16. TIME SERIES

PREVIOUSLY

- Topic Modeling
- LDA, LSA
- Gensim, NLTK

TODAY

- Time series
- Modeling
- Stationarity
- Trending and seasonality
- Forecasting 101

PYTHON

- Python Koans
  https://github.com/gregmalcolm/python_koans
- Hackerrank: https://www.hackerrank.com/domains

TIME SERIES [TS]

A time series is a series of data points listed in time order.

A sequence taken at successive equally spaced points in time.

A sequence of **discrete-time data**.

=> The **time interval** is key

TIME SERIES [TS]

Any domain of applied science and engineering which involves temporal measurements.

- **IoT**
- **econometrics**
- mathematical finance / trading / markets
- intelligent transport and trajectory forecasting
- weather forecasting and Climate change research
- earthquake prediction, astronomy
- electroencephalography, control engineering, communications
- **signal processing**

TS

- Modeling
- Forecasting
- Pattern detection
- Detection of a change in the parameters of a static or dynamic system

TIME VS FREQUENCY DOMAIN: FOURIER TRANSFORM

The Fourier transform decomposes a function of time (a signal) into the frequencies that make it up. A Fourier transform takes a time series or a function of continuous time, and maps it into a frequency spectrum.

- Jean-Baptiste Joseph Fourier, 1807 Treatise on the *propagation of heat in solid bodies*.

SO MANY TS!

- Great source of Time series
- Google trends

TS: COMPONENTS

- Trend:
  - Gradual long term evolution
  - Easiest to detect
- Cycle or Seasonal Variation
  - Up and down repetitive movement
  - Repeats itself over a long period of time
- Random Variations
  - Erratic movements that do not follow a pattern
  - Not predictable

TS: EXAMPLES

- https://datamarket.com/data/set/22ox/monthly-milk-production-pounds-per-cow-jan-62-dec-75#!ds=22ox&display=line
- https://datamarket.com/data/set/22vd/quarterly-production-of-clay-bricks-million-units-mar-1956-sep-1994#!ds=22vd&display=line
- https://datamarket.com/data/set/235d/mean-daily-temperature-fisher-river-near-dallas-jan-01-1988-to-dec-31-1991#!ds=235d&display=line

LOADING AND INDEXING IN PANDAS

Simple loading

```
ts = pd.read_csv('../data/Dow-Jones.csv', parse_dates = ['Date'], infer_datetime_format = True)
ts[ts.Date > '2010-01-01']
```

Make the date the index:

```
ts = pd.read_csv('../data/Dow-Jones.csv', parse_dates=['Date'], index_col='Date', infer_datetime_f
ts['2010-12-31':'2010-01-01']
ts[:'2010-01-01']
```

FORECASTING 101

- Smoothing the data
- Simple forecasting technique
- Tree rings dataset

```
ts = pd.read_csv('../data/tree-rings.csv', parse_dates = ['year'], index_col = 'year', infer_datetime_for
```

PLOT THE TREE RINGS

Notebook - Smoothing and Forecast 101

VERY SIMPLE FORECASTING TECHNIQUE

$$\hat{Y}_{n+1} = Y_n$$

- load the ts
- create a new column 1 period gap

How good is that predictor?

TS METRICS

Forecasting error: $e_i = y_i - \hat{y}_i$

Metrics to compare TS techniques

- Mean Absolute Error: $MAE = mean(|e_i|)$
- Mean Absolute Deviation: $MAD = \frac{1}{n} \sum_{i=1}^{n} |e_i|$
- Root mean squared error: $RMSE = \sqrt{mean(e_i^2)}$
- Mean absolute percentage error: $MAPE = \frac{100}{n} \sum_{i=1}^{n} \frac{|e_i|}{y_i}$

MOVING AVERAGE

Smoothed over a window of n samples

$$SMA = \frac{p_M + p_{M-1} + \cdots + p_{M-(n-1)}}{n} = \frac{1}{n}\sum_{i=0}^{n-1} p_{M-i}$$

and center

$$SMA = \frac{p_{M+n/2} + \cdots + p_{M+1} + p_M + p_{M-1} + \cdots + p_{M-(n/2)}}{n} = \frac{1}{n}\sum_{i=-\frac{n}{2}}^{\frac{n}{2}} p_{M+i}$$

MOVING AVERAGE

Use SMA to forecast

EXPONENTIAL WEIGHTED MOVING AVERAGE

EXPONENTIAL WEIGHTED MOVING AVERAGE

Introduce a Decay

The EWMA for a series Y may be calculated recursively:

- $S_1 = Y_1$
- $t > 1, \quad S_t = \alpha \cdot Y_t + (1 - \alpha) \cdot S_{t-1}$

Where:

- The coefficient α represents the degree of weighting decrease, a constant smoothing factor between 0 and 1. A higher α discounts older observations faster.
- Yt is the value at a time period t.
- St is the value of the EWMA at any time period t.

http://pandas.pydata.org/pandas-docs/version/0.17.0/generated/pandas.ewma.html

http://pandas.pydata.org/pandas-docs/version/0.17.0/generated/pandas.tools.plotting.autocorrelation_plot.html

AUTO CORRELATION

A measure of how much is the current value influenced by the previous values in a time series.

Autocorrelation measures the linear relationship between lagged values of a time series.

```
pandas autocorrelation_plot()
```

PARTIAL AUTO CORRELATION

Let's say you have a TS with a high correlation 1 sample apart

- 1) $Y_{n+1}|$ very correlated with $Y_n|$
- 2) $Y_n|$ very correlated with $Y_{n-1}|$
- etc ...

That correlation impacts the correlation between samples that are further apart

- $Y_{n+1}|$ appears very correlated with $Y_{n-1}|$ since 1_ and
  2)

The partial autocorrelation function (PACF) can be thought of as the correlation between two points that are separated by some number of periods n, BUT with the effect of the intervening correlations removed.

The PACF removes the intervening correlation between the samples

STATSMODELS

http://statsmodels.sourceforge.net/stable/tsa.html

PROPERTIES: STATIONARITY

The concept of **stationarity** is very important in modeling non-independent data

Many results which holds for independent random variables (law of large numbers, central limit theorem, ...) hold for stationary random variables.

A time serie is said to be stationnary if

- **Constant mean**: The mean of the series should not be a function of time rather should be a constant.
- **Constant variance**: Homoscedasticity: The variance of the series should not a be a function of time.
- **Fix lagged covariance**: The covariance of the i th term and the (i + m) th term should only depend on i and not m.

PROPERTIES: STATIONARITY

Constant mean

PROPERTIES: STATIONARITY

Constant variance

PROPERTIES: STATIONARITY

Fix lagged covariance

PROPERTIES: STATIONARITY

- Tree rings?
- Dow Jones?
- Average Water Temp?

STATIONARITY TEST

- Dickey-Fuller Test: This is one of the statistical tests for checking stationarity.
- Here the null hypothesis is that the TS is non-stationary.
- The test results comprise of a Test Statistic and some Critical Values for difference confidence levels.
- If the 'Test Statistic' is less than the 'Critical Value', we can reject the null hypothesis and say that the series is stationary.

  For instance: the Dickey-Fuller test statistic is less than the 10% critical value, thus the TS is stationary with 90% confidence.

DICKEY FULLER TEST

- from statsmodels.tsa.stattools import adfuller
- dftest = adfuller(timeseries, autolag='AIC')

Returns: ['Test Statistic','p-value','#Lags Used','Number of Observations Used']

HOW TO MAKE A TS STATIONARY

There are 2 major reasons behind non-stationarity of a TS:

1. Trend – varying mean over time. On average, the number of passengers was growing over time.
2. Seasonality – variations at specific time-frames. eg people might have a tendency to buy cars in a particular month because of pay increment or festivals.

So remove trend and seasonality and apply prediction on resulting TS

TREND

Estimating & Eliminating Trend

One of the first tricks to reduce trend can be transformations such as log, square root, cube root, etc. Lets take a log transform here for simplicity:

NOTEBOOK

- Load the Monthly milk production dataset
- Fit a linear regression line
- Remove Moving Average
- Fit a linear regression line
- plot before after

DIFFERENCING

Differencing – taking the difference with a particular time lag

One of the most common methods of dealing with both trend and seasonality is differencing. In this technique, we take the difference of the observation at a particular instant with that at the previous instant. This mostly works well in improving stationarity.

DECOMPOSING

Both trend and seasonality are modeled separately

```
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(ts_log)

trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.resid

# Then plot
plt.subplot(411)
plt.plot(ts_log, label='Original')
plt.legend(loc='best')
plt.subplot(412)
plt.plot(trend, label='Trend')
plt.legend(loc='best')
plt.subplot(413)
plt.plot(seasonal,label='Seasonality')
plt.legend(loc='best')
plt.subplot(414)
plt.plot(residual, label='Residuals')
```

TRANSFORMATIONS

https://www.otexts.org/fpp/2/4

- Mathematical: Box plot
- Calendar Adjustments monthly milk production on a farm,
- Population adjustments per 1000 you remove the effect of population changes by consi
- Inflation adjustments

```
for money related series
 a price index is used. If ztzt denotes the price index and ytyt denotes the original house price
```

RESIDUALS

A good forecasting method will yield residuals with the following properties:

- The residuals are uncorrelated. If there are correlations between residuals, then there is information left in the residuals which should be used in computing forecasts.
- The residuals have zero mean. If the residuals have a mean other than zero, then the forecasts are biased.

In addition to these essential properties, it is useful (but not necessary) for the residuals to also have the following two properties.

- The residuals have constant variance.
- The residuals are normally distributed.

Look at

- Histograms of residuals
- QQ plots
- ACF of the residuals
- Box-Pierce test
- Ljung-Box

LINKS

- Datasets
- Seasonal ARIMA with Python
- Time Series Analysis using iPython
- Complete guide to create a Time Series Forecast
- A Complete Tutorial on Time Series Modeling in R
- A Simple Time Series Analysis Of The S&P 500 Index
- Identifying the order of differencing in an ARIMA model
- fecon235 : Computational data tools for financial economics

Others:

- https://www.otexts.org/fpp/2/5