Forecasting at Scale

Sean J. Taylor*†
Facebook, Menlo Park, California, United States
sjt@fb.com

and

Benjamin Letham†
Facebook, Menlo Park, California, United States
bletham@fb.com

Abstract

Forecasting is a common data science task that helps organizations with capacity planning, goal setting, and anomaly detection. Despite its importance, there are serious challenges associated with producing reliable and high quality forecasts – especially when there are a variety of time series and analysts with expertise in time series modeling are relatively rare. To address these challenges, we describe a practical approach to forecasting "at scale" that combines configurable models with analyst-in-the-loop performance analysis. We propose a modular regression model with interpretable parameters that can be intuitively adjusted by analysts with domain knowledge about the time series. We describe performance analyses to compare and evaluate forecasting procedures, and automatically flag forecasts for manual review and adjustment. Tools that help analysts to use their expertise most effectively enable reliable, practical forecasting of business time series.

Keywords: Time Series, Statistical Practice, Nonlinear Regression

*To whom correspondence should be addressed.
†The authors contributed equally to this work.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

1 Introduction

Forecasting is a data science task that is central to many activities within an organization. For instance, organizations across all sectors of industry 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. First, completely automatic forecasting techniques can be hard to tune and are often too inflexible to incorporate useful assumptions or heuristics. Second, the analysts responsible for data science tasks throughout an organization typically have deep domain expertise about the specific products or services that they support, but often do not have training in time series forecasting. Analysts who can produce high quality forecasts are thus quite rare because forecasting is a specialized skill requiring substantial experience.

The result is that the demand for high quality forecasts often far outstrips the pace at which they can be produced. 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 notions of scale.

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 3 we present a time series model which is flexible enough for a wide range of business time series, yet 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 4 we describe a forecast evaluation system

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

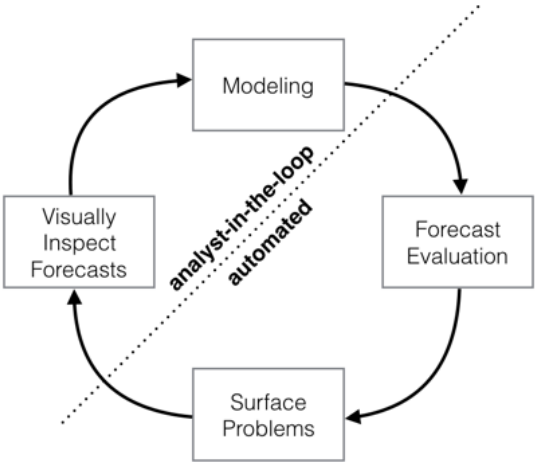

Figure 1: Schematic view of the analyst-in-the-loop approach to forecasting at scale, which best makes use of human and automated tasks.

that uses simulated historical forecasts to estimate out-of-sample performance and identify problematic forecasts for a human analyst to understand what went wrong and make necessary model adjustments.

It is worth noting that we are not focusing on the typical considerations of scale: 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 analyst-in-the-loop approach to business forecasting *at scale* in Fig. 1. We start by modeling the time series using a flexible specification that has a straightforward human interpretation for each of the parameters. We then produce forecasts for this model and a set of reasonable baselines across a variety of historical simulated forecast dates, and evaluate forecast performance. When there is poor performance or other aspects of the forecasts warrant human intervention, we flag these potential problems to a human analyst in a prioritized order. The analyst can then inspect the forecast and potentially adjust the model based on this feedback.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

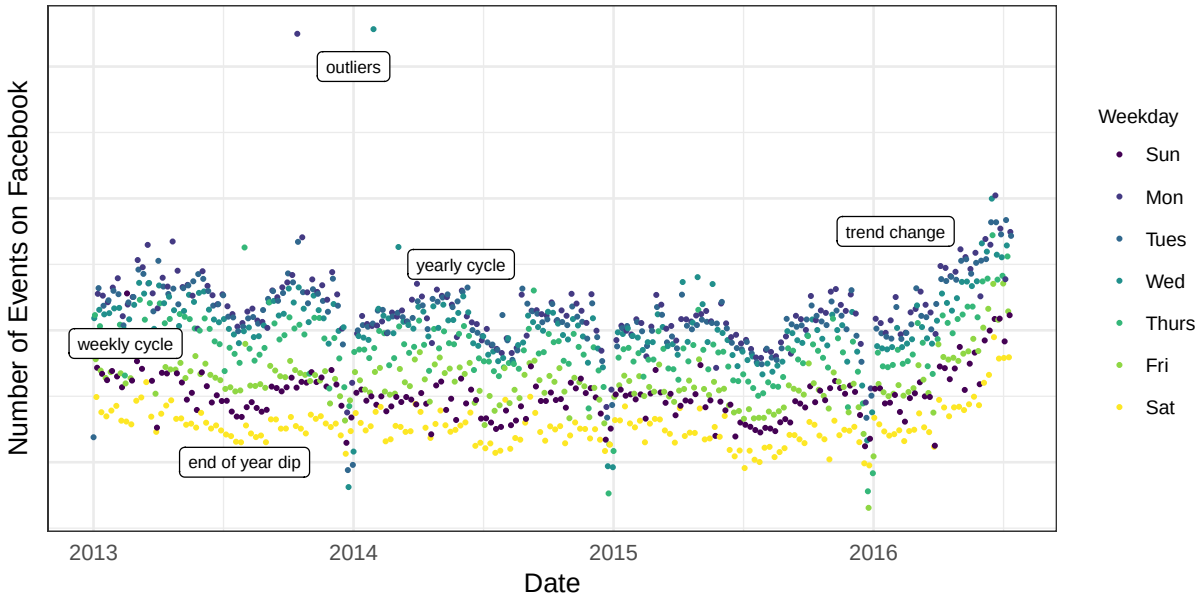

Figure 2: The number of events created on Facebook. There is a point for each day, and points are color-coded by day-of-week to show the weekly cycle. The features of this time series are representative of many business time series: multiple strong seasonalities, trend changes, outliers, and holiday effects.

2 Features of Business Time Series

There is a wide diversity of business forecasting problems, however there are some features common to many of them. Fig. 2 shows a representative Facebook time series for Facebook Events. Facebook users are able to use the Events platform to create pages for events, invite others, and interact with events in a variety of ways. Fig. 2 shows daily data for the number of events created on Facebook. There are several seasonal effects clearly visible in this time series: weekly and yearly cycles, and a pronounced dip around Christmas and New Year. These types of seasonal effects naturally arise and can be expected in time series generated by human actions. The time series also shows a clear change in trend in the last six months, which can arise in time series impacted by new products or market changes. Finally, real datasets often have outliers and this time series is no exception.

This time series provides a useful illustration of the difficulties in producing reasonable forecasts with fully automated methods. Fig. 3 shows forecasts using several automated

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

procedures from the `forecast` package in R, described in Hyndman et al. (2007). Forecasts were made at three points in the history, each using only the portion of the time series up to that point to simulate making a forecast on that date. The methods in the figure are: `auto.arima`, which fits a range of ARIMA models and automatically selects the best one; `ets`, which fits a collection of exponential smoothing models and selects the best (Hyndman et al. 2002); `snaive`, a random walk model that makes constant predictions with weekly seasonality (seasonal naive); and `tbats`, a TBATS model with both weekly and yearly seasonalities (De Livera et al. 2011).

The methods in Fig. 3 generally struggle to produce forecasts that match the characteristics of these time series. The automatic ARIMA forecasts are prone to large trend errors when there is a change in trend near the cutoff period and they fail to capture any seasonality.[1] The exponential smoothing and seasonal naive forecasts capture weekly seasonality but miss longer-term seasonality. All of the methods overreact to the end-of-year dip because they do not adequately model yearly seasonality.

When a forecast is poor, we wish to be able to tune the parameters of the method to the problem at hand. Tuning these methods requires a thorough understanding of how the underlying time series models work. The first input parameters to automated ARIMA, for instance, are the maximum orders of the differencing, the auto-regressive components, and the moving average components. A typical analyst will not know how to adjust these orders to avoid the behavior in Fig. 3 – this is the type of expertise that is hard to scale.

3 The Prophet Forecasting Model

We now describe a time series forecasting model designed to handle the common features of business time series seen in Fig. 2. Importantly, it is also designed to have intuitive parameters that can be adjusted without knowing the details of the underlying model. This is necessary for the analyst to effectively tune the model as described in Fig. 1. Our implementation is available as open source software in Python and R, called Prophet

[1] ARIMA models are capable of including seasonal covariates, but adding these covariates leads to extremely long fitting times and requires modeling expertise that many forecasting novices would not have.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

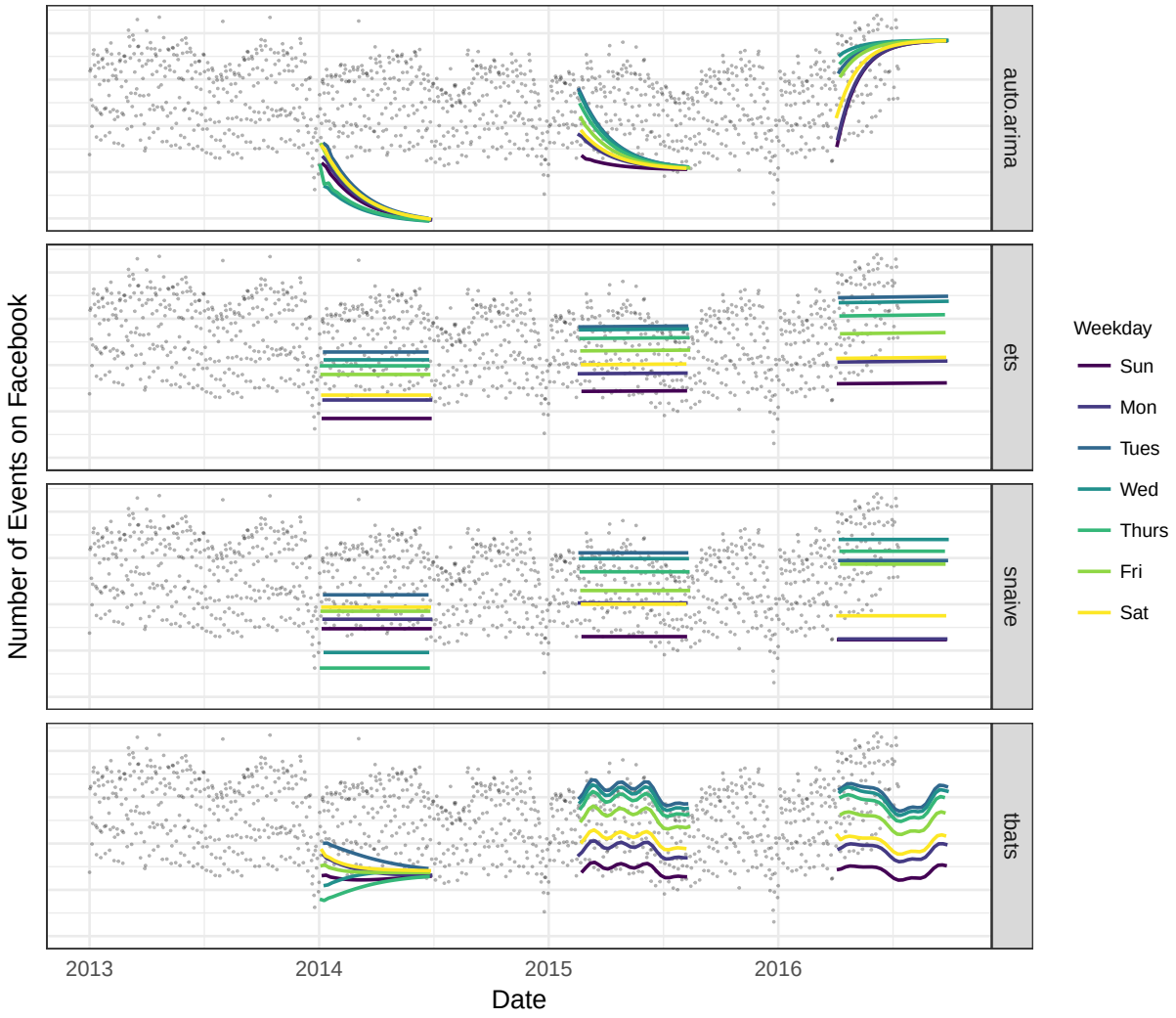

Figure 3: Forecasts on the time series from Fig. 2 using a collection of automated forecasting procedures. Forecasts were made at three illustrative points in the history, each using only the portion of the time series up to that point. Forecasts for each day are grouped and colored by day-of-week to visualize weekly seasonality. We removed the outliers during plotting to allow for more vertical space in the figure.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

(https://facebook.github.io/prophet/).

We use a decomposable time series model (Harvey & Peters 1990) with three main model components: trend, seasonality, and holidays. They are combined in the following equation:

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

Here $g(t)$ is the trend function which models non-periodic changes in the value of the time series, $s(t)$ represents periodic changes (*e.g.*, weekly and yearly seasonality), and $h(t)$ represents the effects of holidays which occur on potentially irregular schedules over one or more days. The error term ϵ_t represents any idiosyncratic changes which are not accommodated by the 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 & 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 seasonality as an additive component is the same approach taken by exponential smoothing (Gardner 1985). Multiplicative seasonality, where the seasonal effect is a factor that multiplies $g(t)$, can be accomplished through a log transform.

The GAM formulation has the advantage that it decomposes easily and accommodates new components as necessary, for instance when a new source of seasonality is identified. GAMs also fit very quickly, either using backfitting or L-BFGS (Byrd et al. 1995) (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:

– Flexibility: We can easily accommodate seasonality with multiple periods and let the analyst make different assumptions about trends.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

– Unlike with ARIMA models, the measurements do not need to be regularly spaced, and we do not need to interpolate missing values *e.g.* from removing outliers.

– Fitting is very fast, allowing the analyst to interactively explore many model specifications, for example in a Shiny application (Chang et al. 2015).

– The forecasting model has easily interpretable parameters that can be changed by the analyst to impose assumptions on the forecast. Moreover, analysts typically do have experience with regression and are easily able to extend the model to include new components.

Automatic forecasting has a long history, with many methods tailored to specific types of time series (Tashman & Leach 1991, De Gooijer & 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.

3.1 The Trend Model

We have implemented two trend models that cover many Facebook applications: a saturating growth model, and a piecewise linear model.

3.1.1 Nonlinear, Saturating 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. 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

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

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

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

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 trend changes in the growth model by explicitly defining changepoints 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 defining 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 m must also be adjusted to connect the endpoints of the segments. The correct adjustment at changepoint j is easily computed as

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

The piecewise logistic growth model is then

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

An important set of parameters in our model is $C(t)$, or the expected capacities of the system at any point in time. Analysts often have insight into market sizes and can set these accordingly. There may also be external data sources that can provide carrying capacities, such as population forecasts from the World Bank.

The logistic growth model presented here is a special case of generalized logistic growth curves, which is only a single type of sigmoid curve. Extensions of this trend model to other families of curves is straightforward.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

3.1.2 Linear Trend with Changepoints

For forecasting problems that do not exhibit saturating growth, a piece-wise constant rate of growth provides a parsimonious and often useful model. Here the trend model is

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

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

3.1.3 Automatic Changepoint Selection

The changepoints s_j could be specified by the analyst using known dates of product launches and other growth-altering events, or may be automatically selected given a set of candidates. Automatic selection can be done quite naturally with the formulation in (3) and (4) by putting a sparse prior on $\boldsymbol{\delta}$.

We often 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 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 or linear growth.

3.1.4 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 forward. The generative model for the trend is that there are S changepoints over a history of T points, each of which has a rate change $\delta_j \sim \text{Laplace}(0, \tau)$. We simulate future rate changes that emulate those of the past by replacing τ with a variance inferred from data. In a fully Bayesian framework this could be done with a hierarchical prior on τ to obtain its posterior, otherwise we can use the maximum likelihood estimate of the rate scale parameter: $\lambda = \frac{1}{S}\sum_{j=1}^{S} |\delta_j|$. Future changepoints are randomly sampled in such a

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

way that the average frequency of changepoints matches that in the history:

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

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 λ has 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, a useful indication of the level of uncertainty, and especially 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.

3.2 Seasonality

Business time series often have multi-period seasonality as a result of the human behaviors they represent. For instance, a 5-day work week can produce effects on a time series that repeat each week, while vacation schedules and school breaks can produce effects that repeat each year. To fit and forecast these effects we must specify seasonality models that are periodic functions of t.

We rely on Fourier series to provide a flexible model of periodic effects (Harvey & 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). We can approximate arbitrary smooth seasonal effects with

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi n t}{P}\right) + b_n \sin\left(\frac{2\pi n t}{P}\right) \right)$$

a standard Fourier series[2]. Fitting seasonality requires estimating the $2N$ parameters $\boldsymbol{\beta} = [a_1, b_1, \ldots, a_N, b_N]^\intercal$. This is done by constructing a matrix of seasonality vectors for each

[2]The intercept term can be left out because we are simultaneously fitting a trend component.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

value of t in our historical and future data, for example with yearly seasonality and $N = 10$,

$$X(t) = \left[\cos\left(\frac{2\pi(1)t}{365.25}\right), \ldots, \sin\left(\frac{2\pi(10)t}{365.25}\right)\right]. \quad (5)$$

The seasonal component is then

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

In our generative model we take $\boldsymbol{\beta} \sim \text{Normal}(0, \sigma^2)$ to impose a smoothing prior on the seasonality.

Truncating the series at N applies a low-pass filter to the seasonality, so increasing N allows for fitting seasonal patterns that change more quickly, albeit with increased risk of overfitting. For yearly and weekly seasonality we have found $N = 10$ and $N = 3$ respectively to work well for most problems. The choice of these parameters could be automated using a model selection procedure such as AIC.

3.3 Holidays and Events

Holidays and events provide large, somewhat predictable 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. Many countries around the world have major holidays that follow the lunar calendar. 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.

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

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

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.

t is during holiday i, and assign each holiday a parameter κ_i which is the corresponding change in the forecast. This is done 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

$$h(t) = Z(t)\boldsymbol{\kappa}. \tag{7}$$

As with seasonality, we use a prior $\boldsymbol{\kappa} \sim \text{Normal}(0, \nu^2)$.

It is often important to include effects for a window of days around a particular holiday, such as the weekend of Thanksgiving. 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.

3.4 Model Fitting

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 in (1) can be expressed in a few lines of Stan code (Carpenter et al. 2017), given in Listing 1. For model fitting we use Stan's L-BFGS to find a maximum *a posteriori* estimate, but also can do

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

Listing 1: Example Stan code for our complete model.

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

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

full posterior inference to include model parameter uncertainty in the forecast uncertainty.

Fig. 4 shows the Prophet model forecast to the Facebook events time series of Fig. 3. These forecasts were made on the same three dates as in Fig. 3, as before using only the data up to that date for the forecast. The Prophet forecast is able to predict both the weekly and yearly seasonalities, and unlike the baselines in Fig. 3, does not overreact to the holiday dip in the first year. In the first forecast, the Prophet model has slightly overfit the yearly seasonality given only one year of data. In the third forecast, the model has not yet learned that the trend has changed. Fig. 5 shows that a forecast incorporating the most recent three months of data exhibits the trend change (dashed line).

An important benefit of the decomposable model is that it allows us to look at each component of the forecast separately. Fig. 6 shows the trend, weekly seasonality, and yearly seasonality components corresponding to the last forecast in Fig. 4. This provides a useful tool for analysts to gain insight into their forecasting problem, besides just producing a prediction.

The parameters `tau` and `sigma` in Listing 1 are controls for the amount of regularization

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

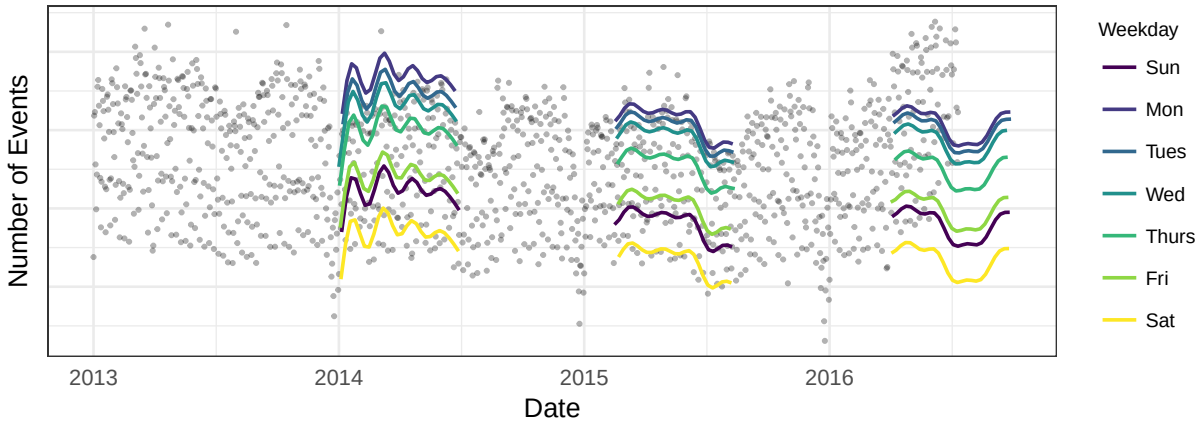

Figure 4: Prophet forecasts corresponding to those of Fig. 3. As before, forecasts are grouped by day-of-week to visualize weekly seasonality.

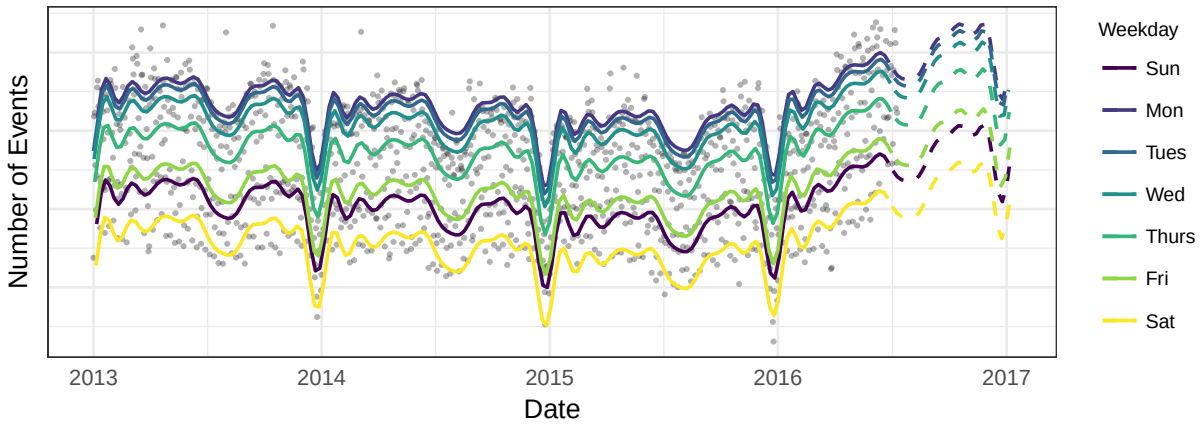

Figure 5: Prophet forecast using all available data, including the interpolation of the historical data. Solid lines are in-sample fit, dashed lines are out-of-sample forecast.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

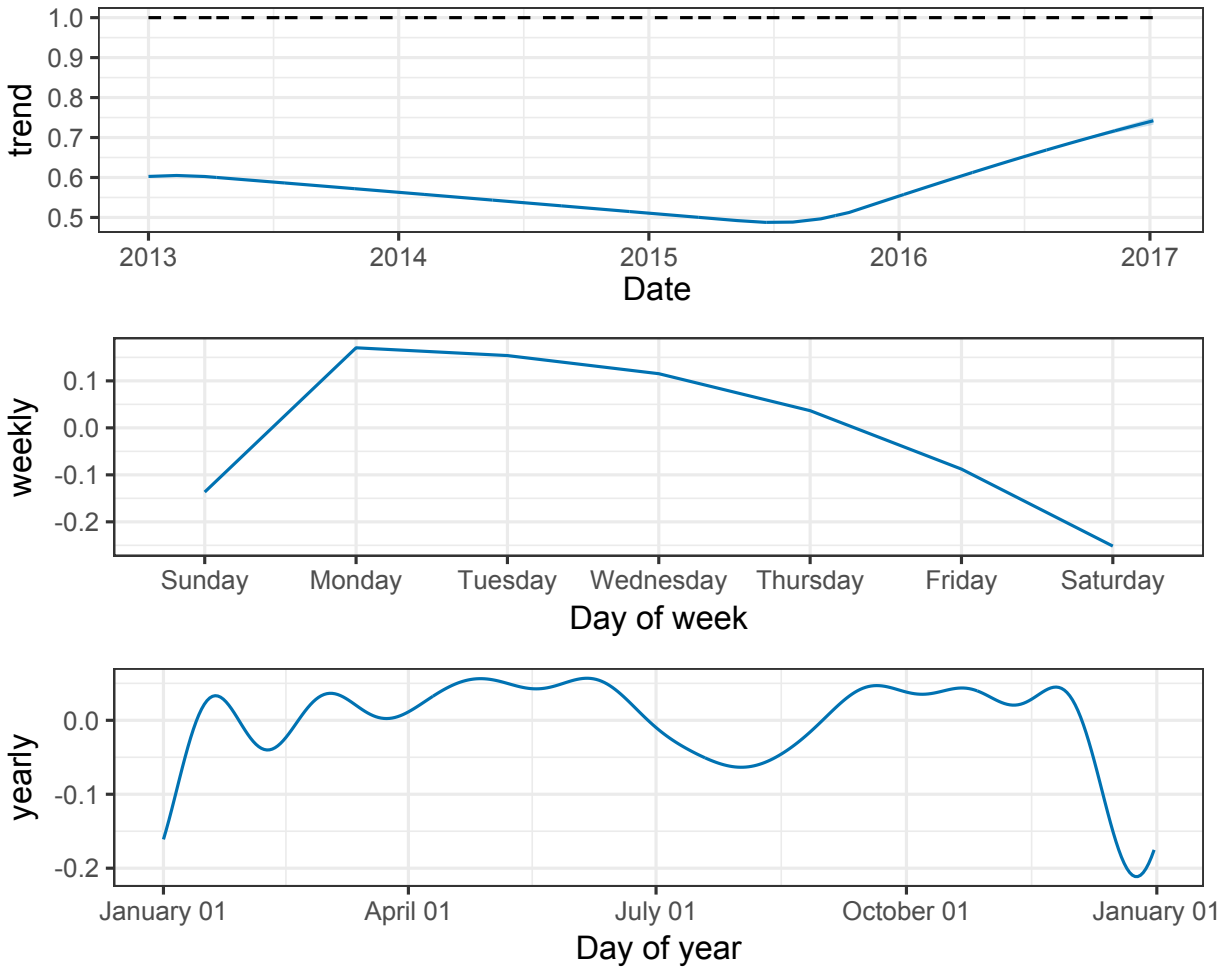

Figure 6: Components of the Prophet forecast in Fig. 5.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

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 set default values that are appropriate for most forecasting problems, and when these parameters need to be optimized it happens with the analyst in the loop.

3.5 Analyst-in-the-Loop Modeling

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

Capacities: Analysts may have external data for the total market size and can apply that knowledge directly by specifying capacities.

Changepoints: Known dates of changepoints, such as dates of product changes, can be directly specified.

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.

With good visualization tools, analysts can use these parameters to improve the model fit. 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. The τ parameter is a single knob that can be turned to increase or decrease the trend flexibility, and σ is a knob to increase or decrease the strength of the seasonality component. Visualization provides many other opportunities for fruitful human intervention: linear trend or logistic

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

growth, identifying 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 analysts to apply their insights or domain knowledge.

The forecasting literature often makes the distinction between statistical forecasts, which are based on models fit to historical data, and judgmental forecasts (also called managerial forecasts), which human experts produce using whatever process they have learned tends to work well for a specific time series. Each of these approaches has their advantages. Statistical forecasts require less domain knowledge and effort from human forecasters, and they can scale to many forecasts very easily. Judgmental forecasts can include more information and be more responsive to changing conditions, but can require intensive work by analysts (Sanders 2005).

Our analyst-in-the-loop modeling approach is an alternative approach that attempts to blend the advantages of statistical and judgmental forecasts by focusing analyst effort on improving the model when necessary rather that directly producing forecasts through some unstated procedure. We find that our approach closely resembles the "transform-visualize-model" loop proposed by Wickham & Grolemund (2016), where the human domain knowledge is codified in an improved model after some iteration.

Typical scaling of forecasting would rely on fully automated procedures, but judgmental forecasts have been shown to be highly accurate in many applications (Lawrence et al. 2006). Our proposed approach lets analysts apply judgment to forecasts through a small set of intuitive model parameters and options, while retaining the ability to fall back on fully automated statistical forecasting when necessary. As of this writing we have only anecdotal empirical evidence for possible improvements to accuracy, but we look forward to future research which can evaluate the improvements analysts can have in a model-assisted setting.

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 identify the most relevant forecasts for analyst input.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

4 Automating Evaluation of Forecasts

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 and contains some best-practices we have settled on while shipping production business forecasts across a variety of applications.

4.1 Use of Baseline Forecasts

When evaluating any forecasting procedure it is important to compare to a set of baseline methods. We prefer using simplistic forecasts that make strong assumptions about the underlying process but that can produce a reasonable forecast in practice. We have found it useful to compare simplistic models (last value and sample mean) as well as the automated forecasting procedures described in Section 2.

4.2 Modeling Forecast Accuracy

Forecasts are made over a certain *horizon*, which we denote H. The horizon is the number of days in the future we care about forecasting – 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.

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. De Gooijer & Hyndman (2006) review several such error metrics – in practice we prefer mean absolute percentage error (MAPE) for its interpretability. We define the empirical accuracy of a forecast of $h \in (0, H]$ periods ahead of time T as:

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

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

In order to form an estimate of this accuracy and how it varies with h, it is common to specify a parametric model for the error term and to 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 $\nu(t) \sim \text{Normal}(0, \sigma_\nu^2)$ and focus on estimating the variance term σ_ν^2 from the data. Then we could form expectations using any distance function through simulation or by using an analytic expression for the expectation of the 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. The approach is similar to 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 fit a model of the expected error we would make at different forecast horizons h:

$$\xi(h) = \mathbf{E}[\phi(T, h)]. \tag{8}$$

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, although this need not be the case for all forecast models. In practice, we use a local regression (Cleveland & 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*.

4.3 Simulated Historical Forecasts

We would like to fit the expected error model in (8) 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 the 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

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

can be evaluated. This procedure is based on classical "rolling origin" forecast evaluation procedures (Tashman 2000), but uses only a small sequence of cutoff dates rather than making one forecast per historical date. The main advantage of using fewer simulated dates (rolling origin evaluation produces one forecast per date) is that it economizes on computation while providing less correlated accuracy measurements.

SHFs simulate the errors we would have made had we used this forecasting method at those points in the past. The forecasts in Figs. 3 and 4 are examples of SHFs. This method has the advantage of being simple, easy to explain to analysts and decision makers, and relatively uncontroversial for generating insight into forecast errors. There are two 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. Although correlated estimates do not introduce bias into our estimation of model accuracy, they do produce less useful information and slow down forecast evaluation.

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.

Fig. 7 shows our estimates of the function $\xi(h)$, the expected mean absolute percentage error across the forecast period using LOESS, for the time series of Figs. 3 and 4. The estimate was made using nine simulated forecast dates, one per quarter beginning after the first year. Prophet has lower prediction error across all forecast horizons. The Prophet forecasts were made with default settings, and tweaking the parameters could possibly further improve performance.

When visualizing forecasts, we prefer to use points rather than lines to represent histor-

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

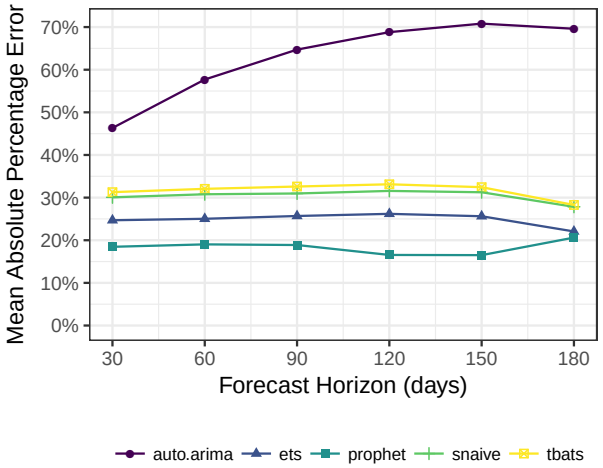

Figure 7: Smoothed mean absolute percentage errors for the forecasting methods and time series of Figs. 3 and 4. Prophet forecasts had substantially lower prediction error than the other automated forecast methods.

ical 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 Fig. 3) and aggregated over SHFs (as in Fig. 7).

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 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.

4.4 Identifying Large Forecast Errors

When there are too many forecasts for analysts to manually check each of them, it is important to be able to automatically identify forecasts that may be problematic. Automatically identifying bad forecasts allows analysts 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 mis-

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

specified. Analysts can adjust the trend model or the seasonality, as needed.

- Large errors for all methods on a particular date are suggestive of outliers. Analysts 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 once the forecast has been flagged for review and visualized.

5 Conclusion

A major theme of forecasting at scale is that analysts with a variety of backgrounds must make 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, modular regression model that often works well with default parameters, and that 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 analysts 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 – what we consider forecasting *at scale*.

6 Acknowledgements

We thank Dan Merl for making the development of Prophet possible and for suggestions and insights throughout the development. We thank Dirk Eddelbuettel, Daniel Kaplan, Rob

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

Hyndman, Alex Gilgur, and Lada Adamic for helpful reviews of this paper. We especially thank Rob Hyndman for insights connecting our work to judgemental forecasts.

References

Byrd, R. H., Lu, P. & Nocedal, J. (1995), 'A limited memory algorithm for bound constrained optimization', *SIAM Journal on Scientific and Statistical Computing* **16**(5), 1190–1208.

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M. A., Guo, J., Li, P. & Riddell, A. (2017), 'Stan: A probabilistic programming language', *Journal of Statistical Software* **76**(1).

Chang, W., Cheng, J., Allaire, J., Xie, Y. & McPherson, J. (2015), *shiny: Web Application Framework for R*. R package version 0.11.
URL: *http://CRAN. R-project. org/package= shiny*

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

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

De Livera, A. M., Hyndman, R. J. & Snyder, R. D. (2011), 'A state space framework for automatic forecasting using exponential smoothing methods', *Journal of the American Statistical Association* **106**(496), 1513–1527.

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

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

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017

Harvey, A. C. & Shephard, N. (1993), Structural time series models, *in* G. Maddala, C. Rao & H. Vinod, eds, 'Handbook of Statistics', Vol. 11, Elsevier, chapter 10, pp. 261–302.

Harvey, A. & Peters, S. (1990), 'Estimation procedures for structural time series models', *Journal of Forecasting* **9**, 89–108.

Hastie, T. & Tibshirani, R. (1987), 'Generalized additive models: some applications', *Journal of the American Statistical Association* **82**(398), 371–386.

Hutchinson, G. E. (1978), 'An introduction to population ecology'.

Hyndman, R. J., Khandakar, Y. et al. (2007), *Automatic time series for forecasting: the forecast package for R*, number 6/07, Monash University, Department of Econometrics and Business Statistics.

Hyndman, R. J., Koehler, A. B., Snyder, R. D. & Grose, S. (2002), 'A state space framework for automatic forecasting using exponential smoothing methods', *International Journal of Forecasting* **18**(3), 439–454.

Lawrence, M., Goodwin, P., O'Connor, M. & Önkal, D. (2006), 'Judgmental forecasting: a review of progress over the last 25 years', *International Journal of Forecasting* **22**(3), 493–518.

Sanders, N. (2005), 'When and how should statistical forecasts be judgementally adjusted?', *Foresight* **1**(1), 5–7.

Tashman, L. J. (2000), 'Out-of-sample tests of forecasting accuracy: an analysis and review', *International journal of forecasting* **16**(4), 437–450.

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

Wickham, H. & Grolemund, G. (2016), 'R for data science'.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3190v2 | CC BY 4.0 Open Access | rec: 27 Sep 2017, publ: 27 Sep 2017