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

Your Name Here

Homework #3 - Spring 2023

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

US Natural Gas Consumption

This exercise is intended to both complement and go beyond the scope of the course materials. In particular, you will be pulling real world data on monthly U.S. Consumption of Natural Gas directly from the U.S. Energy Information Administration (EIA)– See website here

Throughout, our focus will be on estimating and evaluating competing models (albeit the most basic). At the end, you will select an "optimal" model for modeling the future value of consumption. This assignment should bridge the gap between the earlier modules and this current one.

Note: Before you proceed, please install the `readxl` package. Be sure to do this in your *console* and not within the .Rmd file.

1. Using my code below, import the NG consumption data from the EIA into `R`.

```
# Store a file to your computer's temporary memory
tmp <- tempfile(fileext = ".xls")
# Download data from EIA and store in tmp
download.file(url = "https://www.eia.gov/dnav/ng/hist_xls/N9140US2m.xls",
              destfile = tmp, mode = "wb")
```

2. Next, using the `read_excel()` function from the `readxl` package you were instructed to install earlier, read in the temporary file, `tmp`, into `R`.

We will need to skip the first 2 rows using `skip = 2`, since they are not our column headings. Also, we will need to assign `sheet = 2` since our data appears on the second worksheet.

Please feel free to manually download the excel file from the EIA link above and explore this on your own.

Task: Your task will be to store this as `ngdata`.

```
____ <- readxl::read_excel(tmp, sheet = ___, skip = ____)
```

3. If you were to view the `ngdata` series **in your console**, you will note that the first column is `Date` and the second is `U.S. Natural Gas Total Consumption (MMcf)`).

a. Using the `mutate()` function, declare the `Date` column as a `yearmonth()` object. It is fine to override `Date`.
b. Next, using the `as_tibble()` function, declare `Date` as the `index`.
c. Next, using the `rename()` function, change `U.S. Natural Gas Total Consumption (MMcf)` to `Demand`.
d. Convert `Demand` to `Bcf` (Billion cubic feet). Keep the column name as `Demand`.
e. Store the results of this step into a variable called `tsng`.

4. Present a `autoplot()` of `Demand`. Briefly comment on your plot. Are there periods when `Demand` appears to be higher or lower? **Be sure to properly label your axes**.

5. Besides the graph above, using **at least 3 graphs** you have learned so far,

- **comment** on possible seasonality and/or trends observed in the consumption data.
- Would you say that the series appears to be white noise? **Remember that a simple yes will not suffice!**
- **Use the `gridExtra` package to create a grid as in HW2.**

6. Let's proceed to forecast the `tsng` series. We will split the dataset into a training (`train.ng`) and testing set period (`test.ng`).

Using the `filter_index()` function, assign the observations up to December 2019 to `train.ng`. All observations afterwards should be assigned to `test.ng`.

7. Confirm that the data is properly split by using the `autoplot()` and `autolayer()` functions. Be sure to include `tsng`, `train.ng` and `test.ng` in this plot. **A title is not necessary. Add colors to your lines to highlight each series**

If your data is properly subsetted then the training and test series should span the entire sample period in `tsng`. You can ignore a small line segment that connects the training and testing datasets.

8. Use the four (4) benchmark models from class to produce an `autoplot()` of the forecasts in the test period. Does any particular forecast method appear to do a better job than others?

- **Remember that you will need to set `h = nrow(test.ng)`. That is `h = 35`.**
- To get a good visual of each model fit, you might want to `autolayer()` the test data, `test.ng`, as well.
- Be sure to turn off the `PI` at this stage.

9. Using the `accuracy()` command extract the `RMSE`, `MAE`, `MAPE`, and `MASE` statistics. Also, I am intrigued to know what the `MSE` value is so use the `mutate()` function to create that column.

- Display your results as a table using the `knitr::kable( )` function with `digits = 2`. Add the argument `format.arg = list(big.mark = ",")` to format your table using thousands separator.
- Lastly, since Henry likes table captions, include **an appropriate one** by adding and modifying the argument `caption = "Your caption Here"`.

Which model is preferred under the each of the 5 selection criteria? Be sure to explain how you came to that conclusion.

10. There are times when the preferred model might not necessarily satisfy some of our necessary assumptions. To illustrate this, use the `gg_tsresiduals()` function, with `type = "innov"`, to comment on the residuals (*in-sample*) from the model preferred by the `MSE`. In particular,

- do they appear to be normally distributed? **Set the lag_max = 36 so that you can see more of the dynamics of the `ACF`.**
- Do you observe any **potential** patterns (seasonality, trend, cycle) in the `autoplot` and `ACF` of the residuals?

11. Using the same model as above, employ the `Ljung-Box` test to conduct an hypothesis test of autocorrelation up to lag $\ell = 2 \times m$. **Where m is the frequency of the data.**

- You are required to explicitly state the null hypothesis being tested and properly state how you came by your conclusion. **Think back to the decision rule of the test.**

12. After showing off your results to a friend, Isaac, he is unimpressed by your point estimates and the simplicity of your "preferred model". He is even more concerned about the potential non-normality of your residuals. Instead, he would like to get a better feel for the uncertainty around your model forecasts.

You decide to create an `autoplot()` of the bootstrapped `95%` prediction interval around your forecasts. Be sure to `autolayer()` the **training** data on this plot as well. **To save you some computation time, conduct only 1,000 draws.**