Handling Shocks / 处理震荡

As a result of the lockdowns caused by the COVID-19 pandemic, many time series experienced “shocks” during 2020, e.g. spikes in media consumption (Netflix, YouTube), e-commerce transactions (Amazon, eBay), whilst attendance to in-person events declined dramatically. 由于COVID-19大流行造成的封锁，许多时间序列在2020年期间经历了 "冲击"，例如媒体消费（Netflix、YouTube）、电子商务交易（亚马逊、eBay）的高峰，而亲自参加活动的人数则急剧下降。

Most of these time series would also maintain their new level for a period of time, subject to fluctuations driven by easing of lockdowns and/or vaccines. 这些时间序列中的大多数也会在一段时间内保持其新的水平，但会受到放松封控和/或疫苗的推动而出现波动。

Seasonal patterns could have also changed: for example, people may have consumed less media (in total hours) on weekdays compared to weekends before the COVID lockdowns, but during lockdowns weekday consumption could be much closer to weekend consumption. 季节性模式也可能发生变化：例如，在COVID封锁之前，人们在工作日消费的媒体（总时数）可能比周末少，但在封锁期间，工作日的消费可能更接近周末的消费

In this page we’ll explore some strategies for capturing these effects using Prophet’s functionality:

1. Marking step changes / spikes due to COVID events as once-off.
2. Sustained changes in behaviour leading to trend and seasonality changes.

在这一页，我们将探讨一些使用先知的功能来捕捉这些效果的策略：

1. 将COVID事件引起的阶跃变化/尖峰标记为一次性的。

2. 导致趋势和季节性变化的行为的持续变化。

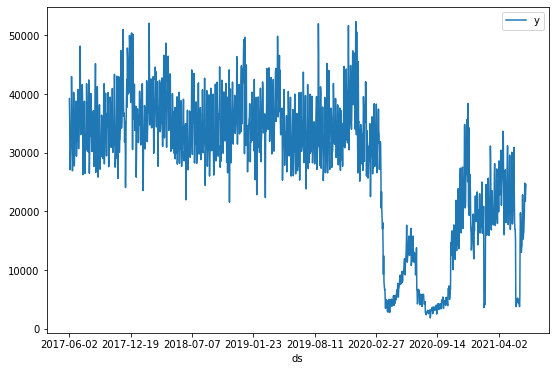
## Case Study - Pedestrian Activity / 案例研究--行人活动

For this case study we’ll use [Pedestrian Sensor data from the City of Melbourne](https://data.melbourne.vic.gov.au/Transport/Pedestrian-Counting-System-Monthly-counts-per-hour/b2ak-trbp). 对于这个案例研究，我们将使用墨尔本市的行人传感器数据。

This data measures foot traffic from sensors in various places in the central business district, and we’ve chosen one sensor (Sensor\_ID = 4) and aggregated the values to a daily grain. 这个数据是通过中央商务区各个地方的传感器来测量人流量的，我们选择了一个传感器（Sensor\_ID=4），并将其数值汇总到每天。

[**The aggregated dataset can be found in the examples folder here. 聚合的数据集可以在这里的例子文件夹中找到**](https://github.com/facebook/prophet/tree/master/examples/example_pedestrians_covid.csv)

1. # Python
2. df = pd.read\_csv('https://raw.githubusercontent.com/facebook/prophet/main/examples/example\_pedestrians\_covid.csv')
3. # Python
4. df.set\_index('ds').plot();



We can see two key events in the time series: 我们可以在时间序列中看到两个关键事件：

* + The initial drop in foot traffic around 21 March 2020, which started to recover around 6 June 2020. This corresponds to the declaration of a pandemic by WHO and subsequent lockdowns mandated by the Victorian government. - 2020年3月21日前后，人流量初步下降，2020年6月6日前后开始恢复。这与世卫组织宣布的大流行病和维多利亚州政府随后授权的封锁相吻合。
  + After some slow recovery, a second drop in foot traffic around July 9 2020, which began to recover around 27 October 2020. This corresponds to the “second wave” of the pandemic in metropolitan Melbourne. 在缓慢恢复之后，2020年7月9日前后，人流量出现第二次下降，2020年10月27日前后开始恢复。这与墨尔本市的大流行病 "第二波 "相对应。

There are also shorter periods of strict lockdown that lead to sudden tips in the time series: 5 days in February 2021, and 14 days in early June 2021. 还有一些较短的严格封锁期，导致时间序列中的突然提示： 2021年2月的5天，以及2021年6月初的14天。

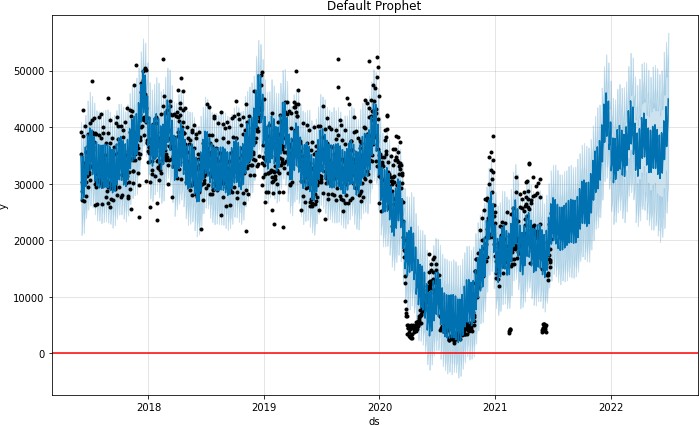
## Default model without any adjustments / 没有任何调整的默认模式

First we’ll fit a model with the default Prophet settings: 首先，我们将用默认的先知设置来拟合一个模型：

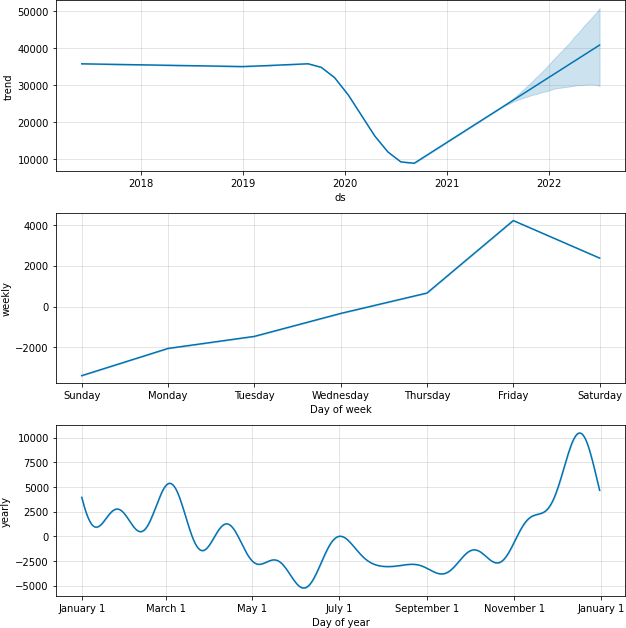
1. # Python
2. m = Prophet() 3 m = m.fit(df)

4 future = m.make\_future\_dataframe(periods=366) 5 forecast = m.predict(future)

1. # Python
2. m.plot(forecast)
3. plt.axhline(y=0, color='red')
4. plt.title('Default Prophet');



1. # Python
2. m.plot\_components(forecast);



The model seems to fit reasonably to past data, but notice how we’re capturing the dips, and the spikes after the dips, as a part of the trend component. 该模型似乎合理地适合过去的数据，但注意到我们是如何捕捉到这些低点，以及低点之后的尖峰，作为趋势成分的一部分。

By default, the model assumes that these large spikes are possible in the future, even though we realistically won’t see something of the same magnitude within our forecast horizon (1 year in this case). This leads to a fairly optimistic forecast of the recovery of foot traffic in 2022. 默认情况下，该模型假设这些大的峰值在未来是可能的。尽管在我们的预测范围内（在这种情况下是1年），我们现实中不会看到同样幅度的东西。这导致了对2022年人流量恢复的相当乐观的预测。

# Treating COVID-19 lockdowns as a one-off holidays

To prevent large dips and spikes from being captured by the trend component, we can treat the days impacted by COVID-19 as holidays that will not repeat again in the future. Adding custom holidays is explained in more detail [here](https://facebook.github.io/prophet/docs/seasonality%2C_holiday_effects%2C_and_regressors.html#modeling-holidays-and-special-events). We set up a DataFrame like so to describe the periods affected by lockdowns: 为了防止大的跌幅和峰值被趋势成分捕获，我们可以把受COVID-19影响的日子当作假期，在未来不会再重复。添加自定义假日在之前的《04\_季节性、假日效应和回归因子》有更详细的解释。我们像这样设置一个DataFrame来描述受锁定影响的时期：

1. # Python
2. lockdowns = pd.DataFrame([

3 {'holiday': 'lockdown\_1', 'ds': '2020-03-21', 'lower\_window': 0, 'ds\_upper': '2020-06-06'},

4 {'holiday': 'lockdown\_2', 'ds': '2020-07-09', 'lower\_window': 0, 'ds\_upper': '2020-10-27'},

5 {'holiday': 'lockdown\_3', 'ds': '2021-02-13', 'lower\_window': 0, 'ds\_upper': '2021-02-17'},

6 {'holiday': 'lockdown\_4', 'ds': '2021-05-28', 'lower\_window': 0, 'ds\_upper': '2021-06-10'},

7 ])

1. for t\_col in ['ds', 'ds\_upper']:
2. lockdowns[t\_col] = pd.to\_datetime(lockdowns[t\_col])
3. lockdowns['upper\_window'] = (lockdowns['ds\_upper'] - lockdowns['ds']).dt.days 11 lockdowns



ckdown\_1 2020-03-21 0

ckdown\_2 2020-07-09 0

ckdown\_3 2021-02-13 0

ckdown\_4 2021-05-28 0

2020-06-06 77

2020-10-27 110

2021-02-17 4

2021-06-10 13

**upper\_window**

**ds\_upper**

**lower\_window**

**ds**

**holiday**

|  |
| --- |
|  |
|  |

**0**lo **1**lo **2**lo **3**lo

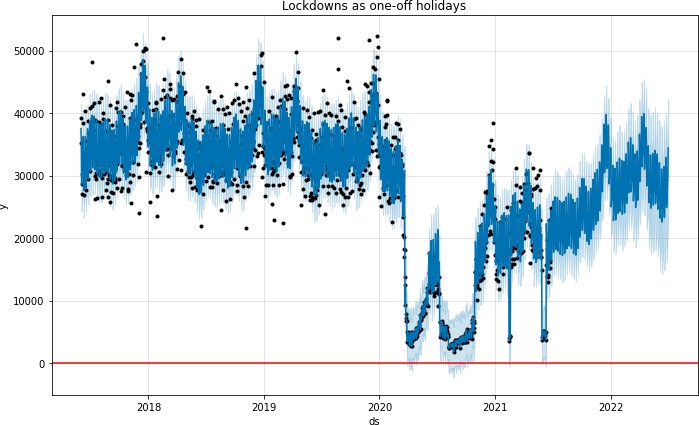
* + We have an entry for each lockdown period, with ds specifying the start of the lockdown. ds\_upper is not used by Prophet, but it’s a convenient way for us to calculate upper\_window. 我们为每个封锁期都有一个条目，ds指定了封锁期的开始。Ds\_upper不被先知使用，但它是我们计算upper\_window的一个方便方法
  + **upper\_window tells Prophet that the lockdown spans for x days after the start of the lockdown. Note that the holidays regression is inclusive of the upper bound.** upper\_window告诉Prophet封锁的时间跨度为封锁开始后的x天**。请注意，假期回归是包括上限的。**

Note that since we don’t specify any future dates, Prophet will assume that these holidays will *not* occur again when creating the future dataframe (and hence they won’t affect our projections). This is different to how we would specify a recurring holiday. 请注意，由于我们没有指定任何未来的日期，Prophet将假设这些假期在创建未来数据框架时不会再发生（因此它们不会影响我们的预测）。

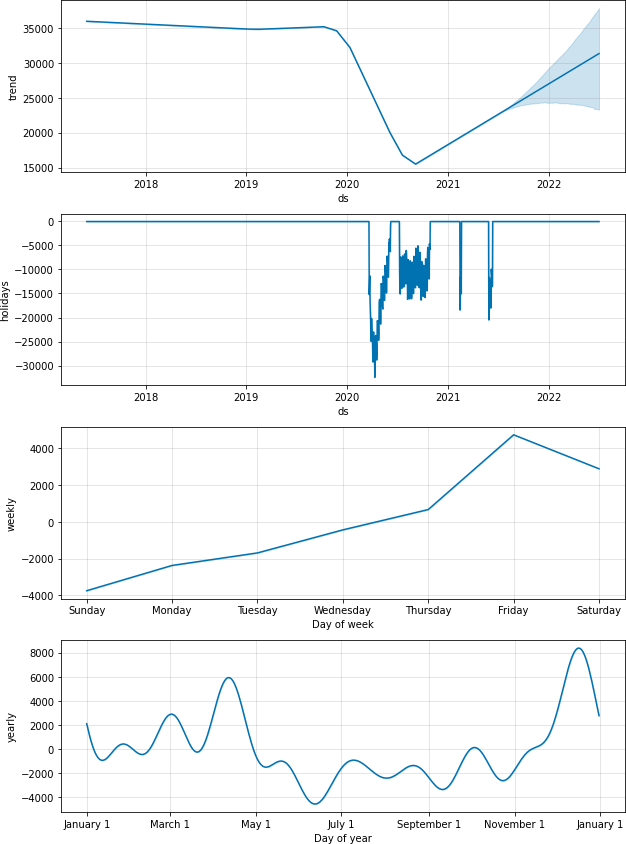
1. # Python
2. m2 = Prophet(holidays=lockdowns) 3 m2 = m2.fit(df)

4 future2 = m2.make\_future\_dataframe(periods=366) 5 forecast2 = m2.predict(future2)

1. # Python
2. m2.plot(forecast2)
3. plt.axhline(y=0, color='red')
4. plt.title('Lockdowns as one-off holidays');



1. # Python
2. m2.plot\_components(forecast2);



* + Prophet is sensibly assigning a large negative effect to the days within the lockdown periods.
  + - 预言家理智地将巨大的负面效应分配给封控期内的日子。
  + The forecast for the trend is not as strong / optimistic and seems fairly reasonable.
  + - 对趋势的预测没有那么强烈/乐观，似乎相当合理。

# Sense checking the trend

In an environment when behaviours are constantly changing, it’s important to ensure that the trend component of the model is capturing to emerging patterns without overfitting to them. 在一个行为不断变化的环境中，重要的是要确保模型的趋势部分能够捕捉到新出现的模式，而不会对其过度拟合。

**The** [**trend changepoints**](https://facebook.github.io/prophet/docs/trend_changepoints.html) **documentation explains two things we could tweak with the trend component: 趋势变化点文档解释了我们可以用趋势组件调整的两件事：**

* + The changepoint locations, 变化点的位置which by default are evenly spaced across 80% of the history. 默认是均匀分布在历史的80%。We should be mindful of where this range ends, and extend the range (either by increasing the % or adding changepoints manually) if we believe the most recent data better reflects future behaviour. 我们应该注意这个范围的终点，如果我们认为最重要的是，我们应该扩大这个范围（通过增加百分比或手动添加变化点）。
  + The strength of regularisation (**changepoint\_prior\_scale**), which determines how flexible the trend is; the default value is

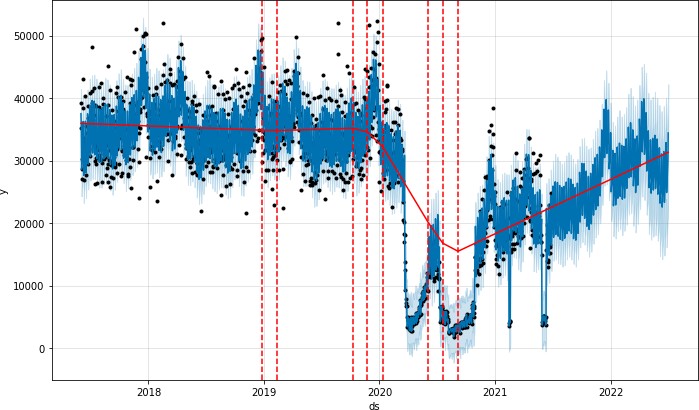
**0.05** and increasing this will allow the trend to fit more closely to the observed data. 正则化的强度（**changepoint\_prior\_scale**），它决定了趋势的灵活程度；默认值是0.05，增加这个值可以使趋势更贴近观测数据。（更好的拟合程度）

We plot the trend component and changepoints detected by our current model below.

我们把我们目前的模型所检测到的趋势成分和变化点绘制在下面。

1. # Python
2. from prophet.plot import add\_changepoints\_to\_plot 3 fig = m2.plot(forecast2)

4 a = add\_changepoints\_to\_plot(fig.gca(), m2, forecast2)



The detected changepoints look reasonable, 检测到的变更点看起来很合理，and the future trend tracks the latest upwards trend in activity, 同时未来趋势追踪到了事件中的最新上升趋势。but not to the extent of late 2020. This seems suitable for a best guess of future activity. 这似乎适合作为对未来活动的最佳预测。

We can see what the forecast would look like if we wanted to emphasize COVID patterns more in model training; we can do this by adding more potential changepoints after 2020 and making the trend more flexible. 我们可以看到，如果我们想在模型训练中更多地强调COVID模式，预测会是什么样子；**我们可以通过在2020年之后增加更多的潜在变化点，使趋势更加灵活。**

1. # Python
2. *m3\_changepoints = (*
3. *# 10 potential changepoints in 2.5 years*

*4 pd.date\_range('2017-06-02', '2020-01-01', periods=10).date.tolist() +*

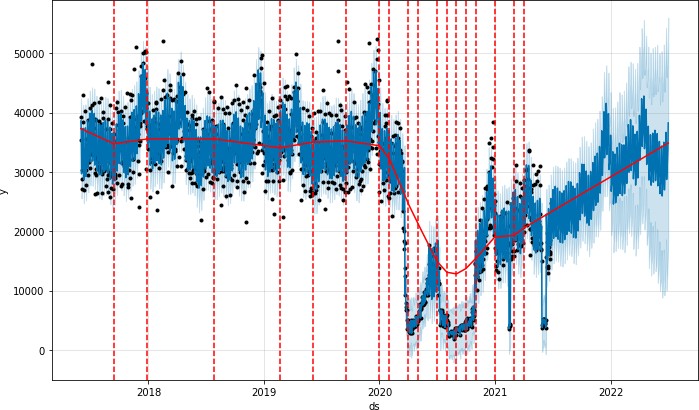
*5 # 15 potential changepoints in 1 year 2 months*

*6 pd.date\_range('2020-02-01', '2021-04-01', periods=15).date.tolist()*

*7 )*

1. # Python
2. # Default changepoint\_prior\_scale is 0.05, so 1.0 will lead to much more flexibility in comparison. 3 m3 = Prophet(holidays=lockdowns, changepoints=m3\_changepoints, changepoint\_prior\_scale=1.0)
3. m3 = m3.fit(df)
4. forecast3 = m3.predict(future2)
5. # Python
6. from prophet.plot import add\_changepoints\_to\_plot 3 fig = m3.plot(forecast3)

4 a = add\_changepoints\_to\_plot(fig.gca(), m3, forecast3)



We’re seeing many changepoints detected post-COVID, matching the various fluctuations from loosening / tightening lockdowns. Overall the trend curve and forecasted trend are quite similar to our previous model, but we’re seeing a lot more uncertainty because of the higher number of trend changes we picked up in the history. 我们看到在COVID之后检测到许多变化点，与松动/紧缩锁定的各种波动相匹配。总的来说，趋势曲线和预测趋势与我们以前的模型相当相似，但我们看到更多的不确定性，因为我们在历史上发现了更多的趋势变化。

We probably wouldn’t pick this model over the model with default parameters as a best estimate, 我们可能不会将一个使用了默认参数的模型作为最终的预测模型，but it’s a good demonstration of how we can incorporate our beliefs into the model about which patterns are important to capture.但是这是一个很好的示范，说明了我们可以将自己的观念加入模型，说明哪些模式是重要的，需要捕捉特征。

# Changes in seasonality between pre- and post-COVID

The seasonal component plots in the previous sections show a peak of activity on Friday compared to other days of the week. If we’re not sure whether this will still hold post-lockdown, we can add *conditional seasonalities* to the model.季节性成分图中显示，在一周中，星期五相对于其他日子，活动达到了高点。如果我们不能确定这样的季节性是否在封控后还会出现，我们就需要在模型中加入季节性。

[有关季节性的描述，参照以下](Prophet%20官方文档/Quick%20Start%20Prophet/Prophet%20Docs/04_季节性、假日效应和回归因子.docx)

First we define boolean columns in the history dataframe to flag “pre covid” and “post covid” periods: 首先，我们在历史数据框架中定义了布尔列，以标记 "pre\_covid "和 "post\_covid "时期：

1. # Python
2. df2 = df.copy()
3. df2['pre\_covid'] = pd.to\_datetime(df2['ds']) < pd.to\_datetime('2020-03-21') 4 df2['post\_covid'] = ~df2['pre\_covid']

The conditional seasonality we’re interested in modelling here is the day-of-week (“weekly”) seasonality. 我们在此模型中加入的季节性是一个周季节性——day\_of\_week，一周中的某一天。To do this, we firstly turn off the default **weekly\_seasonality** when we create the Prophet model.为了做这件事，我们首先将模型的默认的季节性关闭。

1. # Python
2. m4 = Prophet(holidays=lockdowns, weekly\_seasonality=False)

We then add this weekly seasonality manually, 手动添加季节性as two different model components - one for pre-covid, one for post-covid. 一个是per\_COVID，一个是post\_COVID

Note that **fourier\_order=3** is the default setting for weekly seasonality. After this we can run **.fit()**.注意fourier\_order=3是对于周季节性的默认参数，以上设置结束后就可以进行训练了

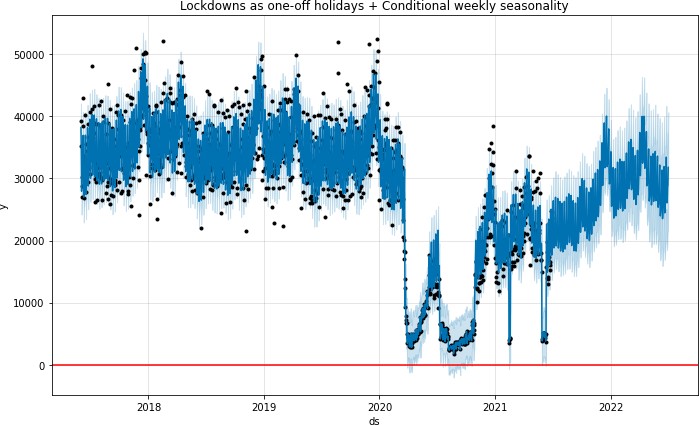
1. # Python
2. m4.add\_seasonality(
3. name='weekly\_pre\_covid',
4. period=7,
5. fourier\_order=3,
6. condition\_name='pre\_covid', 7 )
7. m4.add\_seasonality(
8. name='weekly\_post\_covid',
9. period=7,
10. fourier\_order=3,
11. condition\_name='post\_covid',

13 );

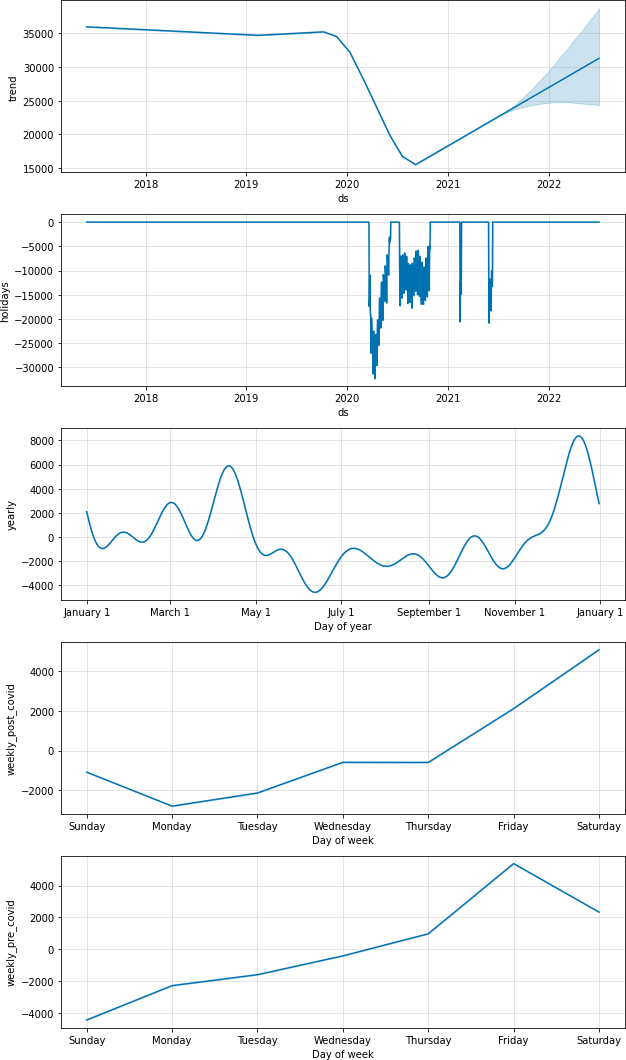
1. # Python
2. m4 = m4.fit(df2)

We also need to create the **pre\_covid** and **post\_covid** flags in the future dataframe. 我们还需要在future DataFrame中创建**pre\_covid** and **post\_covid** 标志，This is so that Prophet can apply the correct weekly seasonality parameters to each future date.这样Prophet才可以正确地处理未来的每个日子的周季节性。

1. # Python
2. future4 = m4.make\_future\_dataframe(periods=366)
3. future4['pre\_covid'] = pd.to\_datetime(future4['ds']) < pd.to\_datetime('2020-03-21') 4 future4['post\_covid'] = ~future4['pre\_covid']
4. # Python
5. forecast4 = m4.predict(future4)
6. # Python
7. m4.plot(forecast4)
8. plt.axhline(y=0, color='red')
9. plt.title('Lockdowns as one-off holidays + Conditional weekly seasonality');



1. # Python
2. m4.plot\_components(forecast4);



Interestingly, the model with conditional seasonalities suggests that, post-COVID, pedestrian activity peaks on Saturdays, instead of Fridays. This could make sense if most people are still working from home and are hence less likely to go out on Friday nights. From a prediction perspective this would only be important if we care about predicting weekdays vs. weekends accurately, but overall this kind of exploration helps us gain insight into how COVID has changed behaviours.

***具体分析：有趣的是，具有条件季节性的模型表明，在COVID之后，行人活动在周六而不是周五达到高峰。这可能是有道理的，如果大多数人仍然在家工作，因此不太可能在周五晚上出去。从预测的角度来看，这只有在我们关心准确预测工作日与周末的情况下才是重要的，但总的来说，这种探索有助于我们深入了解COVID是如何改变人们的行为。***

**Further reading / 延申**

A lot of the content in this page was inspired by this [GitHub discussion](https://github.com/facebook/prophet/issues/1416). We’ve covered a few low hanging fruits for tweaking Prophet models when faced with shocks such as COVID,我们已经给出在类似于COVID中一些比较容易实现的调整Prophet的方法 but there are many other possible approaches as well, such as:但是任然有一些其他的可能的方式。

* + Using external regressors (e.g. the lockdown stringency index).使用外部回归因子（在COVID中表现为封锁的严格程度）

This would only be fruitful if we 这只有在以下情况下，才会有相应成功

a) have regressor data that aligns well (in terms of location) with the series we’re forecasting 有回归数据，且在我们正在预测的序列中一致性较好（就位置而言）

b) have control over or can predict the regressor much more accurately than the time series alone. 与单独的时间序列相比，对回归者有控制权或能更准确地预测回归者。

* + Detecting and removing outlier data from the training period, or throwing away older training data completely. This might be a better approach for sub-daily time series that don’t have yearly seasonal patterns.

Overall though it’s difficult to be confident in our forecasts in these environments when rules are constantly changing and outbreaks occur randomly. In this scenario it’s more important to constantly re-train / re-evaluate our models and clearly communicate the increased uncertainty in forecasts. 总的来说，虽然在这些环境中，当规则不断变化和爆发随机发生时，我们很难对预测有信心。在这种情况下，更重要的是不断地重新训练/重新评估我们的模型，并清楚地传达预测中增加的不确定性。

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