Trend Changepoints

You may have noticed in the earlier examples in this documentation that real time series frequently have abrupt changes in their trajectories.在本文档前面的例子中，实时时间序列的轨迹经常会突然变化。

By default, Prophet will automatically detect these changepoints and will allow the trend to adapt appropriately. 默认情况下，Prophet将自动检测这些变化点，并允许趋势适当地进行调整。

However, if you wish to have finer control over this process (e.g., Prophet missed a rate change, or is overfitting rate changes in the history), then there are several input arguments you can use.但是，如果你希望对这个过程有更精细的控制(例如，Prophet错过了一个斜率变化，或者过拟合了历史上的斜率变化)，那么可以使用几个输入参数。

# 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. 在Prophet中，其会首先检测数量很大一些***潜在的变化点***，这些***潜在变化点***的变化斜率是被允许改变的。

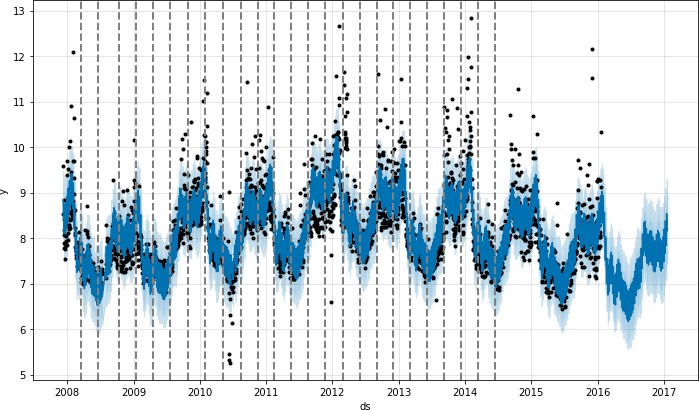
It then puts a sparse prior on the magnitudes of the rate changes (equivalent to L1 regularization) - this essentially means that Prophet has a large number of *possible* places where the rate can change, but will use as few of them as possible. 然后，它对斜率变化的幅度进行稀疏先验(相当于L1正则化)——这本质上意味着Prophet有大量可能的斜率变化的地方，但将使用尽可能少的地方。

Consider the Peyton Manning forecast from the Quickstart. 在第一项 Quick\_Start实验中，给出的Peyton Manning 实验就是存在许多的可变点，但是在拟合过程中只使用了其中很少的一部分。

By default, Prophet specifies 25 potential changepoints which are uniformly placed in the first 80% of the time series. 默认情况下，Prophet指定25个潜在的变化点，这些变化点均匀地放置在时间序列的前80%

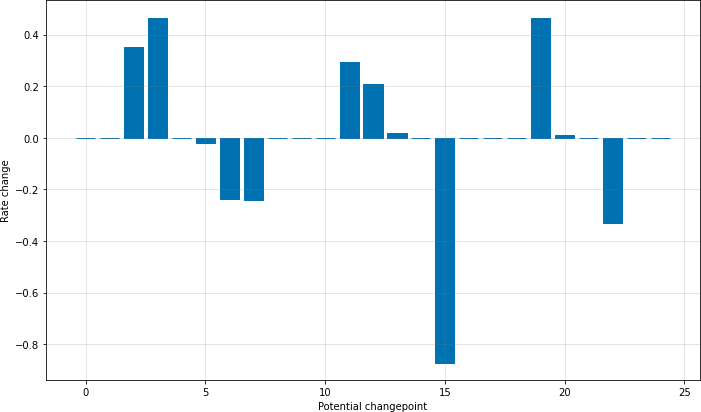
The vertical lines in this figure indicate where the potential changepoints were placed:

如下图。



Even though we have a lot of places where the rate can possibly change, because of the sparse prior, most of these changepoints go unused. 因为先验，很多的斜率变换点未被使用

We can see this by plotting the magnitude of the rate change at each changepoint:下图显示了斜率变化幅度。



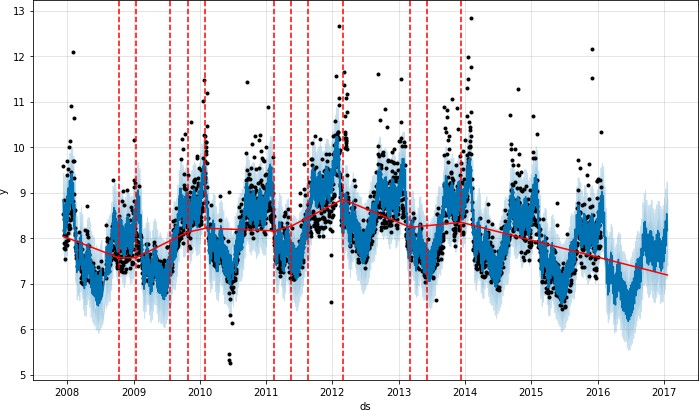
The number of potential changepoints can be set using the argument **n\_changepoints**, but this is better tuned by adjusting the regularization.可变点的数量可以使用参数**n\_changepoints**设置，但最好通过调整正则化来调整。

The locations of the signification changepoints can be visualized with:

以下是可变点位置的可视化代码

1. # Python
2. from prophet.plot import add\_changepoints\_to\_plot 3 fig = m.plot(forecast)

4 a = add\_changepoints\_to\_plot(fig.gca(), m, forecast)



By default changepoints are only inferred for the first 80% of the time series in order to have plenty of runway for projecting the trend forward and to avoid overfitting fluctuations at the end of the time series. 默认情况下，changepoint只在时间序列的前80%被推断出来，这是为了有足够的跑道来预测未来的趋势，并避免时间序列结束时的过度拟合波动。This default works in many situations but not all, 大多数情况下使用默认值即可。

and can be changed using the changepoint\_range argument. For example, **m =Prophet(changepoint\_range=0.9)** in Python or **m <- prophet(changepoint.range = 0.9)** in R will place potential changepoints in the first 90% of the time series.

# Adjusting trend flexibility

**If the trend changes are being overfit (too much flexibility) or underfit (not enough flexibility), you can adjust the strength of the sparse prior using the input argument changepoint\_prior\_scale. By default, this parameter is set to 0.05. Increasing it will make the trend *more* flexible:**

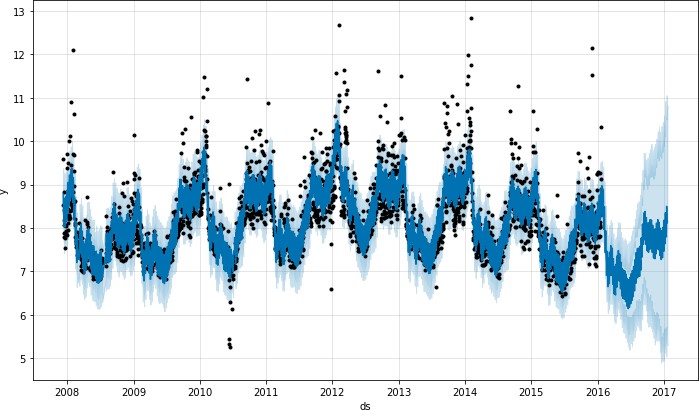
**如果趋势变化是过拟合(灵活性太大)或欠拟合(灵活性不够)，**

**# 则可以使用输入参数changepoint\_prior\_scale调整稀疏先验的强度。**

**# 默认情况下，这个参数为0.05**

**# changepoint\_prior\_scale 越大会使得趋势变化更加灵活**

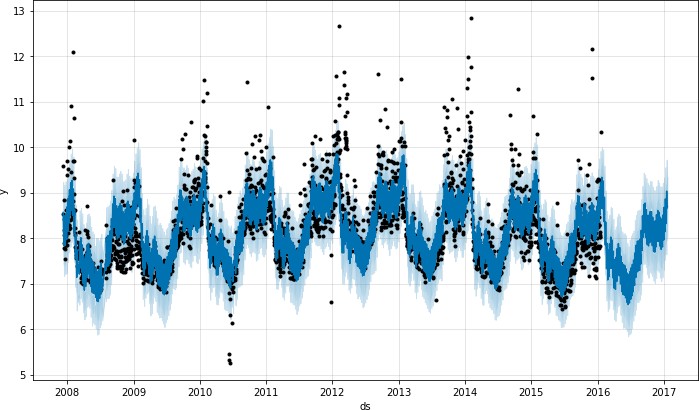
1. # Python
2. m = Prophet(changepoint\_prior\_scale=0.5) 3 forecast = m.fit(df).predict(future)
3. fig = m.plot(forecast)
4. fig.show()



Decreasing it will make the trend *less* flexible:

1. # Python
2. m = Prophet(changepoint\_prior\_scale=0.001) 3 forecast = m.fit(df).predict(future)

4 fig = m.plot(forecast)



When visualizing the forecast, this parameter can be adjusted as needed if the trend seems to be over- or under-fit. In the fully-automated setting, see the documentation on cross validation for recommendations on how this parameter can be tuned.

# Specifying the locations of the changepoints

If you wish, rather than using automatic changepoint detection you can manually specify the locations of potential changepoints with the **changepoints** argument. 如果你愿意，可以使用changepoints参数手动指定潜在变化点的位置，而不是自动检测。

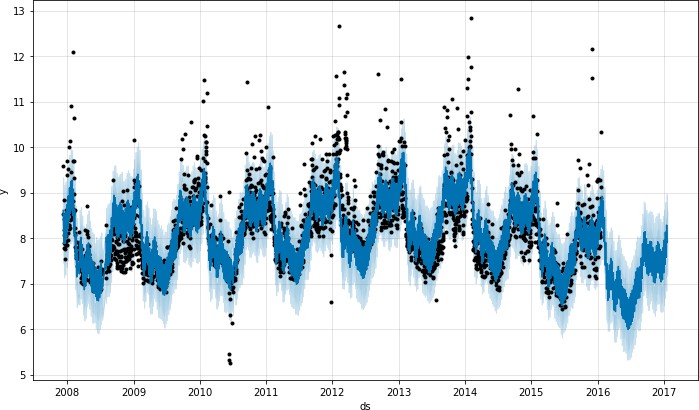
Slope changes will then be allowed only at these points, with the same sparse regularization as before. 与之前稀疏正则化相同，只允许在这些点上改变斜率。

One could, for instance, create a grid of points as is done automatically, but then augment that grid with some specific dates that are known to be likely to have changes. 例如，可以创建一个自动完成的点网格，然后用一些已知可能有变化的特定日期来扩充网格。

As another example, the changepoints could be entirely limited to a small set of dates, as is done here: 再举一个例子，changepoint可以完全限制为一小部分日期，如下所示:

1. # Python
2. m = Prophet(changepoints=['2014-01-01']) 3 forecast = m.fit(df).predict(future)

4 fig = m.plot(forecast)



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

**Facebook Open Source**

[Open Source Projects](https://code.facebook.com/projects/) [GitHub](https://github.com/facebook/) [Twitter](https://twitter.com/fbOpenSource) [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)