D2T13 Task 1 - Time Series Modeling

David Gerard
Data Analytics Time Explorer
October 2021

A-1. Summarize one research question that is relevant to a real-world organizational situation captured in the selected time series

Time series analysis allows us to determine the trend of trends and seasonal flux and recurring in a dataset over time. Revealing these attributes allows for a more accurate understanding of the underlying phenomena that the data represents. Understanding these underlying phenomena can help us predict, with confidence, future trends and shapes, materially improve our operating efficiencies, and gain competitive and operational advantages.

For example, we are examining today's airline tickets demand by the in-flight company. **Can we predict the future revenue trends for the company?** Knowing the question will be the guide for the rest of this analysis.

A-2. Define the objectives or goals of the data analysis.

Our primary goal is to predict the future revenue trends for the relevant company. This will allow the company to adequately plan for operational and financial requirements between the now and the future.

B-1. Summarize the assumptions of a time series model including stationarity and autocorrelated data.

The most critical assumption for the time series data is that it is a stream of events spaced in time and a period of the lifespan. D2T2, The Time Series Data Structure, discusses the time series data structure. The time series data structure is critical for the analysis of trends and seasonalities and autocorrelations.

Stationarity assumes that the points do not contain a trend. The can be calculated by making sure the mean and variance do not vary across the time series. If the mean and variance do not change across the time series, we have a stationary time series. If the mean or variance do not stay constant, we can also work to derive seasonal trends.

Autocorrelation assumes that each data point is not influenced by a previous point or a set thereof (Pearson, 2021). This could be the point immediately before or points that are a set number of points before the current point.

C-1. Provide a line graph visualizing the realization of the time series.

```
data_ts_read.head()
data_ts_read.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 132 entries, 0 to 131
Data columns (total 2):
 #   Year   Monthly
 0   1949-01  112.0
 1   1949-02  118.0
 2   1949-03  132.0
 3   1949-04  129.0
 4   1949-05  121.0
dtype: float64
```

```
print(data_ts_read[ '1' ])
```

C-2. Describe the time step formatting of the realization, including any gaps in measurement and the length of the sequence

Missing Values

To begin analysis on the data, we will check for missing points. We will first check the Year column to see if the day counter matches the row counter.

```
data_ts_read.isna.sum()
```

```
Year    False
1       False
dtype: bool
Year    False
1       False
dtype: bool
Year    False
1       False
dtype: bool
Year    False
1       False
dtype: bool
Year    False
1       False
dtype: bool
```

Duplicate Values

Now that we have checked the data set, there are still check for duplicate entries for a single day.

```
data_ts_read.duplicated(keep='last').sum()
```

```
0
```

0 daily totals over 5 minutes and 1hr ceiling

Since no values were returned in our calculation, we can continue with confirmation that the series is continuous without duplicates or missing values.

C-3. Evaluate the stationarity of the time series.

We will use the Augmented Dickey-Fuller test in order to check for stationarity. If each one of the data sound suggest that it doesn't have a consistent mean and variance, we can also utilize the data to confirm with our data. We can use a lag value of 30, as we are the data at a larger granularity than 30 days.

Library Calculation

```
from statsmodels.tsa.stattools import adfuller
print("Augmentation Dickey-Fuller Test for stationarity")
adf, pvalue, usedlag, nobs, critval, icbest = \
    adfuller(data_ts[0])
print("ADF : %f" % adf)
print("P-value : %f" % pvalue)
print("Number of lags used : %d" % usedlag)
print("Number of observations used : %d" % nobs)
for key, value in critval.items():
    print("Critical value %s : %f" % (key, value))
print("Maximum information criterion : %f" % icbest)
```

```
Augmentation Dickey-Fuller Test
ADF : 0.815369
P-value : 0.991880
Number of lags used : 13
Number of observations used : 118
Critical value 1% : -3.487022
Critical value 5% : -2.886151
Critical value 10% : -2.580009
Maximum information criterion : 1055.358711
```

Since this value is greater than 0.05, we fail to reject the null hypothesis and determine the data is not stationary. Since it doesn't appear to grow in magnitude over time, we should be able to see the difference in means for our analysis.

```
# p-h-d defined
# p=30, h=1, d=1, data_ts[0] is column value for 1st column
adf,pvalue,usedlag,nobs,critval,icbest = \
    adfuller(data_ts[0],30,1,1)

print("Augmentation Dickey-Fuller Test")
print("ADF : %f" % adf)
print("P-value : %f" % pvalue)
print("Number of lags used : %d" % usedlag)
print("Number of observations used : %d" % nobs)
for key, value in critval.items():
    print("Critical value %s : %f" % (key, value))
print("Maximum information criterion : %f" % icbest)
```

Now our p-value is well under 0.05, so we can reject the null hypothesis and determine that our data is now stationary.

C-4. Explain the steps used to prepare the data for analysis, including the training and test set split

We will use the data for training and testing since we are attempting to predict the future. We will keep the data in the training set, and set aside a portion for the testing set.

Library: Pandas, Numpy
Note: Set the training data size, **n** (used hereafter) = 0.8 (80%)
Split in **dp1**(Training) and **dp2**(Testing)
Training Dimensions
Testing Dimensions

```
# n, n-1, n-2
n = 0.8
```

```
Training Dimensions
```

```
n-1, n-2, 0
```

```
Testing Dimensions
```

```
print("Training Dimensions")
dp1.shape
print("Testing Dimensions")
dp2.shape
```

```
Training Dimensions
(105, 2)
Testing Dimensions
(27, 2)
```

After the split, we have a training set with 105 records and a testing set with 105 records.

C-5. Provide a copy of the cleaned dataset.

Next, we will export the cleaned time series so that it can be uploaded with the paper.

```
dp1.to_csv(path, "Cleaned_time_series.csv")
```

C-6. Report the annotated findings with visualizations of your data analysis, including the following elements:

- trends (including seasonal and aperiodic components)
- cycles
- autocorrelation
- spectral density
- mean and variance over time (parts)
- combination of the trend of trends in the residuals of the decomposed series

Let us now review the decomposed elements of the dataset in review seasonality, trends, and noise.

```
Series = dp1.Monthly
```

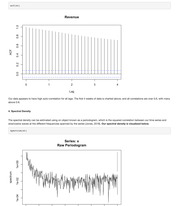
1. Stationarity

```
print(sm.tsa.stattoolsadfuller(Series, 30))
```

```
(0.8153692992235483, 0.991880013389821, 13, 118, {'1%': -3.487022163120315, '5%': -2.886151496432075, '10%': -2.580009026141835}, 1055.3587108966428)
```

Three dates appear to be a seasonal component which we note this date with a seasonity frequency. The most noticeable that each day rates similar sales week over week since the trend and noise data is removed.

```
print(sm.tsa.stattoolsadfuller(Series, 1, 1))
```



```
##
## $AICc
## [1] 1.62872
##
## $BIC
## [1] 1.6849
```

2. Prediction Interval

```
#First prediction
test_forecast$pred[1]
```

```
## [1] 12.1331
```

```
#1.96 Standard deviations above and below (using standard error)
test_forecast$pred[1] - (1.96 * test_forecast$se[1])
```

```
## [1] 11.08216
```

```
test_forecast$pred[1] + (1.96 * test_forecast$se[1])
```

```
## [1] 13.18404
```

We can derive the confidence interval by using the standard errors returns in the predicitons. **For the first predicted point of 12.1331, we find that the confidence interval at 98% is between 11.082 and 13.184.** See calculations above.

3. Forecast Length

```
month_forecast$pred[31] - (1.96 * month_forecast$se[31])
```

```
## [1] 12.92541
```

```
month_forecast$pred[31] + (1.96 * month_forecast$se[31])
```

```
## [1] 21.18617
```

Since this model starts opening up on the range of confidence intervals pretty quickly, I decided to limit the prediction to one month. That already puts the prediction in a range of 12.925 to 21.186, which will not be very helpful in planning fiscal endeavors.

4. Forecast Evaluation

```
library(Metrics)
```

```
##
## Attaching package: 'Metrics'
```

```
## The following object is masked from 'package:forecast':
##
##     accuracy
```

```
mse(test_forecast$pred, test)
```

```
## [1] 17.89029
```

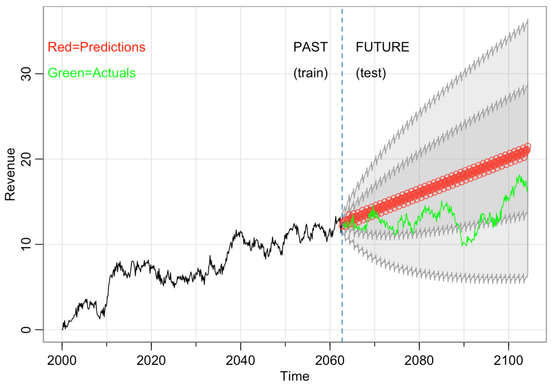
```
rmse(test_forecast$pred, test)
```

```
## [1] 4.229691
```

Our model is not very trustworthy in its current state. The RMSE of 4.29 is rather large considering the values are currently under 15.

E-2. Provide an annotated visualization of the forecast of the final model compared to the test set.

```
Revenue<-train
pass <- sarima.for(Revenue, n.ahead=length(test), 2, 0, 0, 2, 1, 0, 7, plot.all=TRUE)
lines(test, col="green", ylab="Revenue")
text(x=2050, y=33, pos=4, "PAST")
text(x=2050, y=30, pos=4, "(train)")
text(2064, y=33, pos=4, "FUTURE")
text(2064, y=30, pos=4, "(test)")
text(1995, y=33, pos=4, "Red=Predictions", col="red")
text(1995, y=30, pos=4, "Green=Actuals", col="green")
abline(v=2000 + (length(train)/7), lty=2, col=4)
```




```
?text()
```

E-3. Recommend a course of action based on your results.

Since our confidence interval was relatively wide, I would not recommend any large change in current investments or plans based on this model. I would expect slight growth and make sure the company is prepared to increase customer and employee counts, but this model is not accurate enough to dictate a severe change in resource or investment allocations.

Appendix I: Code References

Identify and Remove Duplicate Data in R. (2018, October 19). Datanovia. Retrieved November 29, 2021, from https://www.datanovia.com/en/lessons/identify-and-remove-duplicate-data-in-r/

RPubs - Time series in R: Stationarity testing. (2017, April 21). RPubs. Retrieved November 30, 2021, from https://rpubs.com/richkt/269797

Time series seasonality test. (2018, May 16). Cross Validated. Retrieved November 30, 2021, from https://stats.stackexchange.com/questions/346497/time-series-seasonality-test

R.C.K. (2020, January 21). Split Time Series Object for Training and Testing Partitions. Rdrr.Io. Retrieved November 30, 2021, from https://rdrr.io/cran/TSstudio/man/ts_split.html

auto.arima function - RDocumentation. (n.d.). RDocumentation. Retrieved December 1, 2021, from https://www.rdocumentation.org/packages/forecast/versions/8.15/topics/auto.arima

Poison, N. (n.d.). astsa/fun_with_astsa.md at master · nickpoison/astsa. GitHub. Retrieved December 1, 2021, from https://github.com/nickpoison/astsa/blob/master/fun_with_astsa/fun_with_astsa.md#2-plotting

Appendix II: Citations

Peixeiro, M. (2021, October 20). The Complete Guide to Time Series Analysis and Forecasting. Medium. Retrieved November 29, 2021, from https://towardsdatascience.com/the-complete-guide-to-time-series-analysis-and-forecasting-70d476bfe775

Wu, S. (2021, July 4). Stationarity Assumption in Time Series Data - Towards Data Science. Medium. Retrieved November 29, 2021, from https://towardsdatascience.com/stationarity-assumption-in-time-series-data-67ec93d0f2f

Jones, J. H. (2018, February 19). Time Series and Spectral Analysis. Stanford. Retrieved November 30, 2021, from http://web.stanford.edu/class/earthsys214/notes/series.html