# Additional models for basil-ts

*Andreas Beger, Predictive Heuristics*

*3/22/2018*

This note gives some details on additional models that could be added to the time series forecaster (basil-ts). Here is a summary:

| Model | Additional regressors | Easy to add |
|---|---|---|
| ARIMA (currently used) | Yes | Yes |
| NNETAR, auto-regressive neural net | Yes | Yes |
| ETS, exponential smoothing | No | Yes |
| Arithmetic/Geometric Brownian motion | Yes | Yes |
| Count models | Mixed | No |
| SGDLM | Yes | No |

## Existing model and easy to add extensions

The current version of the forecaster fits a seasonal ARIMA model. The number of observations per year is automatically determined from the input time series if possible, and if not a non-seasonal **ARIMA** model is fitted. There are also some heuristics for taking the log or square root of a input time series before applying the model algorithm, e.g. for count series this can help with non-stable variance.

A full model is specified as ARIMA(p,d,q)(P,D,Q)$_m$, where p/P is how many (seasonal) autoregressive terms are included, d/D is the order of (seasonal) differencing, and q/Q is the number of (seasonal) moving average terms. All forms allow for the inclusion of covariates.

Other models: the package used for the ARIMA models includes several other time series model types, all with options to automatically determine model order. Since they share the same API these would be easy to incorporate into the existing code.

**NNETAR**: a neural network with a single hidden layer and inputs that consist of lagged y values, including seasonal lags. There is a provision for automatically determining the AR order. I used this in the food price forecast examples from a while ago.

**ETS**: Exponential smoothing state space models. Forecasts are based on different kinds of weighted averages of past y values. A simple version of the model that forecasts based on a smoothed level can be written as:

$$\begin{aligned}\hat{y}_{t+1|t} &= \ell_t \\ \ell_t &= \alpha y_t + (1-\alpha)\ell_{t-1}\end{aligned}$$

where $\ell_t$ is the smoothed level at time $t$ and the smoothing parameter $\alpha$ governs how much weight is given to more recent observations versus past observations. Extensions incorporate different kinds of trend (none, additive, or multiplicative; damped or not), seasonal components (none, additive, or multiplicative), and error structure (additive or multiplicative).

A full model has the form ETS(Error, Trend, Seasonality). This does not allow for the inclusion of additional covariates.

As a reference for these, see Hyndman & Athanasopoulos, 2018, *Forecasting: principles and practice* at https://www.otexts.org/book/fpp.

## Brownian motion

David's email from 21 March 2018 references two models mentioned by Ali, arithmetic and geometric Brownian motion. I believe that the discrete time equivalents of these are already approximated by ARIMA. For arithmetic Brownian motion, ARIMA(0, 1, 0) is a random walk with drift and thus should be similar; it can be written as $y_t = y_{t+1} + c + \epsilon_t$, where $c$ is drift. If I understand right geometric brownian motion should correspond to a random walk on the log scale, i.e. an ARIMA(0, 1, 0) model of the logged input time series, or $\ln(y_t) = \ln(y_{t+1}) + c + \epsilon_t$.

There is a heuristic for logging some input time series before applying the ARIMA model, so models similar to arithmetic and geometric Brownian motion are already within the scope of models considered by the automatic model order algorithm for ARIMA. I can add explicit versions of both to the list of models returned by default.

## Count TS models

There are several extensions of ARIMA and state space models specifically for modeling count time series, i.e. in R packages "tscount", "GLARMA", and "KFAS". This would take some work to identify which of these options to incorporate.

## SGDLM

Simultaneous graphical dynamic linear model. This is a state space model with flexible componenets, e.g. different kinds of trends, and based on the inclusion of co-evolving other time series (the graph part). We implemented a version of this for the Mercury project to forecast protests in several countries.

This would be the most difficult to include since we would have to develop an automatic approach for specifying the state space form and dependency graph used to select other time series. (We could let forecasters construct the dependency graph.) It also takes more time than the other approaches to estimate, which may be a constraint for forecasting large numbers of IFPs.