Final Project Paper

Warner Rose

# Introduction

This paper explores the process for developing models to forecast economic time series. The focus is on building many models to be tested on out-of-sample data and refined on a regular basis.

While there is an extensive literature on time series models, this paper will narrow its focus to some **exponential smoothing** methods and their performance forecasting the US unemployment rate.

Models will be compared for 2 types of forecasting scenarios: **1)** medium term (up to 6 months ahead) and **2)** short term (1 month ahead). The latter is how the **fredcast** forecasting competition works.

# Data

The Federal Reserve maintains a site that hosts economic data called [FRED (Federal Reserve Economic Data)](https://fred.stlouisfed.org/). They also have a monthly forecasting contest called [**fredcast**](https://research.stlouisfed.org/useraccount/fredcast/). The purpose of fredcast is to forecast the following 4 time series:

* [**Unemployment Rate**](https://fred.stlouisfed.org/series/UNRATE)
* [**Real GDP**](https://fred.stlouisfed.org/series/GDPC1)
* [**Payroll Employment**](https://fred.stlouisfed.org/series/PAYEMS)
* [**Consumer Price Index**](https://fred.stlouisfed.org/series/CPIAUCSL)

While the techniques to be discussed can be applied to all 4 series, this paper will focus only on the **unemployment rate** for brevity.

# Packages

The package **forecast** is an extensive library developed by [Rob Hyndman](https://robjhyndman.com/) supporting a variety of time series models. **fredr** is a package for accessing the FRED api. It can be used to download data from FRED. **zoo** provides some helpful functions for date-formatted data. **dygraphs** is a great javascript-based library for visualizing time series data.

# Unemployment rate time series from FRED

Using my FRED api key and **fredr\_key** function from the *fredr* package I authenticate my credentials:

fredr\_key(api\_key)

Download the monthly unemployment time series from FRED using **fredr\_series**.

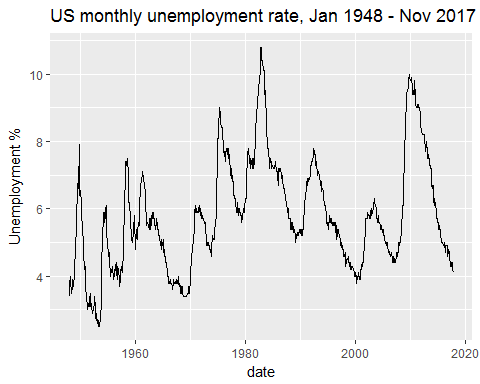
unr <- fredr\_series(series\_id='UNRATE')  
head(unr)

## Jan Feb Mar Apr May Jun  
## 1948 3.4 3.8 4.0 3.9 3.5 3.6

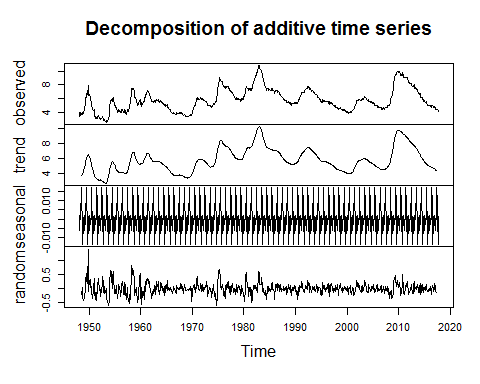
tail(unr)

## Jun Jul Aug Sep Oct Nov  
## 2017 4.4 4.3 4.4 4.2 4.1 4.1

The plot of the unemploymet rate shows varying **seasonality**.



We can decompose the time series into **trend**, **seasonal**, and **random/error** components.



Decomposing a series into appropriate terms is an essential part of exponential smoothing. We will see how the choice of either additive or multiplicative terms in the model impacts forecasting accuracy.

## Modeling

### Exponential smoothing basics

Modeling a time series using exponential smoothing techniques consists of decomposing a time series into additive or multiplicative terms in order to account for seasonality and trend. Below are 2 simple examples of **strictly** additive and multiplicative models on a series .

(Additive)

(Multiplicative)

Where is the random error term, is the trend term, and is the seasonal term.

There are **15** possible models based on components:

### Fitting models

**6 month hold out sample**

A holdout sample consisting of the last 6 data points in the series will be used to measure the forecasting accuracy of the models. The models will be fit using the series **unr2**:

unr2 <- head(unr, -6)  
tail(unr)

## Jun Jul Aug Sep Oct Nov  
## 2017 4.4 4.3 4.4 4.2 4.1 4.1

tail(unr2)

## Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec  
## 2016 4.7  
## 2017 4.8 4.7 4.5 4.4 4.3

**Fit multiple models**

The *forecast* package function **ets** can fit any of the models in the above table. The **E** stands for “error”, the **T** for “trend”, and the **S** for “seasonality”. As we will see below, these terms need to be specified when fitting a model.

A good strategy in R for fitting multiple models is to apply functions to lists:

# forecast::ets(series, model='ETS')  
# A = additive, M = multiplicative, N = none, Z = automatic  
model\_type <- c('ANN','AAA','MMM')   
ets\_models <- purrr::map(model\_type, function(x) forecast::ets(unr2, x))  
names(ets\_models) <- model\_type  
summary(ets\_models)

## Length Class Mode  
## ANN 19 ets list  
## AAA 19 ets list  
## MMM 19 ets list

The first model type **ANN** is the simplest exponential smoothing-based model with no trend or seasonal term. **AAA** contains additive error, trend, and seasonal terms. **MMM** contains multiplicative error, trend, and seasonal terms.

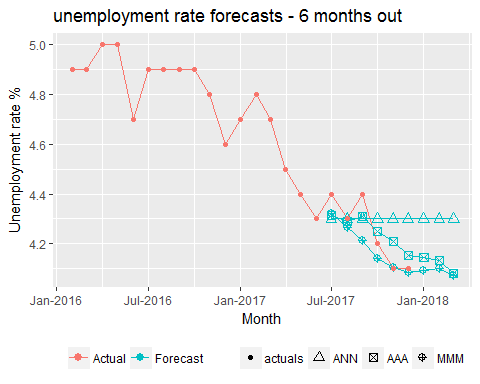
**Getting forecasts in from multiple models**

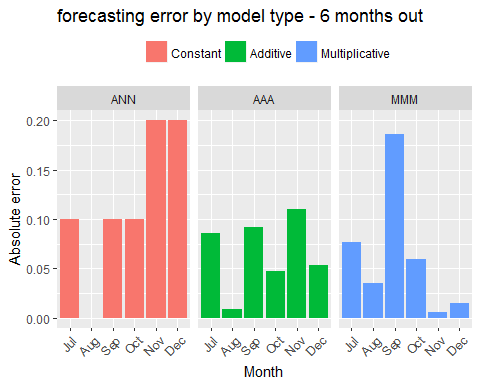
All model forecasts and actuals are combined into formats **ggplot2** (long form data frame) and **dygraphs** (matrix array consisting of multiple time series) can work with. The heavy lifting is is done by the function **gen\_forecast\_array**. See source code for details.

ets\_forecasts1 <- gen\_forecast\_array(ets\_models)

## 6 month forecast performance

The following plots show short to medium term forecasts from models fit with **May 2017** as the latest month of data. The holdout sample is included in the plot.





The model consisting of multiplicative terms **MMM** had the lowest error for medium term (3-6 month) forecasts. The constant model **ANN** was the worst performing model.

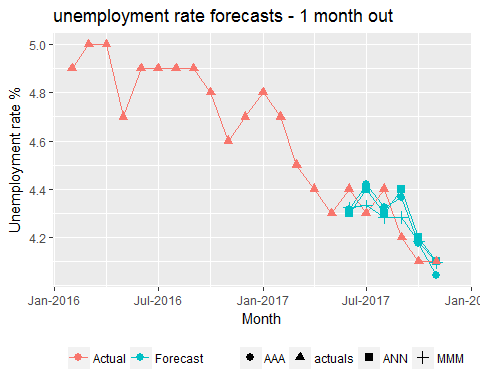
In general, time series model error increases rapidly the further we forecast into the future.

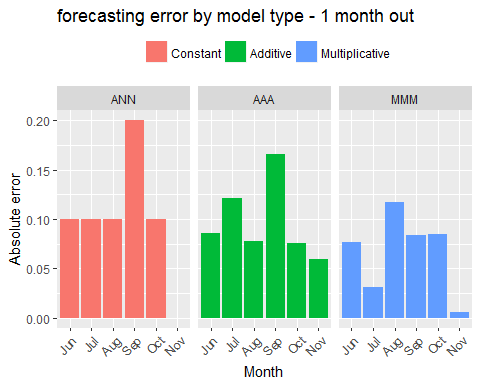
## 1 month forecast performance

The above scenario of fitting models each forecasting 6 months out is not the same scenario as in fredcast. Fredcast is always based on forecasting 1 month into the future.

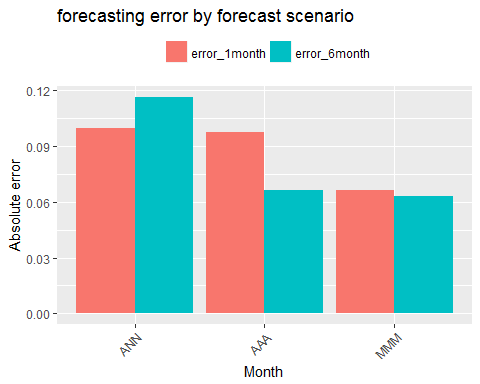
We will now fit the same models but this time simulate 6 separate fittings of models June-November. These models each forecast 1 month ahead.

Details of the algorithm for fitting multiple models on a rolling hold-out sample are in the source code.





# Conclusion



We see that the models with seasonal and trend terms perform better than the simpler constant model in terms of forecasting accuracy.

A surprising finding is that, with the exception of the constant model, the models performed better on medium-term forecasts compared to short term forecasts.

I began this paper expecting to prove that constantly refitting time series models with the latest data would improve forecasting accuracy. These results show that this is not always the case. **We have now seen that a model fit to forecast 6 months out can outperform an equivalent model forecasting 1 month out recalibrated every month.**

This finding has implications about which models are ideal to use in the fredcast competition. Furthermore, in the case of *unemployment* the data is rounded to the nearest 1 decimal place (4.3, 3.9 etc). Error is further increased when the forecasts are rounded.

In future iterations of this work I intend to test more time series models and varying forecast scenarios. I also will apply these techniques to the other time series in the fredcast competitions, all of which exhibit very different behavior from the unemployment rate.