STAT302: Time Series Analysis

Chapter 6. Decomposition and Smoothing of Time Series

Sangbum Choi, Ph.D

Department of Statistics, Korea University

Outline

Introduction

- Time series data can exhibit a variety of patterns, and it is often helpful to split a time series into several components, each representing an underlying pattern category.
- Often this is done to help improve understanding of the time series, but it can also be used to improve forecast accuracy.
- Also, this section covers exponential smoothing, which has motivated some of the most successful forecasting methods.
- Forecasts produced using exponential smoothing methods are weighted averages of past observations, with the weights decaying exponentially as the observations get older.
- This framework generates reliable forecasts quickly and for a wide range of time series.

Time series patterns

Trend	pattern exists when there is a long-term increase or decrease in the data.
Cyclic	pattern exists when data exhibit rises and falls that are *not of fixed period* (duration usually of at least 2 years).
Seasonal	pattern exists when a series is influenced by seasonal factors (e.g., the quarter of the year, the month, or day of the week).

Time series decomposition

We may consider a time series model:

$$Y_t = f(S_t, T_t, R_t)$$

- $Y_t =$ time series data at period t
- $T_t =$ trend-cycle component at period t
- $S_t =$ seasonal component at period t
- $R_t =$ remainder component at period t

Additive decomposition: $Y_t = S_t + T_t + R_t$.
Multiplicative decomposition: $Y_t = S_t \times T_t \times R_t$.

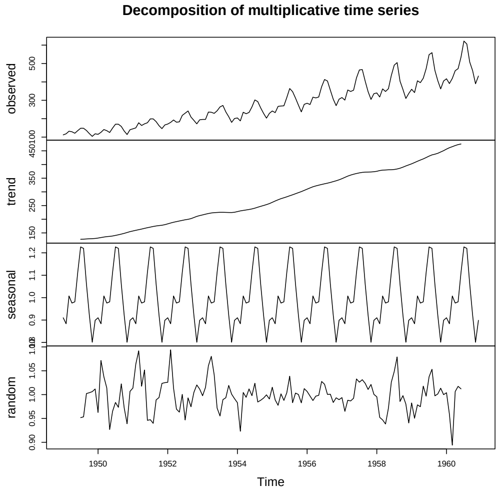
Time series decomposition

- Additive model appropriate if magnitude of seasonal fluctuations does not vary with level.
- If seasonal are proportional to level of series, then multiplicative model appropriate.
- Multiplicative decomposition more prevalent with economic series.
- Alternative: use a Box-Cox transformation, and then use additive decomposition.
- Logs turn multiplicative relationship into an additive relationship:

$$Y_t = S_t \times T_t \times R_t \quad \Rightarrow \quad \log Y_t = \log S_t + \log T_t + \log R_t.$$

Classical time series decomposition

```
plot( decompose(AirPassengers, type="multiplicative") )
```

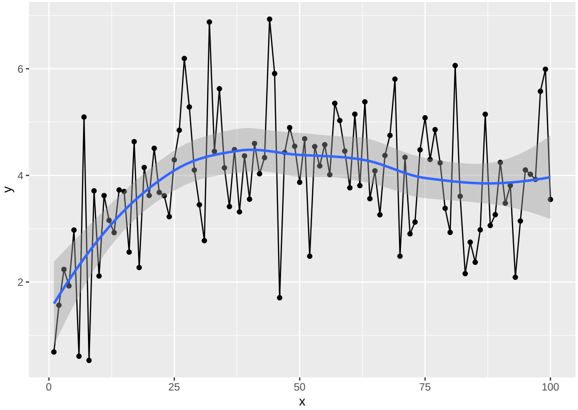


Seasonal and Trend Decomposition Using LOESS (STL)

- STL is a method for decomposing a time series into its trend, seasonality, and residual components.
- It entails applying a smoothing function to the data and then iteratively deleting the fitted curve from the data to extract the trend and seasonality components.
- The residuals, or the difference between the original data and the fitted curve, show noise or irregularity in the data.
- Smoothing functions that can be employed in the STL approach include the LOESS, which is a non-parametric smoothing approach that fits a smooth curve to the data using local regression.

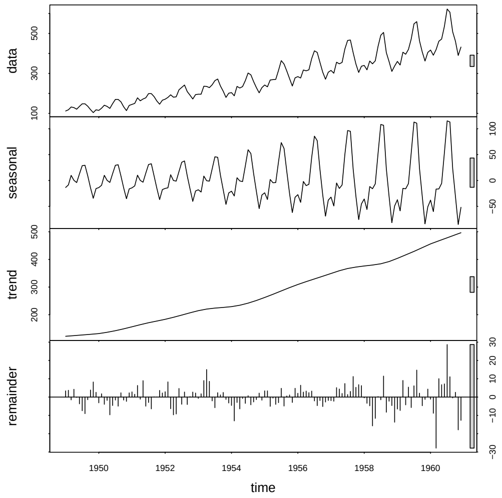
Local polynomial regression fitting with `loess`

```r
library(ggplot2); library(tidyverse)
data = tibble(x = 1:100, y= log(x)+sin(x/20)+rnorm(100))
ggplot(data, aes(x, y)) + geom_point() + geom_line() +
  geom_smooth(method = "loess")
```



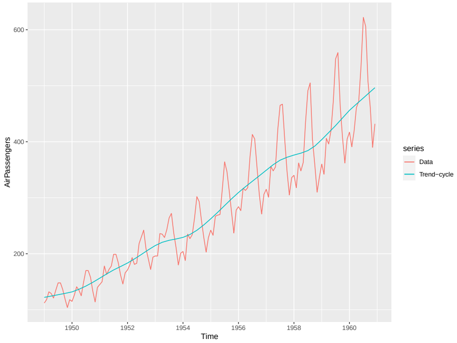
Time series decomposition with stl

```r
plot( fit <- stl(AirPassengers, s.window=6) )
```



Time series decomposition with `stl`

```r
autoplot(AirPassengers, series="Data") +
  autolayer(trendcycle(fit), series="Trend-cycle")
```



More helper functions in `library(forecast)`:

- `seasonal()` extracts the seasonal component
- `trendcycle()` extracts the trend-cycle component
- `remainder()` extracts the remainder component.
- `seasadj()` returns the seasonally adjusted series.

Smoothing time series

- Many TS decomposition methods are based on basic smoothing techniques.
- Smoothing is usually done to help us better see patterns, trends for example, in time series.
- Generally smooth out the irregular roughness to see a clearer signal. For seasonal data, we might smooth out the seasonality so that we can identify the trend.
- Smoothing does not provide us with a model, but it can be a good first step in describing various components of the series.
- The term **filter** is sometimes used to describe a smoothing procedure.

Moving average (MV) filter

- Consider the simple moving average (MV) filter:

$$W_t = \frac{1}{2q+1} \sum_{j=-q}^{q} Y_{t+j}$$

- This is simply the average of $2q+1$ terms of Y_t's, centered at Y_t, which can be implemented by using
  - `rollmean(xt,k=2q+1,align="center")` in `library(xts)`
  - `ma(xt,k)` in `library(forecast)`
- MA smoothing methods can be used to approximate TS components (T_t, S_t, R_t) in a systematic way.

Moving average (MA) filter

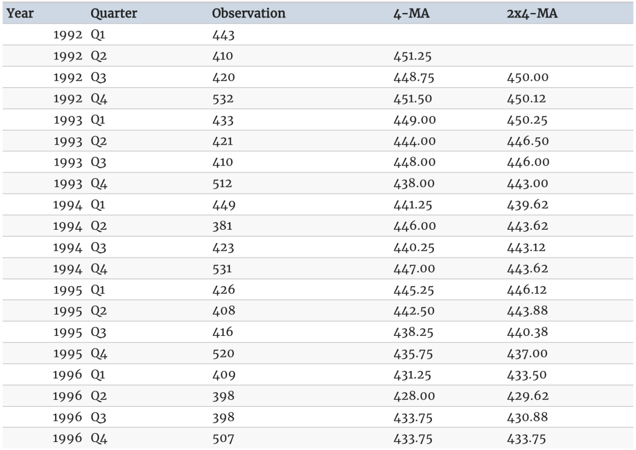
We can write a MA smoothing operation for time trend T_t as

$$\hat{T}_t = \sum_{j=-\infty}^{\infty} a_j Y_{t+j}$$

- $\{a_j\}$ determines filter. It is a weighted average.
- For example, a simple MA filter is given by $a_j = 1/(2q+1)$, $-q \le j \le q$ and 0 elsewhere.
- MA filter is a low-pass filter, since it filters out high frequency variation.
- There are lots of other choices of filter.

Moving average (MA) filter

A moving average of order 4, followed by a moving average of order 2.



Year	Quarter	Observation	4-MA	2x4-MA
1992	Q1	443		
1992	Q2	410	451.25	
1992	Q3	420	448.75	450.00
1992	Q4	532	451.50	450.12
1993	Q1	433	449.00	450.25
1993	Q2	421	444.00	446.50
1993	Q3	410	448.00	446.00
1993	Q4	512	438.00	443.00
1994	Q1	449	441.25	439.62
1994	Q2	381	446.00	443.62
1994	Q3	423	440.25	443.12
1994	Q4	531	447.00	443.62
1995	Q1	426	445.25	446.12
1995	Q2	408	442.50	443.88
1995	Q3	416	438.25	440.38
1995	Q4	520	435.75	437.00
1996	Q1	409	431.25	433.50
1996	Q2	398	428.00	429.62
1996	Q3	398	433.75	430.88
1996	Q4	507	433.75	433.75

Moving average (MA) filter

- When a 2-MA follows a moving average of an even order (such as 4), it is called a **centred moving average of order 4**.
- This is because the results are now symmetric. To see that this is the case, we can write the 2×4-MA as follows:

$$\begin{aligned}\hat{m}_t &= \frac{1}{2}\left[\frac{y_{t-2}+y_{t-1}+y_t+y_{t+1}}{4}+\frac{y_{t-1}+y_t+y_{t+1}+y_{t+2}}{4}\right]\\ &= \frac{1}{8}y_{t-2}+\frac{1}{4}y_{t-1}+\frac{1}{4}y_t+\frac{1}{4}y_{t+1}+\frac{1}{8}y_{t+2}\end{aligned}$$

- It is now a weighted average of observations that is symmetric.
- By default, the `ma()` function in R will return a centred moving average for even orders (unless `center=FALSE` is specified).

Additive decomposition

Step 1: If k is an even number, compute the trend-cycle component $\hat{T}_t$, using a $2 \times k$-MA. Ifk is an odd number, compute $\hat{T}_t$, using a k-MA.

Step 2: Calculate the detrended series: $Y_t - \hat{T}_t$

Step 3: To estimate the seasonal component for each season, simply average the detrended values for that season. For example, with monthly data, the seasonal component for March is the average of all the detrended March values in the data. These seasonal component values are then adjusted to ensure that they add to zero. The seasonal component is obtained by stringing together these monthly values, and then replicating the sequence for each year of data. This gives $\hat{S}_t$.

Step 4: The remainder component is calculated by subtracting the estimated seasonal and trend-cycle components:
$\hat{R}_t = Y_t - \hat{T}_t - \hat{S}_t$.

Multiplicative decomposition

Step 1: If k is an even number, compute the trend-cycle component $\hat{T}_t$, using a $2 \times k$-MA. Ifk is an odd number, compute $\hat{T}_t$, using a k-MA.

Step 2: Calculate the detrended series: $Y_t/\hat{T}_t$

Step 3: To estimate the seasonal component for each season, simply average the detrended values for that season. For example, with monthly data, the seasonal index for March is the average of all the detrended March values in the data. These seasonal indexes are then adjusted to ensure that they add to k. The seasonal component is obtained by stringing together these monthly indexes, and then replicating the sequence for each year of data. This gives $\hat{S}_t$.

Step 4: The remainder component is calculated by dividing out the estimated seasonal and trend-cycle components: $\hat{R}_t = Y_t/(\hat{T}_t\hat{S}_t)$.

Outline

Simple methods

Time series $Y_1, Y_2, \ldots, Y_T$.

- Random walk forecasts

$$\hat{Y}_{T+h|T} = Y_T$$

- Average forecasts

$$\hat{Y}_{T+h|T} = \frac{1}{T}\sum_{t=1}^{T} Y_t$$

- Want something in between that weights most recent data more highly.
- Simple exponential smoothing (SES) uses a weighted moving average with weights that decrease exponentially.

Simple exponential smoothing (SES)

Forecast equation:

$$\hat{Y}_{T+1|T} = \alpha Y_T + \alpha(1-\alpha) Y_{T-1} + \alpha(1-\alpha)^2 Y_{T-2} + \cdots$$

where $0 \le \alpha \le 1$.

	Weights assigned to observations for:			
Observation	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.6$	$\alpha = 0.8$
Y_T	0.2	0.4	0.6	0.8
Y_{T-1}	0.16	0.24	0.24	0.16
Y_{T-2}	0.128	0.144	0.096	0.032
Y_{T-3}	0.1024	0.0864	0.0384	0.0064
Y_{T-4}	$(0.2)(0.8)^4$	$(0.4)(0.6)^4$	$(0.6)(0.4)^4$	$(0.8)(0.2)^4$
Y_{T-5}	$(0.2)(0.8)^5$	$(0.4)(0.6)^5$	$(0.6)(0.4)^5$	$(0.8)(0.2)^5$

Simple exponential smoothing

Component form

$$\begin{aligned} \text{Forecast equation} \qquad & \hat{Y}_{t+h|t} = \ell_t \\ \text{Smoothing equation} \qquad & \ell_t = \alpha Y_t + (1-\alpha)\ell_{t-1} \end{aligned}$$

- ℓ_t is the level (or the smoothed value) of the series at time t.

$$\hat{Y}_{t+1|t} = \alpha Y_t + (1-\alpha)\hat{Y}_{t|t-1}$$

- Iterate to get exponentially weighted moving average form:

$$\hat{Y}_{T+1|T} = \sum_{j=0}^{T-1} \alpha(1-\alpha)^j Y_{T-j} + (1-\alpha)^T \ell_0$$

Optimization

- Need to choose value for α and ℓ_0
- Similarly to regression analysis, we may choose α and ℓ_0 by minimizing SSE:

$$\text{SSE} = \sum_{t=1}^{T} (Y_t - \hat{Y}_{t|t-1})^2.$$

- Unlike regression there is no closed form solution – use numerical optimization.

Holt's linear trend

Component form

$$
\begin{aligned}
\text{Forecast} \quad & \hat{Y}_{t+h|t} = \ell_t + hb_t \\
\text{Level} \quad & \ell_t = \alpha Y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\
\text{Trend} \quad & b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1},
\end{aligned}
$$

- Two smoothing parameters α and β^* $(0 \leq \alpha, \beta^* \leq 1)$.
- ℓ_t level: weighted average between Y_t and one-step ahead forecast for time t, $(\ell_{t-1} + b_{t-1} = \hat{Y}_{t|t-1})$
- b_t slope: weighted average of $(\ell_t - \ell_{t-1})$ and b_{t-1}, current and previous estimate of slope.
- Choose $\alpha, \beta^*, \ell_0, b_0$ to minimise SSE.

Damped trend method

Component form

$$\begin{aligned}\hat{Y}_{t+h|t} &= \ell_t + (\phi + \phi^2 + \cdots + \phi^h) b_t \\ \ell_t &= \alpha Y_t + (1-\alpha)(\ell_{t-1} + \phi b_{t-1}) \\ b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)\phi b_{t-1}.\end{aligned}$$

- Damping parameter $0 < \phi < 1$.
- If $\phi = 1$, identical to Holt's linear trend.
- As $h \to \infty$, $\hat{Y}_{T+h|T} \to \ell_T + \phi b_T/(1-\phi)$.
- Short-run forecasts trended, long-run forecasts constant.

Livestock data

```
livestock2 = window(livestock, start=1970,end=2000)
fit1 = ses(livestock2)
fit2 = holt(livestock2)
fit3 = holt(livestock2, damped = TRUE)
```

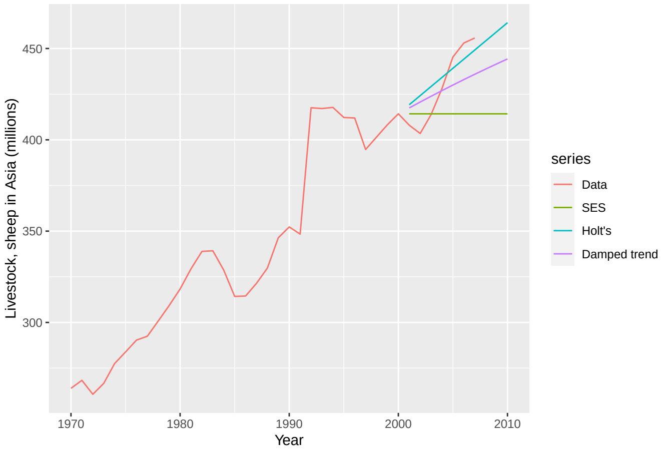
```
accuracy(fit1, livestock)
##                    ME     RMSE      MAE      MPE    MAPE      MASE        ACF1
## Training set  4.85012 14.76861  8.72406 1.375038 2.51993 0.9677984 -0.02497161
## Test set     15.39520 25.46249 20.37887 3.368226 4.59780 2.2607181  0.67966218
##              Theil's U
## Training set        NA
## Test set       2.42332
accuracy(fit2, livestock)
##                      ME     RMSE       MAE         MPE     MAPE      MASE
## Training set  0.2657316 13.98205  7.808464  0.01732775 2.222262 0.8662273
## Test set     -4.5449751 11.88321 10.712284 -1.17754923 2.543186 1.1883609
##                      ACF1 Theil's U
## Training set -0.006192567        NA
## Test set      0.682173277   1.13546
accuracy(fit3, livestock)
##                     ME     RMSE       MAE        MPE     MAPE      MASE
## Training set 0.6250276 13.99691  7.743533 0.08683575 2.200726 0.8590242
## Test set     2.8076440 14.73387 13.302537 0.49686048 3.072831 1.4757091
##                     ACF1 Theil's U
## Training set 0.001678999        NA
## Test set     0.691994623  1.394747
```

Livestock data

	SES	Linear trend	Damped trend
α	1.00	0.98	0.97
β^*		0.00	0.00
ϕ			0.98
ℓ_0	263.90	251.46	251.89
b_0		4.99	6.29
Training RMSE	14.77	13.98	14.00
Test RMSE	25.46	11.88	14.73
Test MAE	20.38	10.71	13.30
Test MAPE	4.60	2.54	3.07
Test MASE	2.26	1.19	1.48

Livestock data

```
data = cbind(Data=window(livestock, start=1970),
  SES=fit1$mean, "Holt's"=fit2$mean, "Damped trend"=fit3$mean)
autoplot(data) + xlab("Year") +
  ylab("Livestock, sheep in Asia (millions)")
```



Additive Holt-Winters for seasonal time series

Holt and Winters extended Holt's method to capture seasonality.

Component form

$$
\begin{aligned}
\hat{Y}_{t+h|t} &= \ell_t + hb_t + s_{t+h-m(k+1)} \\
\ell_t &= \alpha(Y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\
b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \\
s_t &= \gamma(Y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m},
\end{aligned}
$$

- $k =$ integer part of $(h-1)/m$. Ensures estimates from the final year are used for forecasting.
- Parameters: $0 \le \alpha \le 1$, $0 \le \beta^* \le 1$, $0 \le \gamma \le 1-\alpha$ and $m =$ period of seasonality (e.g. $m = 4$ for quarterly data).

Multiplicative Holt-Winters for seasonal time series

Component form

$$\begin{aligned}\hat{Y}_{t+h|t} &= (\ell_t + hb_t)s_{t+h-m(k+1)}.\\ \ell_t &= \alpha\frac{Y_t}{s_{t-m}} + (1-\alpha)(\ell_{t-1} + b_{t-1})\\ b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}\\ s_t &= \gamma\frac{Y_t}{(\ell_{t-1} + b_{t-1})} + (1-\gamma)s_{t-m}\end{aligned}$$

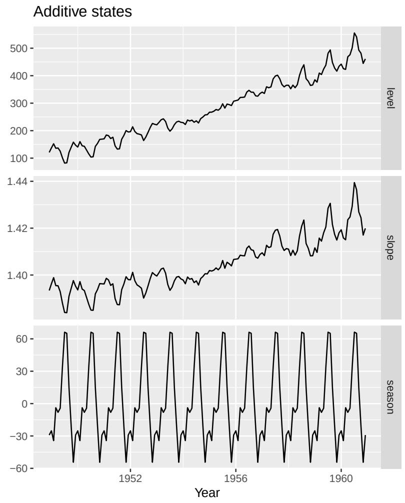
- k is integer part of $(h-1)/m$.
- With additive method s_t is in absolute terms: within each year $\sum_i s_i \approx 0$.
- With multiplicative method s_t is in relative terms: within each year $\sum_i s_i \approx m$.

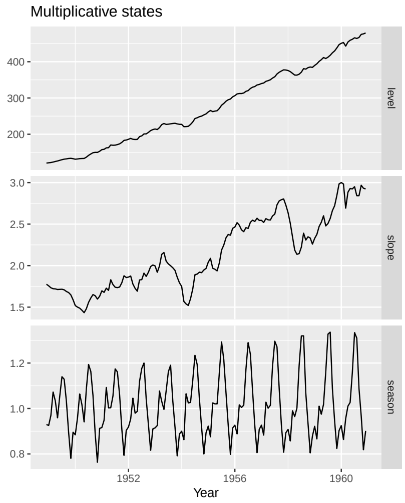
Holt-Winters seasonal method

One can use hw function to implement Holt-Winters in R.

```
fit1 = hw(AirPassengers, seasonal="additive")
fit2 = hw(AirPassengers, seasonal="multiplicative")

addstates = fit1$model$states[,1:3]
multstates = fit2$model$states[,1:3]
colnames(addstates) <- colnames(multstates) <- c("level","slope","season")

p1 = autoplot(addstates, facets=TRUE) + xlab("Year") +
  ylab("") + ggtitle("Additive states")
p2 = autoplot(multstates, facets=TRUE) + xlab("Year") +
  ylab("") + ggtitle("Multiplicative states")

gridExtra::grid.arrange(p1,p2,ncol=2)
```

Holt-Winters seasonal method





Exponential smoothing state space model

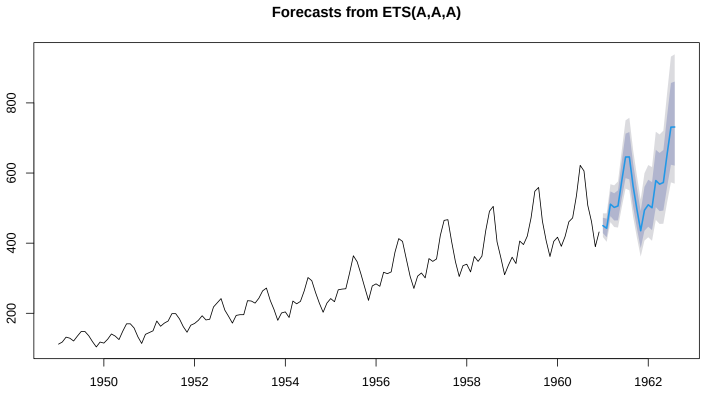
- The ETS models are a family of time series models with an underlying state space model consisting of a level component, (i) trend component (T), (ii) seasonal component (S), and (iii) error term (E).
- Use the `ets` function of `forecast` library in R. When specificying the model type you always specificy the **error, trend, then seasonality** (hence `ETS`).
- The options you can specify for each component is as follows:
  - error: additive (A), multiplicative (M), unknown (Z)
  - trend: none (N), additive (A), multiplicative (M), unknown (Z)
  - seasonality: none (N), additive (A), multiplicative (M), unknown (Z)

Exponential smoothing with ets

```
print(fit <- ets(AirPassengers, lambda=0))
## ETS(A,A,A)
##
## Call:
##  ets(y = AirPassengers, lambda = 0)
##
##   Box-Cox transformation: lambda= 0
##
##   Smoothing parameters:
##     alpha = 0.6975
##     beta  = 0.0031
##     gamma = 1e-04
##
##   Initial states:
##     l = 4.7925
##     b = 0.0111
##     s = -0.1045 -0.2206 -0.0787 0.0562 0.2049 0.2149
##            0.1146 -0.0081 -0.0059 0.0225 -0.1113 -0.0841
##
##   sigma:  0.0383
##
##       AIC      AICc       BIC
## -207.1694 -202.3123 -156.6826
```

Exponential smoothing with ets

```r
forecast(fit, h=20) %>% plot()
```



```r
accuracy(fit)
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.4749774 10.45392 7.539845 -0.2483612 2.779556 0.2353973
##                    ACF1
## Training set 0.06986078
```

Exponential smoothing with ets

- ets() also allows refitting model to new data set.

```
train = window(AirPassengers, end=c(1958,12))
test = window(AirPassengers, start=1959)
fit1 = ets(train)
fit2 = ets(test, model = fit1)
accuracy(fit2)
##                     ME     RMSE      MAE        MPE     MAPE      MA
## Training set -1.135865 10.37451 7.593459 -0.3045808 1.760667 0.15874
##                    ACF1
## Training set -0.3826292
accuracy(forecast(fit1,10), test)
##                     ME      RMSE      MAE       MPE     MAPE      MAS
## Training set  1.208427  8.898232  6.65393 0.3935913 2.777234 0.23286
## Test set     41.380872 50.088276 41.38580 8.7646149 8.766056 1.44836
##              Theil's U
## Training set        NA
## Test set     0.9112882
```

Outline

Time series analysis with facebook `prophet`

- Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects.
- It works best with time series that have strong seasonal effects and several seasons of historical data.
- Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.
- Prophet is open source software in R/Python released by Facebook Core Data Science team; Use `library(prophet)` in R.
- Also, see
  - https://facebook.github.io/prophet/
  - Forecasting at Scale

Motivation

The authors of “Forecasting at Scale” mentioned that there are 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.

Business forecasting at scale

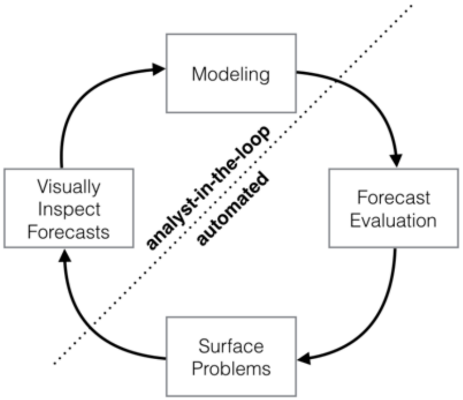


Figure 1: Schematic view of our "forecasting at scale". We propose an analyst-in-the-loop formulation of the problem that best makes use of human and automated tasks.

Decomposition in `prophet`

- `prophet` uses a decomposable time series model with three main model components: growth, seasonality, and holidays.

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

  - $g(t)$ 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,
  - $h(t)$ represents the effects of holidays which occur on potentially irregular schedules over one more days.
  - ϵ_t represents any idiosyncratic changes which are not accommodated by our model.

Model components in `prophet`

- A non-linear growth is typically modeled using the logistic growth model, which in its most basic form is

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

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

- Fourier series can be used to approximate periodic seasonality:

$$s(t) = a_0 + \sum_{k=1}^{K} \left(a_k \cos \frac{2\pi kt}{T} + b_k \sin \frac{2\pi kt}{T} \right)$$

- The holidays component $h(t)$ of the time series is often modeled using a generative additive model (GAM), a statistical model that allows for the insertion of additional elements that may influence the time series.

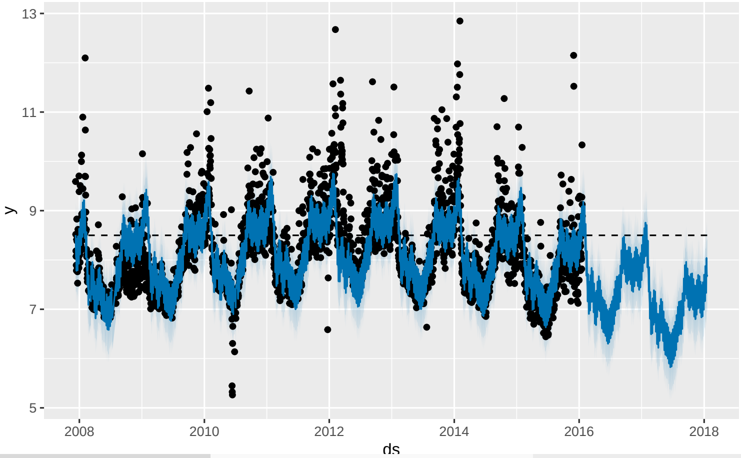
prophet example in R

- First we read in the data and create the outcome variable, which is a dataframe with columns `ds` and `y`, containing the date and numeric value respectively.
- The `ds` column should be `YYYY-MM-DD` for a date, or `YYYY-MM-DD HH:MM:SS` for a timestamp.
- We use the log number of views to Peyton Manning's Wikipedia page, available here.

```r
library(prophet)
library(tidyverse)
df = read_csv("data/fbdata.csv")
df$cap <- 8.5
```

prophet example in R

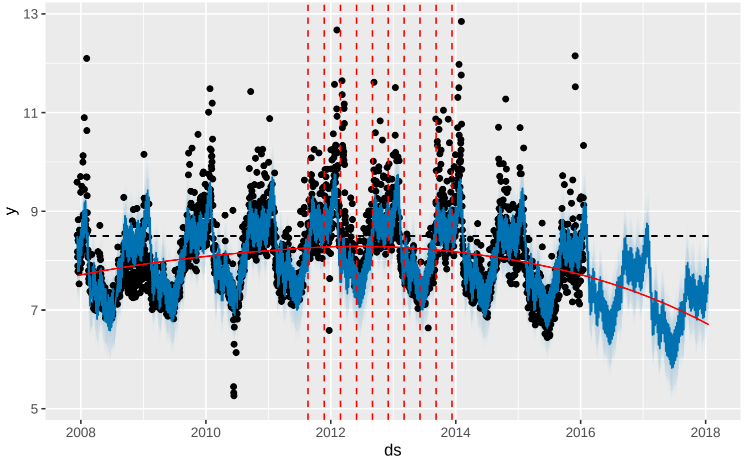
```r
m = prophet(df, growth = 'logistic')
future = make_future_dataframe(m, periods = 365*2)
future$cap = 8.5
fcst = predict(m, future)
plot(m, fcst)
```



Automatic changepoint detection in prophet

Prophet detects changepoints by first specifying a large number of potential changepoints at which the rate is allowed to change.

```r
plot(m, fcst) + add_changepoints_to_plot(m)
```



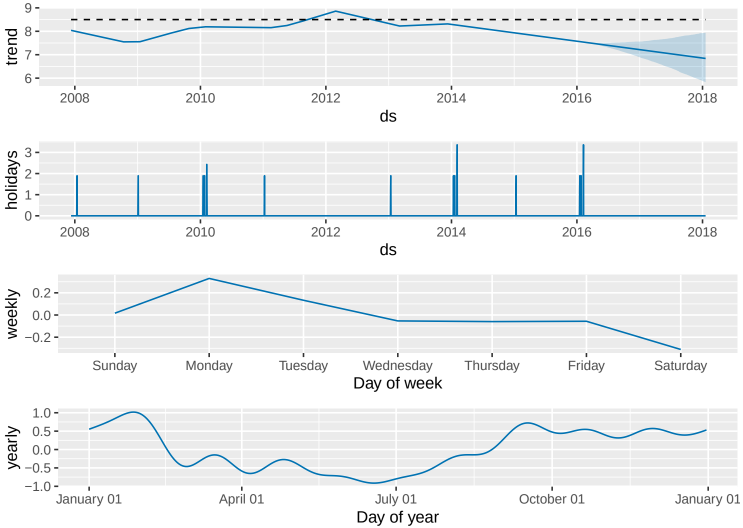
Modeling holidays and special events

Here we create a dataframe that includes the dates of all of Peyton Manning's playoff appearances:

```r
playoffs = data_frame(
  holiday = 'playoff',
  ds = as.Date(c('2008-01-13', '2009-01-03', '2010-01-16',
                 '2010-01-24', '2010-02-07', '2011-01-08',
                 '2013-01-12', '2014-01-12', '2014-01-19',
                 '2014-02-02', '2015-01-11', '2016-01-17',
                 '2016-01-24', '2016-02-07')),
  lower_window = 0,
  upper_window = 1
)
superbowls = data_frame(
  holiday = 'superbowl',
  ds = as.Date(c('2010-02-07', '2014-02-02', '2016-02-07')),
  lower_window = 0,
  upper_window = 1
)
holidays = bind_rows(playoffs, superbowls)

m = prophet(df, holidays = holidays)
fcst = predict(m, future)
```

prophet components

```r
prophet_plot_components(m, fcst)
```


trend
ds
holidays
ds
weekly
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of week
yearly
January 01
April 01
July 01
October 01
January 01
Day of year