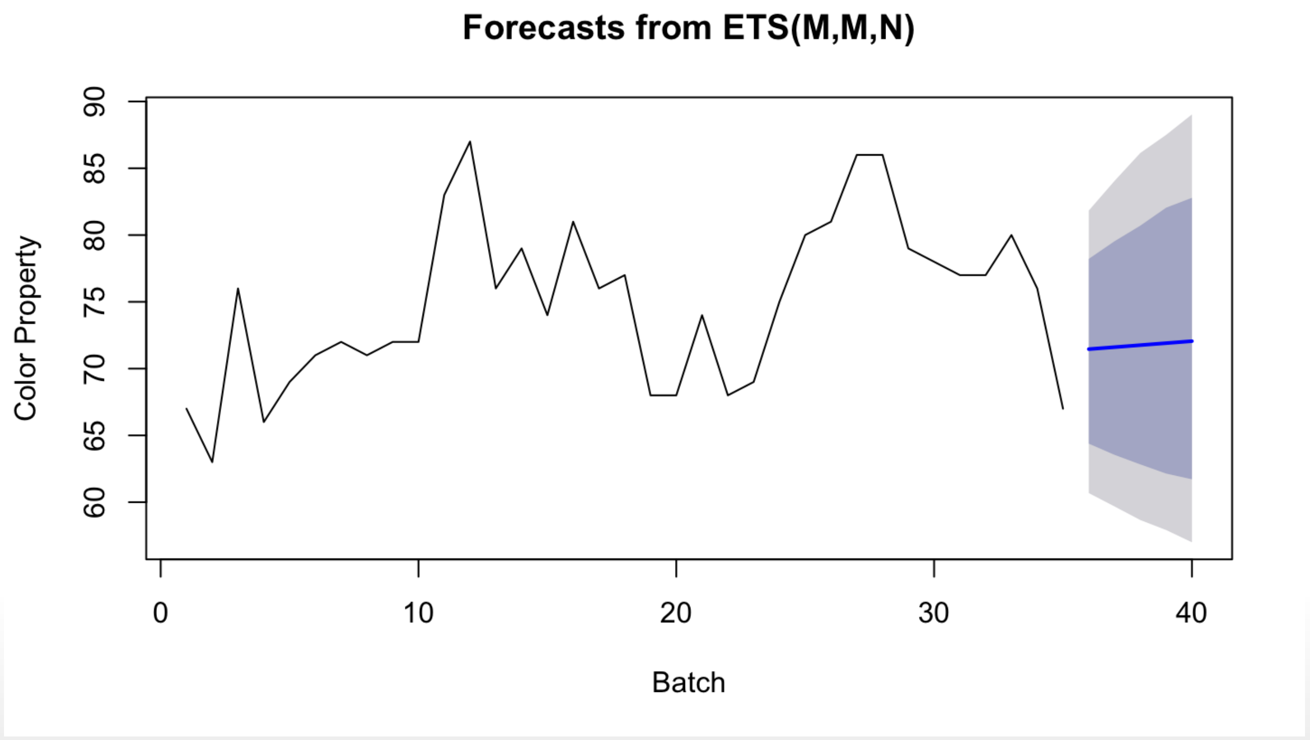
```r
fit.color.MMN = ets(color, model="MMN")
summary(fit.color.MMN)
```

```
## ETS(M,M,N)
##
## Call:
##  ets(y = color, model = "MMN")
##
##   Smoothing parameters:
##     alpha = 0.5888
##     beta  = 1e-04
##
##   Initial states:
##     l = 68.8338
##     b = 1.0021
##
##   sigma:  0.075
##
##      AIC     AICc      BIC
## 250.9714 253.0404 258.7481
##
## Training set error measures:
##                      ME    RMSE     MAE        MPE     MAPE      MASE
## Training set -0.1465168 5.24939 4.22827 -0.5306543 5.640999 0.9846657
##                    ACF1
## Training set 0.04694801
```

As we can observe from the MASE values ETS(M,M,N) performs better than the exponential trend model with this data.

The forecasts from the MMN model are shown below:

```r
plot(forecast(fit.color.MMN,h=5),ylab='Color Property',xlab='Batch')
```



Nonlinear Seasonal Models

When we use the linear structure with seasonal components we assume that the pattern of seasonality remains through time.

However, the seasonal pattern may change over time. In such cases, the need for evolving seasonal patterns is evident.

In this section, we will consider capturing these **evolving seasonal patterns**.

A Multiplicative Seasonal and Error Model: ETS(M,A,M)

The seasonal variations are usually more dramatic within a short time frame than the longer-term changes in trend so that the focus is primarily on the correct specification of the seasonal structure.

The model ETS(M,A,M) consider multiplicative effects for both the seasonal and error components.

This model is formulated as follows:

$$\begin{aligned} Y_t &= (\ell_{t-1} + b_{t-1}) s_{t-m} (1 + \epsilon_t), \\ \ell_t &= (\ell_{t-1} + b_{t-1})(1 + \alpha \epsilon_t), \\ b_t &= b_{t-1} + \beta(\ell_{t-1} + b_{t-1})\epsilon_t, \\ s_t &= S_{t-m}(1 + \gamma \epsilon_t). \end{aligned} \tag{13}$$

The recursive relationships for this model are

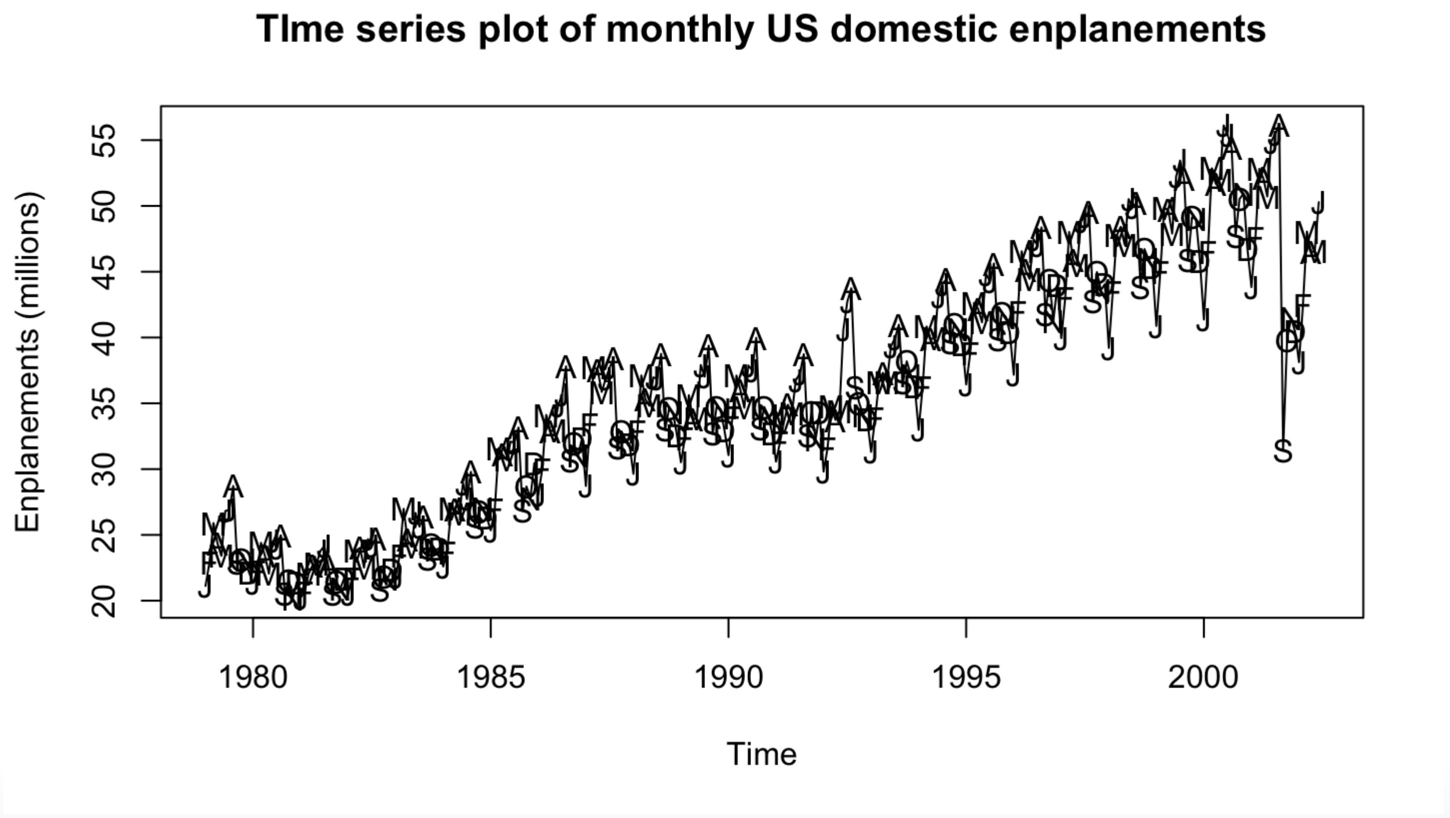
$$\begin{aligned} \ell_t &= (1-\alpha)(\ell_{t-1} + b_{t-1}) + \alpha Y_t / S_{t-m]}, \\ b_t &= \beta^* (\ell_t - \ell_{t-1}) + (1-\beta^*) b_{t-1}, \\ S_t &= (1-\gamma) S_{t-m} + \gamma Y_t / (\ell_{t-1} + b_{t-1}). \end{aligned} \tag{14}$$

Users of this method usually recommend the parameter ranges $0 < \alpha, \beta, \gamma < 1$, but the exact specification of an acceptable parameter space is extremely difficult.

Consider domestic revenue enplanements (millions) series between 1996 and 2000 from the `expsmooth` package.

Time series plot with labels is shown in the next slide.

```
data("enplanements")
plot(enplanements, ylab= "Enplanements (millions)", xlab="Time", main="TIme series plot of
points(y=enplanements,x=time(enplanements), pch=as.vector(season(enplanements)))
```



The seasonal pattern in this series changes after 1985 and also at the end of the observation period.

Changing variance and trend are also present in these series.

Following code chunk applies multiplicative seasonal Holt-Winters’ model and MAM models.

```r
fit3 <- hw(enplanements,seasonal="multiplicative", h=5*frequency(ukcars))
summary(fit3)
```

```
##
## Forecast method: Holt-Winters' multiplicative method
##
## Model Information:
## Holt-Winters' multiplicative method
##
## Call:
##  hw(y = enplanements, h = 5 * frequency(ukcars), seasonal = "multiplicative")
##
##   Smoothing parameters:
##     alpha = 0.3974
##     beta  = 0.0173
##     gamma = 1e-04
##
##   Initial states:
##     l = 25.0534
##     b = -0.3081
##     s = 0.9442 0.96 0.9737 0.9215 1.1088 1.0904
##            1.0913 1.0112 1.0295 1.0514 0.948 0.8701
##
##   sigma:  0.0386
##
##      AIC     AICc      BIC
## 1767.646 1769.964 1829.558
##
## Error measures:
##                      ME     RMSE       MAE       MPE     MAPE      MASE
## Training set 0.04856096 1.418854 0.8935157 0.1822001 2.616095 0.4582219
##                   ACF1
## Training set 0.2774771
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jul 2002       49.44557 46.99799 51.89315 45.70233 53.18881
## Aug 2002       50.19912 47.50815 52.89008 46.08364 54.31459
## Sep 2002       41.65291 39.24767 44.05816 37.97441 45.33142
## Oct 2002       43.93620 41.21544 46.65695 39.77516 48.09723
## Nov 2002       43.24466 40.38394 46.10538 38.86957 47.61975
## Dec 2002       42.46690 39.47597 45.45783 37.89267 47.04114
```

```r
fit.enplanements.MAM = ets(enplanements, model="MAM")
summary(fit.enplanements.MAM)
```

```
## ETS(M,Ad,M)
##
## Call:
##  ets(y = enplanements, model = "MAM")
##
##   Smoothing parameters:
##     alpha = 0.6165
##     beta  = 9e-04
##     gamma = 1e-04
##     phi   = 0.9779
##
##   Initial states:
##     l = 24.9866
##     b = -0.0053
##     s = 0.9474 0.9541 0.9735 0.9355 1.1092 1.0846
##            1.0908 1.0056 1.0269 1.0498 0.9459 0.8768
##
##   sigma:  0.0363
##
##      AIC     AICc      BIC
## 1733.955 1736.556 1799.509
##
## Training set error measures:
##                     ME     RMSE       MAE       MPE     MAPE      MASE
## Training set 0.1192211 1.383477 0.8220167 0.2355971 2.405797 0.4215551
##                   ACF1
## Training set 0.0598574
```

According to MASE value MAM model performs better for this series.

Also, it is possible to implement a model with the multiplicative trend, multiplicative seasonal component and multiplicative errors.

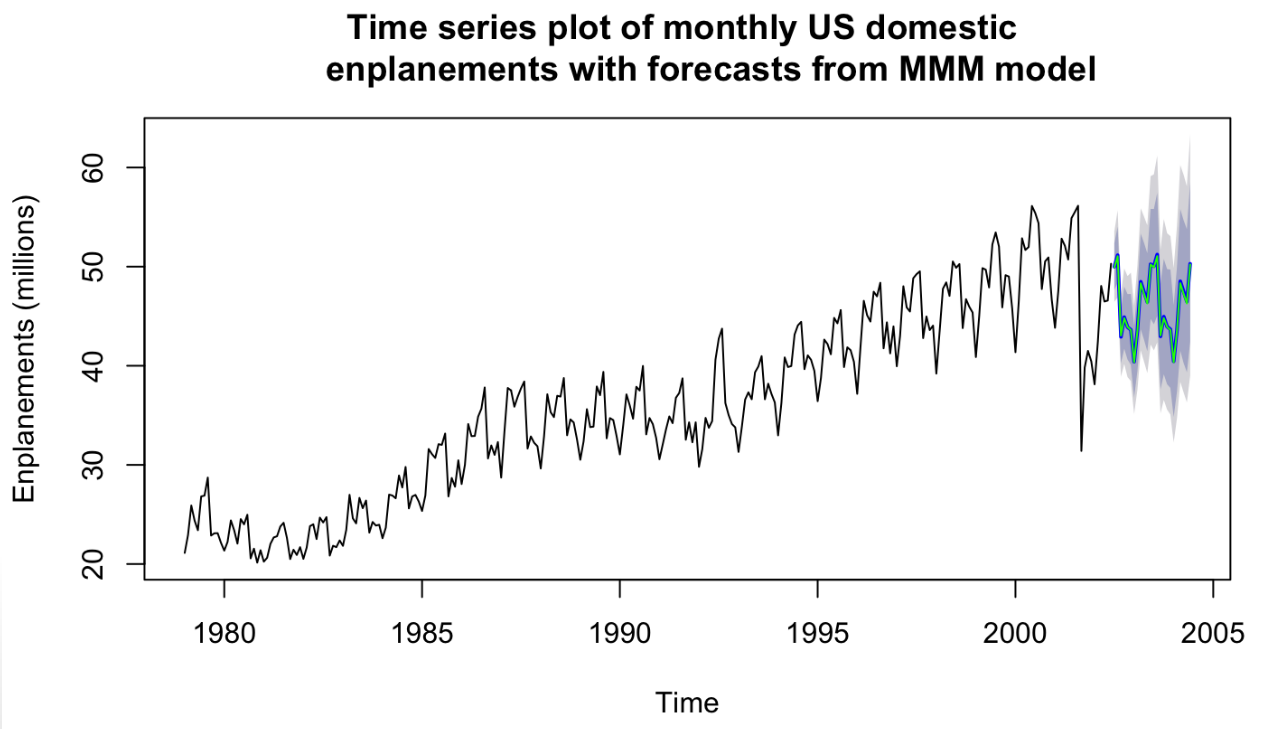
We fit this model to revenue series as well.

```r
fit.enplanements.MMM = ets(enplanements, model="MMM")
summary(fit.enplanements.MMM)
```

```
## ETS(M,Md,M)
##
## Call:
##  ets(y = enplanements, model = "MMM")
##
##   Smoothing parameters:
##     alpha = 0.6397
##     beta  = 0.0038
##     gamma = 1e-04
##     phi   = 0.9753
##
##   Initial states:
##     l = 25.0103
##     b = 0.9983
##     s = 0.9466 0.9526 0.9744 0.9317 1.1104 1.0851
##            1.0895 1.0071 1.029 1.0514 0.9454 0.8767
##
##   sigma:  0.0363
##
##      AIC     AICc      BIC
## 1732.988 1735.589 1798.542
##
## Training set error measures:
##                      ME     RMSE       MAE       MPE     MAPE      MASE
## Training set 0.09982204 1.376452 0.8183127 0.1982424 2.395143 0.4196555
##                   ACF1
## Training set 0.0419811
```

So we get the smalls MASE and AIC from this model. The forecasts from both models are displayed in the following plot. They successfully reflect the seasonal behaviour of the series.

```
plot(forecast(fit.enplanements.MMM),ylab= "Enplanements (millions)", xlab="Time", main="Ti
     enplanements with forecasts from MMM model")
lines(forecast(fit.enplanements.MAM)$mean,col="green", type="l")
```



Variations on the Common Models

We may incorporate damping factors or set specific parameters to special values in the models discussed in the previous sections.

Local Level Model with Drift

If the growth rate is steady over time, we may simplify the local trend model ETS(M,N,N) by setting $\beta = 0$.

This modification is often effective when the forecasting horizon is fairly short and growth is positive:

Damped Trend Model: ETS(M,Ad,N)

The damped trend model has the reduced growth rate $\phi b_t - 1$ at time t, where $0 \leq \phi < 1$.

The level tends to flatten out as time increases, and this feature can be useful for a series whose trends are unlikely to be sustained over time.

In particular, when the variable of interest is non-negative, a negative trend has to flatten out sooner or later.

Local Multiplicative Trend with Damping: ETS(M,Md,N)

If we try to introduce similar damping coefficients into multiplicative models, the models are not well-behaved.

In particular, such damping forces the expected value towards a limiting value of zero.

To avoid this difficulty, we raise the growth rate to a fractional power, $0 \leq \phi < 1$.

We will apply these model to US net electricity generation series.

```r
fit.elec.MNN.drift = ets(enplanements, model="MNN", beta = 0.0001)
summary(fit.elec.MNN.drift)
```

```
## ETS(M,N,N)
##
## Call:
##  ets(y = enplanements, model = "MNN", beta = 1e-04)
##
##   Smoothing parameters:
##     alpha = 0.7105
##
##   Initial states:
##     l = 21.781
##
##   sigma:  0.0777
##
##      AIC     AICc      BIC
## 2146.888 2146.974 2157.813
##
## Training set error measures:
##                     ME     RMSE    MAE         MPE     MAPE     MASE
## Training set 0.1368492 3.014019 2.0456 -0.02894235 5.831527 1.049045
##                    ACF1
## Training set 0.01727761
```

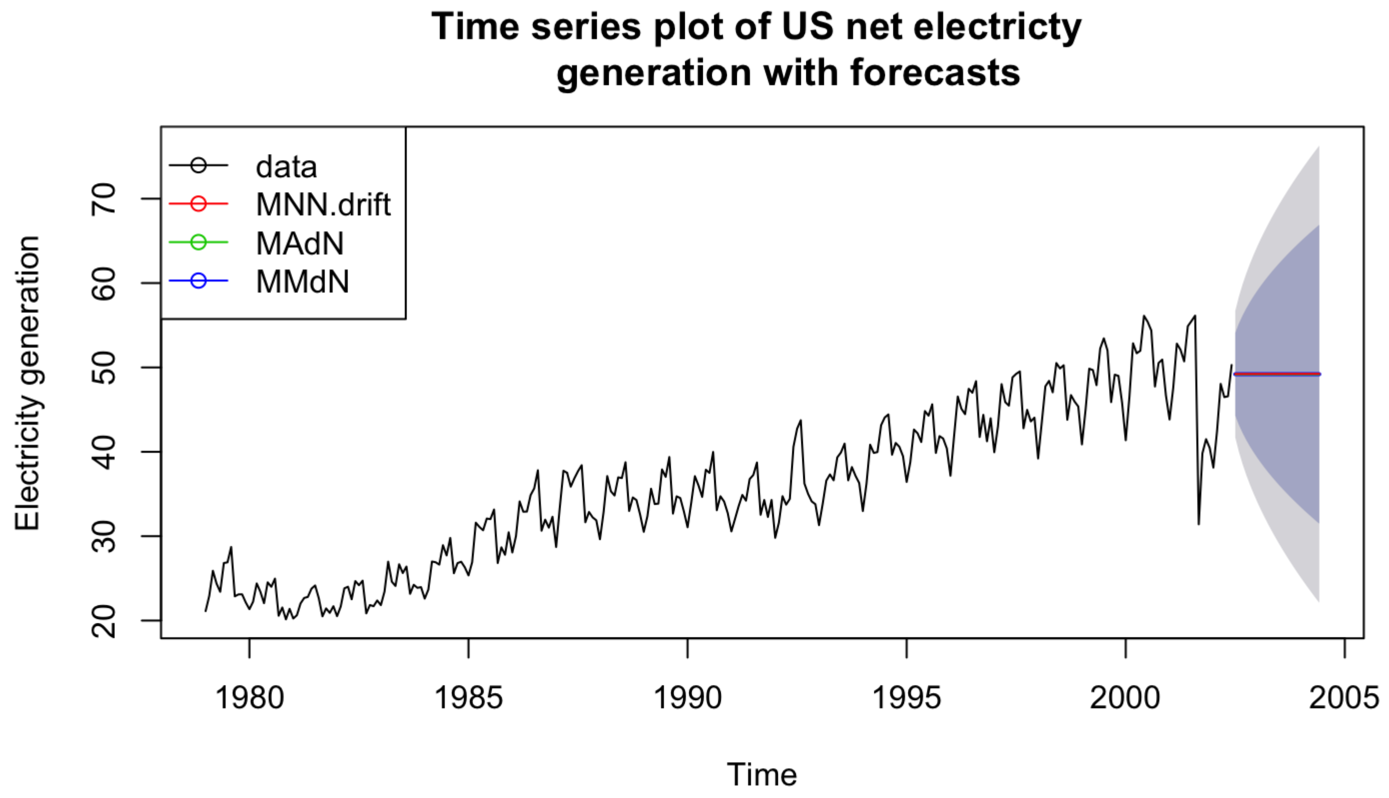
```r
fit.elec.MAdN = ets(enplanements, model="MAN", damped = TRUE)
summary(fit.elec.MAdN)
```

```
## ETS(M,Ad,N)
##
## Call:
##  ets(y = enplanements, model = "MAN", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.7083
##     beta  = 1e-04
##     phi   = 0.8
##
##   Initial states:
##     l = 20.4971
##     b = 2.5616
##
##   sigma:  0.0776
##
##      AIC     AICc      BIC
## 2151.428 2151.734 2173.280
##
## Training set error measures:
##                      ME    RMSE      MAE        MPE    MAPE     MASE
## Training set 0.09230923 3.01287 2.043673 -0.2139939 5.83239 1.048057
##                    ACF1
## Training set 0.02006619
```

```r
fit.elec.AMdN = ets(enplanements, model="MMN", damped = TRUE)
summary(fit.elec.AMdN)
```

```
## ETS(M,Md,N)
##
## Call:
##  ets(y = enplanements, model = "MMN", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.7148
##     beta  = 1e-04
##     phi   = 0.9716
##
##   Initial states:
##     l = 21.6127
##     b = 1.008
##
##   sigma:  0.0778
##
##      AIC     AICc      BIC
## 2152.436 2152.742 2174.288
##
## Training set error measures:
##                     ME     RMSE      MAE        MPE     MAPE     MASE
## Training set 0.1027982 3.014435 2.044552 -0.1635734 5.834396 1.048508
##                    ACF1
## Training set 0.01425891
```

```
plot(forecast(fit.elec.MNN.drift),ylab= "Electricity generation", xlab="Time", main="Time series plot
     generation with forecasts")
lines(forecast(fit.elec.MAdN)$mean,col="green", type="l")
lines(forecast(fit.elec.AMdN)$mean,col="red", type="l")
legend("topleft",lty=1, pch=1, col=1:4, c("data","MNN.drift", "MAdN", "MMdN"))
```



Various Seasonal Models

A variety of special seasonal models may be obtained as special cases of Holt- Winters' multiplicative scheme.

Purely Seasonal Levels

We can create a purely seasonal model with multiplicative errors by reducing the general innovations state-space model to m distinct models with a common parameter.

The model formulation is given below:

$$\begin{aligned} Y_t &= \ell_{t-m}(1+\epsilon_t), \\ \ell_t &= \ell_{t-m}(1+\alpha\epsilon_t). \end{aligned} \tag{15}$$

This model would be useful when periods with higher levels are likely to display greater variability in the series.

Fixed Seasonality

Some series, particularly in the area of macroeconomics, possess quite stable seasonal patterns.

In such cases, it may be desirable to set $\gamma = 0$ and treat seasonal factors as constant.

The corresponding model with multiplicative errors and constant sesonal factors is formulatied as follows:

$$\begin{aligned} Y_t &= (\ell_{t-1} + b_{t-1}) s_j (1 + \epsilon_t), \\ \ell_t &= (\ell_{t-1} + b_{t-1})(1 + \alpha \epsilon_t), \\ b_t &= b_{t-1} + \beta(\ell_{t-1} + b_{t-1})\epsilon_t, \end{aligned} \tag{16}$$

where $j = t \mod m$.

Other Heteroscedastic Models

The simplest way of modifying the variance structure this is to incorporate an additional exponent in a similar manner to the Box-Cox transformation without applying a transformation.

Over the model ETS(M,A,N), we separate out the error terms and modify them by using a power of the trend term, $0 \leq \theta < 1$:

$$\begin{aligned} Y_t &= (\ell_{t-1} + b_{t-1}) + (\ell_{t-1} + b_{t-1})^{\theta}\epsilon_t, \\ \ell_t &= (\ell_{t-1} + b_{t-1}) + \alpha(\ell_{t-1} + b_{t-1})^{\theta}\epsilon_t, \\ b_t &= b_{t-1} + \beta(\ell_{t-1} + b_{t-1})^{\theta}\epsilon_t. \end{aligned} \tag{17}$$

For example, $\theta = 1/3$ would produce a variance proportional to the 2/3rds power of the mean, much as the cube-root transformation does.

The present formulation enables us to retain the linear structure for the expectation, which in many ways is more plausible than the transformation.

Practical Application

For this practical application, we will consider the daily USD/HKD (U.S. dollar to Hong Kong dollar) exchange rate from January 1, 2005 to March 7, 2006, altogether 431 days of data.

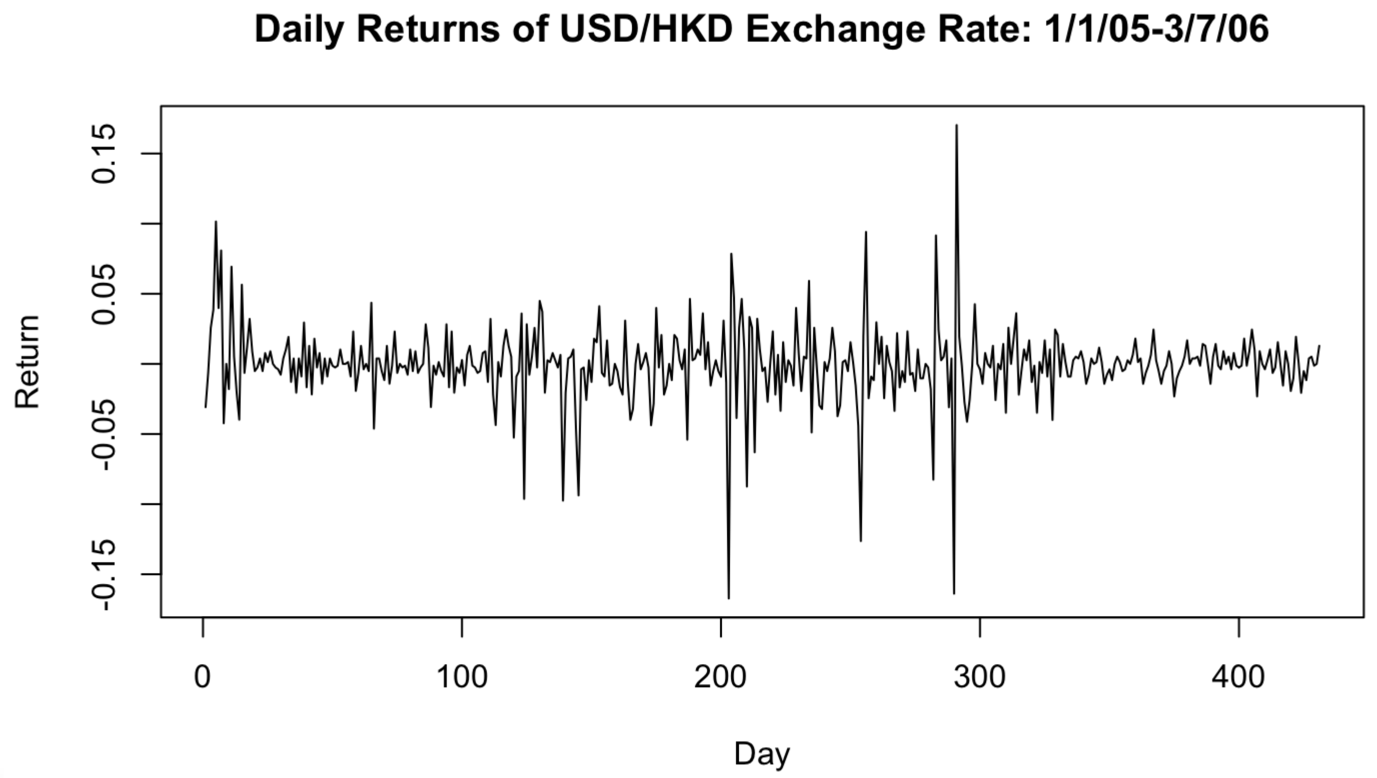
This series is available by the `usd.hkd` dataset of `TSA` package. You need to use variable `hkrate` to get the daily returns.

The returns of the daily exchange rates are shown in the following time series plot.

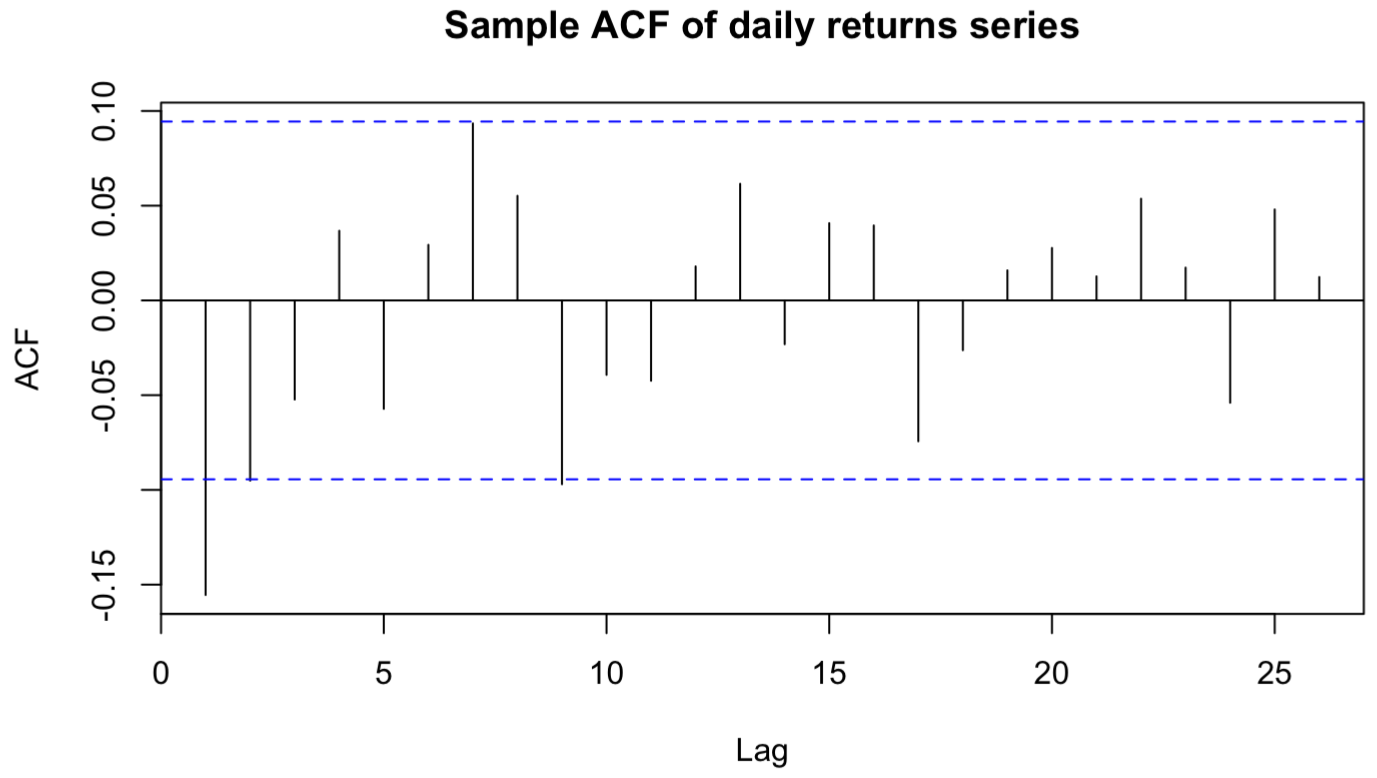
Let’s display time series and sample ACF plots.

We will get a sense of suitable models for this series according to the existence of trend, seasonality and changing variance.

```
data(usd.hkd) # Available in TSA package
returns = ts(usd.hkd$hkrate)
plot(returns, type='l', xlab='Day', ylab='Return',main="Daily Returns of USD/HKD Exchange
```



```r
acf(returns, main = "Sample ACF of daily returns series")
```



There is no trend or seasonality in this series.

Also, there is no evidence of intervention.

But there is an obvious changing variance present in the series.

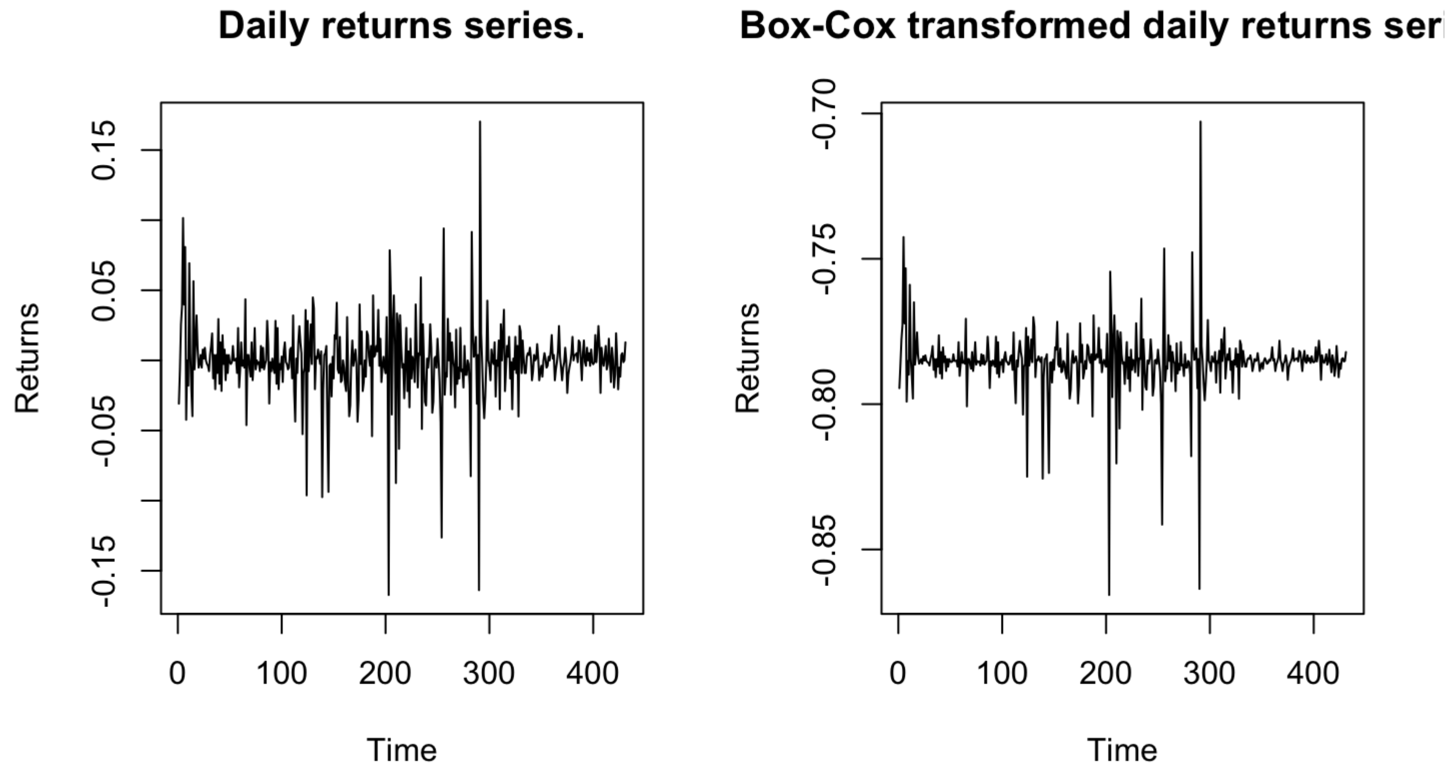
Based on these comments, the best choice of state space models would include only multiplicative errors.

We will start the modelling task with MNN model and compare this model with the ANN model.

But before going on with fitting models, we try to do transformation and see if it helps to stabilise the series.

```
lambda = BoxCox.lambda(returns)
lambda
returns.BC = BoxCox(returns, lambda = lambda)
par(mfrow=c(1,2))
plot(returns, ylab = "Returns", main = "Daily returns series.")
plot(returns.BC, ylab = "Returns", main = "Box-Cox transformed daily returns series.")
```

```
## [1] 1.273643
```



Daily returns series.
Box-Cox transformed daily returns seri
Returns
Time
0.15
0.05
-0.05
-0.15
-0.70
-0.75
-0.80
-0.85
0
100
200
300
400

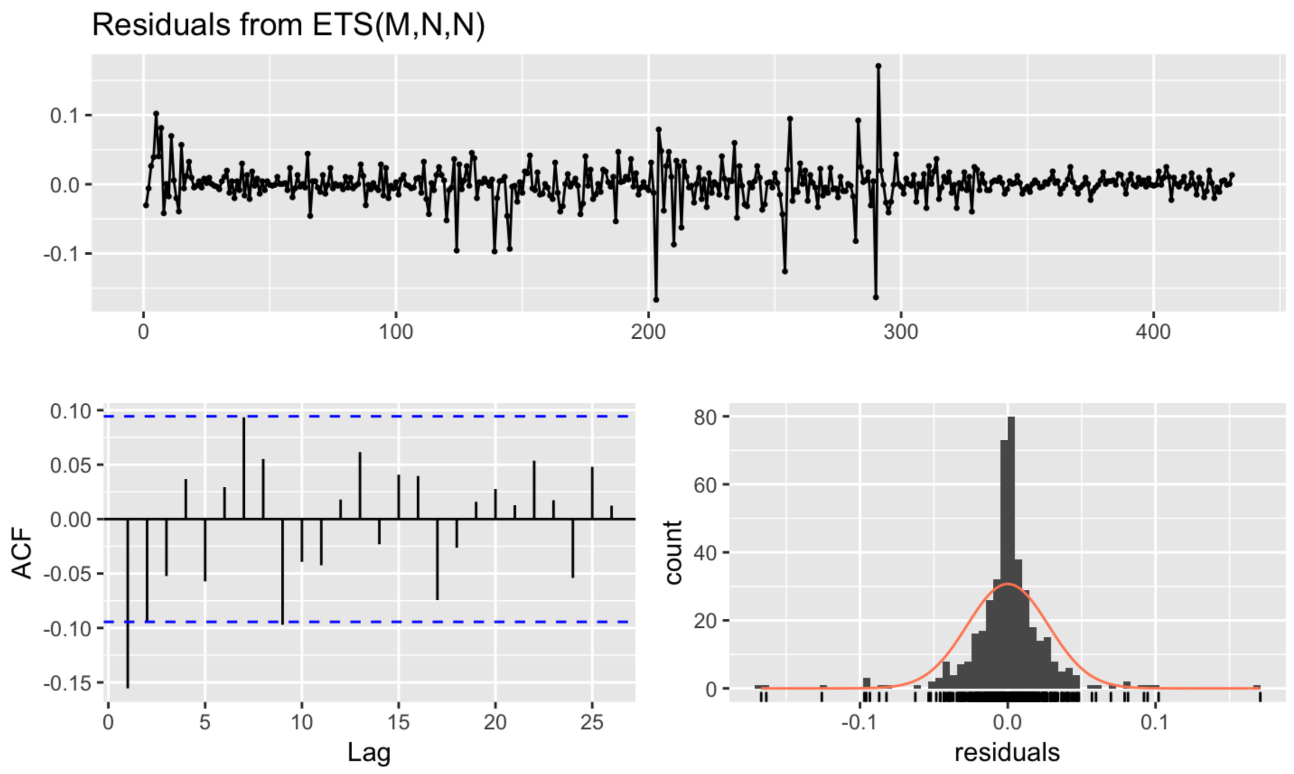
The Box-Cox transformation with $\lambda = 1.27$ did not work for stabilising the changing variables in the series.

We will go on with the original series and fit MNN and ANN models.

```r
fit.HKD.MNN = ets(returns + 1, model="MNN")
summary(fit.HKD.MNN)
```

```
## ETS(M,N,N)
##
## Call:
##  ets(y = returns + 1, model = "MNN")
##
##   Smoothing parameters:
##     alpha = 1e-04
##
##   Initial states:
##     l = 0.9995
##
##   sigma:  0.0275
##
##       AIC      AICc       BIC
## -481.0174 -480.9612 -468.8191
##
## Training set error measures:
##                         ME       RMSE       MAE         MPE     MAPE
## Training set -1.168023e-05 0.02737722 0.0162447 -0.07837285 1.639809
##                   MASE       ACF1
## Training set 0.6338297 -0.1554724
```

```
checkresiduals(fit.HKD.MNN)
```

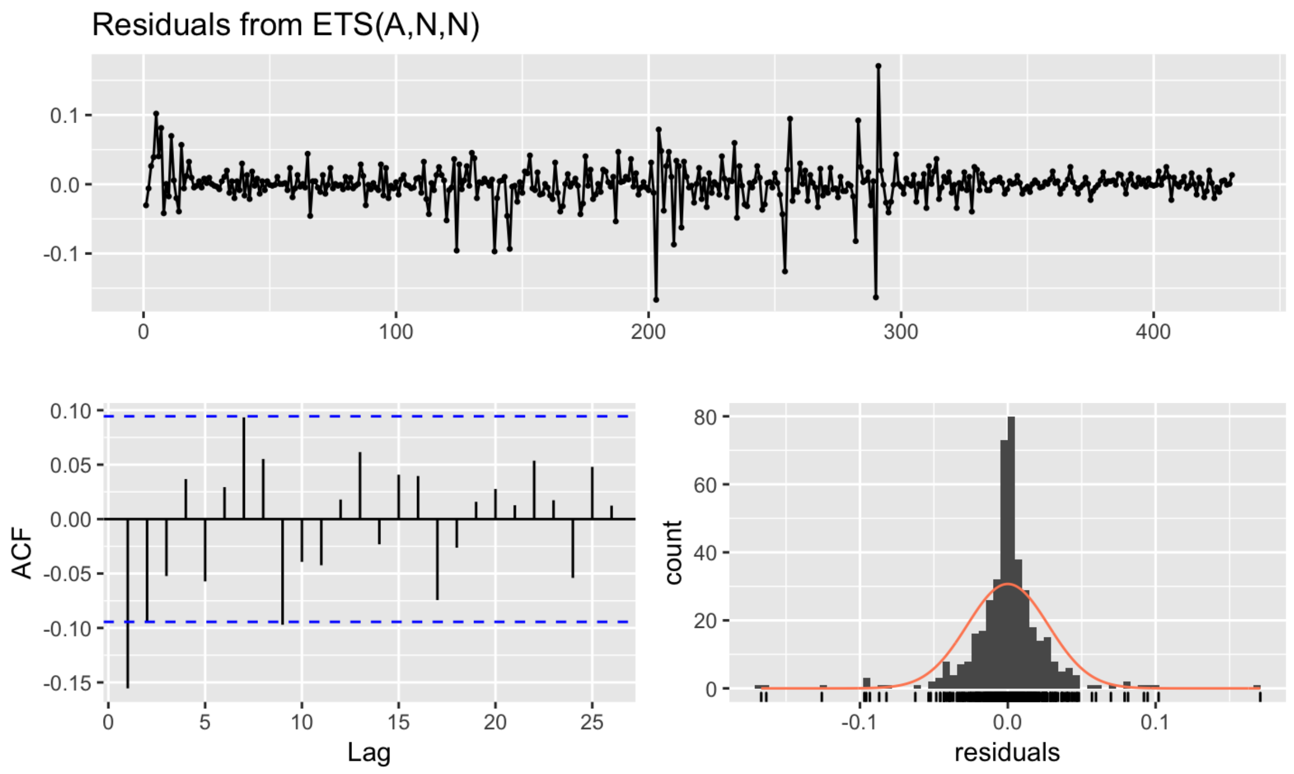


```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,N,N)
## Q* = 28.056, df = 8, p-value = 0.0004637
##
## Model df: 2.   Total lags used: 10
```

```r
fit.HKD.ANN = ets(returns + 1, model="ANN")
summary(fit.HKD.ANN)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = returns + 1, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 1e-04
##
##   Initial states:
##     l = 0.9995
##
##   sigma:  0.0274
##
##       AIC      AICc       BIC
## -481.0213 -480.9651 -468.8230
##
## Training set error measures:
##                        ME       RMSE        MAE         MPE     MAPE
## Training set -1.192409e-05 0.02737722 0.01624468 -0.07839727 1.639808
##                   MASE       ACF1
## Training set 0.6338292 -0.1554725
```

```r
checkresiduals(fit.HKD.ANN)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,N,N)
## Q* = 28.055, df = 8, p-value = 0.0004638
##
## Model df: 2.   Total lags used: 10
```

We got very similar AIC, AICc, BIC, and MASE values for both of MNN and ANN models. The MNN model is slightly better than the ANN in terms of these measures.

Residual diagnostics are also very similar for both models. There are several outlier values seen in the residual time series plot.

These residuals make the residual histogram very long-tailed.

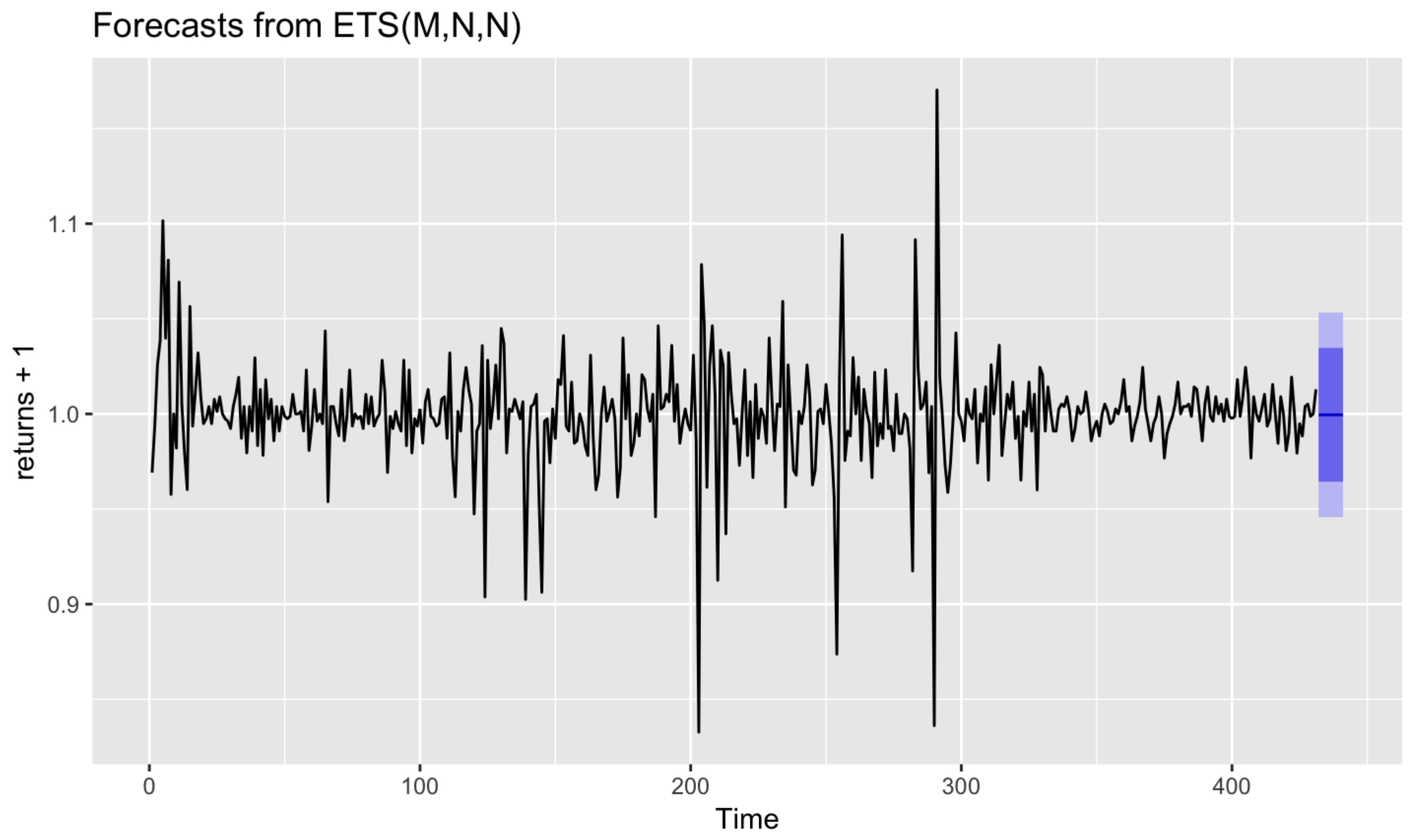
There is also one significant autocorrelation left in the residuals, which needs some attention.

Last, we will let the `ets()` fucntion to select the best model.

```r
fit.HKD.ZZZ = ets(returns + 1)
summary(fit.HKD.ZZZ)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = returns + 1)
##
##   Smoothing parameters:
##     alpha = 1e-04
##
##   Initial states:
##     l = 0.9995
##
##   sigma:  0.0274
##
##       AIC      AICc       BIC
## -481.0213 -480.9651 -468.8230
##
## Training set error measures:
##                         ME       RMSE        MAE         MPE     MAPE
## Training set -1.192409e-05 0.02737722 0.01624468 -0.07839727 1.639808
##                   MASE       ACF1
## Training set 0.6338292 -0.1554725
```

```
frc.HKD = forecast(fit.HKD.MNN)
autoplot(frc.HKD)
```



Summary

In this module, we focused on the nonlinear innovations state-space models with multiplicative errors to handle heteroscedasticity along with this trend and seasonality.

We formulated different models to handle different cases of seasonality and heteroscedasticity.

To identify the best model for these cases, a descriptive analysis including a decomposition of the series would be helpful.

What's next?

In the next module, we will focus on

- estimation of parameters in the models,
- prediction distributions and intervals
- lead-time forecasting.

References

Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D. (2008). Forecasting with exponential smoothing: the state space approach, Springer-Verlag.

Hyndman, R.J., Athanasopoulos, G. (2014). *Forecasting: Principles and Practice*. OTEXTS.

Thanks for your attendance! Pleas
follow the link
https://forms.gle/oE67NUczuvX2P
to give feedback!