AAEC 4484/ STAT(AAEC) 5484: Applied Economic Forecasting

Your Name Here

Homework #5 - Spring 2023

Instructions: In all cases, please ensure that your graphs and visuals have proper titles and axes labels, where necessary. Refer to the output, whenever appropriate, when discussing the results. **Lastly, remember that creativity (coupled with relevance) will be rewarded.**

Question 1: Classical Decomposition

In the class notes, we explored the manual computation/coding of the Classical decomposition of an Additive series. Here, I would like you to reproduce that same idea but for a series with a multiplicative pattern.

Using your in-class notes, use the `quantmod` package to pull the data on "Retail Sales: Beer, Wine, and Liquor Stores" from the FRED website (https://fred.stlouisfed.org/series/MRTSSM4453USN) into `R`.

1. In a single step, pull the relevant data and store it as a `tsibble()` object called `liquor`.
2. Plot a graph of the `liquor` series and **briefly** comment on and pattern you observe. *To help with your titling, please see the series definition and units here (https://fred.stlouisfed.org/series/MRTSSM4453USN)*
3. With these observations, we are now ready to perform our decomposition. Recall that with a multiplicative decomposition, the functional form is:

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

Step 1. Using an appropriate *m-MA* model, extract the `trend-cycle` component, T_t. Recall that we might need to doubly smooth the series to get it to be symmetric.

Present a plot of your trend-cycle element on top of the original data.

Step 2. Using the trend computed above, compute and plot the detrended series

Step 3. Using regression analysis, extract the seasonal component, S_t. Also, plot your seasonal component.

Step 3b. Using the extract seasonal component, S_t, *compute* and *plot* the `seasonally-adjusted` variable. **Overlay the original data as well.**

Step 4. Lastly, compute and plot the `remainder` component.

Step 5. Produce a plot of the data and three components. Store your plot as `plot1`.

Please order your variables (graphs) as Data » Trend » Seasonal » Remainder.

This will require that you convert your variable names to factors so that `R` respects the "hierarchy" and not order alphabetically in the faceted graphs. Your graphs should be stacked in a single column (so it resembles the results of the built in function).

Step 5b. Use the `classical_decomposition()` function to create the **multiplicative** decomposition of the series. Store this plot as `plot2`.

Step 6. Use the `gridarrange()` function to create a 2 column plot of your two plots(`plot1`,`plot2`) stored above.

4. Confirm that your decomposition was correct by plotting the multiplication of the three extracted components (y-axis) against the actual data (x-axis). Be sure to add in a 45^o line to see how the scatter plot stacks up against this line.

Question 2: Forecasting with Time Series Decompositions

Returning to the `liquor` series, we can use our forecast decomposition to make forecasts of the future. As we discussed in class, this would require us to `deseasonalize` the variables, estimate the `seasonal` and `deseasonalized` components separately then add them back together, `reseasonalize`.

1. Your friend Paula is adamant that the seasonality in this model is additive since it appears to be trending upwards (therefore, adding each year). **Explain to her exactly why the data could be best modelled using a multiplicative decomposition.**

`STL` Decomposition

For the questions that follow, we will employ a STL decomposition and then compare model fits depending on the method used to forecast the seasonally adjusted data.

2. Recall from our discussions in class that the STL model only allows for an **additive decomposition**. This means that we will need to transform the data first, then back transform after forecasting.

Using the `guerrero` feature, determine the optimal lambda, `l.opt`, needed to stabilize the variable. Next use the `box_cox()` and `autoplot()` functions to plot the stabilized series.

Add information to your subtitle to let me know what the optimal lambda is. I would like to see you pass your optimal lambda variable to the `bquote()` function. Use `l.opt[[1]]` so that it extracts the value from the tibble

3. Admittedly, it can get a bit tedious to perform backtransformation when you use the Box-Cox method to stabilize the data. I suggest that we could potentially get away with using logs instead. You might want to first use the mutate function to create a new column called `lsales` which is the logged sales value.

Produce a plot of the `lsales` and comment on whether the transformation appears to stabilize your variance.

Forecasting with Decomposition

We are interested in forecasting using these decompositions. That will require that we forecast the seasonally adjusted data (using a nonseasonal method) and the seasonal component using a seasonal method, such as the seasonal naïve approach.

4. Assign the last 2 years (24 data points) of data to a `test` data set. All others will serve to `train` our model.

Note: There are a few nifty function for indexing data. Instead of using the `filter_index()` or `filter()` commands, I would like you to explore some of these. Remember, Google is your friend!

- Extract the last 2 years of data using the `slice_tail()` command.
- We can extract the data for the training using the `slice_head()` command but it requires a bit more mental math than I would like. **Instead, use the `slice()` function and extract the first `N-24` observations. The following syntax will help: `1:(n()-24)`. Here `n()` is the built in argument for the total number of observations.**

Use a graph to confirm that you have properly subset the `lsales` data series.

5. Using the `STL()` function, fit the model on the `lsales` series in the `training` data and the produce the relevant forecast. **Please use a trend window of 13 and allow your seasons to be fixed over time, ensure that your model is robust to outliers.**

- You will find it best to first store the model fit.
- Also, produce an `autoplot` of the `components` of this decomposition .

6. Next, produce a plot of the seasonally adjusted data and overlay the training data. **Ensure you are staying consistent with your units.**

7. Following our notes, use a `naïve` model to forecast the seasonal adjusted data. Produce the plot to see how it actually works.

8. Using the seasonal naïve model, produce a plot of the forecast of the seasonal component.

Using the `decomposition_model()` function

This approach will require that we add these components together then exponentiate them to backtransform the results. On the other hand, we can use the `decomposition_model()` function to dictate the forecast method to be used on the seasonally adjusted component then the forecasts are conducted (using the additive model) and then transformed with proper confidence intervals added. Our "dilemma" however, is which model should we choose?

9. Use the `naive`, `drift`, and `ETS(A,N,N)` methods to forecast the seasonally adjusted component. Save your models into a variable called `mod.fit` before forecasting. Store your forecasts as `mod.all`.

Note:

- **You do not want to use the `lsales` data as the `decomposition_model()` function will find the appropriate lamba value for you and compute accordingly.**
- **Please name each model appropriately in a single `model()` command.**
- You are not required to plot the forecasts here, just store them.

10. Produce a plot of the model forecasts (stored in mod.all) for each of the three models and the *test data*. **Be sure to turn the PIs off.**

Which method appears to do the best job (visually) of forecasting the test portion of the data.

11. Compare the forecast accuracy of the three forecasts.

- Which model is chosen by the `RMSE`? How about `MAE`? As usual, be sure to explain how you arrived at your conclusion.
- Produce your model accuracy statistics as a `kable` with digits rounded to 3 dps.

12. Do the residuals from the preferred model stored in `mod.fit` appear to be white noise?