Classical decomposition - Seasonality

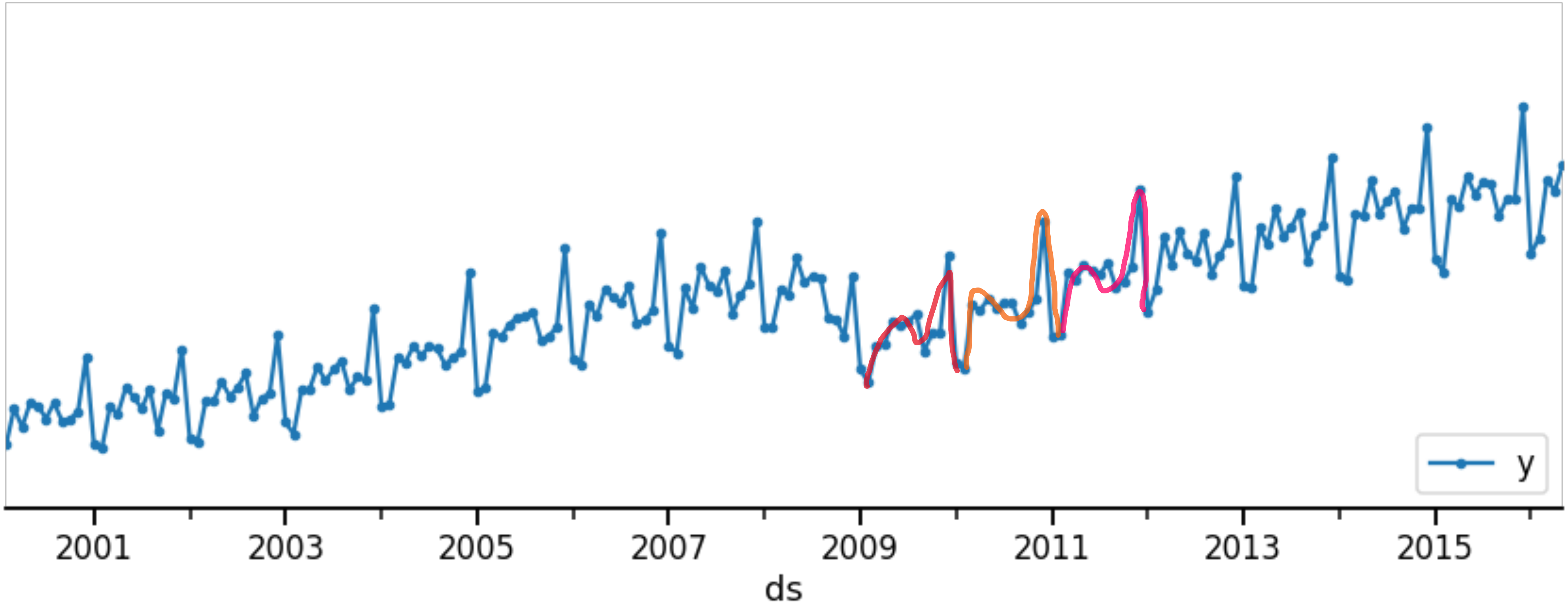
Time series decomposition

Contents

MOVING AVERAGES TO EXTRACT THE SEASONALITY

DISCUSS LIMITATIONS

How can we extract the seasonality?



How can we extract the seasonality?

Additive

$y(t) = trend(t) + seasonal(t) + residual(t)$

$seasonal(t) = y(t) – trend(t) - residual(t)$

Multiplicative

$y(t) = trend(t) \times seasonal(t) \times residual(t)$

$seasonal(t) = y(t) \times residual(t) / trend(t)$

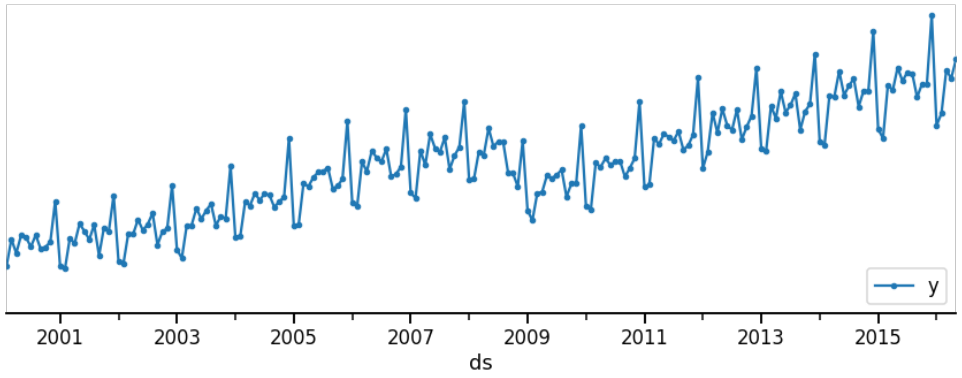
Remove impact of residuals by averaging over multiple seasonal periods

Estimate trend using moving averages

Classical decomposition: Seasonality

1. Identify order of seasonality T

2. Compute trend using T-MA (if odd) or 2 x T-MA (if even)



- Monthly granularity
- Yearly seasonality
- Therefore, T = 12

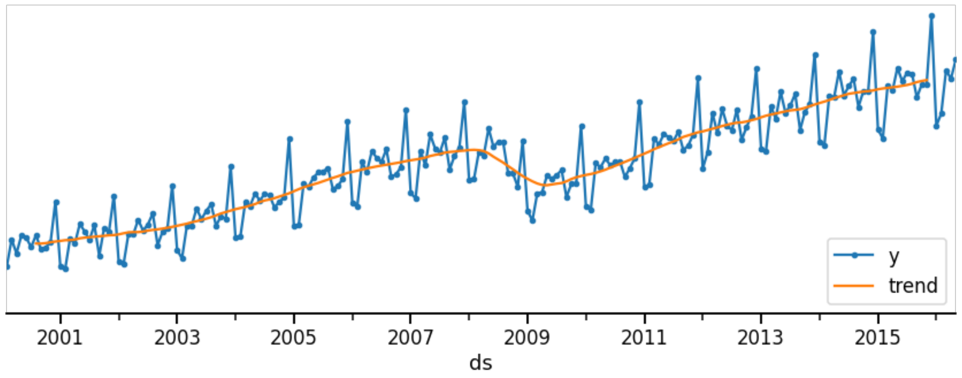
Classical decomposition: Seasonality

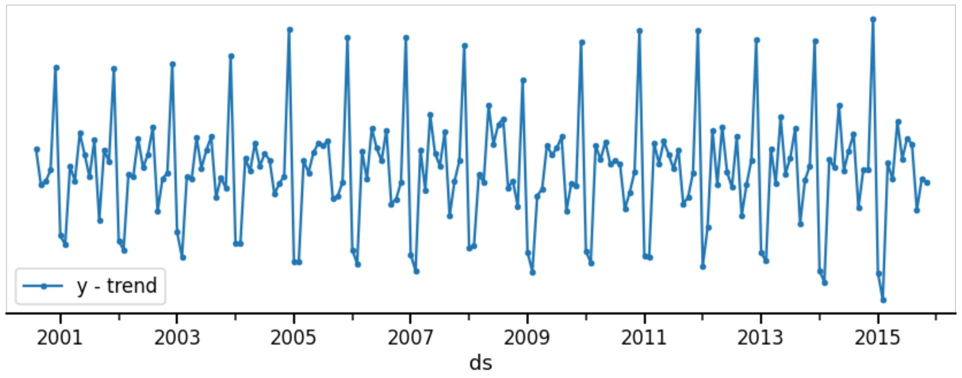
1. Identify order of seasonality T

2. Compute trend using T-MA (if odd) or 2 x T-MA (if even)

3. De-trend the data:
   1. If additive: $y_t - trend_t$
   2. If multiplicative: $y_t \ / \ trend_t$

4. Average the de-trended data over each seasonal index to remove noise (e.g., for monthly data average all the May months)





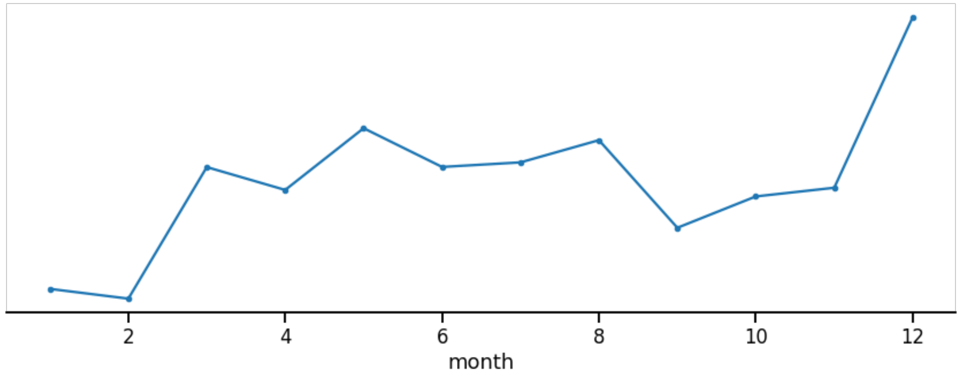
Classical decomposition: Seasonality

1. Identify order of seasonality T

2. Compute trend using T-MA (if odd) or 2 x T-MA (if even)

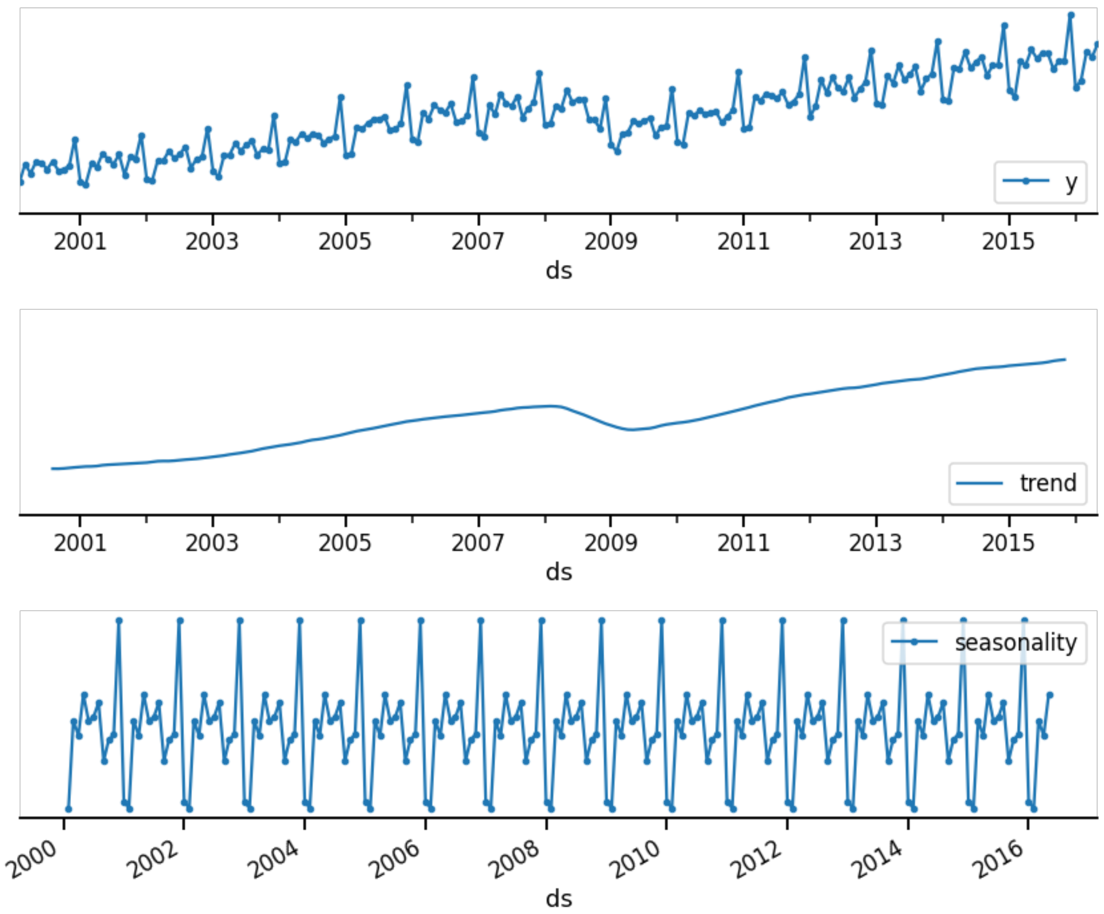
3. De-trend the data:
    1. If additive: $y_t - trend_t$
    2. If multiplicative: $y_t \,/\, trend_t$

4. Average the de-trended data over each seasonal index to remove noise (e.g., for monthly data average all the May months)



Classical decomposition: Seasonality

- The seasonal pattern is fixed each year
- We can repeat the seasonal pattern each year to get $seasonal_t$
- We can plot $seasonal_t$ alongside $trend_t$ and y_t



Implementation

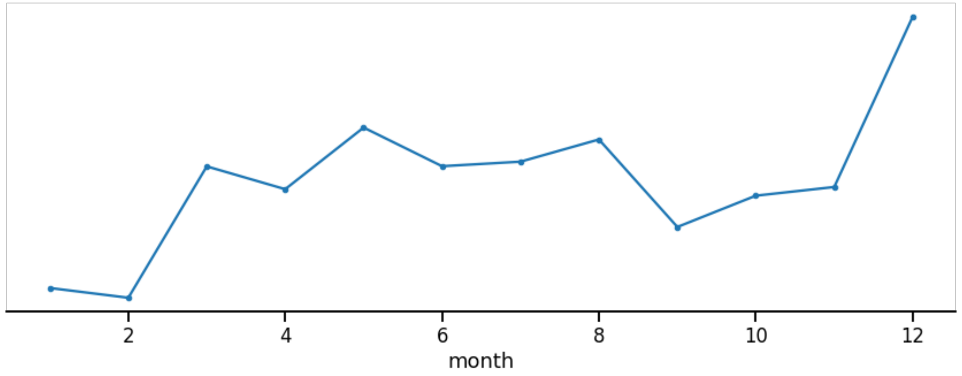
```python
# Compute trend via 2X12-MA
df['trend'] = (df['y'].rolling(window=12).mean()
                      .rolling(window=2).mean()
                      .shift(-12 // 2).values)

# De-trend the data
df['y_detrended'] = df['y'] - df['trend']

# Average over each month
df['month'] = df.index.month
seasonality = df.groupby('month').mean()['y_detrended']
```

```python
seasonality
```

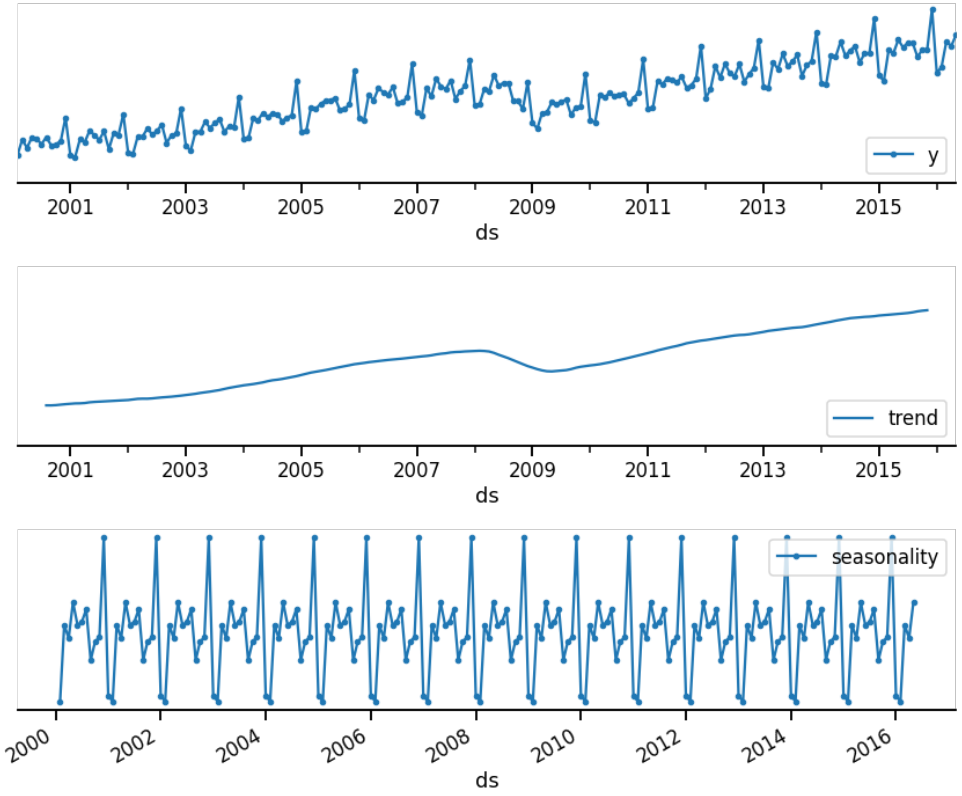
```
month
1     -34017.155556
2     -37090.133333
3       4371.583333
4      -2827.122222
5      16630.911111
6       4424.891667
7       5860.566667
8      12867.078125
9     -14751.338542
10     -4875.880208
11     -2136.195313
12     51604.936111
Name: y_detrended, dtype: float64
```



Implementation

```
(
    df.merge(right=seasonality, left_on='month', right_index=True) # Join on the month to repeat seasonal pattern
      .sort_index() # Need to re-sort by index after joining
      .iloc[12:26] # Subsample to show example
)
```

ds	y	trend	y_detrended	month	seasonality
2001-02-01	247772	278563.041667	-30791.041667	2	-37090.133333
2001-03-01	280449	278712.541667	1736.458333	3	4371.583333
2001-04-01	274925	279273.625000	-4348.625000	4	-2827.122222
2001-05-01	296013	280595.583333	15417.416667	5	16630.911111
2001-06-01	287881	281315.333333	6565.666667	6	4424.891667
2001-07-01	279098	281777.666667	-2679.666667	7	5860.566667
2001-08-01	294763	282201.250000	12561.750000	8	12867.078125
2001-09-01	261924	282623.166667	-20699.166667	9	-14751.338542
2001-10-01	291596	283232.916667	8363.083333	10	-4875.880208
2001-11-01	287537	283825.041667	3711.958333	11	-2136.195313
2001-12-01	326202	284048.458333	42153.541667	12	51604.936111
2002-01-01	255598	284769.625000	-29171.625000	1	-34017.155556
2002-02-01	253086	285970.791667	-32884.791667	2	-37090.133333
2002-03-01	285261	286960.708333	-1699.708333	3	4371.583333



Implementation

statsmodels.tsa.seasonal.seasonal_decompose

`statsmodels.tsa.seasonal.seasonal_decompose`(*x, model='additive', filt=None, period=None, two_sided=True, extrapolate_trend=0*)
[source]

Seasonal decomposition using moving averages.

Parameters

x : array_like

Time series. If 2d, individual series are in columns. x must contain 2 complete cycles.

model : {"additive", "multiplicative"}, `optional`

Type of seasonal component. Abbreviations are accepted.

filt : array_like, `optional`

The filter coefficients for filtering out the seasonal component. The concrete moving average method used in filtering is determined by two_sided.

period : `int`, `optional`

Period of the series. Must be used if x is not a pandas object or if the index of x does not have a frequency. Overrides default periodicity of x if x is a pandas object with a timeseries index.

```python
from statsmodels.tsa.seasonal import seasonal_decompose
```

```python
res = seasonal_decompose(x=df['y'],
                         model='additive',
                         period=12)
res.seasonal.head()
```

```
ds
2000-02-01   -37095.311820
2000-03-01     4366.404847
2000-04-01    -2832.300709
2000-05-01    16625.732624
2000-06-01     4419.713180
Name: seasonal, dtype: float64
```

Implementation

statsmodels.tsa.seasonal.seasonal_decompose

`statsmodels.tsa.seasonal.seasonal_decompose`(*x, model='additive', filt=None, period=None, two_sided=True, extrapolate_trend=0*)
[source]

Seasonal decomposition using moving averages.

Parameters

x : array_like

Time series. If 2d, individual series are in columns. x must contain 2 complete cycles.

model : {"additive", "multiplicative"}, `optional`

Type of seasonal component. Abbreviations are accepted.

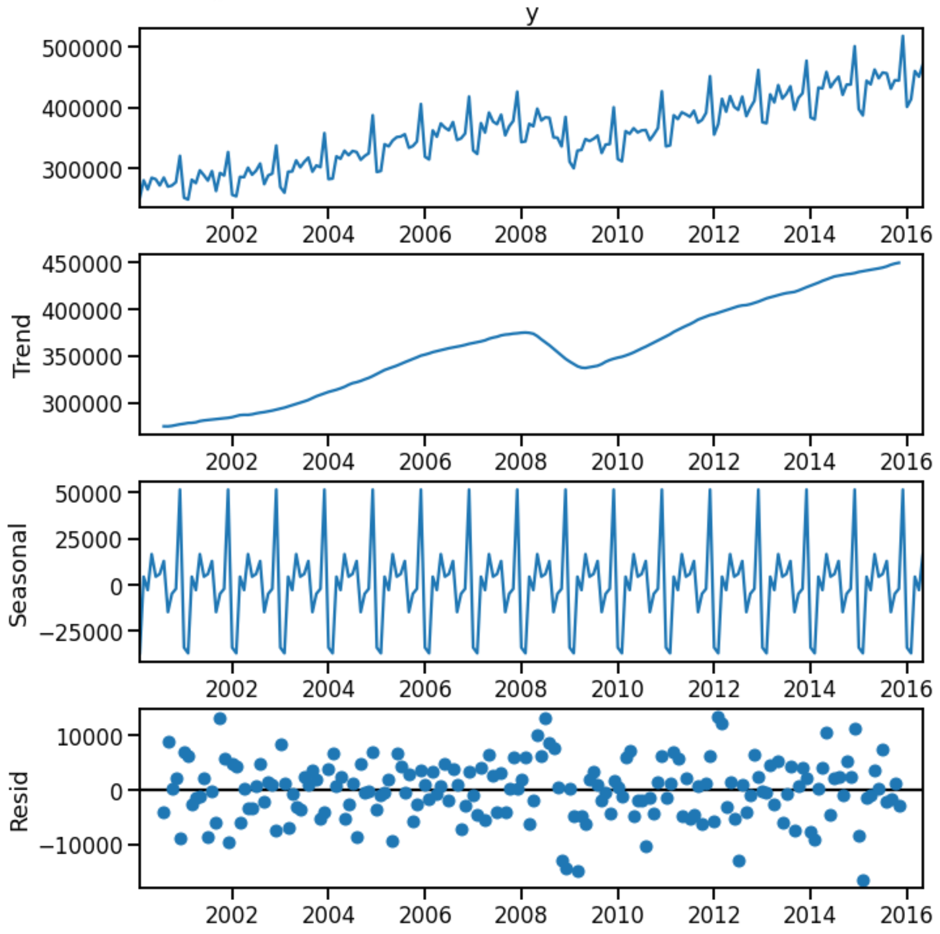
filt : array_like, `optional`

The filter coefficients for filtering out the seasonal component. The concrete moving average method used in filtering is determined by two_sided.

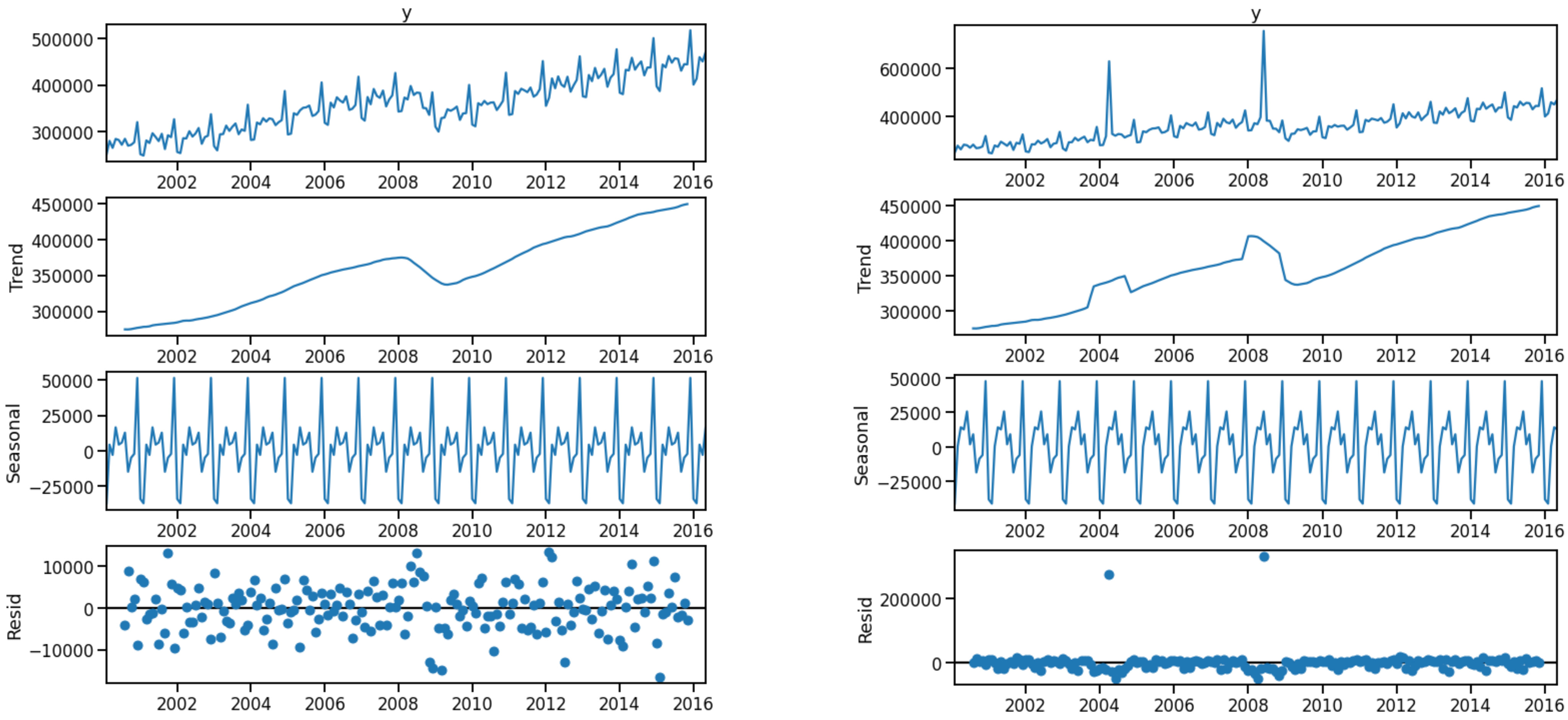
period : `int`, `optional`

Period of the series. Must be used if x is not a pandas object or if the index of x does not have a frequency. Overrides default periodicity of x if x is a pandas object with a timeseries index.

```python
res = seasonal_decompose(x=df['y'],
                         model='additive',
                         period=12)
res.plot();
```



Outliers will distort seasonal component



Discussion

- The seasonal component is a useful feature for forecasting as we will see later in the course

- Outliers can distort the trend and hence also the estimated seasonal component

- The classical approach assumes the seasonal component is fixed and does not change with time

Summary

Seasonality can be extracted by de-trending and averaging over a known seasonal index

This method is not robust to outliers and also assumes a fixed seasonal pattern