Decomposition

Handout

▼ Time Series Components

Additive Decomposition

$$y_t = S_t + T_t + R_t$$

- S_t - Seasonal Component
- T_t - Trend-Cycle Component
- R_t - Remainder Component

▼ **Purpose**

When seasonal variation doesn't change with the level of the time series

▼ **Note**

- We assume that the seasonal effect does not change over the time series.

  i.e. $S_t = S_{t+12}$ (where the period is 12)

  $S_t = S_{t+4}$ (where the period is 4)

Multiplicative Decomposition

$$y_t = S_t \times T_t \times R_t$$

▼ **Purpose**

When the trend cycle seems to vary proportionally to the level of the time series

- More prevalent with economic series

▼ **Alternative**

Note that we can do a log transform for this and it will become an **additive model**

$$\log y_t = \log S_t + \log T_t + \log R_t$$

STL Model

Seasonal and **T**rend decomposition using **L**oess

▼ **Loess**

Method for estimating nonlinear relationships

▼ **Benefits of the STL Model**

- Can handle any type of seasonality, not only monthly and quarterly data
- Seasonal component is allowed to change over time, and the rate can be controlled by the user
- Smoothness of the trend cycle can be controlled by the user
- Robust to outliers (i.e. the user can specify a robust decomposition)
  - Ignores the effect of the outliers
  - Therefore, we could see a large variation within our remainders

▼ **Disadvantages**

- Does not handle trading day or calendar variation automatically, it only provides facilities to additive decompositions

▼ **How to get a STL model**

Pass the data into the `model()`

`model(stl = STL(<var>))`

Within the `components()` function

- `trend` → T_t
- `seasonal_year` - S_t

- `remainder` - R_t
- `season_adjust` - $y_t - \hat{S}_t$
    - Useful when we want to observe the trend without the estimated seasonal effects

R Tools

`components()` - The object returned contains the estimates for the trend-cycle, seasonal, remainder

To determine which component provides the greatest variation

- `autoplot()` the result from `components()`
- The grey bar for each of the components can help us to analyze how much each of the components vary with respect to the scale of the original value
    - We can see which one varies the most with respect to the grey bar and the one that varies the most is the “driving force”. If the grey bar is very small then it should seem to be the one that varies the most since it will look like it keeps going out of the grey bar

Decomposition Features

Note that F_t and F_S could be close to 1. They are both compared to the remainder term and it is comparing between each other.

▼ **Strength of Trend**

$$F_t = \max\left(0, 1 - \frac{Var(R_t)}{Var(T_t + R_t)}\right)$$

Note

- If the value of F_t is small, it will indicate that $Var(R_t) \approx Var(T_t + R_t)$

    - This implies that trend is not a “**driving force**” compared to residual noise
    - This is because the variance of the seasonally adjusted trend is similar to the remainder term
- If the value of F_t is close to 1, it indicates that trend is very important compared to residual noise
    - This means that the variance of the trend contributes to a bigger variance and therefore it will give a larger denominator and therefore, $F_t \approx 1$

▼ **Strength of Seasonality**

$$F_S = \max\left(0, 1 - \frac{Var(R_t)}{Var(S_t + R_t)}\right)$$

Note

- If the value of F_S is small, it will indicate that $Var(R_t) \approx Var(S_t + R_t)$
    - This implies that seasonality is not a “**driving force**” compared to residual noise
- If the value of F_S is close to 1, it indicates that seasonality is very important compared to residual noise

▼ **Seasonal Features**

Note that the following features are derived from the `features(feature_set=c("seasonal"))`

- `seasonal_peak_year` - Indicates the timing of the peaks - which month or quarter contains the largest seasonal component. This tells us something about the nature of the seasonality.
    - However, note that this is taken from the season with the largest average seasonal component across the whole time period
    - Therefore, it may not be indicative of which month has the **highest** seasonal component since it is averaged out.
- `seasonal_trough_year` - Indicates the timing of the troughs - which month or quarter contains the smallest seasonal component.
    - However, note that this is taken from the season with the smallest average seasonal component across the whole time period.

    - Therefore, it may not be indicative of which month has the **lowest** seasonal component since it is averaged out.
- `spikiness` - Measures the prevalence of spikes in the remainder component R_t of the STL decomposition. It is the variance of the leave-one-out variances of R_t
- `linearity` - Measures the linearity of the trend component of the STL decomposition. It is based on the coefficient of a linear regression applied to the trend component
    - This essentially measures how much of a straight line this has.
    - If it has a **positive** linearity it means that there is a **positive trend** upwards
    - If it is has a **negative** linearity it means that there is a **negative trend** downwards
- `curvature` - Measures the curvature of the trend component of the STL decomposition. It is based on the coefficient from an orthogonal quadratic regression applied to the trend component
    - This essentially measures how much of a curve the trend has.
    - If there is a **positive** curvature, we can expect a **concave** shape. We could say more about this with information about the trend (i.e. If the trend is increasing, we can say that the trend is increasing at an increasing rate)
    - If there is a **negative** curvature, we can expect a **convex** shape. We could say more about this with information about the trend (i.e. If the trend is increasing, we can say that the trend is increasing at a decreasing rate)
- `stl_e_acf1` - First autocorrelation coefficient of the remainder series
- `stl_e_acf10` - Sum of squares of the first ten autocorrelation coefficients of the remainder series

▼ Moving Average Filters

Estimates the trend-cycle

The easiest way to estimate the trend-cycle is to make use of a moving average filter

▼ Moving Average Smoother

We can see that when we increase the window range:

1. We will miss observations at the ends of our time series
2. The trend estimate will get smoother since we are taking in more points

MA Order Consideration

1. Larger order → Smoother, flatter curve
2. Larger order → More points lost at the ends
3. If order = length of season or cycle → Removes pattern

Case when m is odd

$$\sum_{j=-k}^{k} \frac{1}{m} y_{t+j}$$

$m = 2k + 1$ is an odd number. This is an m-MA

We are taking t to be the center and look k before and k after

Case when m is even

$$\sum_{j=-(k-1)}^{k} \frac{1}{m} y_{t+j}$$

Note that for the even case, we will take one more observation from the future instead by convention.

▼ Symmetrizing the even case

First we will apply the $m - MA$ to the time series first with one more from the future first. Then we will apply a $2 - MA$ to the $m - MA$ time series after which will allow it to be symmetric.

Note that when we do the $2-MA$, we are using the past previous data instead of the future one instead now

> 💡 **Why does this work?**
>
> By doing the 2-MA afterwards, we taking in the missing part of the previous part and balancing out the effect from taking 1 more observation in the future.
>
> However, note that the weights will no longer be equal but will be symmetric

1. Apply $m-MA$ with 1 more in the future to the original time series $\{y_t^R\}$
2. Apply $m-MA$ with 1 more in the past to the original time series $\{y_t^L\}$
3. Take the average $y_t' = \frac{1}{2}(y_t^R + y_t^L)$

▼ Weighted Moving Averages

In the middle there are heavier weights and it dies off at the ends

$$\hat{T} = \sum_{j=-k}^{k} a_j y_{t+j}$$

where $k = \frac{m-1}{2}$

Note

- This yields smoother estimates since the weights are slowly increased and slowly decreased. Therefore, we will not suddenly add in any outliers.
- We will not see sudden jumps in the estimates since outliers will not have as much weight. Therefore, the change will look more smooth

▼ Designing the Filter

1. We will begin with a window of size m . Then we can try to fit a polynomial trend within the window that minimises the least squares error.
2. We then replace the observation in the middle of that window with the point on the fitted polynomial
3. We only care about b_0 which is the intercept because it will be the estimate for $t = 0$. Therefore, we don't have to care about the value for $b_1, \cdots b_n$

▼ Decomposition Algorithms

Classical Decomposition

General Algorithm

1. Estimate the **trend cycle** using moving-average (**MA**)
2. Compute the **de-trended** series which leaves us with S_t, R_t
   a. **Additive Decomposition**: $y_t - \hat{T}_t$
   b. **Multiplicative Decomposition**: $y_t / \hat{T}_t$
3. Estimate the Seasonal component, $\hat{S}_t$
   a. For each Seasonal component index, we just take an average across all years for instance for a particular month
4. Compute the remainder component which is to just take $y_t - \hat{T}_t - \hat{S}_t$

▼ Additive Decomposition

1. If m is even, compute $\hat{T}_t$ using the $2 \times m - MA$. Otherwise, use the simple $m - MA$.

    a. This provides us with $\hat{T}_t$

2. Calculate the de-trended series: $y_t - \hat{T}_t$
3. Estimating the seasonal component:
    a. Average the detrended values for each month
        i. E.g. Average all the de-trended March values to obtain the estimate of the effect of the March season
            1. Note that this allows us to get each of the $\hat{S}_t$ since our seasonal effect is assumed to be the same for each season for an Additive Decomposition
        ii. Adjust the seasonal component so that they sum to 0 to get $\hat{S}_t$.
            1. Note that what we do here is to subtract them by the average seasonal effect to make sure they sum to 0

$$\hat{S}_t = \hat{S}_t^0 - \frac{1}{m}\sum_{k=1}^{m} \hat{S}_k^0$$

            where $t = 1, \cdots, m$ and m is the period

        iii. This last step ensures that there is no confounding of the seasonal effects with the level of the time series, and allows us to view the seasonal effects as deviations from the trend-cycle
4. Calculate the remainder component using the following formula

$$\hat{R}_t = y_t - \hat{T}_t - \hat{S}_t$$

▼ **Formula**

$$y_t = T_t + S_t + R_t$$

▼ Multiplicative Decomposition

Steps:

1. If m is even, compute $\hat{T}_t$ using the $2 \times m - MA$. Otherwise, use the simple $m - MA$.

    a. This provides us with $\hat{T}_t$

2. Calculate the de-trended series $\frac{y_t}{\hat{T}_t}$
3. Estimate the seasonal component by averaging the de-trended values for that month
    a. Adjust the seasonal component so that they sum to m to get $\hat{S}_t$
    b. This ensures that the average of the seasonal effects if 1; each is then a multiplicative deviation from the trend-cycle
4. Calculate the remainder component using

$$\hat{R}_t = \frac{y_t}{\hat{T}_t \hat{S}_t}$$

Comments on Classical Decomposition

1. Estimate of trend is **unavailable** for first few and last few observations
2. **Seasonal components repeats** from year to year since we assume the same seasonal component across years. (**This may not be realistic**)
3. **Not robust to outliers** since we make use of averages and averages are sensitive to outliers

▼ Other New Methods

▼ X11 Decomposition

Based on the classical decomposition but with some improvements

▼ Improvements

1. Relatively robust to outliers
2. Completed automated choices for trend and seasonal changes
3. Very widely tested on economic data over a long period of time

▼ Cons

1. Did not provide a good adjustment for data at the ends
2. No prediction/confidence intervals
3. Ad hoc method with no underlying model
4. Only developed for quarterly and monthly data

▼ X11-ARIMA

- Models the original time series with ARIMA
- Extends the time series in the past and the future and then use X11

▼ X12-ARIMA

- Used ARIMA
- Used regression ARIMA
  - β's are the outlier effects, level shifts, holiday effects

▼ X13-ARIMA-SEATS Seasonal Adjustment

- Very useful when we have monthly data

SEATS - Uses frequency domain techniques

- Uses this to decompose the ARIMA models

Advantages

- Model-based
- Smooth trend estimate
- Allows estimates at end points
- Allows changing seasonality
- Developed for economic data

Disadvantages

- Developed only for quarterly and monthly data

▼ R Code

```
tmp1 <- model(granite,
              plain=X_13ARIMA_SEATS(units ~ regression(variables="td",
                                                       aictest=NULL)))

# calls the x11 method instead
x11_dcmp <- model(us_retail_employment,
```

```
                plain=X_13ARIMA_SEATS(Employed ~ x11())) |>
              components()
```

`X_13ARIMA_SEATS` - Using the X13-ARIMA-SEATS model

`variables="td"` - Trading Day effect is used as the regression variables

`aictest=NULL` - Does AIC comparison and if it is not significant, the variables will be dropped

Note that it could give us:

1. Additive Outlier - Points where there is significant variation
2. Level Shift - Points where there are significant level shifts

▼ STL Decomposition

STL: “Seasonal and Trend decomposition using Loess”

▼ Advantages

- Versatile and robust
- Can handle **any type of seasonality**
- Seasonal component can change over time and rate of change controlled by the user (**Note that classical decomposition assumes constant seasonality**)
- Smoothness of trend-cycle is also controlled by the user
- Robust to outliers

▼ Disadvantages

- No trading day or calendar adjustments
- Only works on additive decompositions (Note that multiplicative time series can be converted to additive ones using the log transformation)

▼ Loess Fitting

The loess regression curve, $\hat{g}(x)$ is a smoothing of y given x that we can use to compute for any x

Things that needs to be decided

1. Window Size
2. Polynomial degree

Steps for computing $\hat{g}(x)$

1. Choose a positive integer q that denotes the span for the points that we want to include
    a. Note that we can denote it as α as well. Which could be a proportion of points that we want to include inside
    b. Bigger q means a bigger neighborhood of points
2. The q nearest values are given weights based on how close they are to x (Note that x is the fixed point that we want to compute)
    a. We can let each of the weights be $v(x_i) \qquad \text{for } i = 1, \cdots, q$
    b. Note that the weights is just kernel function
    c. If x_i is closer to x, it will have a larger $v(x_i)$ than one that is further from x
3. We will then perform a weighted least squares regression with the above weights

$$\arg\min \sum_{i=1}^{q} (y_i - \beta_0 + \beta_1 x_i - \beta_2 x_i^2) v(x_i)$$

    Note that the degree polynomial is a parameter that we can tune as well but we can normally just make use of a degree 1 or 2 polynomial to estimate it.

    - Note that the $v(x_i)$ makes the error from those points that are closer to be higher

4. Note that our estimate will just be the following

$$\hat{g}(x) = \hat{\beta}_0 + \hat{\beta}_1(x) + \hat{\beta}_2(x)^2$$

- Noting that each of the $\hat{\beta}$'s will be different at different points and we will just join across all the points

▼ R Code

`gg_smooth(span = <>, method="loess", method.args=list(degree=<>))` -

- `span` - Note that the span parameter just tells us how much of the points we will include
  - `span=1.0` - Will use all the points
  - `span=0.5` - Uses 0.5 of the points
- `degree` - States the degree of the polynomial that we are going to be using

▼ STL Algorithm

General Idea

- **Inner Loop**: Smooths the trend and seasonal components (Note that the seasonal component can vary over time)
  - We will make use of a de-trended and seasonally-adjusted time series and compute a smoothed estimate of both the seasonal and trend components respectively and update it
- **Outer Loop:** Computes the remainder component then assigns robustness weights to each observation. These weights will be used in the inner loop to downplay the importance of outliers in the time series
  - Robustness just means that the estimation of other parameters are not affected by the outliers. It does not mean that it can predict the outliers well
- Loess algorithm is repeatedly used as the smoother, except in one portion of the algorithm

Steps:

Inner Loop: For estimation of the $\hat{T}$ and $\hat{S}$ (Makes use of the robustness weights that are given in the outer loop and an initial trend-cycle estimate)

- When we enter the inner loop, we have some preliminary estimate of the trend and the seasonal

- At the end of each loop, there will be an update of the trend and the seasonal effect

1. De-trend the original time series using the current estimate of $\hat{T}_t^{(k)}$, where k is the iteration
2. Estimate the Loess for the 12 sub-series, $\hat{S}_t^{(k)}$ (Fit it for each of the of sub series) using the de-trended series
    a. This comes under the **Season** window (`s.window`)
    b. This means that for each of the month, we will compute a Loess estimate for them.
3. Combine the smooth estimates into 1 series. Estimate a Loess for this combined series, $T_t^{'(k)}$
    a. Note that we will make use of each of the Loess 12 sub-series. The 12 sub series will be combined in chronological ordering
    b. Note that for the second estimation of the loess is just to remove other possible trends that could be introduced when we re-combine them. (We also note that even though this is a de-trended series to begin with, it is still possible that some trend is added)
    c. This comes under the **Low Pass** window. (Default exists)
4. We will then get the following where we de-trend the seasonal estimate in **step 2**

$$S_t^{(k+1)} = S_t^{(k)} - T_t^{'(k)}$$

5. We can compute the following to get a seasonally-adjusted time series

$$y_t - \hat{S}_t^{(k+1)}$$

    1. Smooth this with loess again to get $\hat{T}_t^{(k+1)}$. This is kinda the same idea for the Seasonal component
    2. This comes under the **Trend** window. (Default exists)
6. Note that we will do this normally for 1 - 2 iterations

Outer Loop: Makes it robust to the influence of the outliers. Estimate the remainder component

- **Note**: This can be skipped. No need if we conclude that there are no outliers
- Points that generates large “residuals” (**remainder term** from the decomposition) are given smaller weights.
  - Note that this weight enters in the Loess Fitting of the polynomial. It is a separate weight that is given from the inner loop Loess weights. This weight helps to ensure that the outliers do not affect the estimates too much
  - We will compute the weights after the run with the Inner Loop and getting the new trend and seasonal estimates
- **Empirically**: Usually 5-10 iterations are run

▼ R Code

```
stl_variants <- model(granite,
                      stl1 = STL(units ~ trend(window=10, degree=1)),
                      stl2 = STL(units ~ trend(window=200, degree=0)),
                      stl_robust = STL(units, robust=TRUE),

                      stl1a = STL(units ~ trend(window=10, degree=1) +
                                    season(window=5, degree=1)),

                      stl_robust2 = STL(units ~ season(window=5),
                                        robust=TRUE))
```

Note that we have different parts to the model specification

Params:

`robust=TRUE` - Runs the outer loop

- When we look at the trend, places where there residuals will not affect our trend but we will be able to see the outlier in the remainder part

`window=k` - Number of points that we use for the estimation of the loess trend

    - Note that the bigger the window, the smoother the estimation will be
    - `"periodic"` - We can let the seasonal window be periodic instead to make the seasonal component constant over time

`trend(window = ?)` - Controls the wiggliness of trend component

`season(window = ?)` - Controls the wiggliness of seasonal component

`season(window = "periodic")` - Equivalent to an infinite window

Determining if the season window is appropriate

- We can try to replot the seasonal trend with the subseries and see if the loess estimation fits our points appropriately
    - If the trend is too smooth then the window is probably too big
    - If the trend is too variable then the window is probably too small