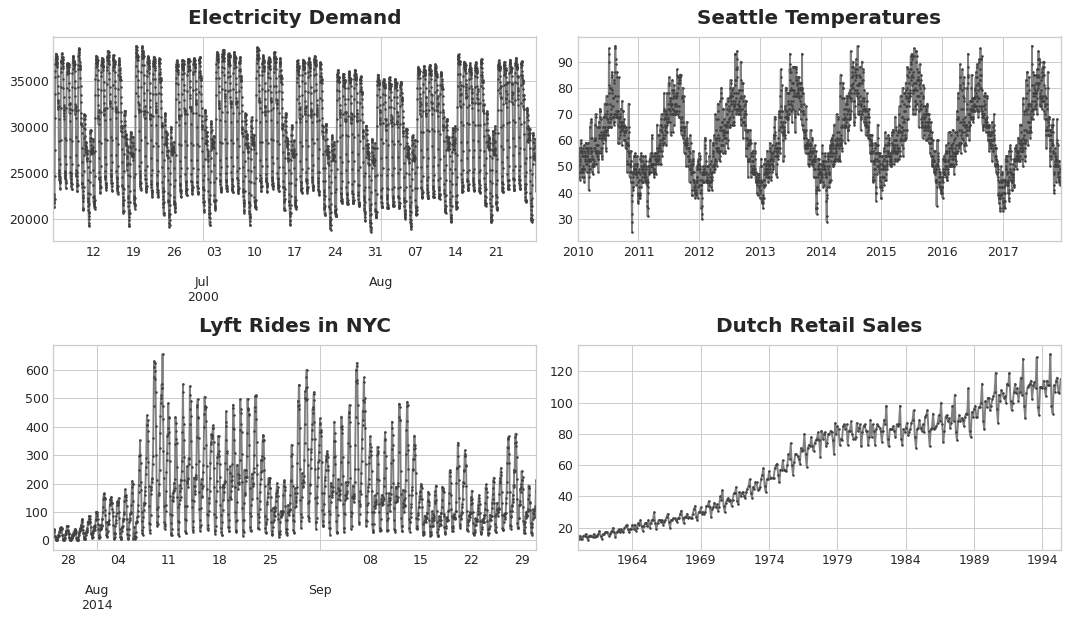
Time Series: 3rd lesson – Seasonality

Seasonality:

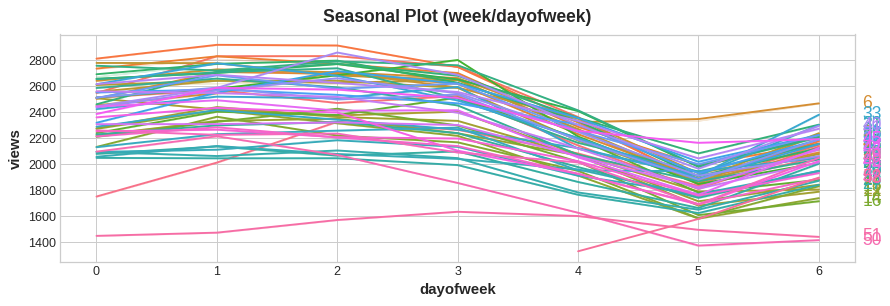
A time series exhibits seasonality whenever there is a regular, periodic change in the mean of the series. Seasonal changes generally follow the clock and calendar -- repetitions over a day, a week, or a year are common. Seasonality is often driven by the cycles of the natural world over days and years or by conventions of social behavior surrounding dates and times.



We will learn two kinds of features that model seasonality. The first kind, indicators, is best for a season with few observations, like a weekly season of daily observations. The second kind, Fourier features, is best for a season with many observations, like an annual season of daily observations.

Seasonal plots and seasonal indicators:

Just like we used a moving average plot to discover the trend in a series, we can use a seasonal plot to discover seasonal patterns. A seasonal plot shows segments of the time series plotted against some common period, the period being the "season" you want to observe. The figure shows a seasonal plot of the daily views of Wikipedia's article on *Trigonometry*: the article's daily views plotted over a common weekly period.

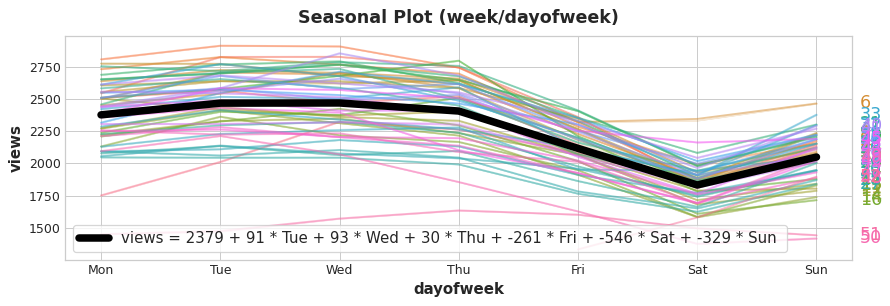


Seasonal indicators:

A group of binary features that represent seasonal differences in the level of a time series. Seasonal indicators are what you get if you treat a seasonal period as a categorical feature and apply one-hot encoding. By one-hot encoding days of the week, we get weekly seasonal indicators. Creating weekly indicators for the Trigonometry series will then give us six new "dummy" features (linear regression works best if you drop one of the indicators; we chose Monday in the frame below).

| Date | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday |
| --- | --- | --- | --- | --- | --- | --- |
| 2016-01-04 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2016-01-05 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2016-01-06 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2016-01-07 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| 2016-01-08 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 2016-01-09 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| 2016-01-10 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| 2016-01-11 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

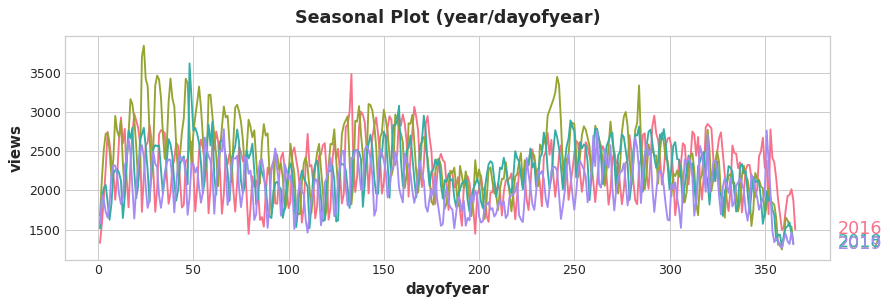
Adding seasonal indicators to the training data helps models distinguish means within a seasonal period:



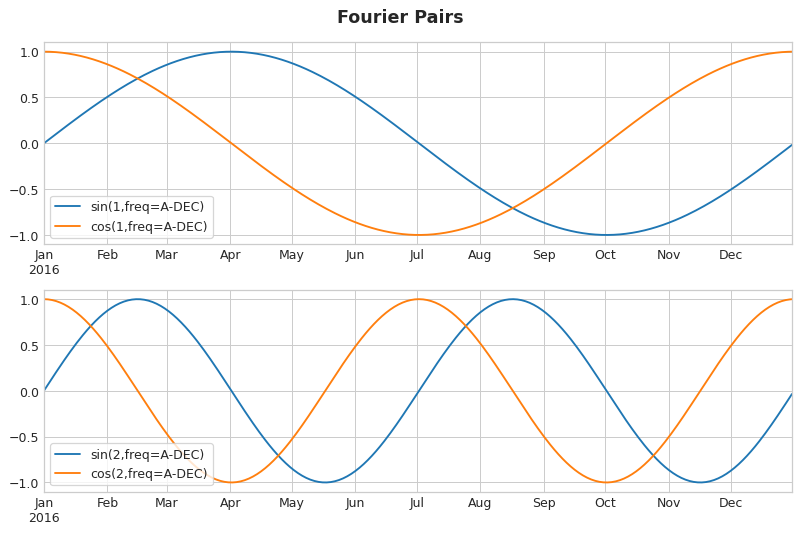
The indicators act as On / Off switches. At any time, at most one of these indicators can have a value of 1 (On). Linear regression learns a baseline value 2379 for Mon and then adjusts by the value of whichever indicator is On for that day; the rest are 0 and vanish.

Fourier features and the periodogram:

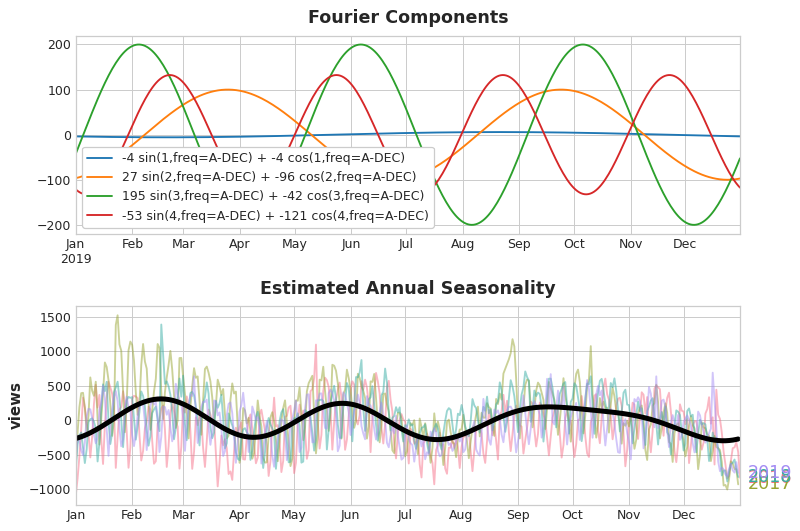
The kind of feature we discuss now are better suited for long seasons over many observations where indicators would be impractical. Instead of creating a feature for each date, Fourier features try to capture the overall shape of the seasonal curve with just a few features. Let's take a look at a plot for the annual season in Trigonometry. Notice the repetitions of various frequencies: a long up-and-down movement three times a year, short weekly movements 52 times a year, and perhaps others.



It is these frequencies within a season that we attempt to capture with Fourier features. The idea is to include in our training data periodic curves having the same frequencies as the season we are trying to model. The curves we use are those of the trigonometric functions sine and cosine. Fourier features are pairs of sine and cosine curves, one pair for each potential frequency in the season starting with the longest. Fourier pairs modeling annual seasonality would have frequencies: once per year, twice per year, three times per year, and so on.



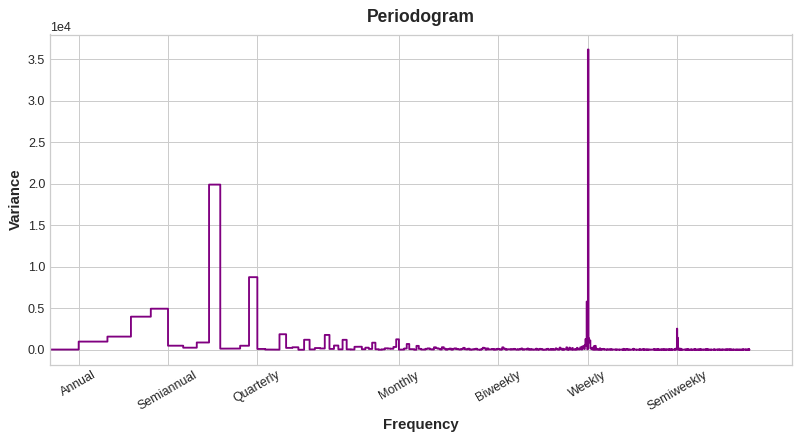
If we add a set of these sine / cosine curves to our training data, the linear regression algorithm will figure out the weights that will fit the seasonal component in the target series. The figure illustrates how linear regression used four Fourier pairs to model the annual seasonality in the Wiki Trigonometry series.



Notice that we only needed eight features (four sine / cosine pairs) to get a good estimate of the annual seasonality. Compare this to the seasonal indicator method which would have required hundreds of features (one for each day of the year). By modeling only the "main effect" of the seasonality with Fourier features, you'll usually need to add far fewer features to your training data, which means reduced computation time and less risk of overfitting.

Choosing Fourier features with the periodogram:

How many Fourier pairs should we actually include in our feature set? We can answer this question with the periodogram. The periodogram tells you the strength of the frequencies in a time series. Specifically, the value on the y-axis of the graph is (a \*\* 2 + b \*\* 2) / 2, where a and b are the coefficients of the sine and cosine at that frequency (as in the Fourier Components plot above).



From left to right, the periodogram drops off after Quarterly, four times a year. That was why we chose four Fourier pairs to model the annual season. The Weekly frequency we ignore since it's better modeled with indicators.

Computing Fourier features:

Knowing how Fourier features are computed isn't essential to using them, but if seeing the details would clarify things, the cell hidden cell below illustrates how a set of Fourier features could be derived from the index of a time series (we'll use a library function from statsmodels for our applications, however).

import numpy as np

def fourier\_features(index, freq, order):

time = np.arange(len(index), dtype=np.float32)

k = 2 \* np.pi \* (1 / freq) \* time

features = {}

for i **in** range(1, order + 1):

features.update({

f"sin\_**{**freq**}**\_**{**i**}**": np.sin(i \* k),

f"cos\_**{**freq**}**\_**{**i**}**": np.cos(i \* k),

})

return pd.DataFrame(features, index=index)

*# Compute Fourier features to the 4th order (8 new features) for a*

*# series y with daily observations and annual seasonality:*

*#*

*# fourier\_features(y, freq=365.25, order=4)*