# Components of a time series

1. Time series components

* Trend
* Seasonal
* Residual

1. Trend

* A long-term increase or decrease of the baseline of the time series
* Not necessarily linear

1. Seasonality

* A fixed periodic pattern in the time series
* Patterns relate to seasonal factors such as time of the year or day of week
* E.g.: Black Friday, Christmas, School holidays, etc.

1. Residual

* Everything leftover after subtracting the other components
* Useful to examine to determine how much structure of the underlying time series is captured in the trend and seasonality
* Large residuals -> there is still information to be extract through the trend or seasonal component or our methods for estimating the trend and seasonal component could be improved

1. Bringing it back together

* Either sum or multiply trend, seasonal, residual components together to recover the original series

1. Why is decomposition useful?

* EDA: to answer questions such as: ‘what was the impact of an ad campaign once we account for seasonality?’
* Pre-processing: useful for identifying outliers and can be used to impute outliers and missing data
* Feature engineering: derive features from the components to use as inputs in ML models
* Forecasting: forecast the components and aggregate to produce the final forecast

1. Decomposition is not always possible

* Not all time series can be easily broken down into components

# White noise

1. Motivation

* We will want to show how tools we introduce later (e.g., lag plots, correlation functions) behave for time series with various properties
* Two properties already covered in the course so far are trend and seasonality
* In this lecture we discuss a new property that can characterize the behavior of a time series – white noise

1. White noise

* Generate a time series, , by taking a random sample from any distribution of our choice (e.g. normal distribution with 0 mean and unit variance N(0, 1) ) at each time step t

A close-up of a math equation

Description automatically generated

A graph of white noise

Description automatically generated

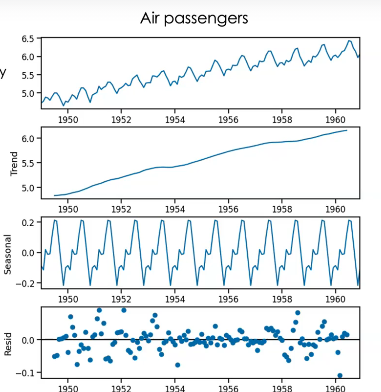
* Each value of is identically and independently distributed. There is no correlation between 2 points in time
* White noise
* If we sample from N(0, 1) -> Gaussian white noise
* Common to see this written as
* White noise has no predictive info in past values as there is no correlation at any 2 points in time
* It is useful to see how the tools we introduce later in the course behave when a time series is just white noise or contains white noise
* An important application of white noise is in the analysis of the residuals of a forecast or time series decomposition
* Residuals – difference between the actual and estimated values obtained through a forecast or time series decomposition

1. White noise applied to residuals

* Its common to think of time series as having a **non-random component** (e.g., trend, seasonality, autoregressive) and a **noise component**

Where is modeled as white noise to represent random effects for example from:

* Imperfect sensors
* Random fluctuations in the underlying process
* A forecast or time series decomposition, can be thought of as estimating the non-random component x(t)
* Assume we estimate perfectly, , then the residual would be equal to white noise:
* If the residuals of our forecast or decomposition look like white noise it means there is no more predictive information to be extracted from the data
* Some simple checks to see if your residuals are like white noise:
* Mean is zero
* Mean and variance don’t change over time – no trend or seasonality in residuals
* No autocorrelation – no correlation between a time point and any other in the past



* This decomposition produces residuals which:
* Have 0 mean
* The mean doesn’t change in time
* The variance does change in time
* This means that there is still some non white noise component left in the time series which could be caused by:
  + Imperfect extraction of trend and seasonality
  + An additional component exists which is not caught be trend or seasonality (e.g., an autoregressive component)

1. Summary

* A white noise time series is one generated by repeatedly sampling from any distribution where each sample is independent
* White noise has no correlation between any 2 time points – past values cannot be used to predict the future
* If the residuals of a decomposition or forecast look like white noise – there is no more info left to extract and we did a good job

# Additive and multiplicative decompositions

1. Time series decomposition

* Aim: decompose a time series into components
* Choice / assumption:
* How to choose?
* Domain knowledge around data generating process
* Looking at certain behaviors of the time series

1. Multiplicative decomposition – e.g.

* T-shirt sales of a growing online clothing brand
* Visitors to website increasing over time ->
* The product has seasonal demand. Each customer has higher prob, p(t), of purchasing during summer than winter

Where:

1. Multiplicative decomposition

A graph of a graph

Description automatically generated with medium confidence

* Magnitude of the seasonal fluctuations is proportional to the level of the time series

1. Additive decomposition - e.g.

* Air pollution on a particular road, pollution(t)
* There is some background level of air pollution, B(t) – long term trend
* Traffic from vehicles can add air pollution, T(t) – daily seasonality
* Traffic is busier during peak hours, hence, there is a daily seasonality
* So we expect:

1. Additive decomposition

A diagram of a graph

Description automatically generated with medium confidence

* The magnitude of the seasonal fluctuations does not change with the level of the time series

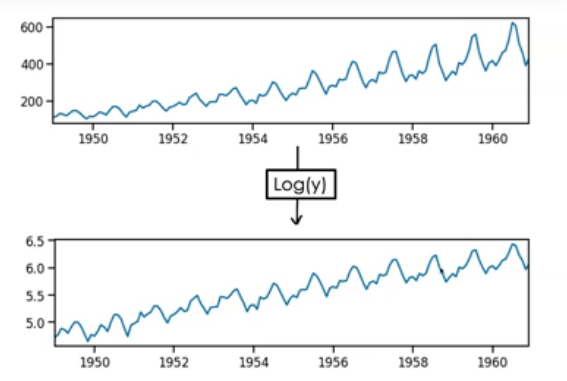
1. Summary

* Time series can be decomposed into a multiplicative or additive decomposition/model
* If the seasonal variation is proportional to the level of the time series -> multiplicative decomposition

# Log transform

1. Log transform

* The transform
* The log transform can be used to stabilize the variance of a time series
* Some forecasting models perform better if the variance of the time series does not increase with time (e.g., ARIMA)
* If the time series goes to 0, y = 0, then is undefined
* Can overcome by adjusting the transform to



1. Converting multiplicative to additive

* A multiplicative decomposition can be converted into an additive decomposition by taking the log of the target y(t)
* Take the log of y(t) for the multiplicative case:
* **If the underlying time series is multiplicative then log of series is additive**
* Provides intuition as to why log transform can stabilize the variance
* This technique is useful because some time series decomposition method (e.g., STL decompositions) only handle the additive case
* So we require log transforming the data first before using certain forecasting and decomposition methods

1. Summary

* The log transform can be used to stabilize the variance of a time series
* The log transform can convert a multiplicative time series to an additive one

# Box Cox transform

1. Motivation

* Some forecasting and decomposition methods perform better if the variance of the time series does not change with the level of the time series (e.g., ARIMA)
* The log transform does not always stabilize the variance of a time series – depends on the time series

A screenshot of a graph

Description automatically generated

1. What kind of transforms are useful?

* What kind of transform of our time series, could be useful in stabilizing the variance? Depends on the time series
* To stabilize the variance we want the transformation to remove the interaction between the trend and seasonality or noise term so we can write them additively
* In our toy time series we had:
* We can think of this as:
* A better transform would be using the square root

A graph of a toy

Description automatically generated with medium confidence

* An example of what is called a power transform. We raised the original variable to some power
* A more general way of writing power transforms are
* Sometimes a power transform can be better at stabilizing the variance than a log transform – depends on the time series
* The Box Cox transform combines both a log transform and a power transform

1. Box Cox transform

* Defined as:
* Different values of – different kinds of transform

A table with mathematical equations

Description automatically generated

* In practice, is typically set between -5 and 5
* y must be positive, if the data has any negative values then it can be transformed to be positive by adding a constant beforehand to the whole time series
* A good value of makes the variance the same size across the time series
* How do we pick a good value for ?
* Try different values, plot the data and check visually that the variance is nearly constant

A graph of a plane

Description automatically generated with medium confidence

* Use a method that automatically selects that optimizes on some criteria
* Maximum likelihood (MLE) – pick that makes transformed data look the most normally distributed
* Guerro: Picks that tries to make the variance constant across the time series

A graph of a graph of a method

Description automatically generated with medium confidence

1. Box Cox implementation in Scipy

A screenshot of a computer program

Description automatically generated

1. Box Cox implementation in sktime – can specify method as Guerrero and seasonal period

A screenshot of a computer

Description automatically generated

1. Summary

* Forecasting and decomposition methods sometimes work better if the variance is stable across the whole time series
* Log and power transforms can help stabilize the variance across the time series
* A Box Cox transform combines the log and power transform into a single method with a parameter
* The best is the one that makes the variance stable across the time series
* Multiple methods exist to automatically select the best . However, they don’t always agree and manual sense checking is advised

# Box Cox transform: Guerrero method

1. Box Cox recap
2. Coefficient of variation

* The coef of variation is a **scaled measure of variability** of a dataset
* Coef of variation:
* Allows us to **compare the variability across datasets** on different scales
* E.g.,

A table with numbers and text

Description automatically generated

1. Guerrero method

* Original paper: “Time-series analysis supported by Power Transformations”
* We want to pick so that the variance of is constant
* Guerrero showed that this requirement implies that:
* Try a range of and check for which value is the most constant over time
* In practice, we have one observation at each t. How do we calculate ?
* Calculate at intervals!
* Split the time series into H evenly sized buckets (subseries), labeled by h
* Compute mean and standard deviation within each subseries
* Compute for each subseries
* How do we measure how constant is across the time series?
* Use the coef of variation of across all subseries,
* If is low -> is more constant across the time series
* Compute at multiple values of between -5 and 5
* Pick which minimizes
* This value creates a time series where is the most constant across time
* Which implies that it’s the best to use to cause the variance of to be constant
* Main parameter is the number of subseries
* If the data has seasonality -> split the subseries by the seasonal period (e.g., 1 subseries for each year if monthly data)
* If no seasonality -> then split the time series into consecutive groups of size 2 to minimize loss of info caused by grouping

1. Why use the Guerrero method?

* Makes no assumptions about the distribution of data
* Directly tries to stabilize the variance across the time series
* More relevant for our time series tasks (i.e., forecasting and decomposition)

1. Implementation in sktime

A screenshot of a computer code

Description automatically generated

1. Summary

* Guerrero method selects that makes the variance of constant by minimizing the coef of variation

# Moving average

1. Example

* Consider window of size 3
* Computer at center of window
* Compute mean
* Move window and iterate
* 3-MA is a shorter time series

A table with numbers and numbers

Description automatically generated

1. Moving average

* Moving average of order m (m-MA)

Where m = 2k+1 is the size of the window where k data points either side of t are included in the average

* Each data point in the window receives equal weight and the window is symmetric

1. Even window size

* Often the window size is selected to be the same as the seasonality to smooth out seasonal variation
* E.g.: monthly data, yearly seasonality T = 12
* With an even window, where do we compute the mean value?
* An odd window size would double count specific months (e.g., counting January twice)
* E.g.

A screenshot of a table

Description automatically generated

* Problem: where should the average value go as there is no obvious center?
* We could think of the value belonging half way between 2 rows
* Apply another moving average of window size 2 to the 4-MA (2x4-MA)
* This gives a symmetric window where the weights still sum to one
* The edges of the window share half the weight, this mitigates the double counting discussed earlier
* Any even order centered MA can be dealt with by applying an additional 2-MA
* E.g.: If we wanted m=6, we can compute a 2x6-MA to get



1. Moving average implementation

* Pandas.DataFrame.rolling
* Odd window size

A screenshot of a computer

Description automatically generated

* Even window size

A computer screen shot of a computer code

Description automatically generated

1. Summary

* Moving average computes the mean of the data over a window across a time series
* The window size, m defines the order of the moving average m-MA
* An even ordered centered moving average can be obtained by applying an additional 2-MA

# Classical decomposition: trend

1. How can we extract the trend

* Moving average
* Window size?
* Assume time series has seasonality with period T (e.g., T=12 for monthly data with a yearly seasonality)
* If T is odd -> T-MA
* If T is even -> 2 x T-MA
* This will smooth over the seasonality
* What if no obvious seasonality?
* Visually inspect different window sizes to ensure that the main trend is captured
  + Too small a window -> capture noise and seasonality rather than overall trend
  + Too large a window-> over smooth variations which might be included in the trend

1. Limitations

* Moving averages are distorted by outliers

A graph showing the growth of a number of people

Description automatically generated with medium confidence

* Trend will miss data points for the first and last few data points
* Rapid changes in trend tend to be oversmoothed

A graph with blue and orange lines

Description automatically generated

* Better methods exist for decomposition
* Useful as inspiration for feature engineering later on

1. Summary

* Moving averages can be used to extract trend
* Due to limitations, other methods are preferred for the purpose of time series decomposition

# Classical decomposition: seasonality

1. How can we extract the seasonality?

A diagram of a graph

Description automatically generated with medium confidence

* Identify order of seasonality T

A graph showing a number of years

Description automatically generated with medium confidence

* Compute trend using T-MA (if odd) or 2xT-MA (if even)
* De-trend the data
  + If additive:
  + If multiplicative:

A graph showing the time line

Description automatically generated with medium confidence

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

A line graph with numbers and a line

Description automatically generated with medium confidence

* Comments
* The seasonal pattern is fixed each year
* We can repeat the seasonal pattern each year to get
* We can plot alongside

1. Implementation

A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

* Implementation in statsmodels

A screenshot of a computer code

Description automatically generated

A screenshot of a graph

Description automatically generated

1. Outliers will distort seasonal component

A close-up of a graph

Description automatically generated

1. 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

1. 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

# LOWESS

1. Limitations of moving averages

* Not robust to outliers
* Missing data at the edges
* Oversmooths rapid changes in the trend
* Order of moving average was set by seasonal period

1. LOWESS

* Robust Locally Weighted Regression and Smoothing Scatterplots

1. The main idea

* Want to compute a smooth curve to a scatterplot

A graph with blue dots

Description automatically generated

* At x consider a window which captures a fraction f of the data

A graph of a function

Description automatically generated

* Fit a **weighted** robust linear regression to this subset of the data

A diagram of a function

Description automatically generated

* The LOWESS curve at x is given by the linear regression

A diagram of a function

Description automatically generated

* Evaluate the same process across many x values to obtain a smooth fit

A line graph with blue dots

Description automatically generated

1. Weight function for LOWESS

* Weights given by tricube function:
* In practice, w(x) is rescaled when evaluating LOWESS at point to fit the window size determined by f. So we compute

Where N is the number of data points and f is the fraction of the dataset that determines the window size

* This ensures the weight function is centered at the point of interest and the function goes to zero at the edge of the window

A diagram of a function

Description automatically generated

1. Robust regression:

* Fit a linear regression multiple time
* On each iteration reweight the data by residuals of the previous fit such that less weight is given to high residual data points
* Outliers produce large residuals. Hence, by reweighting the data we can minimize their impact on the fit
* Steps
* Fit a weighted linear regression:
* Compute residuals:
* Compute weights: , where s is median

A blue line graph with numbers

Description automatically generated

* Refit linear regression with weights:
* Repeat t times

1. LOWESS

* The LOWESS fit can be used as an estimate of the trend of a time series

A graph of a number of objects

Description automatically generated with medium confidence

# LOWESS – practice

1. LOWESS

* LOWESS curve at x is given by the weighted robust linear regression

1. Parameters

* : Fraction of data for window size
* Determines smoothness of the fit
* Good value depends on data
* Trying to have as smooth a fit without distorting patterns in the data
* Can set visually or using cross-validation

A screenshot of a graph

Description automatically generated

* : number of iterations for robust regression
* Ensures robustness to outliers
* Typically set to 2 or 3

1. Implementation

* Statmodels

A screenshot of a computer

Description automatically generated

1. Discussion

* Pros
* Robust to outliers
* No missing data at the edges
* Interpretable
* A good f will avoid over smoothing the data
* No assumptions made about the data
* Cons:
* Slow fit on large datasets
* Requires access to entire dataset to evaluate curve at any point
* Selecting f often requires manual inspection

1. Summary

* LOWESS is a method to fit a smooth curve to a scatter plot
* LOWESS can be used to extract the trend term in the time series

# LOWESS vs. LOESS

1. LOWESS vs LOESS

* LOWESS – locally weighted scatterplot smoothing
* LOESS – Locally estimated scatterplot smoothing
* Similarity
* Fits a locally **weighted** curve using regression
* Uses tri-cubic weight function
* Uses bi-square weights for robustness (depends on implementation)

A diagram of a function

Description automatically generated

* Difference
* LOWESS:
  + Uses **linear regression**
  + Fits a curve to univariate data (y is 1-D)
* LOESS:
  + Uses **polynomial regression**
  + Fits a surface to multivariate data (y is N-D)

1. LOESS fits a local polynomial regression

* Previously showed that LOWESS uses a local, robust and weighted linear regression
* LOESS uses a polynomial regression
* Most applications use d = 1 and in some rarer cases d = 2 or d = 0

A screenshot of a graph

Description automatically generated

1. Summary

* LOWESS and LOESS fit a curve to local partitions of the data
* LOWESS fit a line and LOESS fits a polynomial
* LOWESS only works with 1-D data and LOESS can work with N-D

# STL overview

1. Recap: Classical decomposition – seasonality

* Identify order of seasonality T
* Compute trend using T-MA (if odd) or 2 x T-MA (if even)
* De-trend the data
* If additive:
* If multiplicative:
* Average the detrended data over each seasonal index to remove noise (e.g., for monthly data average all the May months)

1. Limitations of moving averages

* Not robust to outliers
* Missing data at the edges
* Over-smooths rapid changes in trend
* The seasonal component does not change in time (e.g., seasonal component for every Jan is the same)

1. STL – A seasonal-trend decomposition procedure based on LOESS

* **Assume**
* Want a way to calculate each individual component
* STL does this using LOESS
* The main idea

A diagram of a complex trend

Description automatically generated

1. Inner loop summary - Iterate

* **Detrend the data**:
* For the first iteration assume trend(t) = 0
* **Extract seasonal(t) using LOESS** on a set of time series derived from y\_detrend(t) called cycle-subseries
* **Extract trend(t)** by subtracting seasonal(t) from y(t) and smoothing using LOESS

1. Outer loop summary – Iterate

* **Extract seasonal(t) and trend(t)** from inner loop
* **Compute residuals**:
* **Compute weights from residuals** to pass to LOESS in the inner loop. This is to down-weight outliers

1. Implementation

A screenshot of a computer

Description automatically generated

1. Example with outliers

A graph of different types of graphs

Description automatically generated with medium confidence

1. Most important parameters

* (**seasonal**): Determines smoothness and uniformity of seasonal component. Default value of 7 typically good enough
* (**period**): The seasonal period (e.g., for daily data and weekly seasonality this would be 7). Normally determined by the use case

1. How to handle multiplicative time series

* STL assumes the time series is additive
* If the time series is multiplicative use log transform to make it additive
* Transform back to original scale by taking inverse log transform

1. Summary

* STL extracts the seasonality and trend iteratively using LOESS
* STL is robust to outliers
* There are 2 main params to set in practice, the remaining default params are normally sufficient
* The seasonal component can vary in time and is not necessarily periodic

# STL theory – LOESS and Cycle-subseries

1. LOESS parameters in STL

|  |  |
| --- | --- |
| LOESS | STL |
| : Fraction of data for window size   * Determines smoothness of data | : Number of data points in window size   * Must be odd so can have centered window |
| : Number of iterations for robust regression | : Number of outer loop iterations   * In a single outer loop we compute robustness weights to input into LOESS in the inner loop * Ensures robustness to outliers |
| : Degree of polynomial to use in LOESS   * Most applications use d = 1 and in some rarer cases d = 2 and d = 0 |  |

1. Cycle subseries

* : Number of data points in one seasonal cycle
* E.g.: monthly data with yearly seasonality ->
* Cycle-subseries is the time series formed from looking at the value of each period withing a seasonal cycle over time
* E.g. the time series of all Jan values over multiple years
* 12 cycle subseries in total for each month

A screenshot of a graph

Description automatically generated

* Note relation to classical decomposition for seasonality:
* De-trend original time series
* Average over seasonal index

A graph and chart on a white background

Description automatically generated

* This view of the data motivates a **seasonality that can change with time** (i.e., each year)
* Useful to enable STL to extract a time varying seasonal component
* If seasonal cycle has periods -> there are cycle subseries
* In this example hence 12 cycle subseries – one for each month
* This view allows us to think about how seasonality changes with time and how we model it
* E.g. Simple average to get constant seasonality over time like in classical decomposition
* Can fit smoother curves instead to model the seasonality

1. Summary

* LOESS params used in STL are different from LOWESS
* Cycle-subseries is a time series of one of the seasonal indexes
* Cycle-subseries is a useful way to look at the data to understand and model seasonality

# STL theory – The inner loop

1. Inner loop summary
2. Extract seasonal component

* Step 1: apply LOESS to cycle subseries
* User specified (period in statsmodels): number of periods in a seasonal cycle
* Compute all the cycle-subseries for y\_detrend(t)
* Apply LOESS to each cycle-subseries with smoothing param specified by the user
* Denote output as
* Step 2: Detrend the cycle subseries
* Apply a series of moving averages to each smooth cycle-subseries
* Apply LOESS, with smoothing param to each smooth cycle subseries
* Acts as a ‘low-pass filter’ meaning that represents any long term trend in the cycle subseries
* Detrend the smoothed cycle subseries -> this gives seasonality
* As the seasonal component is derived from cycle subseries -> it can vary in time

A group of graphs showing different types of data

Description automatically generated with medium confidence

* Seasonal component isn’t exactly periodic over each cycle

1. Extract trend component

* De-seasonalize the original series
* Smooth using LOESS, with smoothing param , to give trend estimate:

1. Inner loop summary

A close-up of a text

Description automatically generated

* Robustness to outliers
* Remember in LOESS that residuals are used to re-weight data during regression
* Outer loop provides these weights for all uses of LOESS in inner loop
* Parameters from inner loop

|  |  |  |  |
| --- | --- | --- | --- |
| **Symbol** | **Statsmodels** | **Description** | **Typical value** |
|  | Inner\_iter | Number of inner loops | 2 or 3 |
|  | period | Number of periods in a seasonal cycle | Depends on data |
|  | seasonal | Window size for LOESS for cycle subseries (seasonal component) | Depends on data |
|  | seasonal\_deg | Degree for LOESS for cycle subseries | 1 or in rarer cases 0 |
|  | low\_pass | Window size for LOESS for low pass filter of cycle subseries | or the next largest odd integer |
|  | Low\_pass\_deg | Degree for LOESS for low pass filter of cycle subseries | 1 |
|  | trend | Window size for LOESS for trend | 1.5 to 2 x |
|  | Trend\_deg | Degree for LOESS for Trend | 1 |

# STL theory – the outer loop

1. Outer loop summary

A close-up of a diagram

Description automatically generated

1. Compute weights from residuals

* Compute residuals:
* Compute robustness weights: where

A black numbers on a white background

Description automatically generated

* Use as robustness weights in LOESS in the inner loop for:
* Cycle subseries smoothing
* Trend smoothing
* For the first iteration set

1. Parameters from outer loop

|  |  |  |  |
| --- | --- | --- | --- |
| **Symbol** | **Statsmodels** | **Description** | **Typical value** |
|  | outer\_iter | Number of iterations in the outer loop | 2 or 3 |
| N/A | robust | A flag to indicate whether to use robustness weights | Set true if suspect outliers exist |

Summary

1. What is time series decomposition?

* Breaking a time series down into components: trend, seasonality, residual
* Additive:
* Multiplicative:

1. Why is decomposition useful?

* Exploratory data analysis
* Pre-processing: identifying outliers and impute outliers/missing data
* Feature engineering: derive features from the components to use as inputs in ML models
* Forecasting: forecast the components and aggregate to produce the final forecast

1. Decomposition is not always possible

* Not all time series can be easily broken down into components

1. Classical decomposition

* Identify order of seasonality T
* Compute trend using moving averages. T-MA if odd or 2xT-MA if even
* De-trend the data
* Average the de-trended data over each seasonal index to remove noise
* Cycle-subseries
* Limitations: outliers, edges, rapid changes

1. LOWESS

* Fit a weighted robust linear regression to this subset of data
* Evaluate the same process across many x values to obtain a smooth fit

1. Seasonal and trend decomposition using LOWESS (STL)

A pink card with blue and black text

Description automatically generated

* Inner loop
* De-trend the data
* Extract seasonal(t) using LOWESS on cycle subseries
* Extract trend
* Example with outliers

1. What if you have long or many time series

* LOWESS and STL are slow to compute as they require fitting a regression model and making predictions
* For long time series it may be faster to use classical decomposition and accept trade off with impact from outliers
* Alternatively using moving median or other filtering methods for trend extraction
* Depends on available computational resource and use case

1. Regular vs. irregularly spaced time series

* Many methods assume evenly spaced (i.e. regular) time series (e.g. STL)
* Some methods can handle irregularly spaced time series (e.g. LOWESS)
* Many methods assume regular times series
* Potential solution – convert the irregularly spaced time series into a regular one (e.g. through aggregating)

1. Summary