Forecasting at Scale

Sean J. Taylor and Benjamin Letham

January 13, 2017

Abstract

There are a variety of challenges that come with producing a large number of forecasts across a variety time series. Our approach to forecasting *at scale* is a combination of configurable models and thorough analyst-in-the-loop performance analysis. We present a forecasting approach based on a decomposable model with interpretable parameters that can be intuitively adjusted by the analyst. We describe performance analysis that we use compare and evaluate forecasting procedures, as well as automatically flag forecasts for manual review and adjustment. Tools that help analysts to use their expertise most effectively enable reliable forecasting of a large variety of business time series.

1 Introduction

Forecasting is a data science task that is central to many activities within an organization. For instance, large organizations must engage in capacity planning to efficiently allocate scarce resources and goal setting in order to measure performance relative to a baseline. Producing high quality forecasts is not an easy problem for either machines or for most analysts. We have observed two main themes in the practice of creating business forecasts:

1. Completely automatic forecasting techniques can be brittle and they are often too inflexible to incorporate useful assumptions or heuristics.

2. Analysts who can produce high quality forecasts are quite rare because forecasting is a specialized data science skill requiring substantial experience.

The result of these themes is that the demand for high quality forecasts often far outstrips the pace at which the organization can produce them. This observation is the motivation for the research we present here – we intend to provide some useful guidance for producing forecasts *at scale*, for several definitions of scale which we discuss in this section.

The first two types of scale we address are that business forecasting methods should be suitable for 1) a large number of people making forecasts, possibly without training in time series methods; and 2) a large variety of forecasting problems with potentially idiosyncratic features. In Section 2 we present a complete, general time series model which is flexible and configurable by non-experts who may have domain knowledge about the data generating process but little knowledge about time series models and methods.

The third type of scale we address is that in most realistic settings, a large number of forecasts will be created, necessitating efficient, automated means of evaluating and comparing them, as well as detecting when they are likely to be performing poorly. When hundreds or even thousands of forecasts are made, it becomes important to let machines do the hard work of model evaluation and comparison while efficiently using human feedback to fix performance problems. In Section 3, we describe a simple and robust forecast evaluation system based on 1) using simulated historical forecasts to estimate out-of-sample performance and surface outliers, 2) a flexible procedure for model comparison, and 3) a simple procedure for surfacing problematic forecasts to allow a human analyst to effectively critique models and understand what went wrong.

It is worth noting that we are ignoring the typical considerations that scale implies: computation and storage. We have found the computational and infrastructure problems of forecasting a large number of time series to be relatively straightforward – typically these fitting procedures parallelize quite easily and forecasts are not difficult to store in relational databases. The actual problems of scale we have observed in practice involve the complexity introduced by the variety of forecasting problems and building trust in a large number of forecasts once they have been produced.

We summarize our approach to business forecasting *at scale* as follows and graphically in Figure 1. We start with modeling the time series using a flexible specification that has a straightforward human interpretation for each of the parameters. We then produce a set of forecasts for this model along with a set of reasonable baselines (including completely automatic techniques, cf. [Hyndman and Khandakar, 2007]) across a variety of simulated forecast dates. We can semi-automate model selection by choosing the best forecast techniques for each problem when the best model is unambiguous. When there is poor performance or other aspects of the forecasts warrent human intervention, we surface these potential problems to a human analyst in a prioritized order. The analyst can then choose whether or not to alter the model in order to incorporate this feedback.

2 Modeling Business Time Series

We use a decomposable time series model [Harvey and Peters, 1990] with three main model components: growth, seasonality, and holidays. They are combined in the following equation:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t. \tag{1}$$

Here $g(t)$ represents our growth function which models non-periodic changes in the value of the time series, $s(t)$ represents periodic changes due to weekly or yearly seasonality, and $h(t)$ represents the effects of holidays which occur on potentially irregular schedules over one more days. The error term ϵ_t represents any idiosyncratic changes which are not accommodated by our model; later we will make the parametric assumption that ϵ_t is normally distributed.

This specification is similar to a generalized additive model (GAM) [Hastie and Tibshirani, 1987], a class of regression models with potentially non-linear smoothers applied to the regressors. Here we use only time as a regressor but possibly several linear and non-linear functions of time as components. Modeling

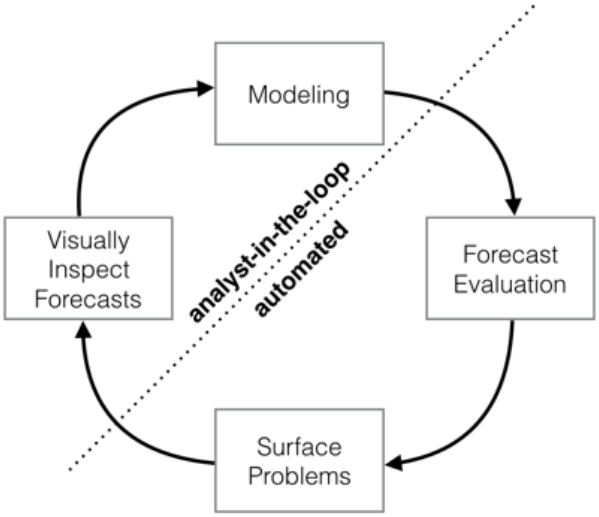

Figure 1: Schematic view of our "forecasting at scale". We propose an analyst-in-the-loop formulation of the problem that best makes use of human and automated tasks.

seasonality as an additive component is the same approach taken by exponential smoothing [Gardner, 1985], and can be adapted to multiplicative seasonality through a log transform.

The GAM formulation has the advantage that it decomposes easily and accommodates new components when necessary, for instance in the case where a new source of seasonality is identified. GAMs also fit very quickly, either using backfitting or L-BFGS (we prefer the latter) so that the user can interactively change the model parameters.

We are, in effect, framing the forecasting problem as a curve-fitting exercise, which is inherently different from time series models that explicitly account for the temporal dependence structure in the data. While we give up some important inferential advantages of using a generative model such as an ARIMA, this formulation provides a number of practical advantages:

1. The formulation is flexible: we can easily accommodate seasonality with multiple periods and different assumptions about trends.

2. Unlike with ARIMA models, the time series measurements need not have a regular period and we do not need to interpolate missing values to fit.

3. Fitting is very fast, allowing the analyst to interactively explore many model specifications, for example in a Shiny application [Chang et al.].

4. The forecasting model has easily interpretable parameters that can be changed heuristically by the analyst to impose assumptions on the forecast.

In addition, when the time series has a regular period, we can accommodate

an ARIMA type error structure for ϵ_t with minimal changes to our modeling procedure.

Automatic forecasting has a long history, with many methods tailored to specific types of time series [Tashman and Leach, 1991, De Gooijer and Hyndman, 2006]. Our approach is driven by both the nature of the time series we forecast at Facebook (piecewise trends, multiple seasonality, floating holidays) as well as the challenges involved in forecasting at scale.

2.1 Growth

2.1.1 Non-linear growth

For growth forecasting, the core component of the data generating process is a model for how the population has grown and how it is expected to continue growing. Modeling growth at Facebook is often similar to population growth in natural ecosystems (e.g. Hutchinson [1978]), where there is nonlinear growth that saturates at a carrying capacity – a natural upper bound on the value of the series. For example, the carrying capacity for the number of Facebook users in a particular area might be the number of people that have access to the Internet. This sort of growth is typically modeled using the logistic growth model, which in its most basic form is

$$g(t) = \frac{C}{1 + \exp(-k(t - b))}, \tag{2}$$

with C the carrying capacity, k the growth rate, and b the offset parameter.

There are two important aspects of growth at Facebook that are not captured in (2). First, the carrying capacity is not constant - as the number of people in the world who have access to the Internet increases, so does the growth ceiling. We thus replace the fixed capacity C with a time-varying capacity $C(t)$. Second, the growth rate is not constant. New products can profoundly alter the rate of growth in a region, so the model must be able to incorporate a varying rate in order to fit historical data.

We incorporate a changing growth rate by explicitly defining changepoints in the model where the growth rate is allowed to change. Suppose there are S changepoints at times s_j, $j = 1, \ldots, S$. We define a vector of rate adjustments $\boldsymbol{\delta} \in \mathbb{R}^S$, where δ_j is the change in rate that occurs at time s_j. The rate at any time t is then the base rate k, plus all of the adjustments up to that point: $k + \sum_{j:t>s_j} \delta_j$. This is represented more cleanly by calculating a vector $\mathbf{a}(t) \in \{0, 1\}^S$ such that

$$a_j(t) = \begin{cases} 1, & \text{if } t \geq s_j, \\ 0, & \text{otherwise.} \end{cases}$$

The rate at time t is then $k + \mathbf{a}(t)^\intercal \boldsymbol{\delta}$. When the rate k is adjusted, the offset parameter b must also be adjusted to connect the endpoints of the segments. The correct adjustment at changepoint j is easily computed as

$$\gamma_j = \left(s_j - b - \sum_{l<j} \gamma_l \right) \left(1 - \frac{k + \sum_{l<j} \delta_l}{k + \sum_{l \leq j} \delta_l} \right).$$

The piecewise logistic growth model is then

$$g(t) = \frac{C(t)}{1 + \exp(-(k + \mathbf{a}(t)^\intercal \boldsymbol{\delta})(t - (b + \mathbf{a}(t)^\intercal \boldsymbol{\gamma})))}. \tag{3}$$

An important set of parameters in our model is $C(t)$, or the expected capacities of the system at any point in time. The simplest possible capacity values are 1) constant capacity ($C(t) = K$) and 2) linear capacity ($C(t) = Mt + K$). We have also found that bringing outside forecasts (e.g. population forecasts from the World Bank) to be fruitful, particularly in cases where the capacity may be likely to diminish over time due to factors not learnable from the historical data.

2.1.2 Linear growth with changepoints

While many applications at Facebook use our non-linear growth specification outlined above, we find that a simpler piece-wise constant rate of growth is often a parsimonious model. In that case we use the following model component:

$$g(t) = (k + \mathbf{a}(t)^\intercal \boldsymbol{\delta})t + (b + \mathbf{a}(t)^\intercal \boldsymbol{\gamma}), \tag{4}$$

where as before k is the growth rate, $\boldsymbol{\delta}$ has the rate adjustments, b is the offset parameter, and γ_j is set to $-s_j \delta_j$ to make the function continuous.

2.1.3 Automatic changepoint selection

The changepoints s_j (used in both our non-linear and linear specifications) could be specified by the analyst using known dates of product launches and other growth-altering events, or they can be automatically selected. Automatic selection can be done quite naturally with the formulation in (3) by putting a sparse prior on $\boldsymbol{\delta}$. We specify a large number of changepoints (*e.g.*, one per month for a several year history) and use the prior $\delta_j \sim \text{Laplace}(0, \tau)$. The single parameter τ directly controls the flexibility of the model in altering its rate. Importantly, a sparse prior on the adjustments $\boldsymbol{\delta}$ has no impact on the primary growth rate k, so as τ goes to 0 the fit reduces to standard (not-piecewise) logistic growth regression.

2.1.4 Trend forecast uncertainty

When the model is extrapolated past the history to make a forecast, the trend will have a constant rate. We estimate the uncertainty in the forecast trend by extending the generative model for the observations forward. The generative model for the trend is that there are S changepoints over a history of T points, at each of which the rate change $\delta_j \sim \text{Laplace}(0, \tau)$. The historical changepoints are at fixed points, but we project this forward by allowing a rate change at any forecast point, with the same average frequency as the rate changes in the history. In a fully Bayesian framework we could put a hierarchical prior on τ so that the rate variance is inferred from data, otherwise we can use the maximum likelihood estimate of the rate scale parameter, which is $\lambda = \frac{1}{S}\sum_{j=1}^{S} |\delta_j|$. The model for rate changes in the extrapolated trend is then

$$\forall j > T, \quad \begin{cases} \delta_j = 0 \text{ w.p. } \frac{T-S}{T}, \\ \delta_j \sim \text{Laplace}(0, \lambda) \text{ w.p. } \frac{S}{T}. \end{cases}$$

We thus measure uncertainty in the forecast trend by assuming that the future will see the same average frequency and magnitude of rate changes that were seen in the history. Once the parameters have been inferred from the data, we use this generative model to simulate possible future trends and use the simulated trends to compute uncertainty intervals.

The assumption that the trend will continue to change with the same frequency and magnitude as it has in the history is fairly strong, so we do not expect the uncertainty intervals to have exact coverage. They are, however, useful indication of the level of uncertainty, and especially as an indicator of overfitting. As τ is increased the model has more flexibility in fitting the history and so training error will drop. However, when projected forward this flexibility will produce wide uncertainty intervals.

2.2 Periodic Seasonality

We have observed that business time series tend to have multi-period seasonality as a result of the human behaviors they usually represent. For instance, weekly periodicity is extremely likely for aggregate measurements of human activity because people to tend to behave differently on different days of the week. Likewise, at different times of the year people can behave quite differently due to school or vacation schedules, weather or temperature variation, and holiday seasons. To model these effects we must specify a seasonality specification, which are periodic functions of t.

We rely on Fourier series to approximate the periodic nature of the data [Harvey and Shephard, 1993]. Let P be the regular period we expect the time series to have (e.g. $P = 365.25$ for yearly data or $P = 7$ for weekly data, when we scale our time variable in days). Let $2N$ be the number of approximation terms we are willing to use. We can then define a seasonality approximation as

$$s(t) = \sum_{n=-N}^{N} c_n e^{i\frac{2\pi nt}{P}},$$

which is a standard Fourier series. To add this to our model we need to estimate the parameters $c_n, n = -N, \ldots, N$.[1] A larger value for N will allow us to fit more complex seasonal functions, with the tradeoff being that we are more likely to overfit. For yearly and weekly seasonality we typically use $N = 10$ and $N = 3$ respectively. We can then construct a matrix of seasonality vectors for each value of t in our historical and future data, for example with yearly seasonality and $N = 10$,

$$X(t) = \left[e^{i\frac{2\pi(-10)t}{365.25}}, \ldots, e^{i\frac{2\pi(10)t}{365.25}}\right]. \tag{5}$$

The seasonal component is then the dot product of $X(t)$ with a parameter vector $\boldsymbol{\beta}$:

$$s(t) = X(t)\boldsymbol{\beta}. \tag{6}$$

In our generative model we take $\boldsymbol{\beta} \sim \text{Normal}(0, \sigma)$ to impose a smoothing prior on the seasonality.

[1] We can see that c_0 is simply an intercept term and not necessary given that we are fitting a growth component.

2.3 Holidays and Events

Holidays and events provide large shocks to many business time series and often do not follow a periodic pattern, so their effects are not well modeled by a smooth cycle. For instance, Thanksgiving in the United States occurs on the fourth Thursday in November. The Super Bowl, one of the largest televised events in the US, occurs on a Sunday in January or February that is difficult to declare programmatically. The impact of a particular holiday on the time series is often similar year after year, so it is important to incorporate it into the forecast.

We allow the analyst to provide a custom list of past and future events, identified by the event or holiday's unique name, as shown in Table 1. We include a column for country in order to keep a country-specific list of holidays in addition to global holidays. For a given forecasting problem we use the union of the global set of holidays and the country-specific ones.

Holiday	Country	Year	Date
Thanksgiving	US	2015	26 Nov 2015
Thanksgiving	US	2016	24 Nov 2016
Thanksgiving	US	2017	23 Nov 2017
Thanksgiving	US	2018	22 Nov 2018
Christmas	*	2015	25 Dec 2015
Christmas	*	2016	25 Dec 2016
Christmas	*	2017	25 Dec 2017
Christmas	*	2018	25 Dec 2018

Table 1: An example list of holidays. Country is specified because holidays may occur on different days in different countries.

Incorporating this list of holidays into the model is made straightforward by assuming that the effects of holidays are independent. For each holiday i, let D_i be the set of past and future dates for that holiday. We add an indicator function representing whether time t is during holiday i, and assign each holiday a parameter κ_i which is the corresponding change in the forecast:

$$h(t) = \sum_{i=1}^{L} \kappa_i \mathbf{1}(t \in D_i). \tag{7}$$

We can generalize Equation 7 to include effects for a window of days around a particular holiday. For instance, people behave differently in the days leading up to Christmas and after it is over. To account for that we include additional parameters for the days surrounding the holiday, essentially treating each of the days in the window around the holiday as a holiday itself.

Holidays are implemented in a similar way as seasonality, by generating a matrix of regressors

$$Z(t) = [\mathbf{1}(t \in D_1), \ldots, \mathbf{1}(t \in D_L)]$$

and taking

$$h(t) = Z(t)\boldsymbol{\kappa}. \tag{8}$$

As with seasonality, we use a prior $\boldsymbol{\kappa} \sim \text{Normal}(0, \nu)$.

2.4 Model fitting

The observed data y is modeled as the sum of each of these components, plus noise:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

When the seasonality and holiday features for each observation are combined into a matrix X and the changepoint indicators $a(t)$ in a matrix A, the entire model can be expressed in a few lines of Stan code [Carpenter et al., 2016], given in Listing 1.

Listing 1: Example Stan code for our complete model.

```
model {
  // Priors
  k ~ normal(0, 5);
  b ~ normal(0, 5);
  epsilon ~ normal(0, 0.5);
  delta ~ double_exponential(0, tau);
  beta ~ normal(0, sigma);

  // Non-linear Likelihood
  y ~ normal(C ./ (1 + exp(-(k + A * delta) .* (t - (b + A * gamma)))) +
             X * beta, epsilon);
  // Linear Likelihood
  y ~ normal((k + A * delta) .* t + (m + A * gamma) + X * beta, sigma);
}
```

For model fitting we use Stan's L-BFGS to find a maximum *a posteriori* estimate. Using Stan for the fitting has the advantage of being capable of full posterior inference to determine forecast uncertainty, if necessary.

Figure 2 provides an example of the model fit to a Facebook time series (the number of Events on Facebook). Figure 3 shows each of the components of the fitted model separately: a piecewise nonlinear growth trend, weekly seasonality, yearly seasonality, and holidays. The model identifies a recent change in the growth trend, strong seasonality from the winter holidays and during the summer, as well as a strong weekly cycle.

The external parameters `tau` and `sigma` are controls for the amount of regularization on the model changepoints and seasonality respectively. Regularization is important for both of these to avoid overfitting, however there likely is not enough historical data to select the best regularization parameters via cross-validation. We can set default values that are appropriate for most forecasting problems, but when these parameters need to be optimized it happens with the analyst-in-the-loop.

2.5 Analyst-in-the-loop modeling

Analysts making forecasts often have extensive domain knowledge about the quantity they are forecasting, but limited statistical knowledge. In our growth model specification there are several places where the analyst can alter the model to apply their expertise and external knowledge without requiring any understanding of the underlying statistics.

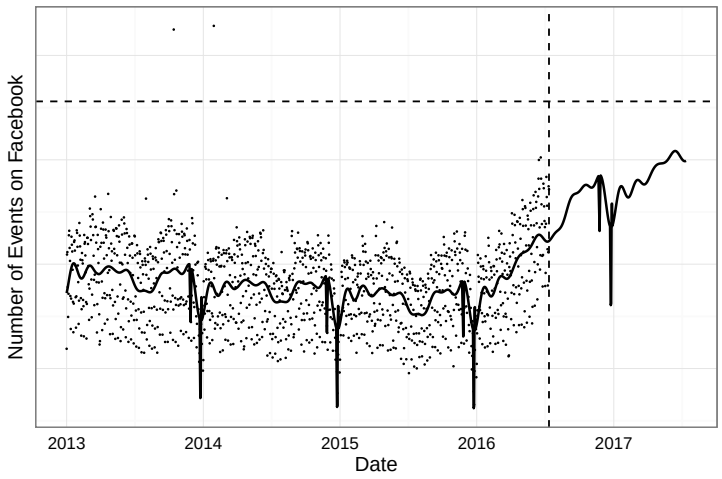

Figure 2: Number of Events on Facebook: an example forecast using nonlinear growth, yearly and weekly seasonality, and holidays. For this figure we removed the weekly component to make it easier to read due to strong weekly periodicity. The horizontal dashed lines indicates our assumed capacity for the nonlinear growth curve. The vertical dashed line indicates the forecast date.

Capacities - Analysts may have external data for the total market size and can apply that knowledge directly by specifying capacities. For instance, a low or declining capacity can be used to adjust the forecast downward or even to produce negative growth.

Changepoints - Known dates of changepoints can be directly specified, for instance when the analyst knows about a product change.

Holidays and seasonality - Analysts that we work with have experience with which holidays impact growth in which regions, and they can directly input the relevant holiday dates and the applicable time scales of seasonality.

Smoothing parameters - By adjusting τ an analyst can select from within a range of more global or locally smooth models. The seasonality and holiday smoothing parameters (σ, ν) allow the analyst to tell the model how much of the historical seasonal variation is expected in the future.

These parameters are all listed in Table 2.

With good visualization tools, the analyst can specify other parts of the model without requiring statistical expertise. When the model fit is plotted over historical data, it is quickly apparent if there were changepoints that were missed by the automatic changepoint selection, or if there are too many changepoints. Missing changepoints can be directly added, and the τ parameter is a single knob that can be turned to increase or decrease the number of changepoints. Similarly, the σ parameter is a single knob to increase or decrease the strength of the seasonality component. Visualization provides many other opportunities for fruitful human intervention: linear trend or non-linear growth, identifying

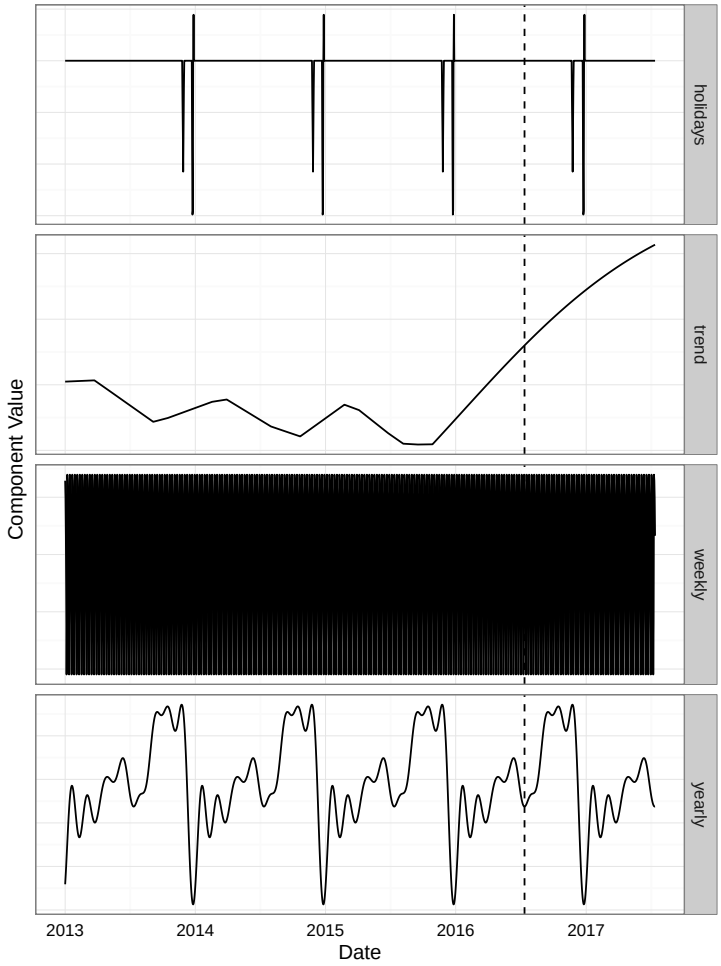

Figure 3: Forecast components for the number of Events on Facebook, corresponding to the forecast in Figure 2. The four main components of the model are depicted here in separate panels.

time scales of seasonality, and identifying outlying time periods that should be removed from fitting are a few. All of these interventions can be made without statistical expertise, and are important ways for the analyst to apply their expertise.

The ability to have an analyst-in-the-loop at scale relies critically on automatic evaluation of forecast quality and good visualization tools. We now describe how forecast evaluation can be automated to surface the most relevant forecasts to the analyst for their input.

3 Automating Evaluation of Forecasts

In Section 2 we proposed a flexible model for the business time series data that we have seen work well in practice. But any new forecast method, particu-

Parameter	Interpretation
$C(t)$	A maximum value for the time series.
s_j	The position of possible time series changepoints.
τ	How likely the model is to update growth rates at changepoints.
σ	How flexibly the model can accommodate seasonality.
ν	How flexibly the model can accommodate holidays.

Table 2: A small set of analyst-interpretable parameters. A main goal of our forecast model is to give the analyst just a few simple ways to change the forecast

Name	Function name
Last value (naive)	`naive`
Mean	`meanf`
ARIMA	`auto.arima`
Exponential smoothing	`ets`
Seasonal Naive	`snaive`
TBATS	`tbats`

Table 3: Baseline Forecasting Procedures, Each of these procedures is available in the `forecast` package in `R` and uses a similar interface.

larly complicated ones, should be carefully evaluated and critiqued before being used in downstream applications such as planning or goal-setting. In this section we outline a procedure for automating forecast performance evaluation, by comparing various methods and identifying forecasts where manual intervention may be warranted. This section is agnostic to the forecasting method used by the analyst and contains some best-practices we have settled on while shipping production business forecasts across a variety of applications.

3.1 Use of Baseline Forecasts

When evaluating any forecasting procedure it is important compare to a set of interpretable baseline methods. We prefer using simplistic forecasts that make strong assumptions about the underlying process but that can to perform well in practice. Table 3 provides a set of baselines that we use when they are applicable. We have found it powerful to compare simplistic models (last value and sample mean) as well as the suite of automated forecasting procedures in the `forecast` package in R, primarily developed by Rob Hyndman and described in detail in Hyndman and Khandakar [2007].

Forecasts are made over a certain *horizon*, which we denote H. The horizon is the number of days in the future we care about estimating - this is typically 30, 90, 180, or 365 days in our applications. Thus for any forecast with daily observations, we produce up to H estimates of future states that will each be associated with some error. We need to declare a forecasting objective to compare methods and track performance. Additionally, understanding how error-prone our forecasting procedure is can allow consumers of the forecasts in a business setting to determine whether to trust it at all.

3.2 Modeling Forecast Errors

Let $\hat{y}(t|T)$ represent a forecast for time t made with historical information up to time T and $d(y, y')$ be a distance metric such as mean absolute error, $d(y, y') = |y - y'|$. Choice of a distance function should be problem-specific and De Gooijer and Hyndman [2006] review several such error metrics – in practice we prefer mean absolute percentage error for its interpretability. We define the empirical error a forecast of $h \in (0, H]$ periods ahead of time T as:

$$\epsilon(T, h) = d(\hat{y}(T + h|T), y(T + h)).$$

In order to form an estimate of this error and how it varies with h, it is common to specify a parametric model for the error term in the model and estimate its parameters from data. For instance if we were using an $AR(1)$ model, $y(t) = \alpha + \beta y(t - 1) + \nu_t$, we would assume that $\nu_t \sim \text{Normal}(0, \sigma_\nu)$ and focus on estimating the variance term $\sigma\nu$ from the data. Then we could form expectations using any distance function through simulation or by using an analytic expression for expectation of sum of the errors. Unfortunately these approaches only give correct estimates of error conditional on having specified the correct model for the process – a condition that is unlikely to hold in practice.

We prefer to take a non-parametric approach to estimating expected errors that is applicable across models which is similar to the idea of applying cross-validation to estimate out-of-sample error for models making predictions on *i.i.d.* data. Given a set of historical forecasts, we can fit a flexible model of the expected error we would make at different forecast horizons h:

$$\xi(h) = \mathbf{E}[\epsilon(T, h)]. \tag{9}$$

This model should be flexible but can impose some simple assumptions. First, the function should be locally smooth in h because we expect any mistakes we make on consecutive days to be relatively similar. Second, we may impose the assumption that the function should be weakly increasing in h. In practice, we use a local regression [Cleveland and Devlin, 1988] or isotonic regression [Dykstra, 1981] as flexible non-parametric models of error curves.

In order to generate historical forecast errors to fit this model, we use a procedure we call *Simulated Historical Forecasts* which we detail in the next subsection.

3.3 Simulated Historical Forecasts

We would like to fit the expected error model in (9) to perform model selection and evaluation. Unfortunately it is difficult to use a method like cross validation because the observations are not exchangeable – we cannot simply randomly partition our data.

We use simulated historical forecasts (SHFs) by producing K forecasts at various cutoff points in the history, chosen such that the horizons lie within the history and the total error can be evaluated.

These SHFs simulate the errors we would have made had we used this forecasting method at those points in the past. This method has the advantage of being simple, easy to explain to analysts and decision makers, and relatively uncontroversial for for generating insight into forecast errors.

Cutoff	Forecast Date	Method	$\hat{y}$	y
2014-01-01	2014-01-01	naive	-1.67	-1.78
2014-01-01	2014-01-02	naive	-1.67	0.28
2014-01-01	2014-01-03	naive	-1.67	-0.13
2014-01-01	2014-01-04	naive	-1.67	-0.98
2014-01-01	2014-01-05	naive	-1.67	-0.56
2014-01-01	2014-01-06	naive	-1.67	0.66

Table 4: Example Simulated Historical Forecast errors. Storing the errors in long format allows us to easily append rows (or partitions in a Hive table) as we generate errors from different methods and for different cutoff dates in parallel.

There are three main issues to be aware of when using the SHF methodology to evaluate and compare forecasting approaches.

First, the more simulated forecasts we make, the more correlated their estimates of error are. In the extreme case of a simulated forecast for each day in the history, the forecasts are unlikely to have changed much given an additional day of information and the errors from one day to the next would be nearly identical. On the other hand, if we make very few simulated forecasts then we have fewer observations of historical forecasting errors on which to base our model selection. As a heuristic, for a forecast horizon H, we generally make a simulated forecast every $H/2$ periods.

Second, forecasting methods can perform better or worse with more data. A longer history can lead to worse forecasts when the model is misspecified and we are overfitting the past, for example using the sample mean to forecast a time series with a trend. We should thus in general expect the distribution of errors to be non-stationary.

Third, if the data generating process changes abruptly, then error estimates based on SHFs are essentially useless. So while we may be able to estimate errors due to systematic variation in the time series process, any structural changes could potentially produce much larger errors than those estimated using SHFs.

Figure 4 illustrates the SHF error estimation for the Facebook time series of Figure 2. For a $H = 180$ day forecast we chose 11 simulated forecast dates ("cutoffs"), once per quarter, beginning one year from the start of the time series to ensure sufficient training data. The error estimation is done for our system from Section 2, which we call Prophet, along with a collection of other automated forecast methods. On this time series, the automatic ARIMA forecasts are fragile, while exponential smoothing and seasonal naive forecasts are fairly robust. Only the Prophet forecasts are able to account for the dip in events around the holidays and adjusting the trajectory upward toward the end of observed data.

Figure 5 shows our estimates of the function $\xi(h)$, the expected mean absolute percentage error across the forecast period using LOESS. We compare our our proposed method "prophet" from Section 2 compared to the baselines we discussed in Section 3.1. This time series is typical of those that we forecast at Facebook, exhibiting strong multiple seasonality, holiday effects, and a piecewise trend. The other automated forecasting methods that we tried do not handle these features well, leading Prophet to have much lower prediction error. The forecasts in Figure 4 were made with default settings, and tweaking the

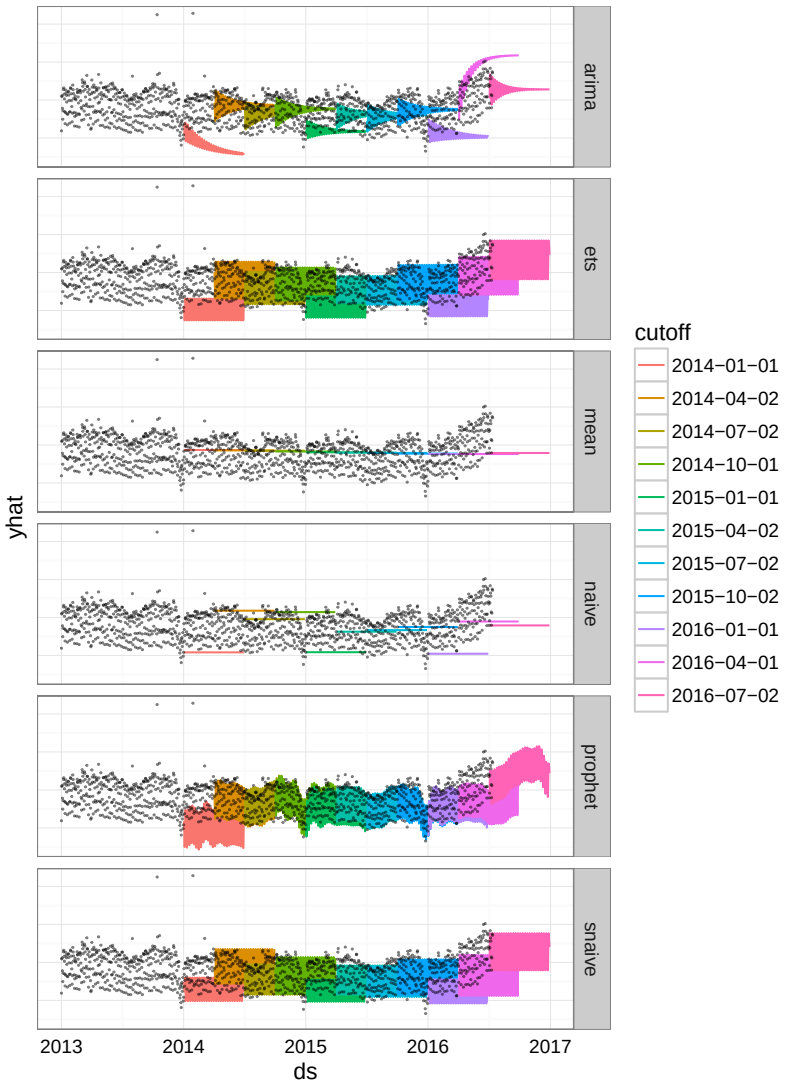

Figure 4: Simulated historical forecasts for the number of events on Facebook, at 11 cutoffs. Prophet, our model from Section 2, is the only method that accounts for the clear dip around the holidays and also the recent change in trend.

parameters could possibly further improve performance.

When visualizing the forecasts, we prefer to use points (rather than lines) to represent historical data, inasmuch as these represent precise measurements that are never interpolated. We then overlay lines with the forecasts. For SHFs, it is useful to visualize the errors the model has made at various horizons, both as a time series (as in Figure 4) and aggregated over SHFs (as in Figure 5).

Even for a single time series SHFs require many forecasts to be computed, and at scale we may want to forecast many different metrics at many different

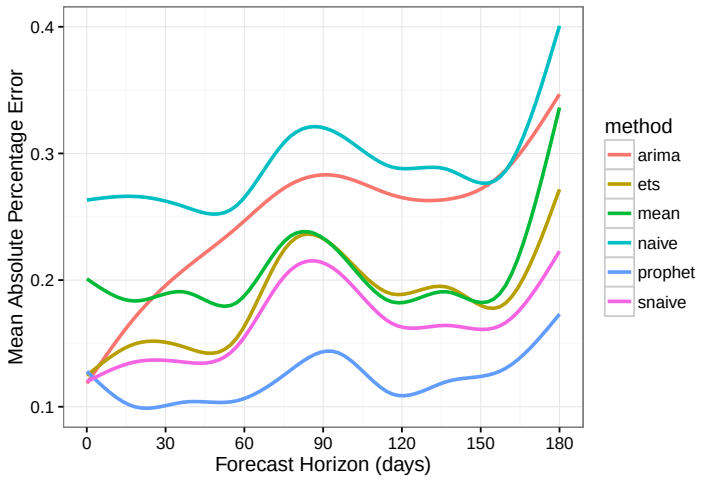

Figure 5: Smoothed mean absolute percentage errors for various forecasting methods, from the SHFs in Figure 4. Prophet forecasts had substantially lower prediction error than the other automated forecast methods.

levels of aggregation. SHFs can be computed independently on separate machines as long as those machines can write to the same data store. We store our forecasts and associated errors in Hive or MySQL depending on their intended use.

3.4 Surfacing large forecast errors

When there are too many forecasts for the analyst to manually check each of them, it is important to be able to automatically identify forecasts that may be problematic. Automatically surfacing bad forecasts allows the analyst to use their limited time most effectively, and to use their expertise to correct any issues. There are several ways that SHFs can be used to identify likely problems with the forecasts.

- When the forecast has large errors relative to the baselines, the model may be misspecified. The analyst can adjust the base growth model (nonlinear, linear) or could adjust the seasonality model, as needed.

- Large errors for all methods on a particular date are suggestive of outliers in the history. The analyst can identify outliers and remove them.

- When the SHF error for a method increases sharply from one cutoff to the next, it could indicate that the data generating process has changed. Adding changepoints or modeling different phases separately may address the issue.

There are pathologies that cannot be easily corrected, but most of the issues that we have encountered can be corrected by specifying changepoints and removing outliers. These issues are easily identified and corrected by an analyst once the forecast has been flagged for review and visualized.

4 Conclusions

A major theme of forecasting at scale is analysts with a variety of backgrounds making more forecasts than they can do manually. The first component of our forecasting system is the new model that we have developed over many iterations of forecasting a variety of data at Facebook. We use a simple, decomposable model, which works reasonable well with default parameters but allows analysts to select the components that are relevant to their forecasting problem and easily make adjustments as needed. The second component is a system for measuring and tracking forecast accuracy, and flagging forecasts that should be checked manually to help analysts make incremental improvements. This is a critical component which allows the analyst to identify when adjustments need to be made to the model or when an entirely different model may be appropriate. Simple, adjustable models and scalable performance monitoring in combination allow a large number of analysts to forecast a large number and a variety of time series – our definition of forecasting *at scale*.

References

B. Carpenter, A. Gelman, M. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. A. Brubaker, J. Guo, P. Li, and A. Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software*, 2016.

W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson. shiny: Web application framework for r, 2015. *URL http://CRAN. R-project. org/package= shiny. R package version 0.11.*

W. S. Cleveland and S. J. Devlin. Locally weighted regression: an approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83(403):596–610, 1988.

J. G. De Gooijer and R. J. Hyndman. 25 years of time series forecasting. *International Journal of Forecasting*, 22(3):443–473, 2006.

R. L. Dykstra. An isotonic regression algorithm. *Journal of Statistical Planning and Inference*, 5(4):355–363, 1981.

E. S. Gardner. Exponential smoothing: the state of the art. *Journal of Forecasting*, 4:1–28, 1985.

A. Harvey and S. Peters. Estimation procedures for structural time series models. *Journal of Forecasting*, 9:89–108, 1990.

A. C. Harvey and N. Shephard. Structural time series models. In G. Maddala, C. Rao, and H. Vinod, editors, *Handbook of Statistics*, volume 11, chapter 10, pages 261–302. Elsevier, 1993.

T. Hastie and R. Tibshirani. Generalized additive models: some applications. *Journal of the American Statistical Association*, 82(398):371–386, 1987.

G. E. Hutchinson. An introduction to population ecology. 1978.

R. J. Hyndman and Y. Khandakar. Automatic time series for forecasting: the forecast package for R. Technical report, Monash University, Department of Econometrics and Business Statistics, 2007.

L. J. Tashman and M. L. Leach. Automatic forecasting software: a survey and evaluation. *International Journal of Forecasting*, 7:209–230, 1991.