

Introduction

Innovations Form of the General State Space Model

Basic Special Cases

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

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

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

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

Nonlinear Seasonal Models

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

Variations on the Common Models

Local Level Model with Drift

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

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

Various Seasonal Models

Purely Seasonal Levels

Fixed Seasonality

Other Heteroscedastic Models

Summary

References

Module 7 - Nonlinear and Heteroscedastic Innovations State Space Models

MATH1307 Forecasting

Prepared by: Dr. Haydar Demirhan based on the textbook by Hyndman et al., *Forecasting with Exponential Smoothing: The State Space Approach*. Springer, 2008.

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 $X_t = (\ell_t, b_t, s_t, s_{t-1}, \dots, s_{t-m+1})'$ and state space equations of the form

$$\begin{aligned} Y_t &= w(X_{t-1}) + r(X_{t-1})\epsilon_t, \\ X_t &= f(X_{t-1}) + g(X_{t-1})\epsilon_t, \end{aligned} \quad (1)$$

where $w(\cdot)$ and $r(\cdot)$ are scalar functions and $f(\cdot)$ and $g(\cdot)$ are vector functions, and ϵ_t is a white noise process with variance σ^2 . In the previous module, the functions r and g were constants, whereas w and f were linear in the state vector. To get the simplest nonlinear form of the model, we set $w(X_{t-1}) = r(X_{t-1}) = f(X_{t-1}) = \ell_{t-1}$ and $g(X_{t-1}) = \alpha \ell_{t-1}$ and obtain the following:

$$\begin{aligned} Y_t &= \ell_{t-1}(1 + \epsilon_t), \\ \ell_t &= \ell_{t-1}(1 + \alpha\epsilon_t). \end{aligned} \quad (2)$$

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:

$$\ell_t = \alpha Y_t + (1 - \alpha)\ell_{t-1}. \quad (3)$$

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

$$\hat{Y}_{t+1|t} = (1 - \alpha)^t \ell_0 + \alpha \sum_{j=0}^{t-1} (1 - \alpha)^j Y_{t-j}. \quad (4)$$

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.

$$\begin{aligned} V_A(Y_t|X_0) &= \sigma_A^2 [1 + (t-1)\alpha^2], \\ V_M(Y_t|X_0) &= X_0^2 [(1 + \sigma_M^2)(1 + \alpha^2 \sigma_M^2)^{t-1} - 1]. \end{aligned} \quad (5)$$

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

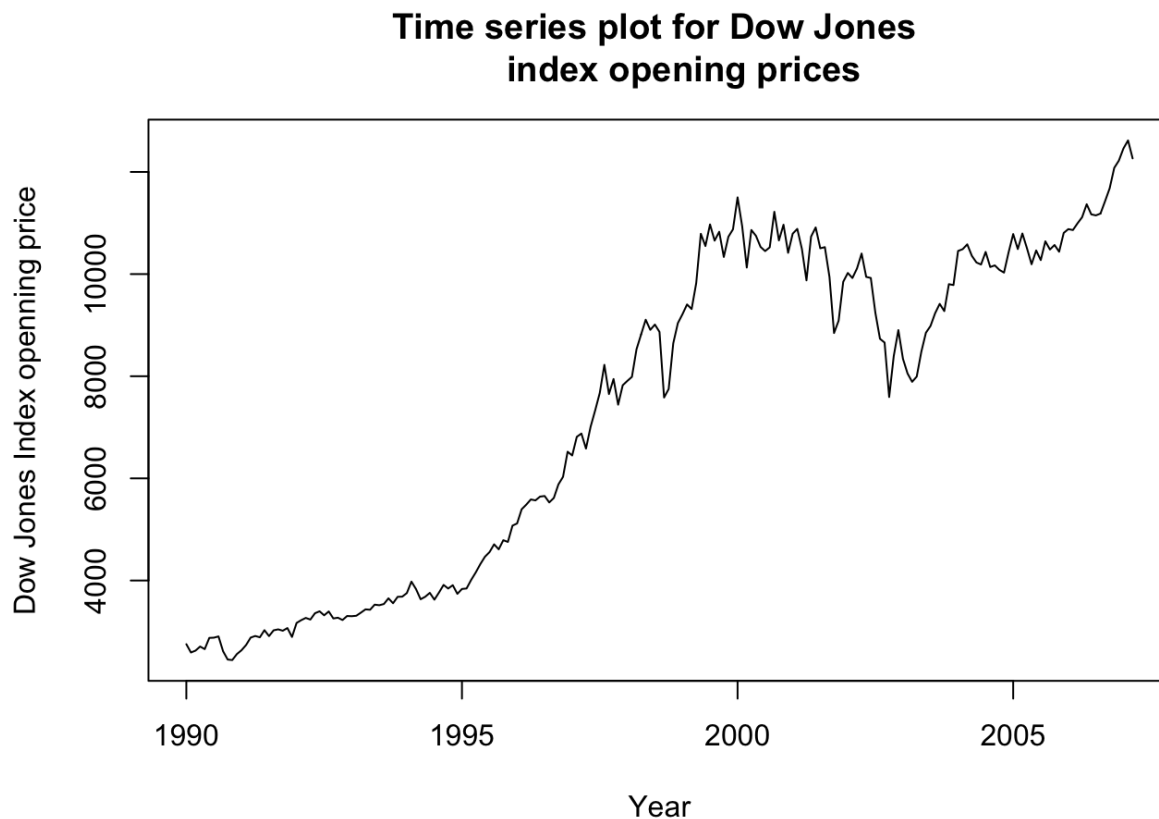
The difference between fitting with additive or multiplicative errors become more apparent as t , α and σ_M increase. When $\sigma_M = 0.30$, the multiplicative error has a mean that is about three times its standard deviation, and the differences become noticeable quite quickly. See the results for $t = 10$ 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 opening price", xlab = "Year",
     , main = "Time series plot for Dow Jones
             index opening prices" )
```



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)
```

```

## 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 0.266
6724
##           ACF1
## Training set -0.02755172

```

```

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

```

```

## 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 0.266
6797
##           ACF1
## Training set -0.02148202

```

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

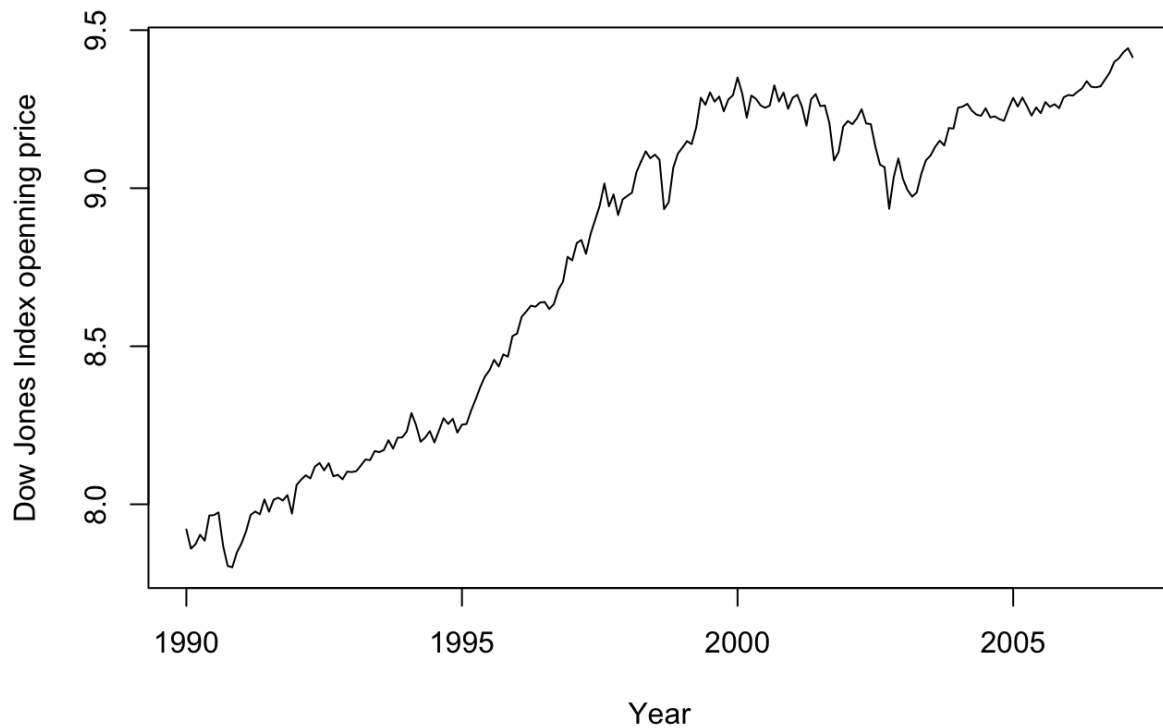
To deal with changing variance, we could apply logarithmic transformation and see if it helps. The following display shows the time series plot of log-transformed opening series.

```

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

```

**Time series plot for log transformed
Dow Jones index opening prices**



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.

```
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)
```

```

## 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 0.360
8663
##
##                               MASE           ACF1
## Training set 0.2530142 -0.0350858

```

```

fit.dji.ANN.log = ets(dji.open, model="ANN",lambda=0) # Le the ets
() apply log transformation for us
summary(fit.dji.ANN.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 0.266
8067
##
##           ACF1
## Training set -0.04499207
```

Even with a more stable variance, we got better 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
```

```
frc.dji.MNN.log = forecast(fit.dji.MNN.log , h=1)
# To take the log transformation back, we will apply the back tran
sformation 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. So, 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 the Holt's linear model with multiplicative errors. We upgrade the local level model by adding an growth rate b_t as follows:

$$\begin{aligned} Y_t &= (\ell_{t-1} + b_{t-1})(1 + \epsilon_t), \\ \ell_t &= (\ell_{t-1} + b_{t-1})(1 + \alpha\epsilon_t), \\ b_t &= b_{t-1} + \beta(\ell_{t-1} + b_{t-1})\epsilon_t. \end{aligned} \quad (6)$$

The state-space structure of this model is

$$\begin{aligned} \mathbf{X}_t &= [\ell_t \quad b_t]' , \omega(\mathbf{X}_{t-1}) = r(\mathbf{X}_{t-1}) = \ell_{t-1} + b_{t-1}, \\ f(\mathbf{X}_{t-1}) &= [\ell_{t-1} + b_{t-1} \quad b_{t-1}]' \text{ and } \mathbf{g} = [\alpha(\ell_{t-1} + b_{t-1}) \quad \beta(\ell_{t-1} + b_{t-1})]'. \end{aligned} \quad (7)$$

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:

$$\begin{aligned} \ell_t &= \alpha Y_t + (1 - \alpha)(\ell_{t-1} + b_{t-1}), \\ b_t &= \beta(Y_t - \ell_{t-1}) + (1 - \beta)b_{t-1}. \end{aligned} \quad (8)$$

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.

```
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

```

```

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 0.26
14204
##
##           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.

```

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

```

```

## ETS(M,A,N)
##
## Call:
## ets(y = dji.open.log, model = "MAN")
##
## Smoothing parameters:
##   alpha = 0.9383
##   beta  = 1e-04
##
## Initial states:
##   l = 7.9086
##   b = 0.0076
##
## sigma: 0.0046
##
##           AIC           AICc           BIC
## -217.9744 -217.6759 -201.3109
##
## Training set error measures:
##
##                               ME           RMSE           MAE           MPE
MAPE
## Training set -0.0003615555 0.04027499 0.03015165 -0.004142343
0.3434768
##
##                               MASE           ACF1
## Training set 0.2409231 0.002697118

```

```

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

```

```

## ETS(A,A,N)
##
## Call:
## ets(y = dji.open, model = "AAN", lambda = 0)
##
## Box-Cox transformation: lambda= 0
##
## Smoothing parameters:
##   alpha = 0.9589
##   beta  = 1e-04
##
## Initial states:
##   l = 7.8794
##   b = 0.0073
##
## sigma: 0.0407
##
##           AIC           AICc           BIC
## -215.2599 -214.9613 -198.5963
##
## Training set error measures:
##
##                               ME           RMSE           MAE           MPE           MAPE
MASE
## Training set -7.349889 325.8395 226.5187 -0.07384788 3.051856
0.2579546
##
##                               ACF1
## Training set -0.02334945

```

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 approaches 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:

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

We have the following recursive relationships for this model:

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

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

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

We have the same recursive relationships for this model as 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.

```
fit5 <- holt(visitors, 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 = 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 1.2
74299
##              ACF1
## Training set 0.05609321
##
## Forecasts:
##      Point Forecast      Lo 80      Hi 80      Lo 95      Hi 95
## May 2005      460.8019 368.6384 553.5833 313.5639 602.2356
## Jun 2005      462.0554 355.2107 565.0545 305.9414 624.9473
## Jul 2005      463.3122 351.4609 577.3204 303.9188 652.5145
## Aug 2005      464.5724 347.1512 588.6150 298.7829 661.2165
## Sep 2005      465.8361 339.9562 595.5655 285.2554 683.6895
```

```
fit.elec.AMN = ets(visitors, model="MMN")
summary(fit.elec.AMN)
```



```

## 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 1.29
7673
##
##           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.

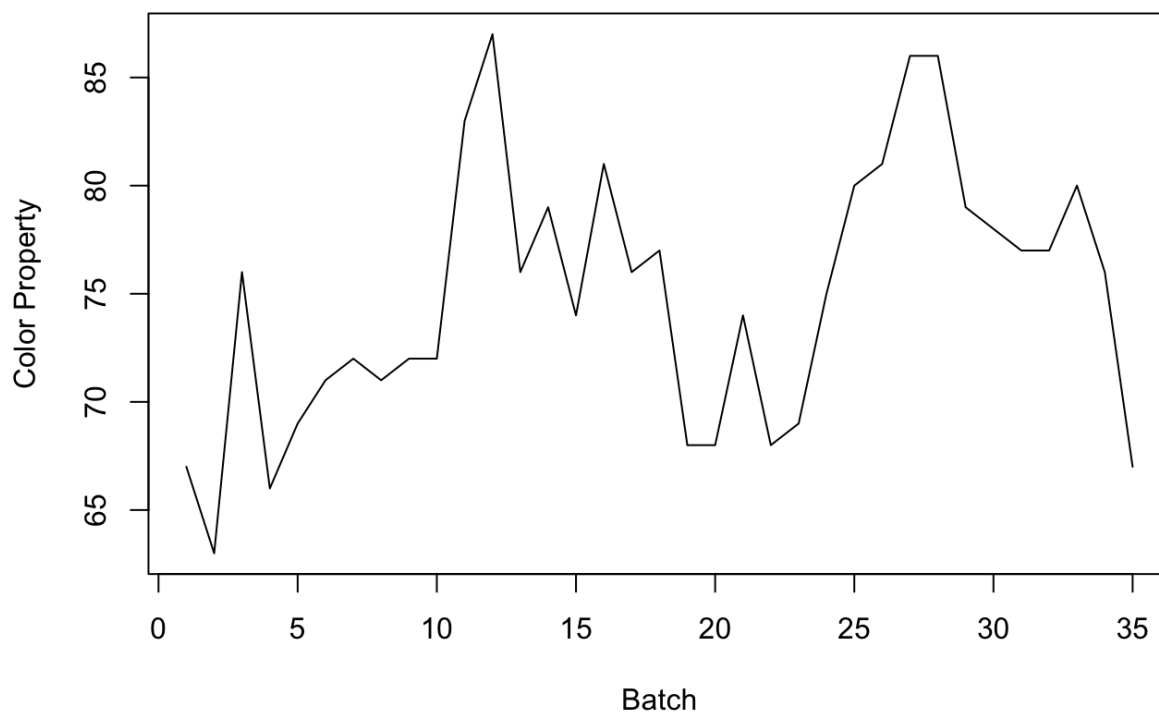
Now, let's consider a time series from an industrial chemical process. A colour property from consecutive batches in the process is measured and given by dataset `color` in the `TSA` package. The following shows the time series plot for this data.

```

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

```

Time series plot of color property series



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

```
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      MA
SE
## Training set 0.5461306 5.85692 4.488297 0.5542036 5.99135 1.045
22
##           ACF1
## Training set -0.02253264
##
## Forecasts:
##   Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 36      67.92926 60.81780 75.03628 57.21614 78.62714
## 37      66.85051 57.15179 76.82476 52.83097 82.20509
## 38      65.78889 54.13000 77.90389 49.33255 85.19672
## 39      64.74414 51.34788 79.94145 45.24914 88.55812
## 40      63.71597 48.35745 81.21112 42.14290 93.05480
```

```
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 0.9
846657
##              ACF1
## Training set 0.04694801

```

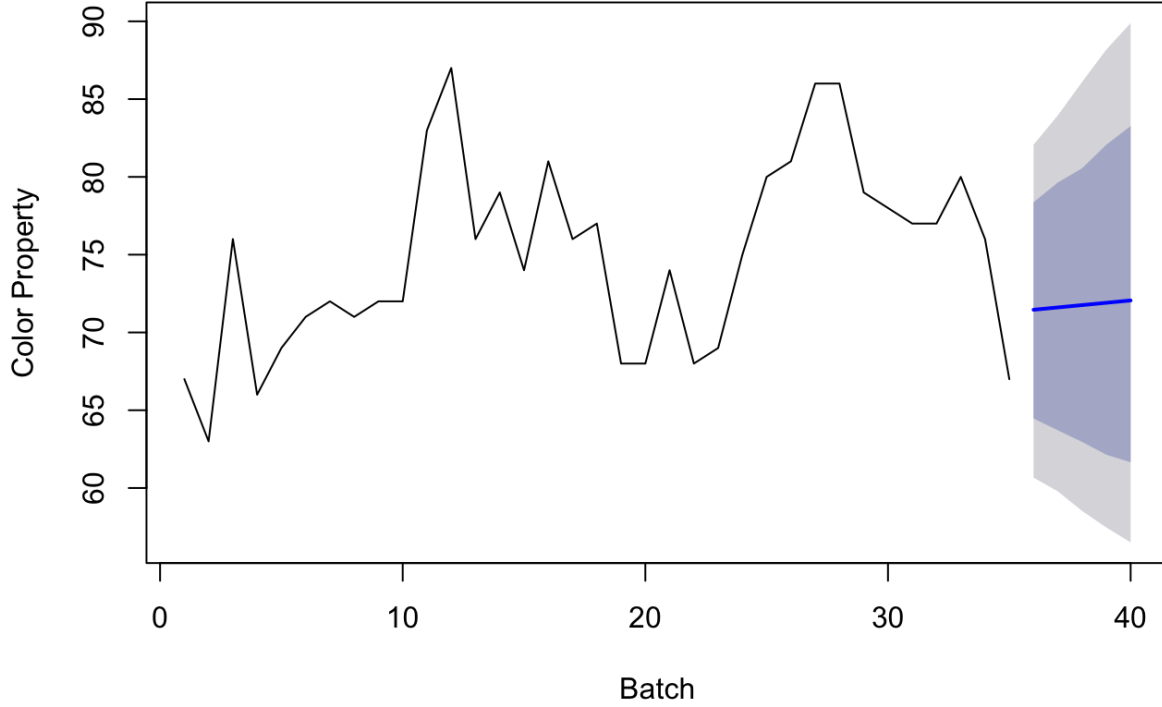
As we can observe from the MASE values ETS(M,M,N) performs better than the exponential trend model with this data. The forecasts from the MMN model are shown below:

```

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

```

Forecasts from ETS(M,M,N)



Nonlinear Seasonal Models

When we use a 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:

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

The recursive relationships for this model are

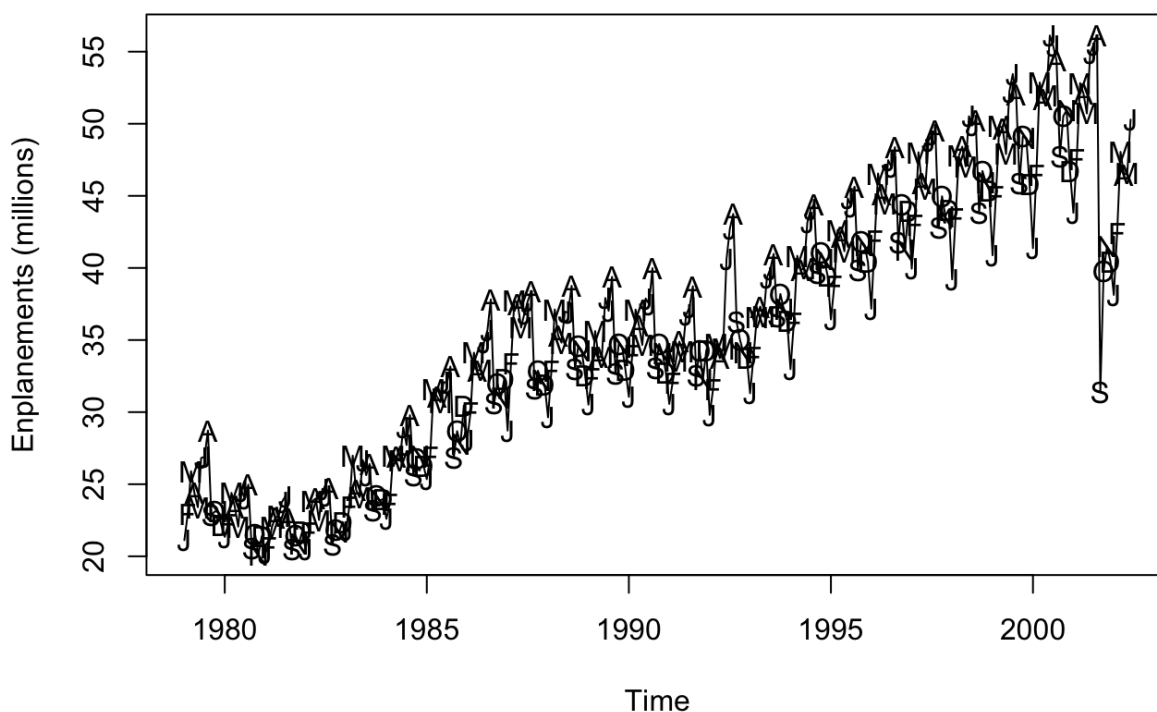
$$\begin{aligned}
 \ell_t &= (1 - \alpha)(\ell_{t-1} + b_{t-1}) + \alpha Y_t / S_{t-m}, \\
 b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1 - \beta^*)b_{t-1}, \\
 S_t &= (1 - \gamma)S_{t-m} + \gamma Y_t / (\ell_{t-1} + b_{t-1}).
 \end{aligned} \tag{13}$$

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 below:

```
data("enplanements")
plot(enplanements, ylab= "Enplanements (millions)", xlab="Time", m
      ain="TIme series plot of monthly US domestic enplanements")
points(y=enplanements,x=time(enplanements), pch=as.vector(season(e
      nplanements)))
```

Time series plot of monthly US domestic 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.

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

```

##
## Forecast method: Holt-Winters' multiplicative method
##
## Model Information:
## Holt-Winters' multiplicative method
##
## Call:
## hw(y = enplanements, h = 5 * frequency(ukcars), seasonal = "mu
ltiplicative")
##
## 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
## Jan 2003 39.06720 36.14647 41.98793 34.60033 43.53407
## Feb 2003 42.49179 39.12855 45.85503 37.34815 47.63543
## Mar 2003 47.04999 43.11670 50.98327 41.03454 53.06543
## Apr 2003 45.99493 41.94249 50.04737 39.79726 52.19260
## May 2003 45.09911 40.91950 49.27873 38.70694 51.49129
## Jun 2003 48.59065 43.86233 53.31897 41.35931 55.82199
## Jul 2003 48.46757 43.52352 53.41162 40.90629 56.02884

```

```
## Aug 2003      49.20457 43.95107 54.45807 41.17003 57.23911
## Sep 2003      40.82632 36.27032 45.38232 33.85851 47.79413
## Oct 2003      43.06285 38.04661 48.07909 35.39117 50.73452
## Nov 2003      42.38363 37.23639 47.53087 34.51161 50.25566
## Dec 2003      41.61995 36.35637 46.88354 33.57000 49.66991
## Jan 2004      38.28676 33.24983 43.32369 30.58344 45.99008
## Feb 2004      41.64152 35.94851 47.33453 32.93481 50.34823
```

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

```
## ETS(M,Ad,M)
##
## Call:
## ets(y = enplanements, model = "MAM", damped = TRUE)
##
## 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 the enplanements series as well.

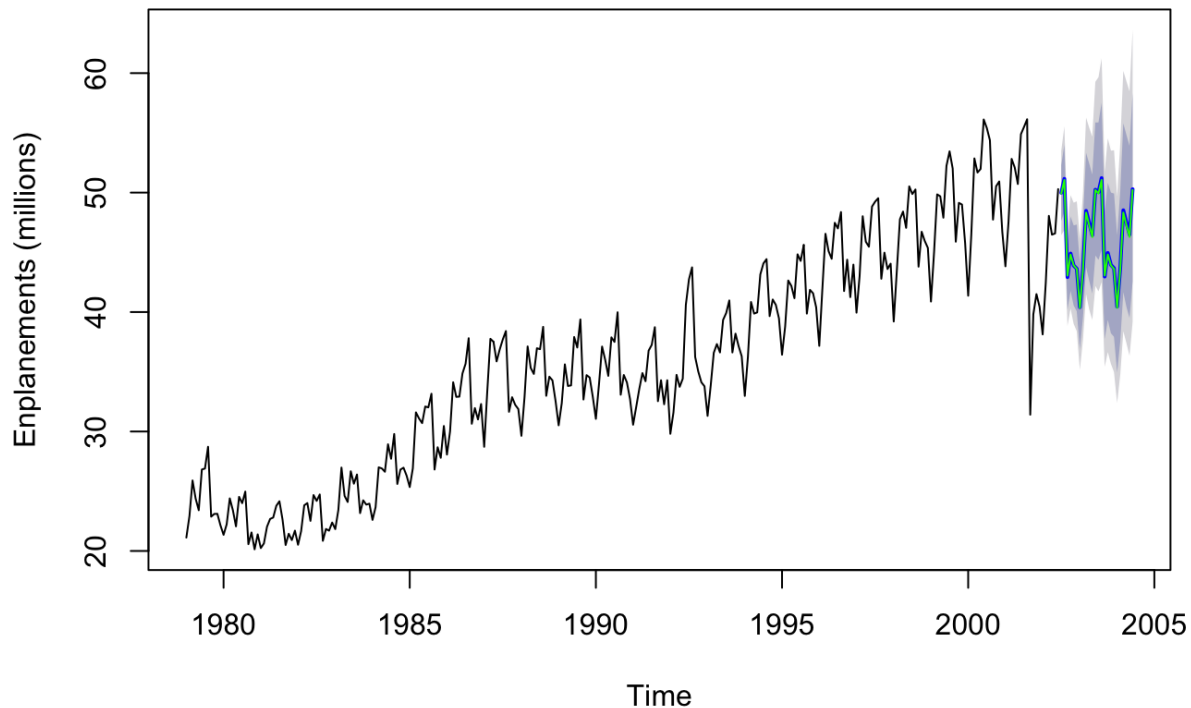

```
fit.enplanements.MMM = ets(enplanements, model="MMM", damped = TRUE)
summary(fit.enplanements.MMM)
```

```
## ETS(M,Md,M)
##
## Call:
## ets(y = enplanements, model = "MMM", damped = TRUE)
##
## 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 small 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 (million
s)", xlab="Time", main="Time series plot of monthly US domestic
enplanements with forecasts from MMM model")
lines(forecast(fit.enplanements.MAM)$mean,col="green", type="l")
```

Time series plot of monthly US domestic enplanements with forecasts from MMM model



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.

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

```
## ETS(M,N,N)
##
## Call:
## ets(y = usnetelec, model = "MNN", beta = 1e-04)
##
## Smoothing parameters:
##   alpha = 0.9999
##
## Initial states:
##   l = 295.0734
##
## sigma: 0.0631
##
##      AIC      AICc      BIC
## 724.1544 724.6250 730.1764
##
## Training set error measures:
##              ME      RMSE      MAE      MPE      MAPE      M
## ASE
## Training set 64.60515 82.08745 69.31701 4.50229 4.662774 0.9821
## 623
##              ACF1
## Training set 0.1320278
```

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

```

## ETS(M,Ad,N)
##
## Call:
## ets(y = usnetelec, model = "MAN", damped = TRUE)
##
## Smoothing parameters:
##   alpha = 0.9999
##   beta  = 0.2901
##   phi   = 0.98
##
## Initial states:
##   l = 253.5082
##   b = 46.8036
##
## sigma: 0.0269
##
##      AIC      AICc      BIC
## 638.8398 640.5898 650.8838
##
## Training set error measures:
##
##           ME      RMSE      MAE      MPE      MAPE
MASE
## Training set 4.015087 52.4398 37.70495 0.3305556 2.005602 0.534
2466
##           ACF1
## Training set -0.04375213

```

```

fit.elec.MMdN = ets(usnetelec, model="MMN", damped = TRUE)
summary(fit.elec.MMdN)

```

```

## ETS(M,Md,N)
##
## Call:
## ets(y = usnetelec, model = "MMN", damped = TRUE)
##
## Smoothing parameters:
##   alpha = 0.9999
##   beta  = 1e-04
##   phi   = 0.9661
##
## Initial states:
##   l = 266.2098
##   b = 1.1186
##
## sigma: 0.0247
##
##      AIC      AICc      BIC
## 630.2382 631.9882 642.2822
##
## Training set error measures:
##
##              ME      RMSE      MAE      MPE      MAPE
MASE
## Training set -2.733449 49.35139 33.62681 -0.1130182 1.792656 0.
4764629
##
##              ACF1
## Training set 0.08107609

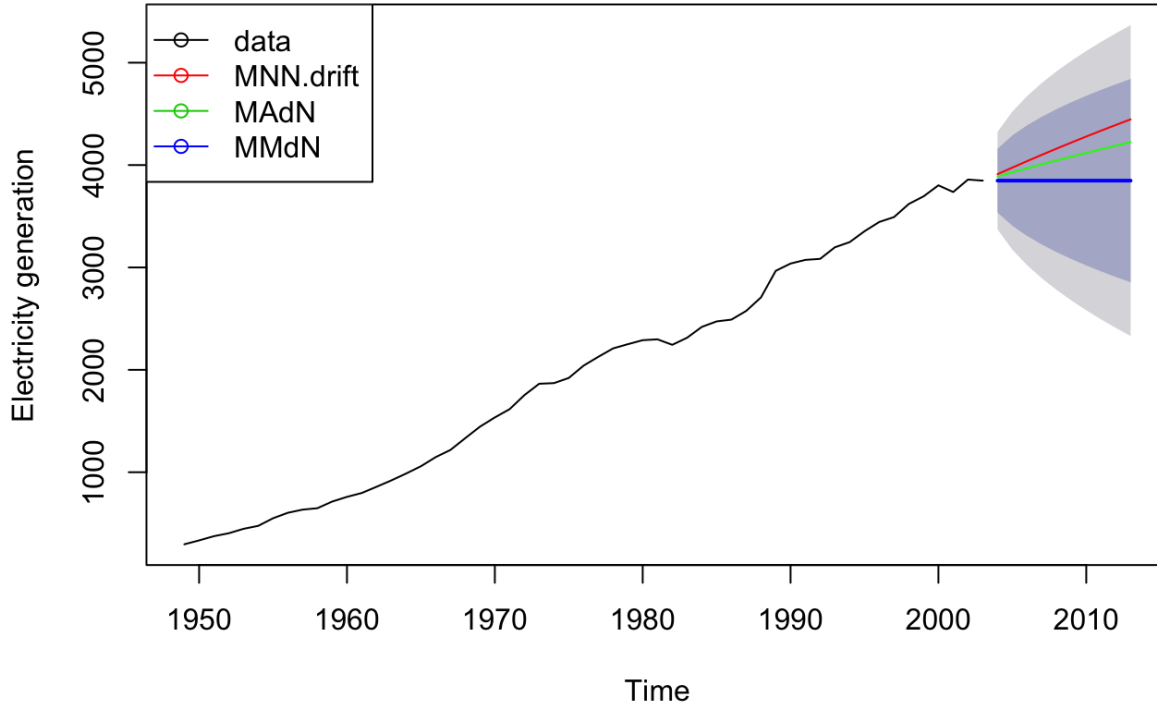
```

```

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

```

Time series plot of US net electricity generation with forecasts



Although all the approaches produce similar forecasts, the ETS(M,Md,N) performs better for this series.

Various Seasonal Models

A variety of special seasonal models may be obtained as special cases of Holt-Winterss' 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:

$$\begin{aligned} Y_t &= \ell_{t-m}(1 + \epsilon_t), \\ \ell_t &= \ell_{t-m}(1 + \alpha\epsilon_t). \end{aligned} \tag{14}$$

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 the seasonal factors as constant. The corresponding model with multiplicative errors and constant seasonal factors is formulated as follows:

$$\begin{aligned}
Y_t &= (\ell_{t-1} + b_{t-1})s_j(1 + \epsilon_t), \\
\ell_t &= (\ell_{t-1} + b_{t-1})(1 + \alpha\epsilon_t), \\
b_t &= b_{t-1} + \beta(\ell_{t-1} + b_{t-1})\epsilon_t,
\end{aligned} \tag{15}$$

where $j = t \bmod m$.

Other Heteroscedastic Models

The simplest way of modifying the variance structure this is to incorporate an additional exponent in a similar mannerto the Box-Cox trnsformation 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, $0 \leq \theta < 1$:

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

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.

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.

References

- Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D. (2008). Forecasting with exponential smoothing: the state space approach (<http://www.exponentialsMOOTHING.net>), Springer-Verlag.
- Hyndman, R.J., Athanasopoulos, G. (2014). *Forecasting: Principles and Practice*. OTEXTS.