



Time Series Analysis and Forecasting

## Chapter 9: Prophet and TBATS

Modern Models for Multiple Seasonalities



# Outline

- 1 Multiple Seasonalities
- 2 TBATS Model
- 3 Facebook Prophet
- 4 Comparison and Guidelines
- 5 Case Study
- 6 Summary

# The Problem: Complex Seasonal Patterns

## Real-World Examples

- **Hourly electricity demand:** Daily + Weekly + Annual patterns
- **Website traffic:** Daily + Weekly + Holiday effects
- **Retail sales:** Weekly + Monthly + Annual + Holiday effects
- **Call center volume:** Hourly + Daily + Weekly patterns

## SARIMA Limitation

Standard  $\text{SARIMA}(p, d, q)(P, D, Q)_s$  handles only **one** seasonal period  $s$ .

For hourly data with daily AND weekly patterns, we need  $s_1 = 24$  and  $s_2 = 168$ .

# Solutions for Multiple Seasonalities

## Traditional Approaches

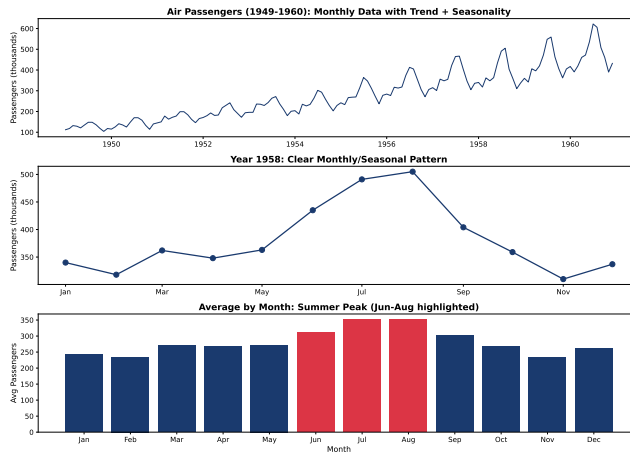
- **Fourier terms:** Add sin/cos regressors
- **Dummy variables:** Many parameters
- **Nested models:** Complex specification

## Modern Approaches

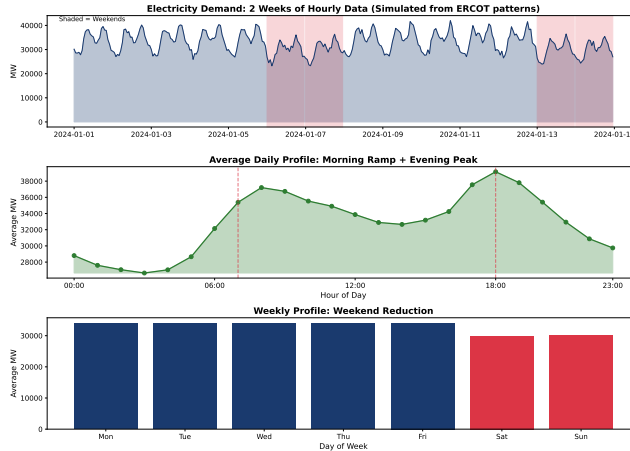
- **TBATS:** Automatic, handles many periods
- **Prophet:** Flexible, interpretable
- **Neural methods:** Deep learning

Method	Max Seasonalities	Interpretable
SARIMA	1	Yes
Fourier + ARIMA	Multiple	Moderate
TBATS	Multiple	Moderate
Prophet	Multiple	Yes

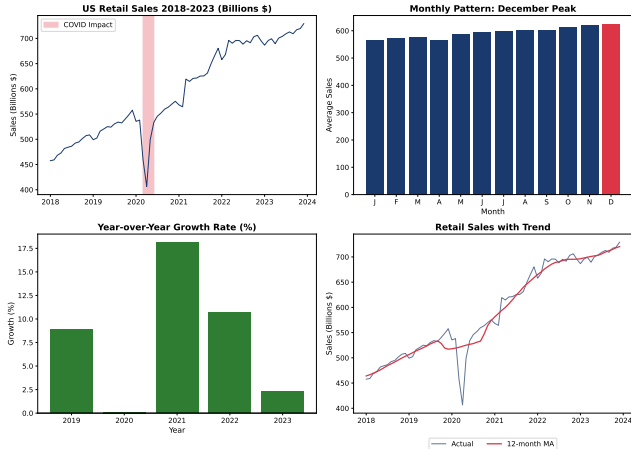
# Example: Hourly Data with Multiple Seasonalities



# Real Example: Electricity Demand



# Real Example: Retail Sales with Holidays



# TBATS: What Does It Stand For?

## TBATS Components

- T** **Trigonometric** seasonality using Fourier terms
- B** **Box-Cox** transformation for variance stabilization
- A** **ARMA** errors for remaining autocorrelation
- T** **Trend** component (possibly damped)
- S** **Seasonal** components (multiple allowed)

## Key Innovation

TBATS uses **trigonometric representation** of seasonality:

$$s_t^{(i)} = \sum_{j=1}^{k_i} \left[ s_j^{(i)} \cos\left(\frac{2\pi jt}{m_i}\right) + s_j^{*(i)} \sin\left(\frac{2\pi jt}{m_i}\right) \right]$$

where  $m_i$  is the  $i$ -th seasonal period and  $k_i$  is the number of harmonics.



## Full Model Specification

$$y_t^{(\omega)} = \ell_{t-1} + \phi b_{t-1} + \sum_{i=1}^T s_{t-m_i}^{(i)} + d_t \quad (1)$$

$$\ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha d_t \quad (2)$$

$$b_t = \phi b_{t-1} + \beta d_t \quad (3)$$

$$d_t = \sum_{i=1}^p \varphi_i d_{t-i} + \sum_{j=1}^q \theta_j \varepsilon_{t-j} + \varepsilon_t \quad (4)$$

Where:

- $y_t^{(\omega)}$  is Box-Cox transformed series (if  $\omega \neq 1$ )
- $\ell_t$  is local level,  $b_t$  is trend with damping  $\phi$
- $s_t^{(i)}$  are  $T$  seasonal components with periods  $m_1, \dots, m_T$
- $d_t$  is  $\text{ARMA}(p, q)$  error process

## Why Fourier/Trigonometric Terms?

- 1 **Parsimonious:** Fewer parameters than dummy variables
- 2 **Smooth:** Captures smooth seasonal patterns naturally
- 3 **Flexible:** Number of harmonics  $k$  controls complexity
- 4 **Non-integer periods:** Can handle  $s = 365.25$  for daily data

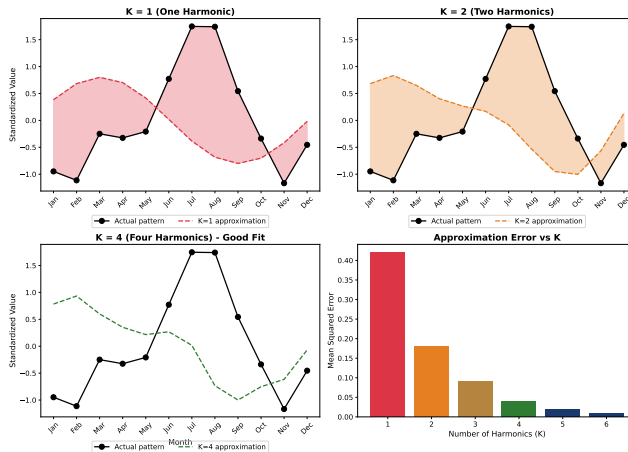
### Low $k$ (few harmonics)

- Smooth pattern
- Fewer parameters
- May miss sharp peaks

### High $k$ (many harmonics)

- Can capture any pattern
- More parameters
- Risk of overfitting

# Fourier Approximation of Seasonality



## Python Implementation

tbats package provides automatic model selection:

- Automatically selects Box-Cox parameter  $\omega$
- Chooses number of harmonics  $k_i$  for each seasonal period
- Selects ARMA orders  $(p, q)$
- Tests damped vs non-damped trend

## Code Example

```
from tbats import TBATS
estimator = TBATS(seasonal_periods=[7, 365.25])
model = estimator.fit(y)
forecast = model.forecast(steps=30)
```

**Note:** BATS is the simpler version without trigonometric terms (uses traditional seasonal states).

# TBATS: Advantages and Limitations

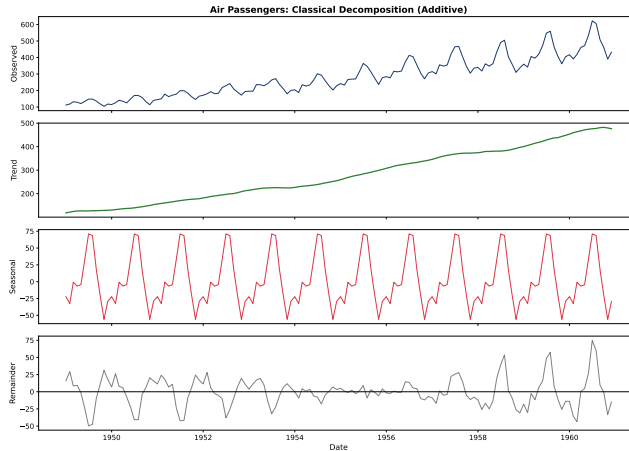
## Advantages

- Handles **multiple** seasonal periods
- **Automatic** model selection
- Handles **non-integer** periods (365.25)
- **Box-Cox** for heteroskedasticity
- Good for **high-frequency** data

## Limitations

- **Computationally intensive**
- No **external regressors**
- Less **interpretable** than Prophet
- Can be **slow** for very long series
- Requires **sufficient data** per season

# TBATS Decomposition Example



## What is Prophet?

Prophet is a forecasting procedure developed by Facebook (Meta) in 2017.

Designed for **business time series** with:

- Strong seasonal effects (daily, weekly, yearly)
- Holiday effects
- Trend changes (change points)
- Missing data and outliers

## Key Philosophy

*“Analyst-in-the-loop” forecasting*

Prophet is designed to be tuned by analysts who have domain knowledge but may not be time series experts.

# Prophet Model Structure

## Decomposition Approach

Prophet uses an **additive decomposition**:

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

### $g(t)$ : Trend

- Linear or logistic
- Automatic changepoints
- Growth saturation

### $s(t)$ : Seasonality

- Fourier series
- Multiple periods
- Custom seasonality

### $h(t)$ : Holidays

- Country holidays
- Custom events
- Window effects



## Linear Trend with Changepoints

$$g(t) = (k + \mathbf{a}(t)^T \boldsymbol{\delta}) \cdot t + (m + \mathbf{a}(t)^T \boldsymbol{\gamma})$$

where:

- $k$  is the base growth rate
- $\boldsymbol{\delta}$  is a vector of rate adjustments at changepoints
- $\mathbf{a}(t)$  indicates which changepoints are active at time  $t$
- $m$  is the offset,  $\boldsymbol{\gamma}$  ensures continuity

## Logistic Growth (for saturating trends)

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

where  $C(t)$  is the (possibly time-varying) carrying capacity.

## Prophet: Seasonality Component

### Fourier Series Representation

For a seasonal period  $P$ , Prophet uses:

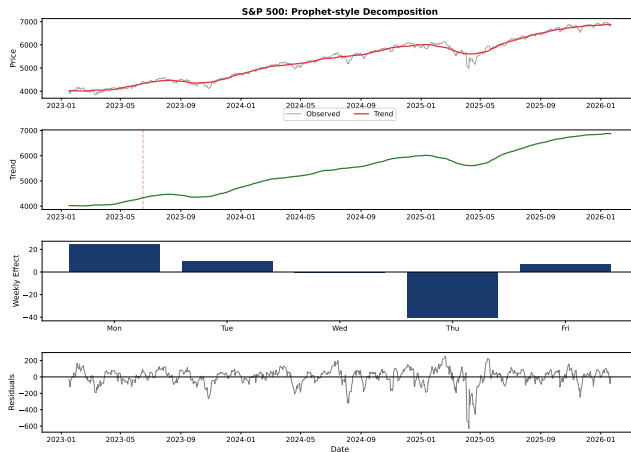
$$s(t) = \sum_{n=1}^N \left[ a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right) \right]$$

### Default Settings

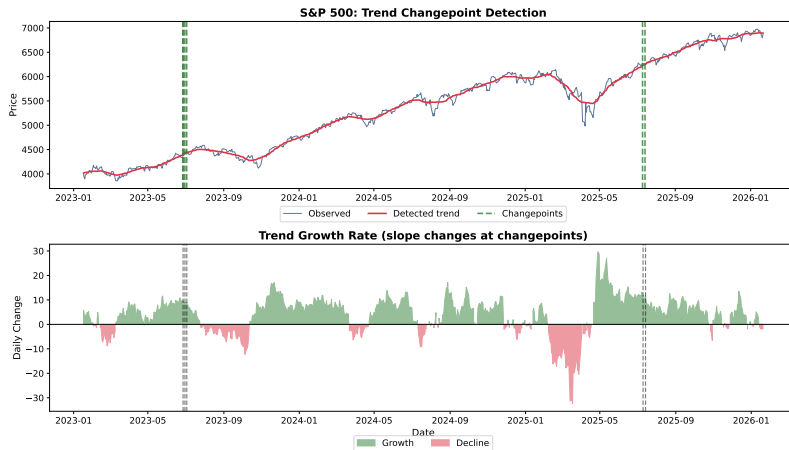
Seasonality	Period	Fourier Order
Yearly	365.25 days	10
Weekly	7 days	3
Daily	1 day	4

Higher Fourier order  $N$  = more flexibility (can fit more complex patterns) but higher risk of overfitting.

# Prophet Component Decomposition



# Trend Changepoint Detection



## Holiday Model

$$h(t) = Z(t) \cdot \kappa$$

where  $Z(t)$  is an indicator matrix for holidays and  $\kappa$  are holiday effects.

## Features

- **Built-in holidays:** 60+ countries supported
- **Custom holidays:** Add your own events (Black Friday, company events)
- **Window effects:** Holidays can affect days before/after
- **Prior scale:** Control regularization of holiday effects

## Code Example

```
holidays = pd.DataFrame({'holiday': 'black_friday', ...})  
model = Prophet(holidays=holidays)
```

## Basic Usage

```
from prophet import Prophet
import pandas as pd

# Data must have columns 'ds' (date) and 'y' (value)
df = pd.DataFrame({'ds': dates, 'y': values})

model = Prophet()
model.fit(df)

future = model.make_future_dataframe(periods=365)
forecast = model.predict(future)
```

## Adding Custom Seasonality

```
model = Prophet(weekly_seasonality=False)
model.add_seasonality(name='monthly', period=30.5, fourier_order=5)
model.add_seasonality(name='quarterly', period=91.25, fourier_order=3)
```

## Three Sources of Uncertainty

- 1 **Trend uncertainty:** Future changepoints are uncertain
- 2 **Seasonality uncertainty:** Parameter estimation uncertainty
- 3 **Observation noise:** Inherent randomness

## Prediction Intervals

Prophet provides:

- Point forecast: `yhat`
- Lower bound: `yhat_lower`
- Upper bound: `yhat_upper`

Default is 80% interval.

Change with `interval_width=0.95`

## Note

Uncertainty grows with forecast horizon, especially for trend uncertainty.

## Key Parameters

Parameter	Effect
<code>changepoint_prior_scale</code>	Trend flexibility (default: 0.05)
<code>seasonality_prior_scale</code>	Seasonality flexibility (default: 10)
<code>holidays_prior_scale</code>	Holiday effect size (default: 10)
<code>seasonality_mode</code>	'additive' or 'multiplicative'
<code>changepoint_range</code>	Portion of history for changepoints

## Practical Tips

- **Overfitting trend?** Decrease `changepoint_prior_scale`
- **Underfitting seasonality?** Increase `seasonality_prior_scale`
- **Seasonal amplitude varies?** Use `seasonality_mode='multiplicative'`



# Prophet: Advantages and Limitations

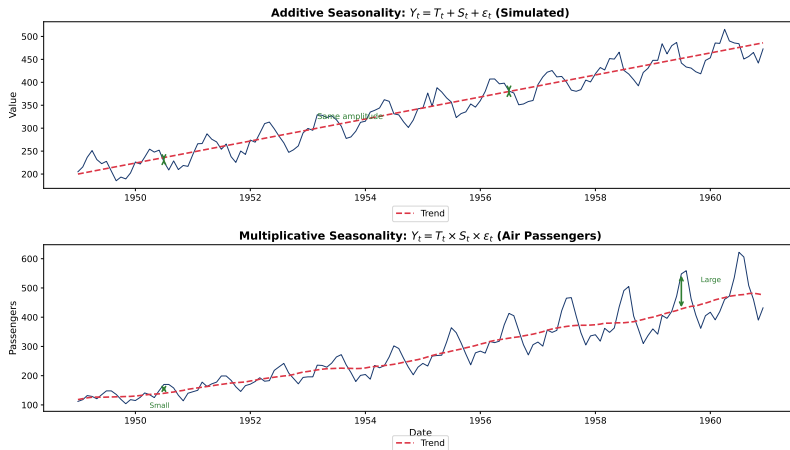
## Advantages

- **Easy to use:** Minimal tuning needed
- **Interpretable:** Clear decomposition
- **Handles missing data** well
- **Holiday effects** built-in
- **Multiple seasonalities**
- **External regressors** supported
- **Fast fitting**

## Limitations

- **Not ARIMA-based:** No autocorrelation modeling
- **Daily data focus:** Less suited for very high frequency
- **Trend assumptions:** Linear/logistic may not fit
- **No built-in CV:** Must implement manually
- **Overfitting risk** with many seasonalities

# Additive vs Multiplicative Seasonality



## TBATS vs Prophet: Head-to-Head

Feature	TBATS	Prophet
Multiple seasonalities	Yes (automatic)	Yes (manual or auto)
Holiday effects	No	Yes (built-in)
External regressors	No	Yes
Trend changepoints	No (smooth)	Yes (automatic)
Missing data	Interpolation needed	Handles natively
Interpretability	Moderate	High
Computation speed	Slow	Fast
High-frequency data	Good	Moderate
Non-integer periods	Yes (e.g., 365.25)	Yes
Uncertainty intervals	Yes	Yes

## When to Use Each Model

### Use TBATS when:

- High-frequency data (hourly, sub-daily)
- Multiple complex seasonal periods
- No external regressors needed
- Automatic model selection preferred
- Traditional state-space framework desired

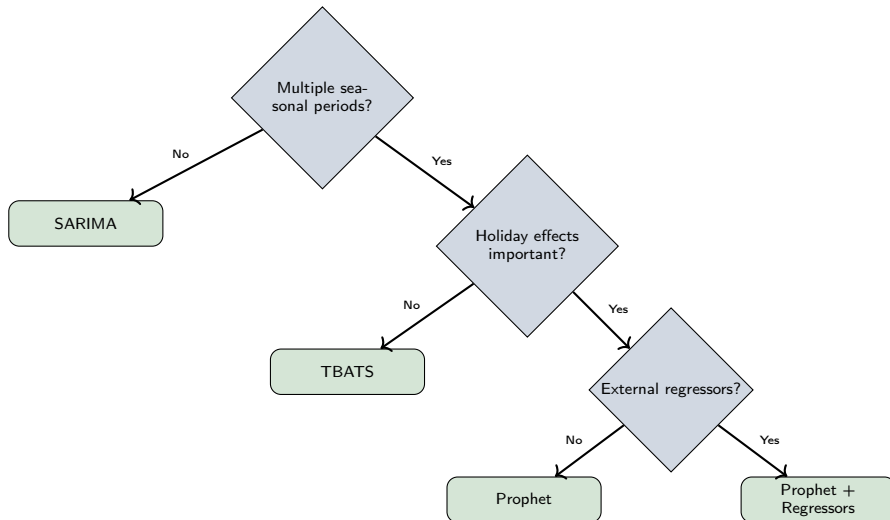
### Use Prophet when:

- Business forecasting (daily/weekly)
- Holiday effects are important
- Trend has structural breaks
- Missing data present
- Interpretability is key
- External regressors available

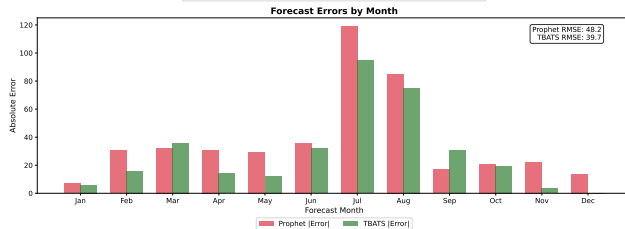
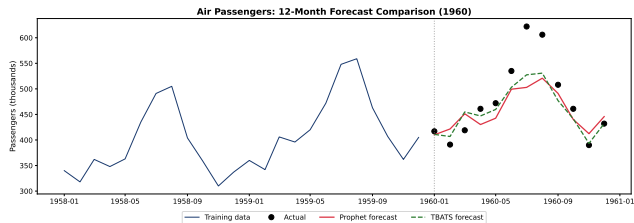
### General Guideline

**Prophet** for business applications with daily data  
**TBATS** for technical applications with high-frequency data

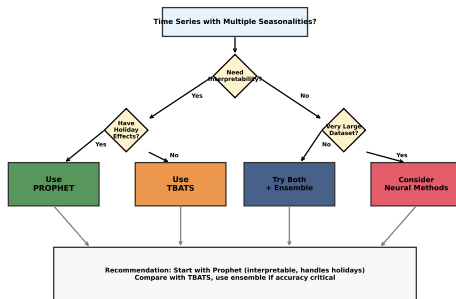
# Decision Flowchart



# Prophet vs TBATS: Forecast Comparison



## Model Selection Guide for Multiple Seasonalities



# Case Study: Energy Demand Forecasting

## Problem

Forecast hourly electricity demand with:

- **Daily pattern:** Peak at noon and evening
- **Weekly pattern:** Lower on weekends
- **Annual pattern:** Higher in summer (AC) and winter (heating)
- **Holiday effects:** Lower demand on holidays

## Approach

- 1 Try TBATS with periods [24, 168, 8766]
- 2 Try Prophet with daily, weekly, yearly seasonality + holidays
- 3 Compare using cross-validation



## Case Study: Results Interpretation

### Evaluation Metrics

- **MAPE:** Mean Absolute Percentage Error
- **RMSE:** Root Mean Square Error
- **Coverage:** % of actuals within prediction interval

### Typical Findings

Model	MAPE	RMSE	Coverage
SARIMA (daily only)	8.5%	450 MW	75%
TBATS	4.2%	220 MW	82%
Prophet	4.8%	250 MW	85%
Prophet + holidays	3.9%	200 MW	88%

Multiple seasonality models significantly outperform single-seasonality SARIMA.

# Key Takeaways

## Multiple Seasonalities

- Real-world data often has multiple seasonal patterns
- Standard SARIMA handles only one seasonal period
- TBATS and Prophet are designed for this challenge

## Model Selection

- **TBATS**: Automatic, handles high-frequency, no external regressors
- **Prophet**: Interpretable, holiday effects, external regressors
- Both use Fourier terms for efficient seasonality representation

## Remember

Always validate with proper time series cross-validation!

Questions?

Questions?

**Next Steps:**

- Practice with the Jupyter notebook
- Try Prophet on your own data
- Explore NeuralProphet for deep learning extension