Marketing Analytics and Forecasting | Lancaster University Management School

# Session 1. Introduction to ETS: Selection and Combinations

*Demand Forecasting with the ADAM*

Ivan Svetunkov (i.svetunkov@lancaster.ac.uk)

## Contents



## Introduction

In this workshop, we will use the `es()` function from the `smooth` package, which implements ETS with normal distribution and supports basic mechanisms for model selection and combinations. We will use several time series from the M3 Competition as examples.

Load the packages `smooth` and `Mcomp`

```
library(smooth)
library(Mcomp)
```

# ETS model selection

The model selection and combinations are supported by ETS only in case of the maximum likelihood estimation. Both of them work with a selected pool of models. `es()` function supports several options for pools, with predefined and custom ones. Here how the basic AICc-based model selection can be done:

```
esModel1 <- es(M3[[2568]]$x)
```

This model has some summary and can be used for analysis or forecasting. Here how the summary looks:

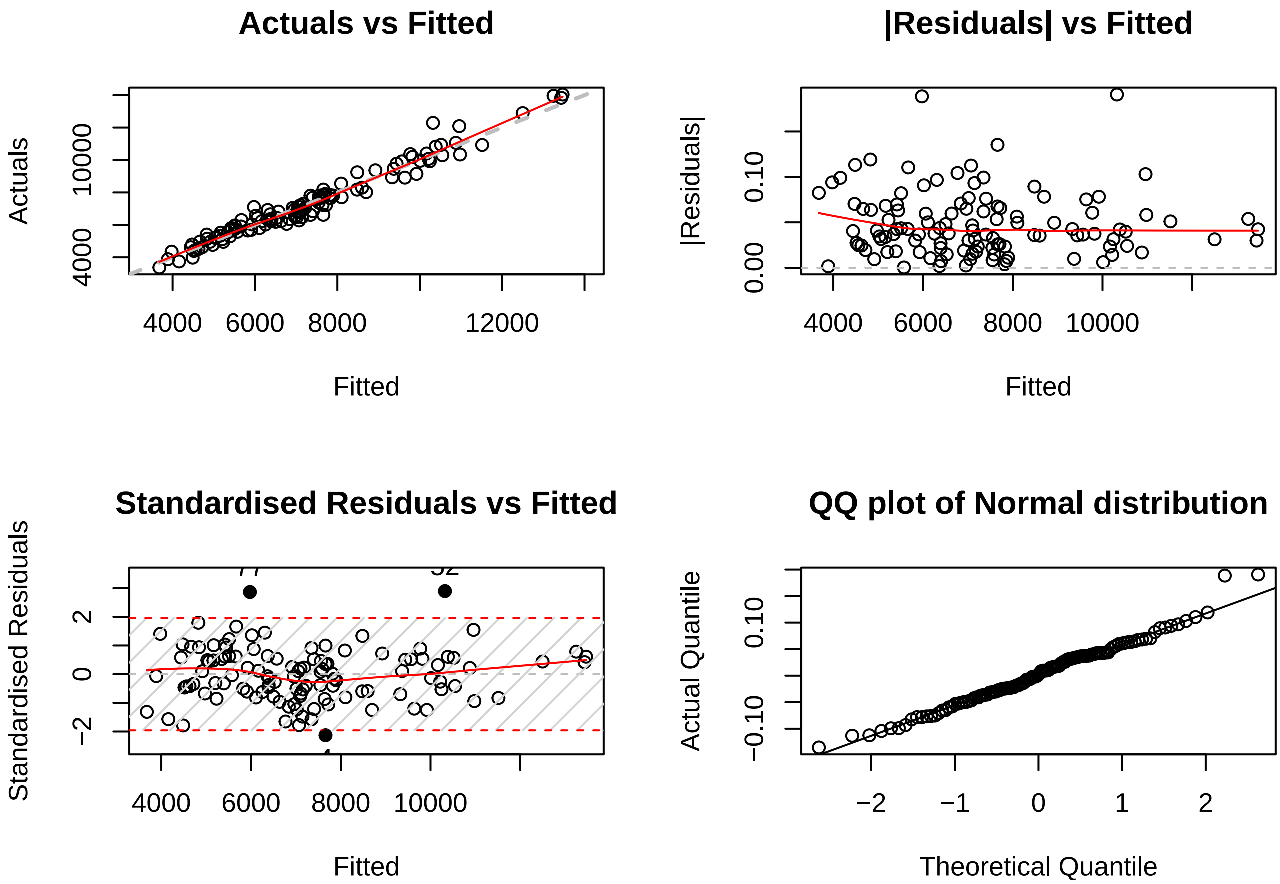
```
summary(esModel1)
```

```
## Warning: Observed Fisher Information is not positive semi-definite, which means
## that the likelihood was not maximised properly. Consider reestimating the
## model, tuning the optimiser or using bootstrap via bootstrap=TRUE.

##
## Model estimated using es() function: ETS(MMdM)
## Response variable: data
## Distribution used in the estimation: Normal
## Loss function type: likelihood; Loss function value: 863.8526
## Coefficients:
##              Estimate Std. Error Lower 2.5% Upper 97.5%
## alpha          0.0209     0.0501     0.0000      0.1199
## beta           0.0209     0.0012     0.0185      0.0209 *
## gamma          0.0000     0.0531     0.0000      0.1050
## phi            0.9688     0.0238     0.9214      1.0000 *
## level       3740.5010   324.6863  3096.1715   4382.6739 *
## trend          1.0226     0.0108     1.0012      1.0440 *
## seasonal_1     1.1825     0.0204     1.1569      1.2359 *
## seasonal_2     0.8204     0.0143     0.7948      0.8739 *
## seasonal_3     0.8266     0.0143     0.8010      0.8800 *
## seasonal_4     1.5629     0.0270     1.5373      1.6164 *
## seasonal_5     0.7427     0.0129     0.7171      0.7962 *
## seasonal_6     1.2745     0.0223     1.2489      1.3279 *
## seasonal_7     0.8922     0.0154     0.8666      0.9457 *
## seasonal_8     0.9119     0.0159     0.8863      0.9654 *
## seasonal_9     1.2282     0.0224     1.2026      1.2817 *
## seasonal_10    0.8828     0.0162     0.8572      0.9362 *
## seasonal_11    0.8383     0.0154     0.8127      0.8918 *
##
## Error standard deviation: 0.0649
## Sample size: 116
## Number of estimated parameters: 18
## Number of degrees of freedom: 98
## Information criteria:
##      AIC     AICc      BIC     BICc
## 1763.705 1770.757 1813.270 1830.030
```

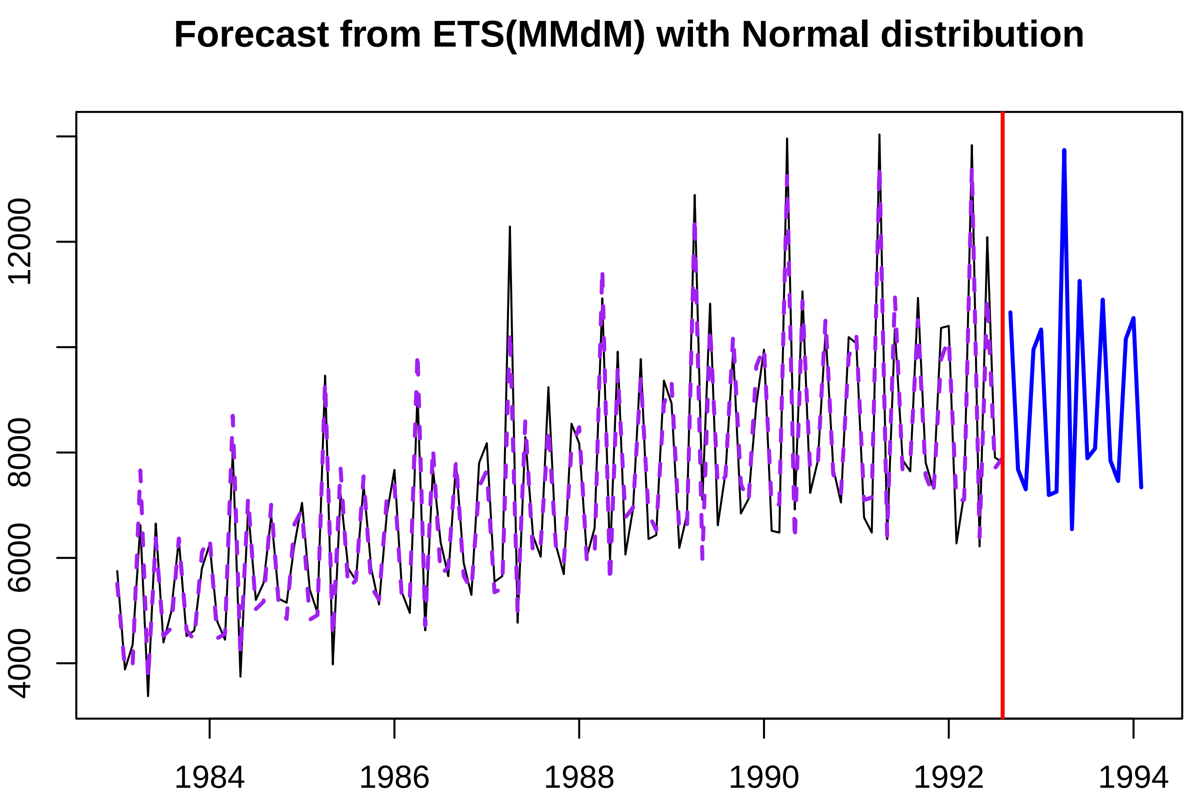
We can also do model diagnostics using the `plot()` method:

```
par(mfcol=c(2,2))
plot(esModel1)
```




And we can produce forecasts:

```
forecast(esModel1, h=18) |>
    plot()
```



The default pool of model is defined by `model="ZXZ"`, which includes all types of error and seasonality and only additive and "none" trends. The supported pools are listed in the documentation of the function. For example, we can ask for pure additive/multiplicative models via `model="XXX"` and `model="YYY"` respectively.

```
es(M3[[2568]]$x, model="YYY")
```

It is also possible to provide a custom pool of models. For example, if we want to have a model with multiplicative error and seasonality, and we know that there is a trend and want to select one, we can choose the following pool of models:

```
es(M3[[2568]]$x, model=c("MAM","MAdM","MMM","MMdM"))
```

In general, before trying any pools, it makes sense to consider, which of the models would make sense for the specific data. For example, for the series N2568, the seasonality should be multiplicative, and there is a trend (not sure what type). To align seasonality with the error term, we will try `model="MZM"`, which will test different types of trends (including "none"), keeping error and seasonality multiplicative:

```
es(M3[[2568]]$x, "MZM")
```

```
## Time elapsed: 0.58 seconds
## Model estimated using es() function: ETS(MMdM)
## Distribution assumed in the model: Normal
## Loss function type: likelihood; Loss function value: 863.8526
## Persistence vector g:
##  alpha   beta  gamma
## 0.0209 0.0209 0.0000
## Damping parameter: 0.9688
## Sample size: 116
## Number of estimated parameters: 18
## Number of degrees of freedom: 98
## Information criteria:
##      AIC     AICc      BIC     BICc
## 1763.705 1770.757 1813.270 1830.030
```

In some cases, this might result in a different model than in case of previous pools.

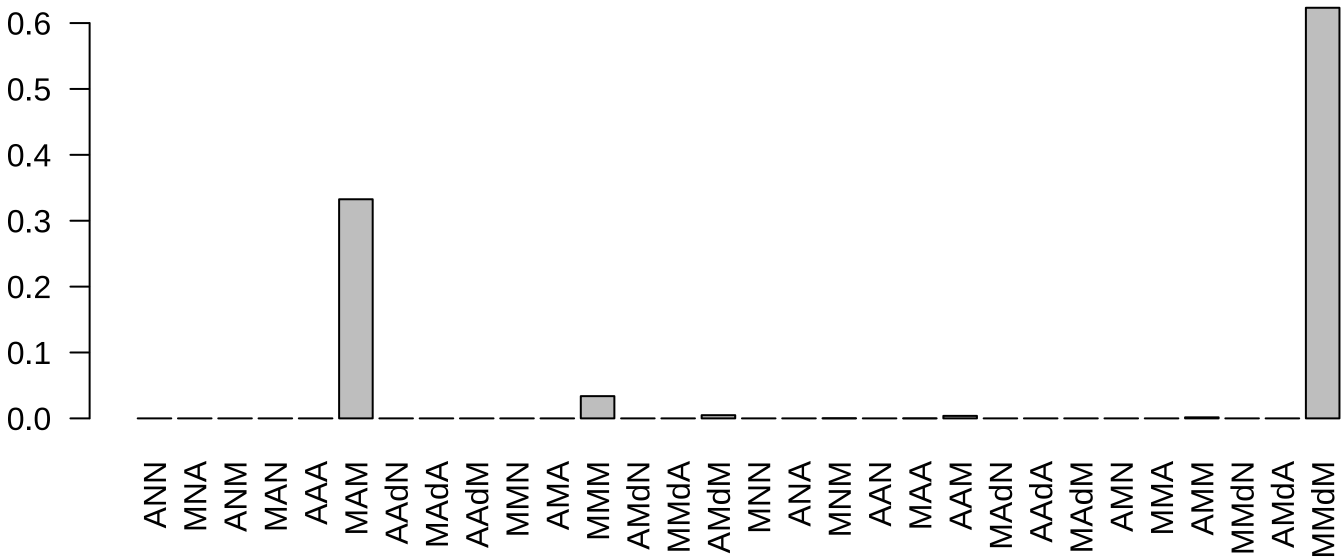
# ETS combinations

While there are many ways for forecasts combinations, the well proven one is to combine them based on AIC weights (Kolassa, 2011). The `es()` function supports forecasts combinations based on that. The simplest pool to have is all the ETS models, which is done via `model="CCC"`:

```
adamCCC2568 <- es(M3[[2568]]$x, "CCC")
```

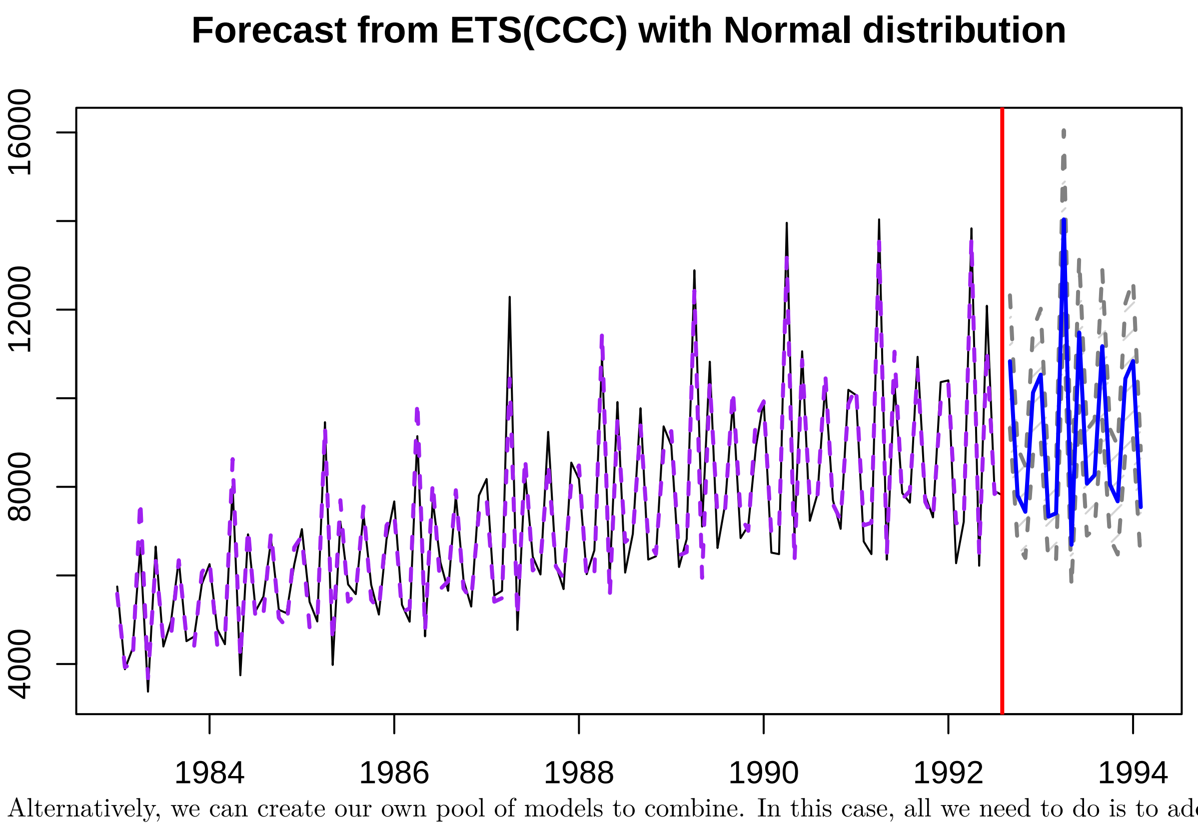
To understand how the weights were distributed between the models, we can extract them the following way:

```
adamCCC2568$ICw |>
    barplot(las=2)
```




As we see, there are two models that had much higher weights than the rest: ETS(M,A,M) and ETS(M,Md,M). As a result, they influence the combined forecasts the most. This can then be used to produce point forecasts and prediction intervals (the models with weights lower than 0.01 will be dropped):

```
forecast(adamCCC2568, h=18, interval="prediction") |>
    plot()
```




Alternatively, we can create our own pool of models to combine. In this case, all we need to do is to add a model with "C" in its name - this will trigger the combination mechanism. Here is an example:

```
adam(M3[[2568]]$x, c("CNN","MAM","MAdM","MMM","MMdM"))
```

The function also supports the predefined pools via "MCM", "YZC" etc. The important part is to have "C" in the model name, which is considered similar to "Z" and tells function to combine all models based on IC weights.

# Additional materials

For some additional examples on ETS implemented in `smooth` run:

```
vignette("adam","smooth")
```

Some additional resources on exponential smoothing:

1. ETS in the blog of Ivan Svetunkov;
2. Chapter 15 of ADAM, discussing the selection and combination mechanisms;
3. Posts on the functions in smooth package;
4. Posts of Nikos Kourentzes on exponential smoothing.