Non-Daily Data

# Sub-daily data

Prophet can make forecasts for time series with sub-daily observations by passing in a dataframe with timestamps in the ds column. Prophet可以通过在ds列中传入带有时间戳的数据框架，对具有亚日观测的时间序列进行预测。

The format of the timestamps should be YYYY-MM-DD HH:MM:SS - see the example csv [here](https://github.com/facebook/prophet/blob/main/examples/example_yosemite_temps.csv). 时间戳的格式应该是YYY-MM-DD HH:MM:SS--见这里的[csv](https://github.com/facebook/prophet/blob/main/examples/example_yosemite_temps.csv)例子。

When sub-daily data are used, daily seasonality will automatically be fit. 不用担心，当对这些数据进行拟合时，Prophet依旧会对每日季节性进行拟合。相同的，你也可以自己指定季节性让Prophet拟合

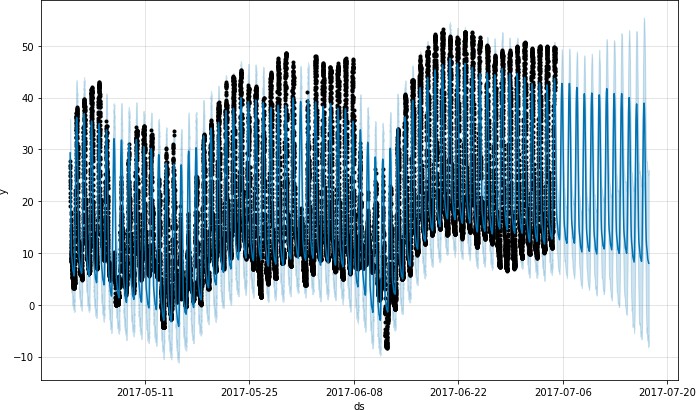
Here we fit Prophet to data with 5-minute resolution (daily temperatures at Yosemite):使用 daily temperatures at Yosemite 数据，其数据时每五分钟一更新。

在实际预测中进场会出现一些存在数据记录的单位时间小于一天的数据，例如，记录某地的温度变化时总是会以小时为单位。

1. # Python
2. df = pd.read\_csv('https://raw.githubusercontent.com/facebook/prophet/main/examples/example\_yosemite\_temps.csv') 3 m = Prophet(changepoint\_prior\_scale=0.01).fit(df)

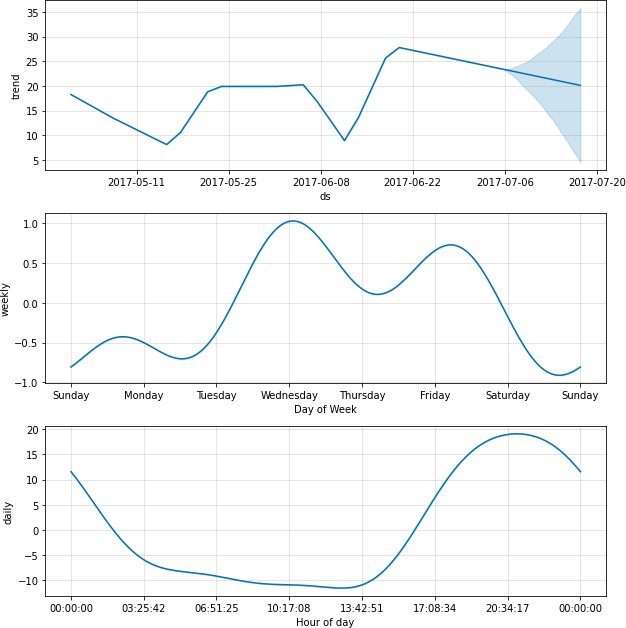
4 future = m.make\_future\_dataframe(periods=300, freq='H') 5 fcst = m.predict(future)

6 fig = m.plot(fcst)



The daily seasonality will show up in the components plot:

1. # Python
2. fig = m.plot\_components(fcst)

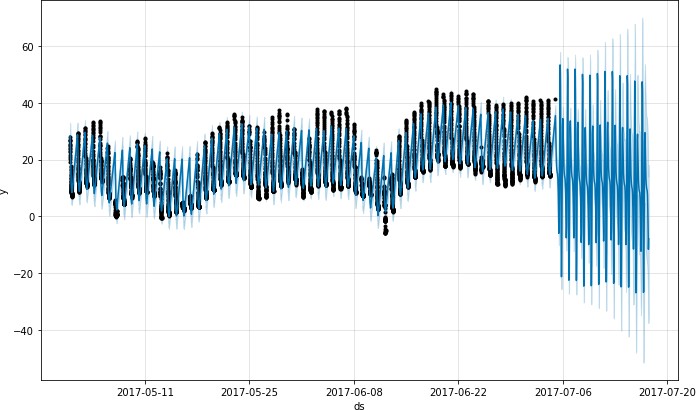


# Data with regular gaps

Suppose the dataset above only had observations from 12a to 6a:假设只对12AM-6AM的数据进行预测

1. # Python
2. df2 = df.copy()
3. df2['ds'] = pd.to\_datetime(df2['ds']) 4 df2 = df2[df2['ds'].dt.hour < 6]
4. m = Prophet().fit(df2)
5. future = m.make\_future\_dataframe(periods=300, freq='H') 7 fcst = m.predict(future)

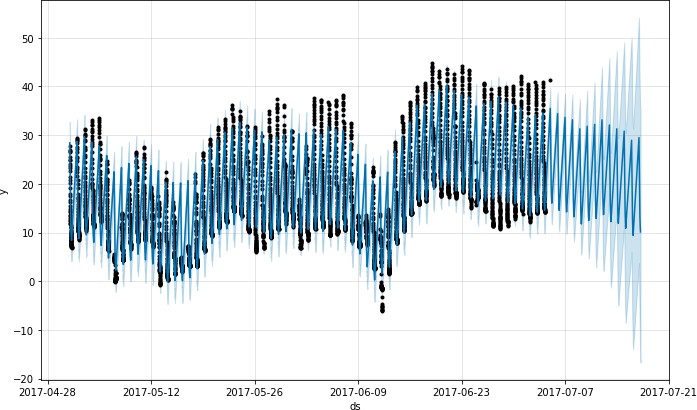
8 fig = m.plot(fcst)



The forecast seems quite poor, with much larger fluctuations in the future than were seen in the history. The issue here is that we have fit a daily cycle to a time series that only has data for part of the day (12a to 6a). The daily seasonality is thus unconstrained for the remainder of the day and is not estimated well. The solution is to only make predictions for the time windows for which there are historical data. Here, that means to limit the **future** dataframe to have times from 12a to 6a: 预测结果似乎相当差，未来的波动比历史上看到的要大得多。这里的问题是，我们对一个只有一天部分时间（12a到6a）数据的时间序列拟合了一个日周期。因此，每天的季节性在一天的剩余时间里是不受约束的，也没有被很好地估计。解决办法是只对有历史数据的时间窗口进行预测。在这里，我们需要先使用 m.make\_future\_dataframe() 方法将 **future** 限制在12AM-6AM

1. # Python
2. future2 = future.copy()
3. future2 = future2[future2['ds'].dt.hour < 6] 4 fcst = m.predict(future2)

5 fig = m.plot(fcst)



The same principle applies to other datasets with regular gaps in the data. For example, if the history contains only weekdays, then predictions should only be made for weekdays since the weekly seasonality will not be well estimated for the weekends.

同样的原则也适用于其他在数据中存在定期空白的数据集。例如，如果历史上只包含工作日，那么就应该只对工作日进行预测，因为每周的季节性将不能很好地估计周末的情况。

# Monthly data / 月度数据

You can use Prophet to fit monthly data. 你可以使用Prophet来拟合月度数据

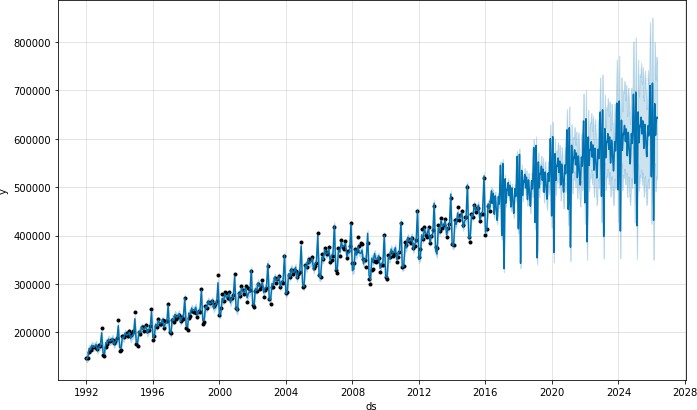
However, the underlying model is continuous-time, which means that you can get strange results if you fit the model to monthly data and then ask for daily forecasts. 然而，底层模型是连续时间的，这意味着如果你对月度数据进行拟合，然后要求进行每日预测，你会得到奇怪的结果。

Here we forecast US retail sales volume for the next 10 years: 这里我们预测美国未来10年的零售量。

1. # Python
2. df = pd.read\_csv('https://raw.githubusercontent.com/facebook/prophet/main/examples/example\_retail\_sales.csv') 3 m = Prophet(seasonality\_mode='multiplicative').fit(df)

4 future = m.make\_future\_dataframe(periods=3652) 5 fcst = m.predict(future)

6 fig = m.plot(fcst)

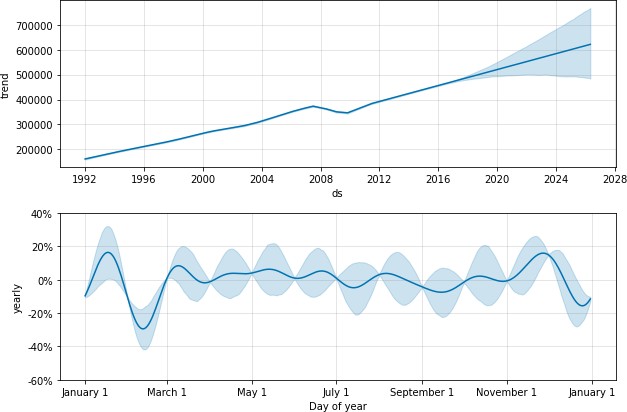


This is the same issue from above where the dataset has regular gaps. When we fit the yearly seasonality, it only has data for the first of each month and the seasonality components for the remaining days are unidentifiable and overfit. This can be clearly seen by doing MCMC to see uncertainty in the seasonality: 这和上面的问题是一样的，数据集有规律的空隙。当我们拟合年季节性时，它只有每个月的第一个月的数据，其余日子的季节性成分是无法识别的，而且是过度拟合。这可以通过做MCMC来清楚地看到季节性的不确定性。

1. # Python
2. m = Prophet(seasonality\_mode='multiplicative', mcmc\_samples=300).fit(df, show\_progress=False) 3 fcst = m.predict(future)

4 fig = m.plot\_components(fcst)

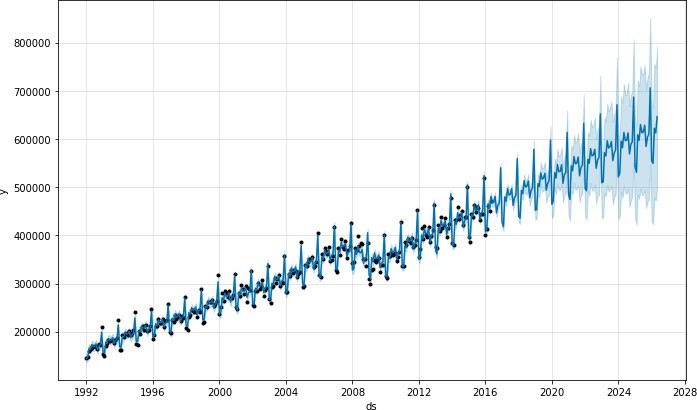
1 WARNING:pystan:481 of 600 iterations saturated the maximum tree depth of 10 (80.2 %) 2 WARNING:pystan:Run again with max\_treedepth larger than 10 to avoid saturation



The seasonality has low uncertainty at the start of each month where there are data points, but has very high posterior variance in between. When fitting Prophet to monthly data, only make monthly forecasts, which can be done by passing the frequency into **make\_future\_dataframe**: 在有数据点的每个月的开始，季节性的不确定性很低，但在两者之间有非常高的后验方差。当只做月度预测，对月度数据拟合 Prophet 时，这可以通过将频率传入 make\_future\_dataframe 来完成

1. # Python
2. future = m.make\_future\_dataframe(periods=120, freq='MS') 3 fcst = m.predict(future)

4 fig = m.plot(fcst)



In Python, the frequency can be anything from the pandas list of frequency strings here:在Pandas中，频率可以是任何在pandas列表中的频率String，以下是他的网站：https://pandas.pydata.org/pandas- docs/stable/user\_guide/timeseries.html#timeseries-offset-aliases .

Note that **MS** used here is month-start, meaning the data point is placed on the start of each month.注意，MS参数指的是month-start，意味着数据点放在每个月的开始

In monthly data, yearly seasonality can also be modeled with binary extra regressors. In particular, the model can use 12 extra regressors like **is\_jan, is\_feb**, etc. where **is\_jan** is 1 if the date is in Jan and 0 otherwise. This approach would avoid the within-month unidentifiability seen above. Be sure to use **yearly\_seasonality=False** if monthly extra regressors are being added. **在月度数据中，年度的季节性**也可以用二进制的额外回归因子来建模。特别是，模型可以使用12个额外的回归因子，如is\_jan、is\_feb等。其中当is\_jan是1意味着如果日期是在1月，否则是0。***这种方法可以避免上面看到的月内无法识别的问题。如果每月添加额外的回归因子，请确保使用yearly\_seasonality=False。***

# Holidays with aggregated data / 有汇总数据的节假日

Holiday effects are applied to the particular date on which the holiday was specified. With data that has been aggregated to weekly or monthly frequency, holidays that don’t fall on the particular date used in the data will be ignored: for example, a Monday holiday in a weekly time series where each data point is on a Sunday. To include holiday effects in the model, the holiday will need to be moved to the date in the history dataframe for which the effect is desired. Note that with weekly or monthly aggregated data, many holiday effects will be well-captured by the yearly seasonality, so added holidays may only be necessary for holidays that occur in different weeks throughout the time series.

[Edit on GitHub](https://github.com/facebook/prophet/blob/main/docs/_docs/non-daily_data.md)

# Facebook Open Source

[Open Source Project](https://code.facebook.com/projects/)s [GitHub](https://github.com/facebook/) [Twitte](https://twitter.com/fbOpenSource)r [Privacy](https://opensource.facebook.com/legal/privacy/) [Terms](https://opensource.facebook.com/legal/terms/) [Contribute to this project on GitHub](https://github.com/facebook/prophet)