Multi-seasonal time series decomposition using MSTL

Time series decomposition

Contents

MSTL ALGORITHM

MSTL: What is it?

- **M**utliple **S**easonal-**T**rend decomposition using **L**oess (MSTL).

- A method to decompose a time series into a trend component, multiple seasonal components, and a residual component by repeatedly applying STL.

$$y_t = \underbrace{\hat{T}_t}_{\text{Trend component}} + \underbrace{\hat{S}_t^{(1)} + \hat{S}_t^{(2)} + \cdots + \hat{S}_t^{(N)}}_{\text{Seasonal components}} + \underbrace{\hat{R}_t}_{\text{Residual component}}$$

- MSTL, like STL, assumes the time series can be broken into an additive decomposition.

- Transform the time series (e.g., using Box Cox) if it is not additive.

MSTL: How does it work?

- User specifies number of seasonal components and their seasonal periods.
- Example: Hourly data and there is daily and weekly seasonality, then periods = (24, 24*7).

if number of seasonal components = 0

Extract trend using a smoothing method (Friedman's Supersmoother)*

Subtract trend to get residuals

if number of seasonal components > 0

Use STL to extract each seasonal component

Use final STL fit to extract trend and residuals

* Learn more about Friedman's Supersmoother and python implementation here and here.

MSTL: How does it work?

- User specifies number of seasonal components and their seasonal periods.
- Example: Hourly data and there is daily and weekly seasonality, then periods = (24, 24*7).

if number of seasonal components = 0

Extract trend using a smoothing method (Friedman's Supersmoother)*

Subtract trend to get residuals

if number of seasonal components > 0

Use STL to extract each seasonal component

Use final STL fit to extract trend and residuals

* Learn more about Friedman's Supersmoother and python implementation here and here.

MSTL: How does it work?

Step 1

- Iteratively extract each seasonal component using STL.

Step 2

- Refine each extracted seasonal component.

Step 3

- Extract the trend from the final STL fit.

Step 4

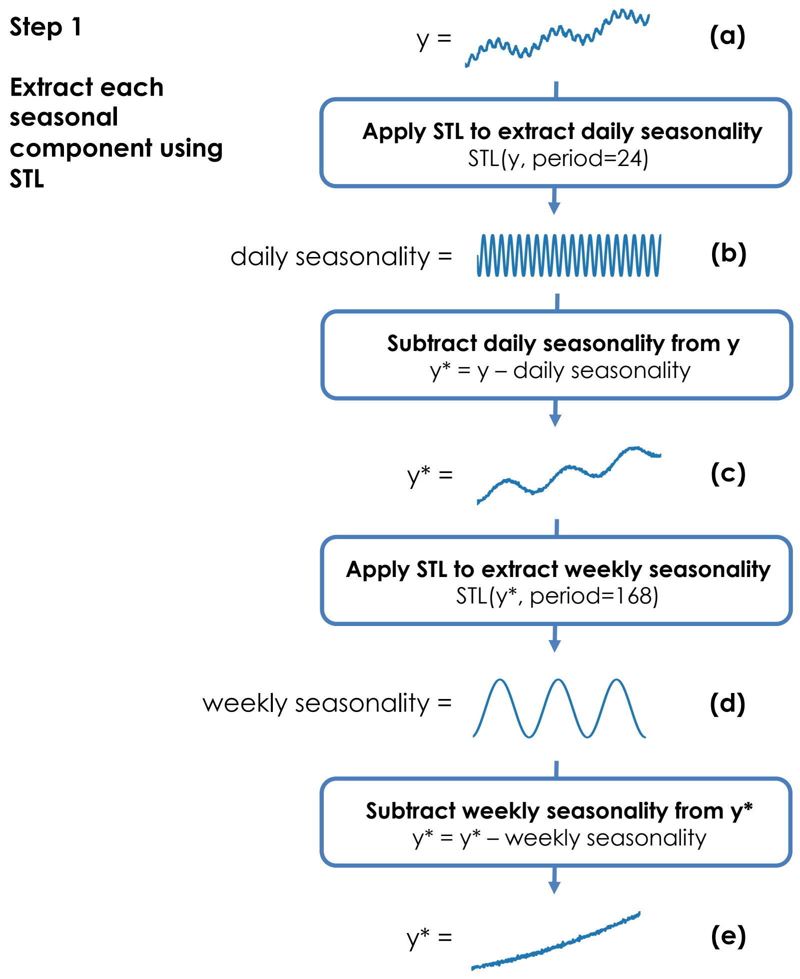
- Extract the residual component.

Step 1: extract seasonal components

- Iterate through each seasonal component starting from the shortest period (e.g., daily) to the longest period (e.g., yearly).

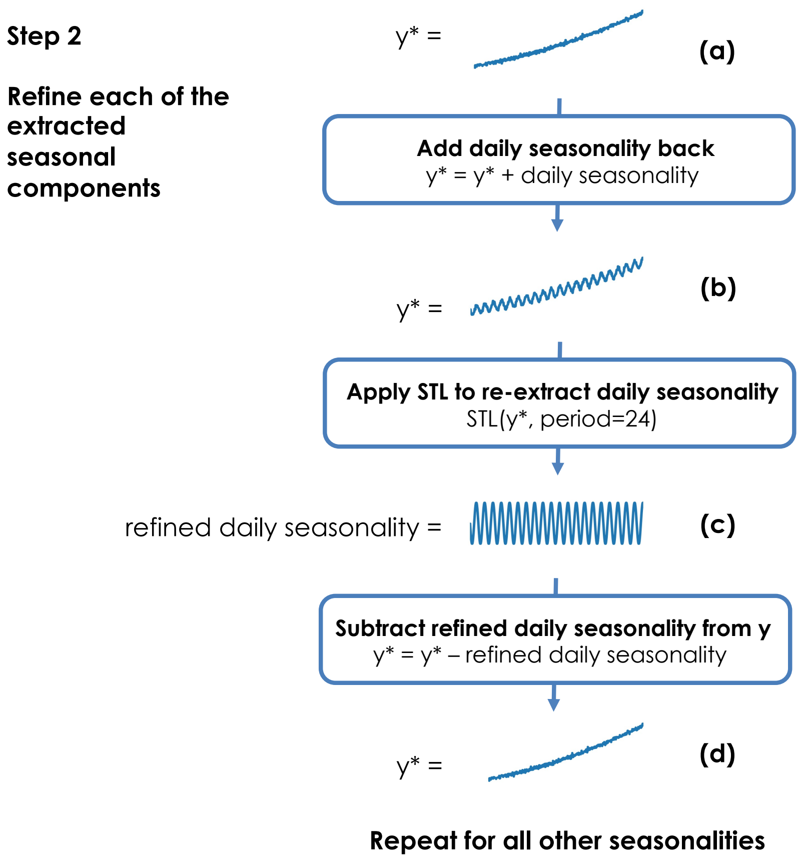
- On each iteration, we extract the seasonal component via STL and then subtract it from the time series.

- Continue until all seasonalities have been extracted.



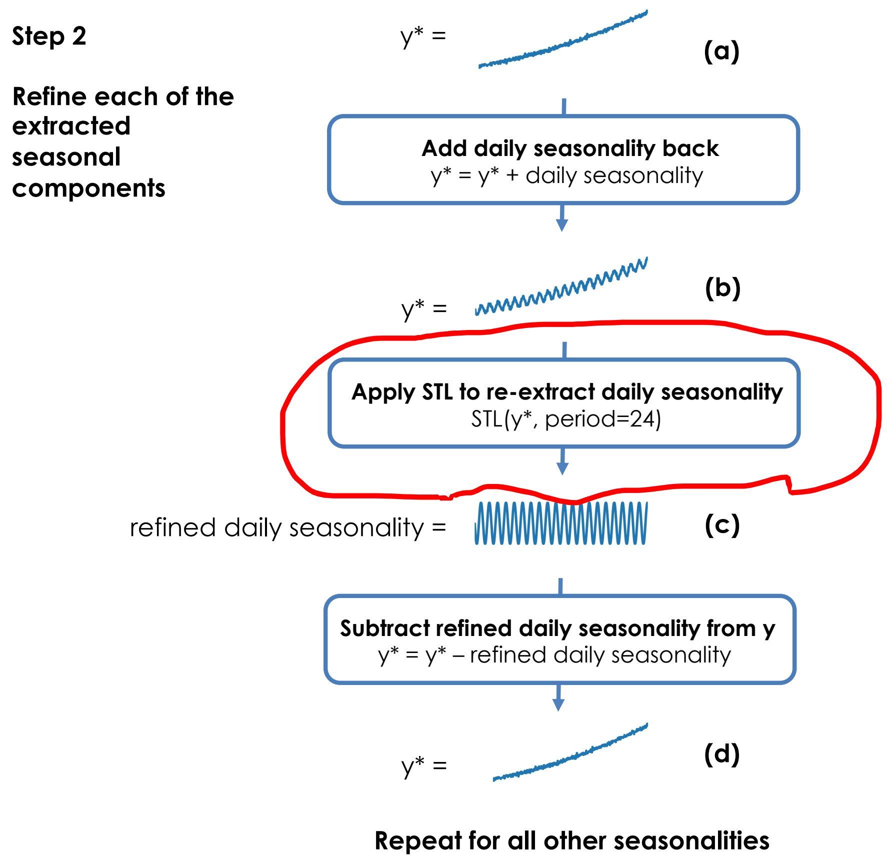
Step 2: refine each seasonal component

- We currently have an estimate for each seasonal component and a de-seasonalized time series.
- Iterate over each seasonal component again.
- Add each seasonal component back to the de-seasonalized time series.
- Extract the same seasonal component back using STL.
- Subtract this new estimate of the seasonal component from the time series.
- Do this for each seasonal component.



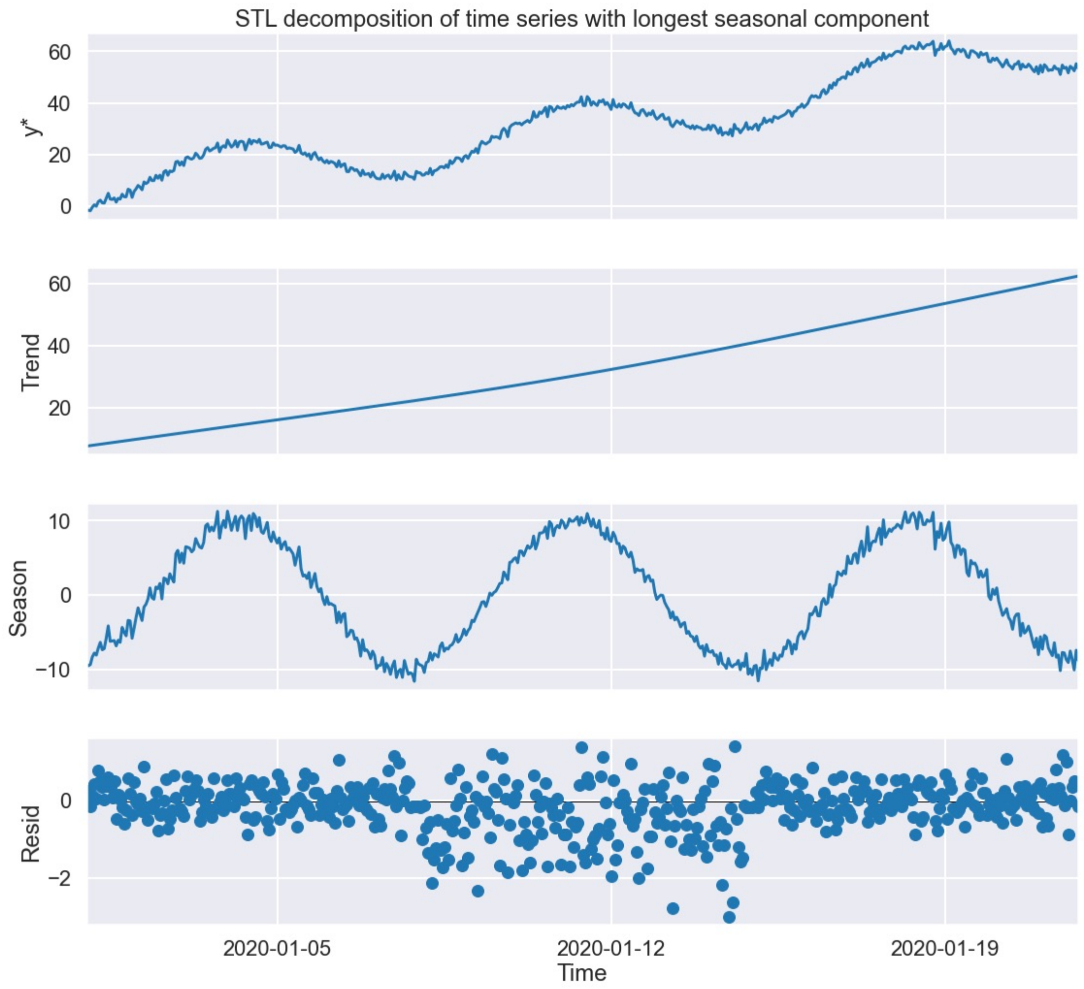
Step 3: extract the trend component

- Extract the trend component from the last STL fit used at end of step 2.
- This is the trend component for MSTL.



Step 3: extract the trend component

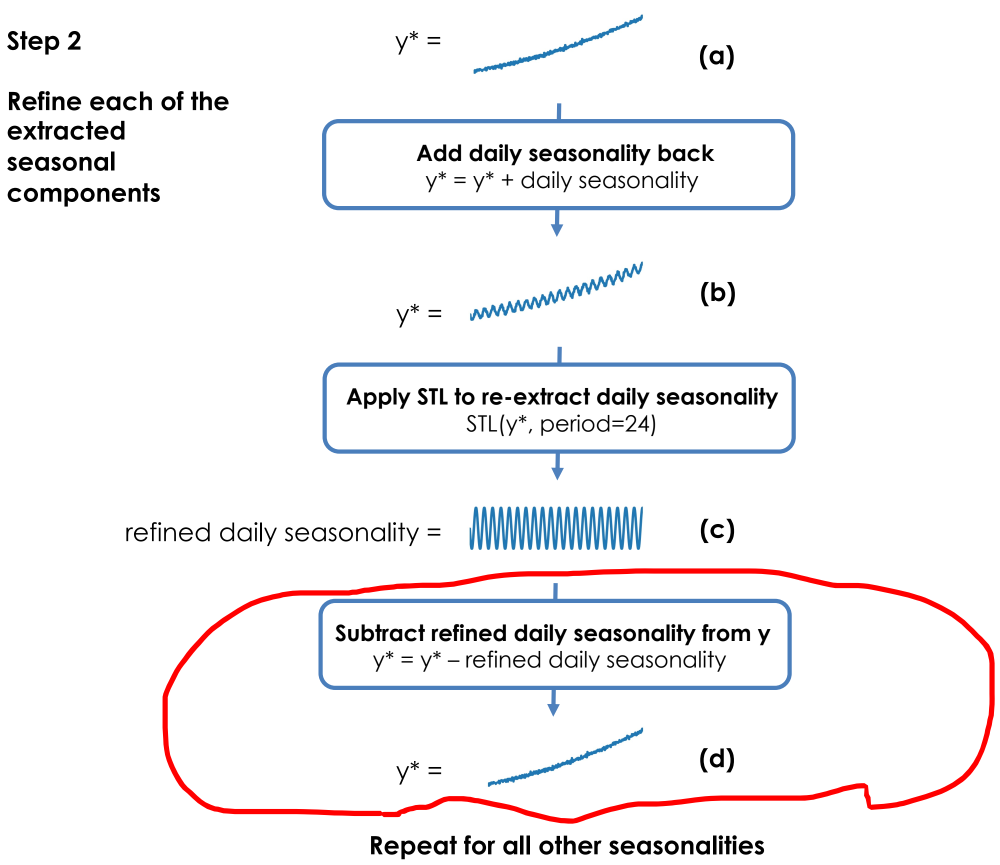
- Extract the trend component from the last STL fit used at end of step 2.
- This is the trend component for MSTL.



Step 4: Extract the residual component

- Subtract the trend component (from step 3) from the de-seasonalized time series at the end of step 2.

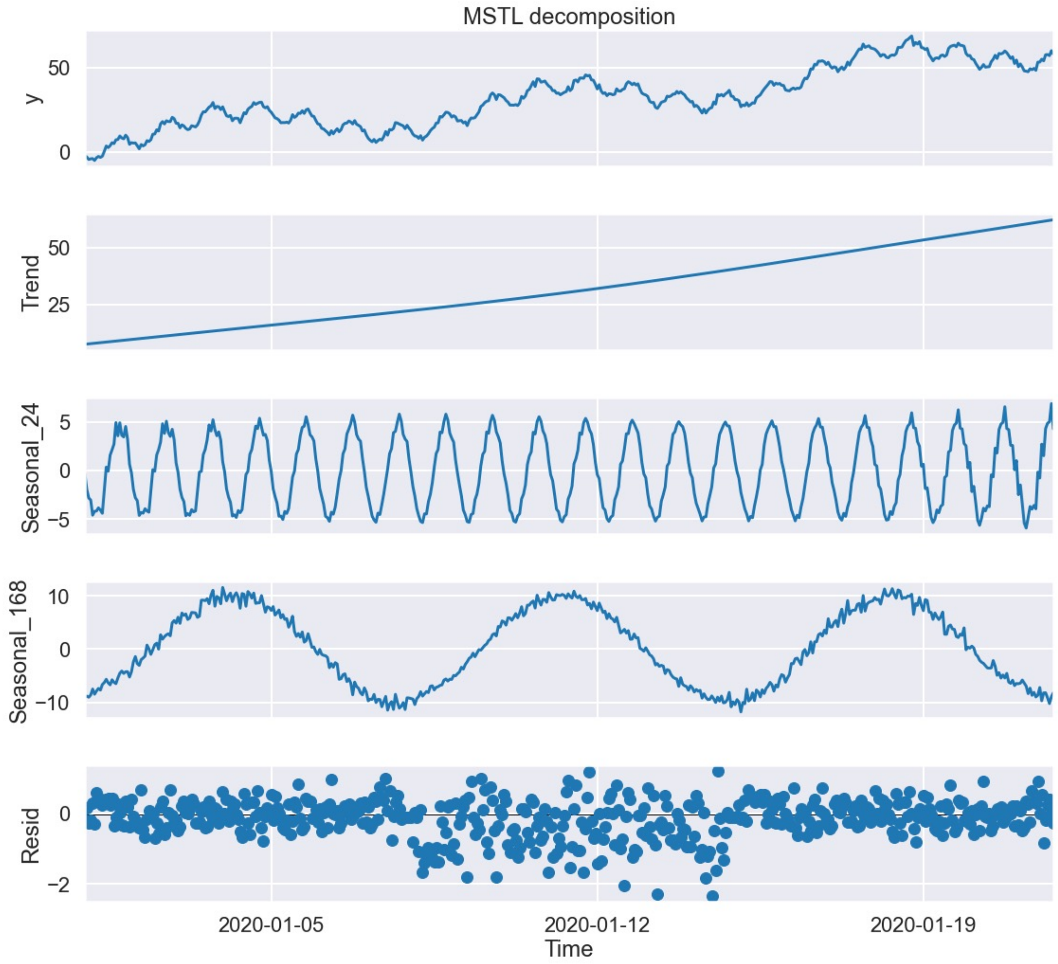
- Now we have:
  - Trend component
  - Daily seasonal component
  - Weekly seasonal component
  - Residual component



Step 4: Extract the residual component

- Subtract the trend component (from step 3) from the de-seasonalized time series at the end of step 2.

- Now we have:
  - Trend component
  - Daily seasonal component
  - Weekly seasonal component
  - Residual component

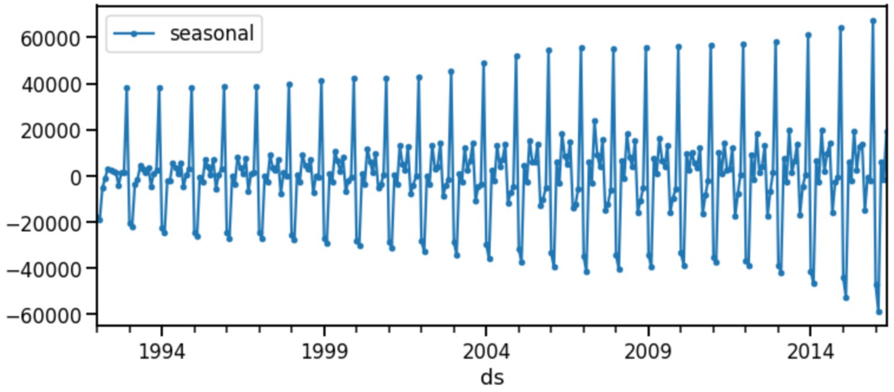


Reminder: STL most important parameters

- n_p (period): The seasonal period (e.g., for daily data and weekly seasonality this would be 7). Normally determined by the use case.

- n_s (seasonal): Determines smoothness and uniformity of seasonal component. Default value of 7 typically good enough.

n_s = 7



MSTL: Parameters

- **Periods**: The period of each seasonal component that we want to extract.
  - Determined by the use case (i.e., you know which seasonalities are important).
  - Use ACF or PACF (next section) to identify seasonalities.
  - Example: periods = (24, 24*7) for daily and weekly seasonality on hourly data.

- **Windows**: The smoothing parameter associated with each seasonal component.
  - Default values from experimentation on weekly and hourly time series [1].

Default value = 7 + 4 * i ; i = 0, 1, …, number of seasonal components

  - Reasoning: Longer seasonal components may require a larger smoothing window.
  - Example: windows = (7, 11)
  - Need to tune and inspect results otherwise, try the default values as a starting point.

[1] Bandara, K., Hyndman, R.J. and Bergmeir, C., 2021. MSTL: A Seasonal-Trend Decomposition Algorithm for Time Series with Multiple Seasonal Patterns. *arXiv preprint arXiv:2107.13462.*

MSTL: Parameters

- **STL Parameters**: All the other STL parameters can also be set, however, the default values are normally used.

Symbol	Statsmodels	Description	Typical value
n_i	inner_iter	Number of inner loops	2 or 3
d_s	seasonal_deg	Degree for LOESS for cycle-subseries (aka seasonal component)	1 or in rare cases 0 (see notebook)
n_l	low_pass	Window size for LOESS for low pass filter of cycle-subseries	n_p or the next largest odd integer
d_l	low_pass_deg	Degree for LOESS for low pass filter of cycle-subseries	1
n_T	trend	Window size for LOESS for Trend	1.5 to 2 x n_p
d_T	trend_deg	Degree for LOESS for Trend	1
n_o	outer_iter	Number of iterations in the outer loop	1 or 2
N/A	robust	A flag to indicate whether to use robustness weights	Set true if suspect outliers exist

Implementation

statsmodels.tsa.seasonal.MSTL

class statsmodels.tsa.seasonal.MSTL(*endog*, *periods=None*, *windows=None*, *lmbda=None*, *iterate=2*, *stl_kwargs=None*)[source]

Season-Trend decomposition using LOESS for multiple seasonalities.

Parameters

endog : array_like

Data to be decomposed. Must be squeezable to 1-d.

periods : {`int`, array_like, `None`}, **`optional`**

Periodicity of the seasonal components. If None and endog is a pandas Series or DataFrame, attempts to determine from endog. If endog is a ndarray, periods must be provided.

windows : {`int`, array_like, `None`}, **`optional`**

Length of the seasonal smoothers for each corresponding period. Must be an odd integer, and should normally be >= 7 (default). If None then default values determined using 7 + 4 * np.arange(1, n + 1, 1) where n is number of seasonal components.

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

```python
mstl = MSTL(timeseries["y"], # Time series
            periods=(24, 24 * 7), # Period of seasonal components
            stl_kwargs={"seasonal_deg": 0}) # Other STL parameters
res = mstl.fit() # Use .fit() to perform and
                 # return the decomposition

res.trend # access trend component
res.seasonal # access seasonal components
res.resid # access residual component
```

```python
res.seasonal.head()
```

	seasonal_24	seasonal_168
ds		
2012-01-01 00:00:00	-1694.799788	-165.282860
2012-01-01 01:00:00	-1602.267142	-231.770996
2012-01-01 02:00:00	-2205.330138	-260.793169
2012-01-01 03:00:00	-2455.880584	-387.594018
2012-01-01 04:00:00	-2372.200885	-656.522701

Summary

MSTL is a method to decompose multi-seasonal time series by repeatedly applying STL.

It is accurate, computationally efficient, outlier robust, models changing seasonality, and relatively simple.