Time Series Models

Kai Ming Lee

Introduction

A time series is a set of observations $\{y_t\}$ indexed by time t. Interesting time series will always contain some uncertainty about future observations, so most time series models are formulated as a set of (time-correlated) random variables $\{Y_t\}$, from which the observations are assumed to be a realisation. Specific time series models place different restrictions on the distributions of Y_t and especially on the dependencies or correlations between Y_t's at different times.

Useful applications of time series models include

- forecasting
- trend estimation and noise filtering
- non-experimental causal inference
- anomaly detection

Forecasting and trend estimation can be done with model-free ad-hoc methods such as rolling averages, but stochastic models can provide estimates of forecast uncertainty and standard statistical diagnostics to signal if the chosen method is inappropriate. For causal inference and anomaly detection, we almost always need a statistical model, because we need to decide what is “normal”.

Overview of Models

I will briefly review four traditional approaches to time series models:

1. regression/curve fitting

2. exponential smoothing

3. ARIMA

4. structural time series / unobserved components models

Most of these approaches have readily usable implementations in both R and Python.

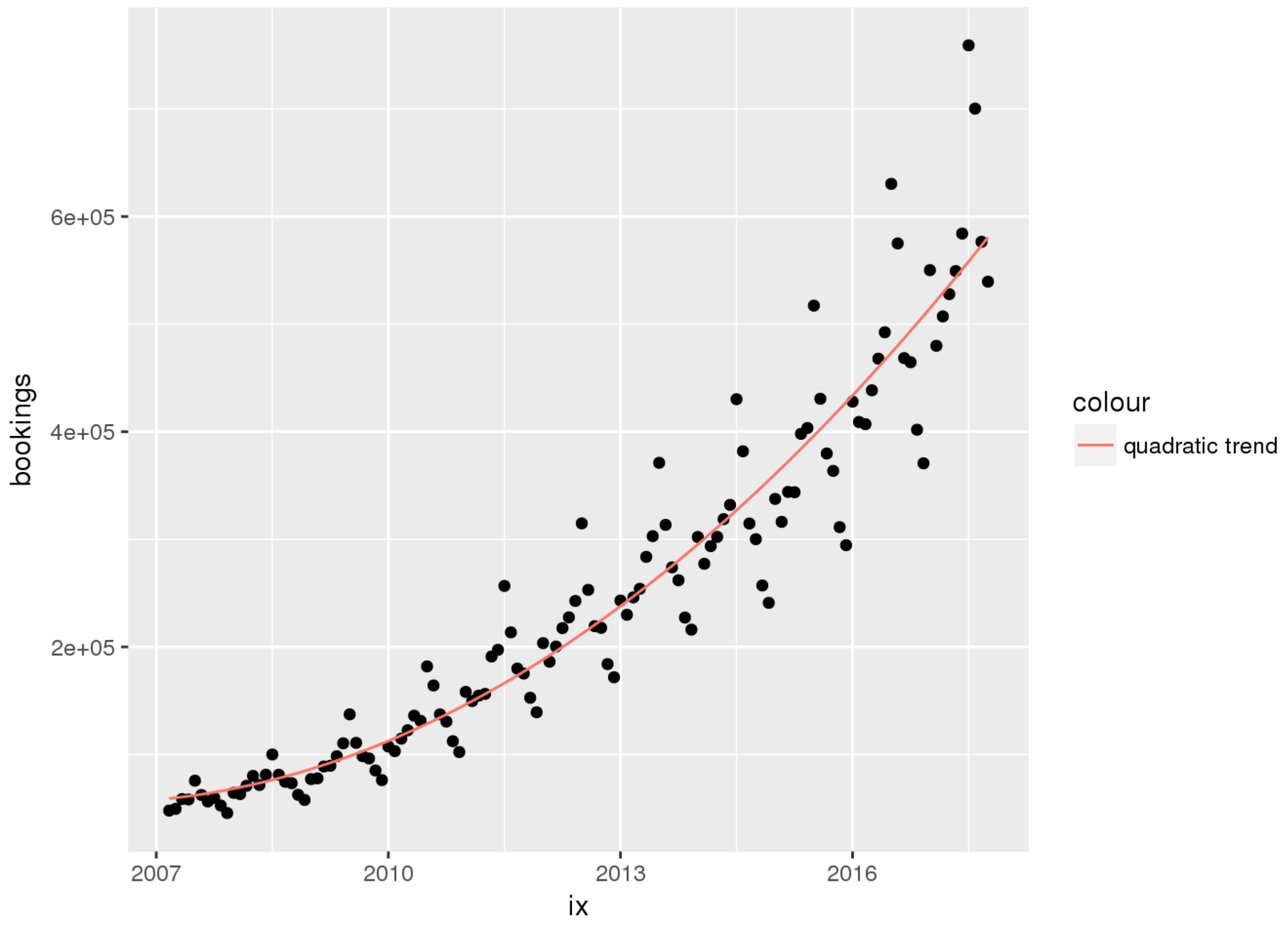
Note that the scope of many of these models is probably more limited that we might initially expect:

- the time index t is usually assumed discrete and sampled at regular intervals (e.g., monthly, weekly, daily, hourly; standard ARIMA and seasonal exponential smoothing models are optimized for quarterly and monthly observations and don’t handle daily or higher frequencies well when there are multiple periodicities like intra-week and intra-year)
- the observations y_t are (approximately) continous, not categorical or low-value counts
- the observations y_t are univariate (scalar), not a vector of values or multiple observations at t

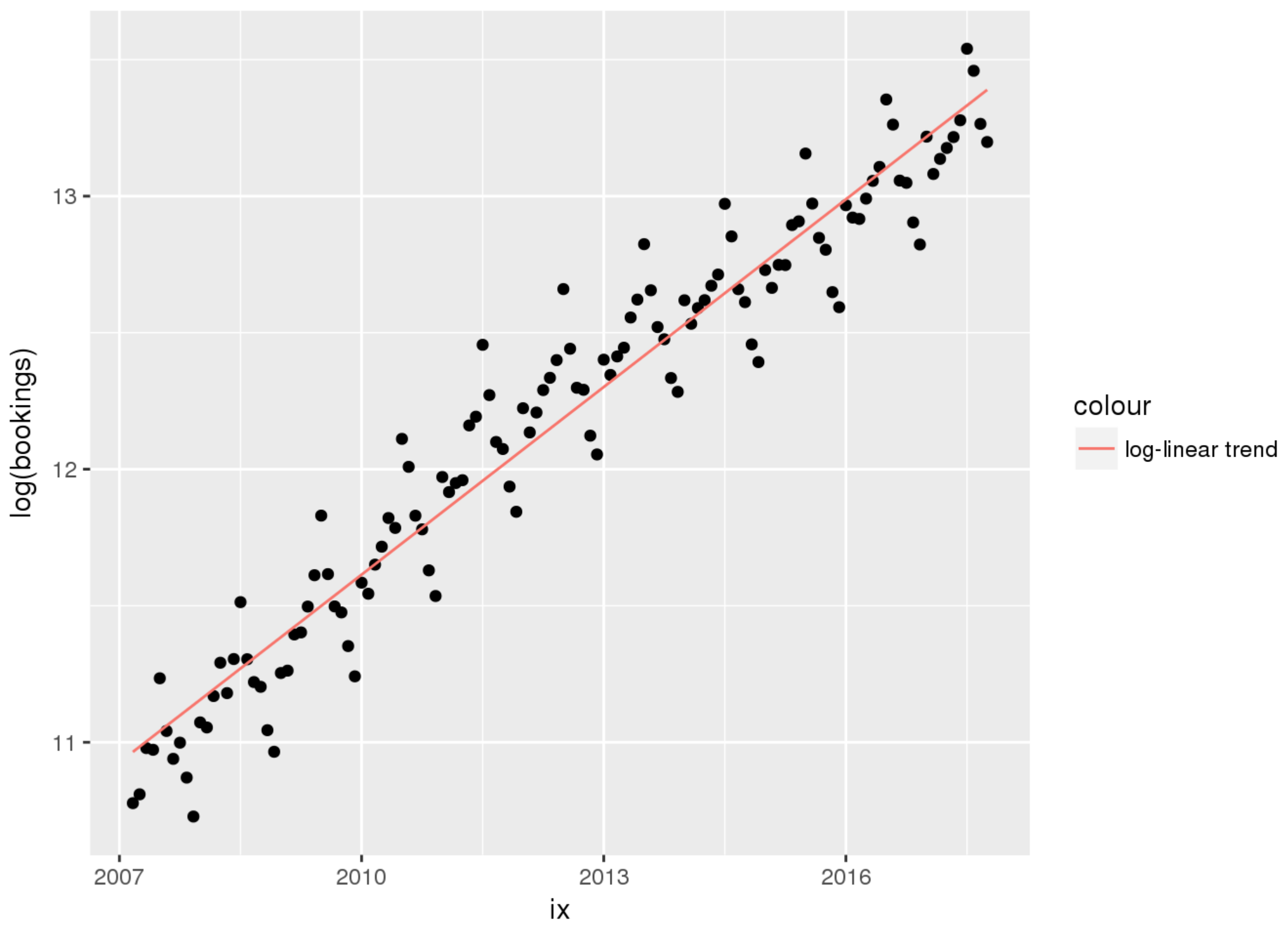
This may all appear very restrictive, but it covers many practical business needs, especially models that do handle weekly and daily observations. At the end of this article, I’ll discuss some approaches for time-indexed data that fall outside this scope.

1. Curve fitting / regressions

The simplest approach is to fit some smooth curve through the observed points (t, y_t). This can be implemented as a simple linear regression of y_t on an intercept and on t, and perhaps also on t^2 to allow for some curvature in the trend. For the number of bookings/month in a certain country a quadratic curve fit looks like this:



However, trend curvature is more commonly handled by fitting a linear model to $\log y_t$ (or more generally a Box-Cox transformed y_t), rather than a non-linear model to y_t (because this will also take care of the proportionaly increasing seasonal variation).



These approaches can work pretty well for pure forecasting applications for time series with prominent trends and seasonality, but they have some important deficiencies:

1. OLS places the same fitting penalty on residual erors near the end as near the start of the dataset
2. residuals are serially correlated: over-/underforecasts are likely followed by over-/underforecast in the same direction in the next period
3. the functional relationship is static in time, and increasingly difficult to maintain over longer time periods (notice that I fitted data starting from 2007 rather than the full history)
4. no prediction intervals are provided (you could get them from standard OLS formulas, but these are unrealistically narrow)

Most of these issues are symptoms of the same cause: standard regression approaches have a limited notion of *locality*. With time series data, observations located close together in time are generally more relevant than distant observations, and in forecasting the recent past is more important than observations from several years ago. This is the main distinguising feature of time series data compared to cross-section data, and largely ignored in simple regressions.

Regression approaches also have important advantages especially over exponential smoothing and ARIMA models: it's easy to add exogenous variables/covariates and custom features (such as holidays, interventions, trend breaks and outliers), and to handle irregular and high frequency data (with multiple and/or non-integer periodicities).

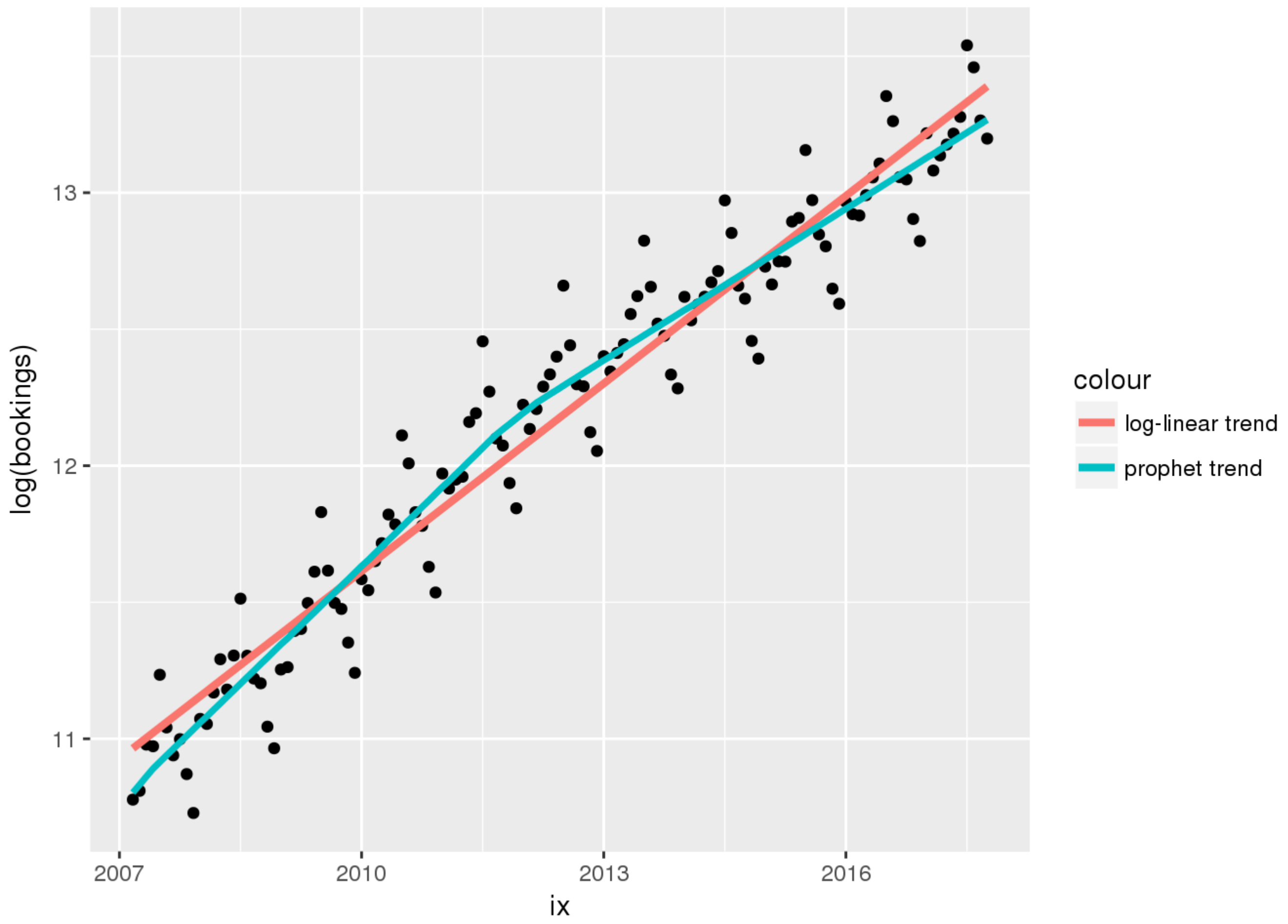
Many variations on regression curve fitting exist, such as piecewise splines, smoothing splines, trigonometric seasonal terms, LOESS. Simple time series regressions are easily implemented using standard R `lm()` , Python sklearn `LinearRegression()` or Python Statsmodels `OLS()` functions. For more complicated features, it takes a bit more effort to build them in Python (with a combination of Pandas, Statsmodels, SciPy and NumPy), than in R, which has many of them ready-made.

One of the most evolved forms of the curve fitting / regression approach is Facebook's recently released Prophet model. Its essential features are:

- Piecewise linear or logistic trends, with automatically selected turning points. Through some Bayesian cleverness, they overcome some the traditional weak points of simple time series regressions (deterministic trends lead to unrealistically narrow prediction intervals)
- Trigonometric functions to fit seasonality. Compared to SARIMA and exponential smoothing/Holt-Winters, these have the advantage that it's easy to specify multiple seasonalities (e.g. both intra-year and intra-week) and non-integer seasonalities (e.g. intra-year seasonality with weekly observations)
- Easy inclusion of covariates (holidays, interventions, outliers)

The model still retains some of the traditional drawbacks of curve fitting methods, and we'll discuss it in detail in the second Journal Club meeting on Time Series Modelling.

```
## Initial log joint probability = -2.31797
## Optimization terminated normally:
##   Convergence detected: relative gradient magnitude is below tolerance
```



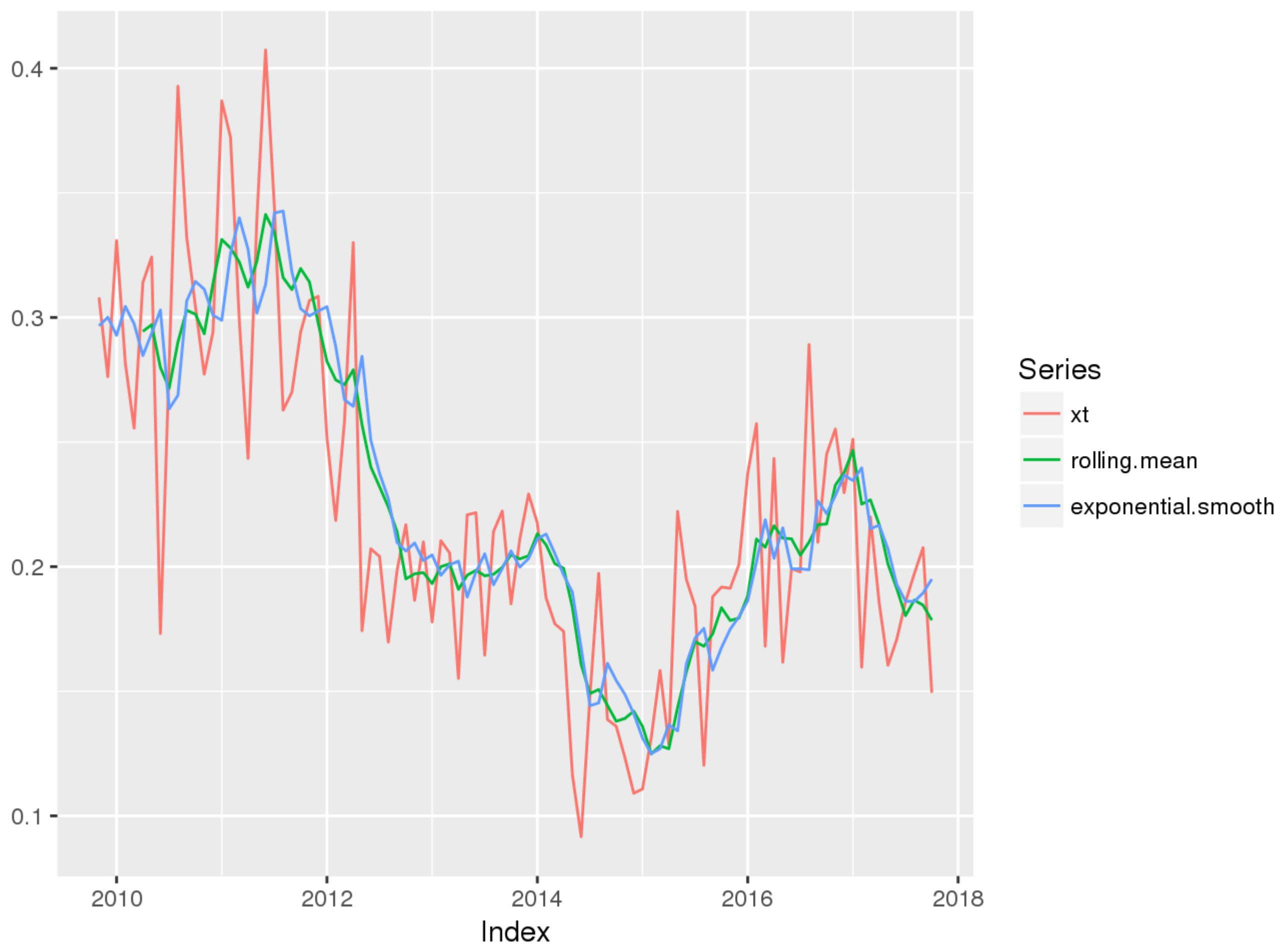
2. Exponential smoothing

For time series without prominent trends or seasonality, most of the modelling deals with *persistence* or serial correlation and with noise reduction. Initially, this was done with various ad-hoc weighted or unweighted rolling average procedures, and later more formally with ARMA models.

Take for instance the following transformation of the previously plotted monthly bookings data series

$$x_t = \log y_t - \log y_{t-12} = \log \frac{y_t}{y_{t-12}}$$

The log-transform stabilizes increasing seasonal and error variances, while the 12-month differencing operation removes most of the trend and seasonality, resulting in a year-over-year growth rate series that looks as follows:



The historical average growth rate in the (post-2009) sample is 0.22. It's clear from the plot that this is not a very good forecast for the near future, since there is a lot of persistence and clustering of periods with higher- and lower-than-average growth. (At the end of the sample the actual value of x_t has been below 0.22 for 9 months in a row). We could take just the very last observation (0.15) as forecasts, but that will be subject to a lot of random fluctuations. The simplest way to balance these two extremes is to take at each point in time the equal-weighted average of only the last k months, resulting in a k-period rolling mean

$$m_t = \sum_{j=0}^{k-1} w_j x_{t-j}, \quad w_j = \frac{1}{k} \text{ for all } j$$

People soon realized that there are few datasets for which equal weights and a discrete cutoff at lag k seems optimal, so a simple alternative with weights declining at an exponential rate became popular:

$$m_t = \sum_{j=0}^{\infty} w_j x_{t-j}, \quad w_j = (1-\alpha)\alpha^j \text{ for all } j, \quad 0 < \alpha < 1$$

This is called exponential smoothing or exponentially weighted moving average. (The term $(1-\alpha)$ is only there to make the weights sum to 1.) Both procedures have one free parameter (k and α), which could be chosen by minimizing the sum of squared in-sample 1-step-ahead forecast errors (or just fixed at some arbitrary value). Emprically both procedures can give pretty similar results, as shown in the plot. In this particular example, the very last estimates of m_t (which we can use as forecasts) are 0.18 and 0.19 respectively.

Exponential smoothing has some nice mathematical properties, for instance, the calculation of m_t can be done recursively as $m_t = \alpha m_{t-1} + (1-\alpha)x_t$ once we have some initial estimate m_1, without having to store the history of x_t or the entire set of weights w_j. This is also the most primitive form of the Kalman filter.

Simple exponential smoothing is easy and intuitive, but extending them to capture trends and seasonality was a major effort (done by Holt and Winters in the 1950s and 60s) and lead to more and more complicated recursions (https://www.otexts.org/fpp/7/5 (https://www.otexts.org/fpp/7/5)). Since these became very difficult to extend with custom features and higher frequency observations, the approach seems somewhat at a dead end, though that didn't stop Hyndman et.al. to write a 386 page book about it in 2008 (http://www.springer.com/gp/book/9783540719168 (http://www.springer.com/gp/book/9783540719168)). For monthly and quarterly observations, forecast performance of exponential smoothing approaches are still hard to beat: http://kourentzes.com/forecasting/2017/07/29/benchmarking-facebooks-prophet/ (http://kourentzes.com/forecasting/2017/07/29/benchmarking-facebooks-prophet/)

On the software side:

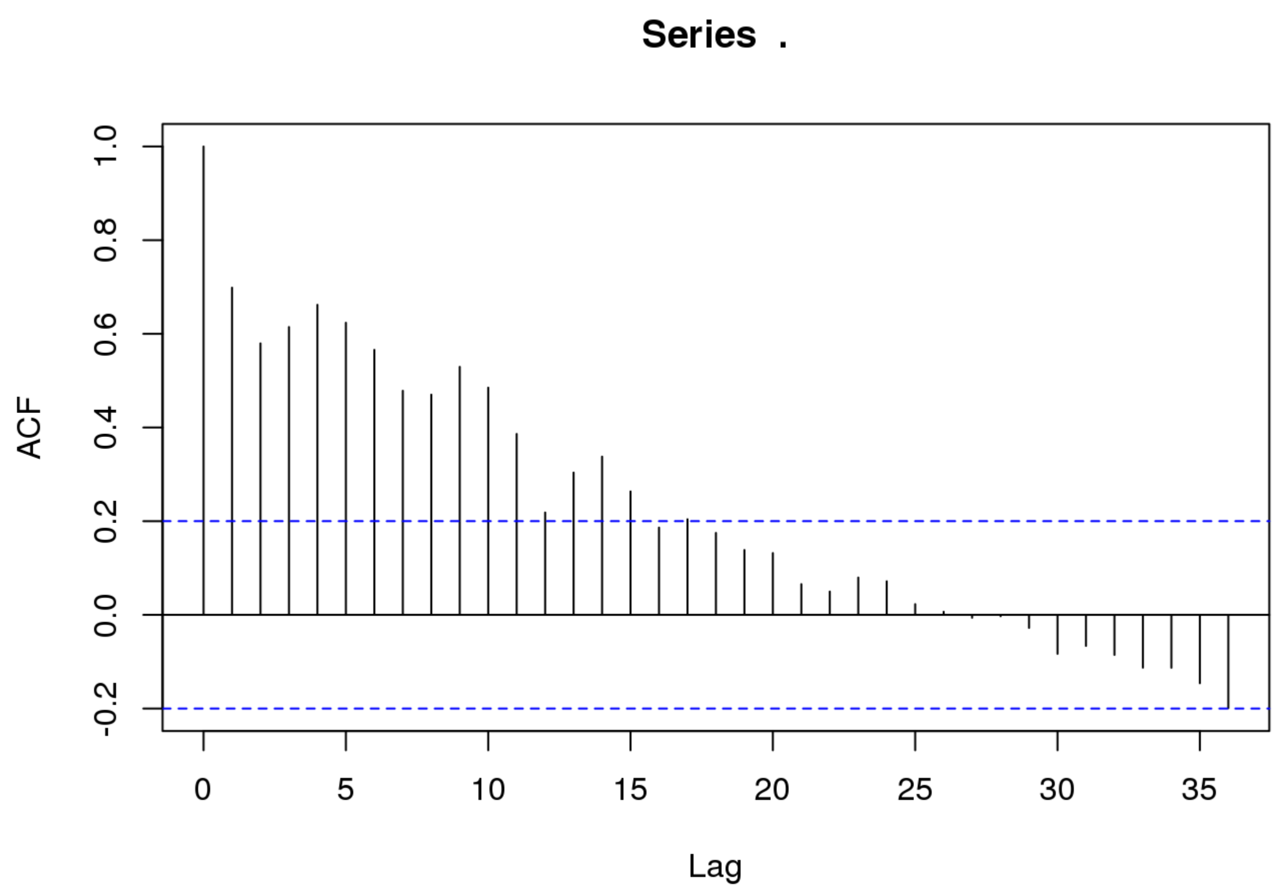
- R forecast has the most developed form of exponential smoothing, in the `ets()` function, with many setting for trends and seasonality. It also has separate Holt-Winters functions, but nowadays these are better viewed as special cases of `ets()` .
- Python Pandas has simple exponential smoothing in the form of `ewm()` functions (and other types of weighted rolling average methods in `rolling()`), but they don't handle seasonality. There are some packages implementing Holt-Winters, but recent Statsmodels versions provide generic state space objects and Unobserved Components models, exponential smoothing with trends and seasons are better treated as special cases of these.

3. ARMA, ARIMA, SARIMA

Curve fitting and exponential smoothing remain popular among forecast practicioners, but academic research has been largely dominated by ARIMA and extensions. The standard reference is Box and Jenkins (1970) (5th ed.: http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1118675029.html (http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1118675029.html)). (Nobody ever reads it, but it's mandatory to cite it in any work relating to ARIMA models.)

There are at least a dozen of textbooks and tutorials on ARIMA models, ranging from quite practical (https://www.otexts.org/fpp (https://www.otexts.org/fpp), http://www.springer.com/la/book/9783319298528 (http://www.springer.com/la/book/9783319298528)), to math-heavy (http://www.springer.com/la/book/9781489900043 (http://www.springer.com/la/book/9781489900043)). (All the extra math won't help much with producing more accurate forecasts, but are nice mental excercises. For the curious, these free lecture notes are pretty similar to the second Brockwell and Davis book: https://staff.fnwi.uva.nl/p.j.c.spreij/onderwijs/master/aadtimeseries2010.pdf (https://staff.fnwi.uva.nl/p.j.c.spreij/onderwijs/master/aadtimeseries2010.pdf))

ARIMA models are not easy to understand, so here's a modest attempt at developing some intuition. Recall the bookings growth rate series $x_t = \log y_t - \log y_{t-12}$ plotted and smoothed in the previous section. The most prominent trend and seasonality in the raw bookings series y_t was removed by the differencing operation. But as we saw, there is still a lot of persistence: periods higher or lower than the full sample average tend to cluster together. Formally, there is significant *autocorrelation*:



This autocorrelation plot shows the sample correlations between historical growth rates and growth rates from *Lag*-months earlier. In this case it tells us that current higher/lower-than-average growth rates are predictive of higher/lower-than-average growth rates up to about 15 months in the future.

In principle we could use the entire sample autocorrelation function to develop forecasts, but that would a lot of parameters (one for each lag). ARMA models are just a way to model the autocorrelation function with a small number of parameters. The simplest example is an AR(1) model $x_t = \phi x_{t-1} + \epsilon_t$, which models autocorrelations declining at exponential rate with a single rate-of-decline parameter ϕ.

Modelling autocorrelations only make sense for series without strong trends, so trending series need to have the trend removed first. This could be done by first fitting a deterministic function of time (like in the regression examples) and modelling deviations from the fitted function as an ARMA process. Alternatively, trends can be removed by taking period-to-period differences and/or seasonal differences (as in our current example), and model the resulting growth rates as ARMA processes. The model on the original (undifferenced) series is then an ARIMA model. The Box-Jenkins tradition strongly favours differencing, mostly because the resulting prediction intervals are more credible than thos from the deterministic trend-fitting approach. But differencing makes the resulting model on the original series more difficult to understand, especially when both single period and seasonal differences are involved (SARIMA). A fitted SARIMA model on log-bookings might look as follows:

$$x_t = (y_t - y_{t-12}) - (y_{t-1} - y_{t-13})$$
$$x_t - 0.12x_{t-1} + 0.29x_{t-12} - 0.21x_{t-13} = 2.13 + \epsilon_t - 0.56\epsilon_{t-1} - 0.41\epsilon_{t-6} - 0.65\epsilon_{t-12} + 0.46\epsilon_{t-13}$$
$$\epsilon_t \sim \text{IID}(0, \sigma^2)$$

which is uninterpretable by humans, and only used as a black-box predictor.

Actual application of (S)ARIMA is nowadays largely automated, at least if you're dealing with a limited number of series without multiple periodicities or very long periods (meaning: no daily or intraday unless you only need only short term forecasts, weekly is probably borderline doable with several years of data). In R using the `forecast` package, the main steps are

1. putting your data in a `ts()` object with appropriate frequency parameter (normally 12, 4 or 1)
2. feeding it to `auto.arima()` , often with `lambda=0` (which performs a log-transform, but the forecasts are automatically back-transformed with a bias correction)
3. plot the resulting forecasts and see if they make sense

What it does under the hood is an automatic selection of the 6 order parameters in a SARIMA $(p, d, q)(P, D, Q)$ by maximizing some penaltized likelihood and running several different statistical tests, and then estimating the remaining ARMA parameters by Maximum Likelihood. If you don't have too many series to fit, it's usually easier to plot some long-horizon forecast (with prediction intervals) to see if they make sense, rather than trying to make sense of estimated coefficients.

The Python Statsmodels implementation of SARIMA is a less developed than the R version. Order selection of is largely non-automated, and can be a hassle in practice.

4. Structural Time Series

Also known as unobserved components models, Bayesian dynamic linear models, or state space models these are less widely known than the previous approaches, but have several advangages:

- they're flexible, interpretable and easy to modify with custom features like regression models (and unlike exponential smoothing and SARIMA models)
- they have stronger locality than regression models, can handle gradual changes with stochastic trends and seasons
- they're fully specified statistical models with prediction intervals and diagnostic tests

These type of models are fairly old (most of the work on structural time series was done by Harrison and Stevens in the 1970s, Harvey in the 1980s and Durbin and Koopman in 1990s, while their superclass, the generic state space model and Kalman filtering were developed in the 1960s), but have been put in the spotlight recently by Google's easy-to-use `CausalImpact` and `bsts` R packages which will be discussed in the first Journal Club on Time Series models.

Seasonal regression and curve fitting models, in contrast to exponential smoothing and SARIMA models, are easily visualised and interpreted as the sum of a slowly moving trend component μ_t, periodically seasonal components τ_t, arbitrary other covariates x_t, and an IID error process ϵ_t:

$$Y_t = \mu_t + \tau_t + \beta' x_t + \epsilon_t$$

Structural time series retain this structure, but replace the deterministic trend μ_t and seasonal components τ_t by stochastic processes. Notice that a deterministic linear trend with fixed intercept μ_0 and slope δ

$$\mu_t = \mu_0 + \delta t$$

can be written as the recursion

$$\mu_{t+1} = \mu_t + \delta$$

In structural time series, the slope and level are made stochastic by adding IID errors $\eta_{0,t}, \eta_{1,t}$ to the slope and the level at each time step:

$$\begin{aligned} \mu_{t+1} &= \mu_t + \delta_t + \eta_{0,t} \\ \delta_{t+1} &= \delta_t + \eta_{1,t} \end{aligned}$$

In the deterministic regression version, we don't observe μ_0, δ directly, but they can be estimated from the data y_t using OLS. In structural time series models, we don't observe the components μ_t or δ_t directly, but they can be estimated from the data y_t using the Kalman filter. Similar to the trend, the seasonal component τ_t can be made stochastic and by writing a regression fixed dummy or trigonometric specification in recursive form and add its own IID error process

$$\tau_{t+1} = -\sum_{s=1}^{S-1} \tau_t + \eta_{2,t}$$

The full specification for Y_t with stochastic trends and seasons has four separate unobserved error processes $\epsilon_t, \eta_{0,t}, \eta_{1,t}, \eta_{2,t}$, each with its own variance parameter. When the variances are known, all the unobserved components μ_t, δ_t, τ_t (or equivalently, their error processes) can be estimated from y_t by the Kalman filter. When the variances are unknown, they can be estimated in a separate step using Maximum Likelihood or MCMC.

On the software side

- Some other R packages beside `bsts` that handle state space and structural time series models include `dlm`, `KFAF`, `FKF`, in various stages of completeness, see (https://www.jstatsoft.org/article/view/v041i04/v41i04.pdf (https://www.jstatsoft.org/article/view/v041i04/v41i04.pdf) for a pre-`bsts` overview. Disclaimer: I have not tried most of them.). The most user friendly implementations are probably `bsts` and `dlm`.
- In Python, Statsmodels recently gained a fairly complete implementation in the `tsa.UnobservedComponents` submodule (http://www.statsmodels.org/dev/statespace.html (http://www.statsmodels.org/dev/statespace.html)).

Out of scope

Numerous types of data with a prominent time index t fall outside the restrictions mentioned in the introduction. They are handled by different model classes, many of which have their own large body of literature. For instance, continuous time or irregularly sampled data could be modelled by specialised continuous time stochastic processes (Brownian motion, Ornstein–Uhlenbeck processes). These are useful to learn if you're into high-frequency trading or modelling derivatives but of limited value in business applications. Various curve fitting techniques (LOWES, smoothing splines, Facebook Prophet) and generic state space models can deal pretty well with irregularly sampled data. For ARIMA and exponential smoothing models, we're mostly limited to aggregating/resampling data to some regular observation frequency.

Time-indexed categorial observations or low-value (discrete) counts are commonly modelled as directly or indirectly observed Markov Chains. Often the interest is in the effect of covariates (which could be treatments or interventions), rather than predictions of the value of y_t. The `bsts` R package implements the state space / hidden Markov model approach for both continuous (Normal, lognormal, student-t) variables and for low-value counts and binary response variables (using Poisson and logit models).

For vector-values observations y_t (e.g. daily aggregates where every vector component in y_t is a country, UFI or property), there are several approaches. Vector Autoregressive Models (VAR or even VARMA) models are generalizations of univariate ARMA models, popular for forecasting macro-economic time series, and some limited causal inference. In unrestricted form, these attempt to capture all cross-correlations in the data, both in the time dimension and between all the different vector components, at all lags and leads. If you think univariate ARIMA models are already overparameterized, this will square your problems. The number of free parameters increases at k^2-rate with k vector components, and placing sensible restrictions on them is difficult, so they're very hard to scale in the number of components. (The typical macro-economic VAR model probably has 3 to 6 components.) A popular way to restrict high-dimensional time series is to model observations as regressions on some latent low dimensional process (extracted from observations using PCA and modelled as as a VAR); known in the literature as Dynamic Factor models. A more practical approach is to treat different components as univariate time series and combine them in a separate step, as implemented in the `hts` R package by Hyndman et.al. Yet more practical approach is to just model the different components as univariate time series and ignore the cross-correlations. In general it's difficult to say whether the efficiency gains from pooling components/series and modelling the cross-correlations are worth the additional complexity and costs of a multi-variate model.

Panel data or longitudinal models offer an alternative approach to modelling non-scalar time-indexed observations, with it's own tradition and large body of literature. These are models for repeated observations on the same individual (where individuals could be persons, cities, countries, hotels or any other identifiable entities). Many important datasets in Booking.com have a clear panel structure, and could probably benefit from a careful panel treatment. (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.465.7329&rep=rep1&type=pdf (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.465.7329&rep=rep1&type=pdf)) Panel data models are used to estimate effects of covariates on y_t, not for forecasting y_t.

Note that in standard time series models, the variable of interest is y_t, some quantity observed at time t, not t itself, nor times between events, nor the effects of covariates on event times. These are handled by survival/duration models.

Pure time series models rely only on time t and historical observations y_t as input. Adding covariates / exogenous variables X_t is straightforward in regression models, structural time series models and basic ARMA models, at least when the covariates are deterministic (such as indicator variables for holidays and or interventions). The iterpretation of a model with stochastic covariates can be quite difficult, especially when they have their own time dynamics (relevant notes for ARIMAX: https://robjhyndman.com/hyndsight/arimax/ (https://robjhyndman.com/hyndsight/arimax/)).