**Time Series Analysis**

1. **What is a Time Series?**

***Time series*** *is a sequence of observations recorded at regular time intervals.*

Depending on the frequency of observations, a time series may typically be hourly, daily, weekly, monthly, quarterly and annual. Sometimes, you might have seconds and minute-wise time series as well, like, number of clicks and user visits every minute etc.

Most problems use time series data. Anything that is observed sequentially over time is a time series.

Examples of time series data include:

* Daily stock prices
* Monthly sea temperature
* Quarterly sales for a company
* Annual company profits

**Time series analysis** involves understanding various aspects about the inherent nature of the series so that you are better informed to create meaningful and accurate forecasts.

Time series analysis is the preparatory step before you develop a *forecast* of the series. The more you learn about the data, the better the forecast.

**2. The Basic Steps in a Forecasting Task**

1. **Problem definition.** Often this is the most difficult part of forecasting. Defining the problem carefully requires an understanding of the way the forecasts will be used, who requires the forecasts, and how the forecasting function fits within the organization requiring the forecasts. A forecaster needs to spend time talking to everyone who will be involved in collecting data, maintaining databases, and using the forecasts for future planning.
2. **Gathering information.** There are always at least two kinds of information required: (a) statistical data, and (b) the accumulated expertise of the people who collect the data and use the forecasts. Often, it will be difficult to obtain enough historical data to be able to fit a good statistical model. Occasionally, old data will be less useful due to structural changes in the system being forecast; then we may choose to use only the most recent data. However, remember that good statistical models will handle evolutionary changes in the system; don’t throw away good data unnecessarily.
3. **Exploratory analysis.** Always start by graphing the data. Are there consistent patterns? Is there a significant trend? Is seasonality important? Is there evidence of the presence of business cycles? Are there any outliers in the data that need to be explained by those with expert knowledge? How strong are the relationships among the variables available for analysis? Various tools have been developed to help with this analysis.
4. **Choosing and fitting models.** The best model to use depends on the availability of historical data, the strength of relationships between the forecast variable and any explanatory variables, and the way in which the forecasts are to be used. It is common to compare two or three potential models. Each model is itself an artificial construct that is based on a set of assumptions (explicit and implicit) and usually involves one or more parameters which must be estimated using the known historical data.
5. **Using and evaluating a forecasting model.** Once a model has been selected and its parameters estimated, the model is used to make forecasts. The performance of the model can only be properly evaluated after the data for the forecast period have become available. A number of methods have been developed to help in assessing the accuracy of forecasts. There are also organizational issues in using and acting on the forecasts.

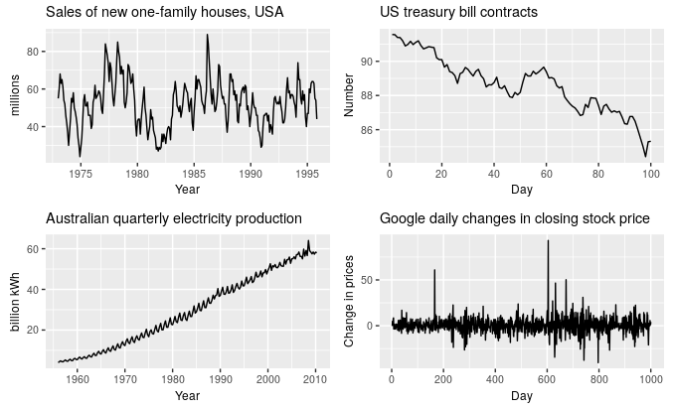
**3. Time Series Graphics**

The first thing to do is to **import the data**. They are usually in a .csv format. They can have a column with the date and multiple columns for the measured values. When having multiple samples, the dataset can have a sample\_id column and for each unique id, the dates will be repeated. This format is called **panel data**.

The next thing to do in any data analysis task is to **plot the data**. Graphs enable many features of the data to be visualized, including patterns, unusual observations, changes over time, and relationships between variables. The features that are seen in plots of the data must then be incorporated, as much as possible, into the forecasting methods to be used. Just as the type of data determines what forecasting method to use, it also determines what graphs are appropriate.

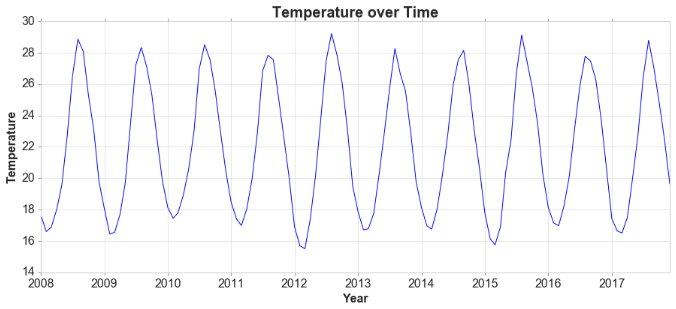
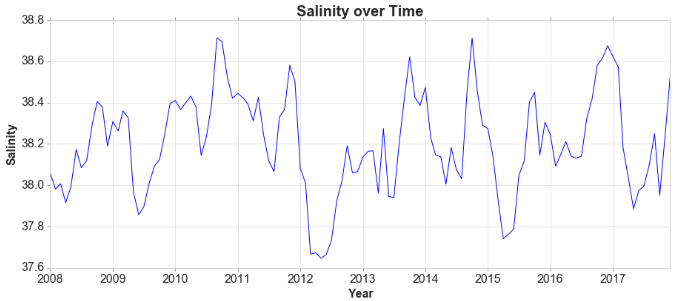
From the plots we can observe several **time series patterns**:

* **Trend.** A trend exists when there is a long-term increase or decrease in the data. It does not have to be linear. Sometimes we will refer to a trend as “changing direction”, when it might go from an increasing trend to a decreasing trend.
* **Seasonal.** A seasonal pattern occurs when a time series is affected by seasonal factors such as the time of the year or the day of the week. Seasonality is always of a fixed and known frequency.
* **Cyclic.** A cycle occurs when the data exhibit rises and falls that are not of a fixed frequency. If the patterns are not of fixed calendar based frequencies, then it is cyclic. Because, unlike the seasonality, cyclic effects are typically influenced by the business and other socio-economic factors.



1. The monthly housing sales (top left) show strong seasonality within each year, as well as some strong cyclic behavior with a period of about 6–10 years. There is no apparent trend in the data over this period.
2. The US Treasury bill contracts (top right) show results from the Chicago market for 100 consecutive trading days in 1981. Here there is no seasonality, but an obvious downward trend. Possibly, if we had a much longer series, we would see that this downward trend is actually part of a long cycle, but when viewed over only 100 days it appears to be a trend.
3. The Australian quarterly electricity production (bottom left) shows a strong increasing trend, with strong seasonality. There is no evidence of any cyclic behavior here.
4. The daily change in the Google closing stock price (bottom right) has no trend, seasonality or cyclic behavior. There are random fluctuations which do not appear to be very predictable, and no strong patterns that would help with developing a forecasting model.

**3.1 Time Plots**



For time series data, the obvious graph to start with is a time plot. That is, the observations are plotted against the time of observation, with consecutive observations joined by straight lines.

In these plots we can see:

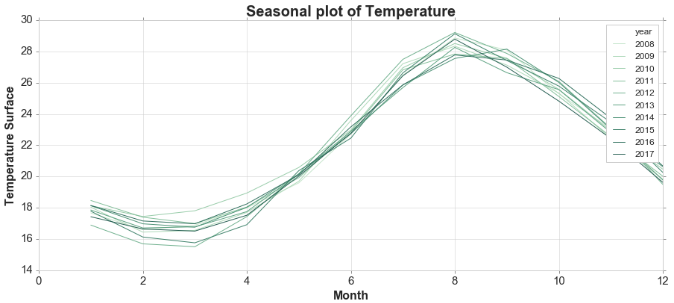
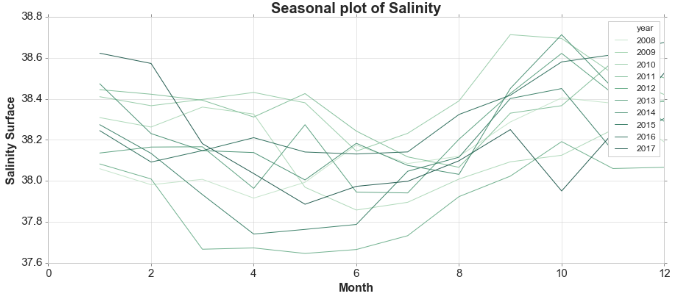
* Strange values, like outliers and values which need to be explained because they differ from the seasonality or trend.
* Periods of missing observations.
* Fluctuations of the data. For example in the salinity, there is a fluctuation which increases in 2008.
* The trend, seasonality and cyclic behavior. For example the temperature had strong seasonality and no trend or cyclic behavior.

In the case of missing observations we can fill them with various methods like:

* Backward Fill
* Linear Interpolation
* Quadratic Interpolation
* Mean of nearest neighbors
* Mean of seasonal counterparts

**3.2 Seasonal Plots**

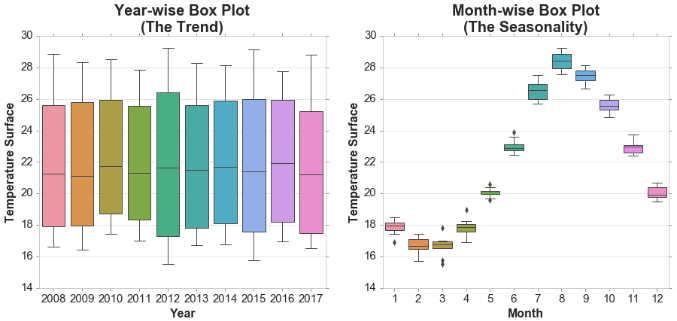
A seasonal plot is similar to a time plot except that the data are plotted against the individual “seasons” in which the data were observed. The data for each season are overlapped.

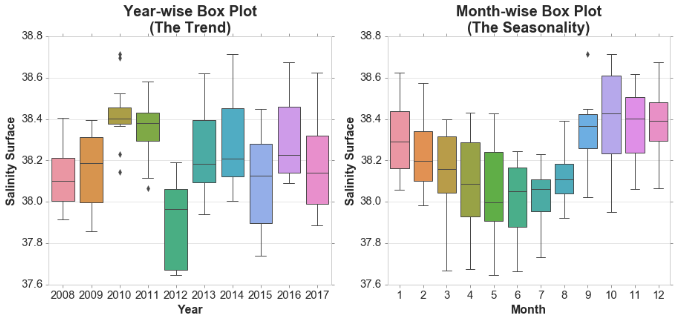


In these plots we can instantly see:

* More clearly the seasonal pattern if it exists.
* Identify years in which the pattern changes.
* Identify large jumps or drops.

**3.3 Box Plots for the Trend and the Seasonality**



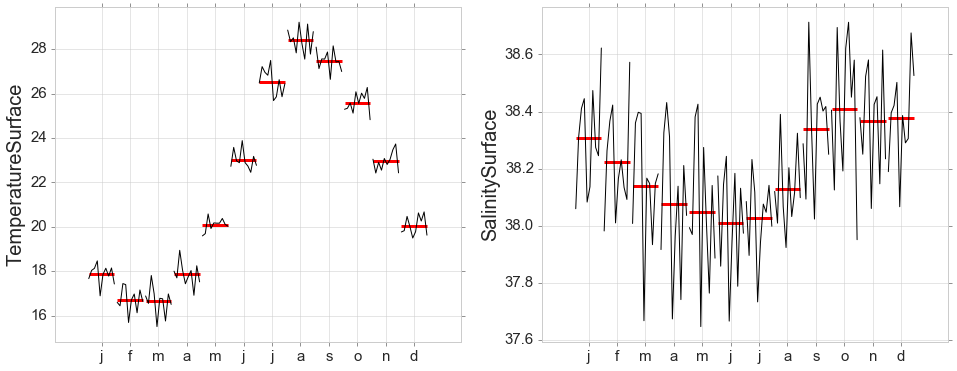


In these plots we can see:

* More clearly the trend and the seasonality. For example, although that we could say salinity had seasonality from the previous plots, now it is more evident that it has.
* Years or months with outliers.
* Compare years or months easier.

**3.4 Seasonal Subseries Plots**

An alternative plot that emphasizes the seasonal patterns is where the data for each season are collected together in separate mini time plots.



It performs a groupby to see more clearly the months of seasonality.

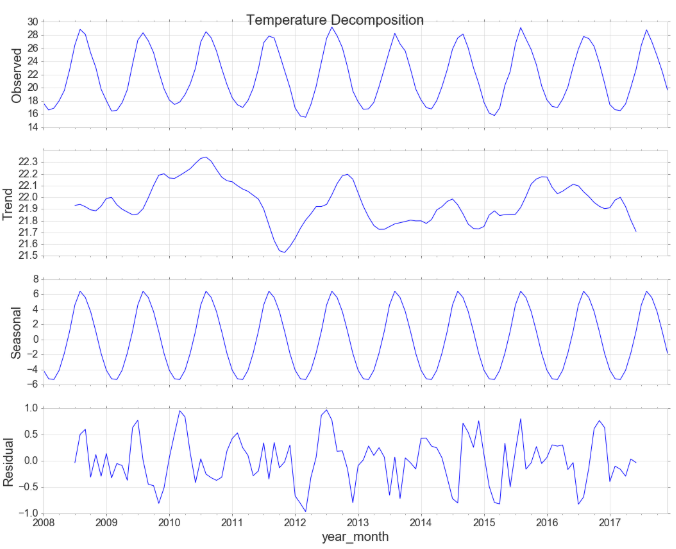
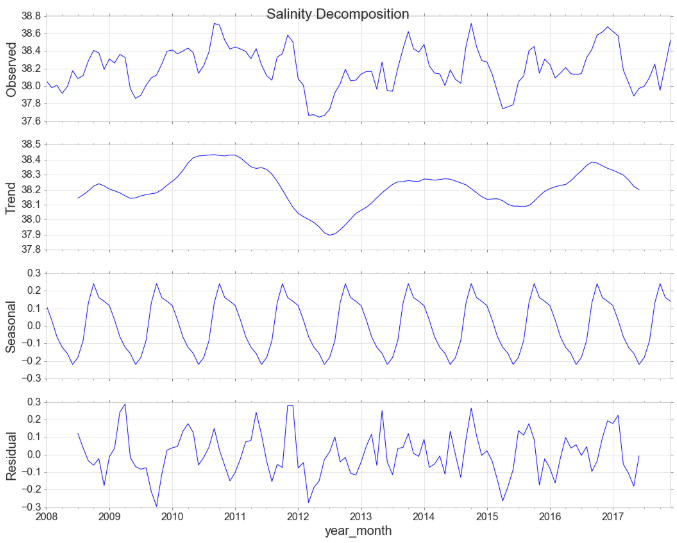
**4. Time Series Components**

Time series data can exhibit a variety of patterns, and it is often helpful to split a time series into several components, each representing an underlying pattern category. When we decompose a time series into components, we usually combine the trend and cycle into a single **trend-cycle** component (sometimes called the **trend** for simplicity). Thus we think of a time series comprising three components: a trend-cycle component, a seasonal component, and a remainder component (containing anything else in the time series).

If we assume an additive decomposition, then we can write

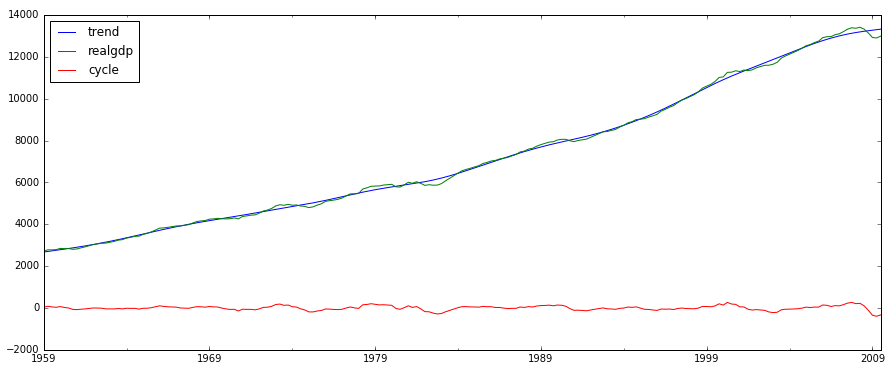
, where yt is the data, St is the seasonal component, Tt is the trend-cycle component, and Rt is the remainder component, all at period t. Alternatively, a multiplicative decomposition would be written as

The additive decomposition is the most appropriate if the magnitude of the seasonal fluctuations, or the variation around the trend-cycle, does not vary with the level of the time series. In other words, we apply an **additive** model when it seems that the trend is more linear and the seasonality and trend components seem to be constant over time. When the variation in the seasonal pattern, or the variation around the trend-cycle, appears to be proportional to the level of the time series, then a multiplicative decomposition is more appropriate. In other words, A **multiplicative** model is more appropriate when we are increasing (or decreasing) at a non-linear rate Multiplicative decompositions are common with economic time series. This decomposition is called **ETS Decomposition** (Error/Trend/Seasonality).



Plotting all these figures together gives huge instant insights on the data. For example although it wasn’t so obvious before, salinity has a clear seasonal pattern.

The **Hodrick-Prescott** filter separates a time-series yt into a **trend component** Tt and a cyclical component Ct. For monthly data lambda=129,600, for annual lambda=6.25, and for quarterly data is 1600.



**4.1 Detrend**

Detrending is the statistical or mathematical operation of removing trend from the series. Detrending is often applied to remove a feature thought to distort or obscure the relationships of interest. In climatology, for example, a temperature trend due to urban warming might obscure a relationship between cloudiness an air temperature. Detrending is also sometimes used as a preprocessing step to prepare time series for analysis by methods that assume stationarity.

**5. Stationarity**

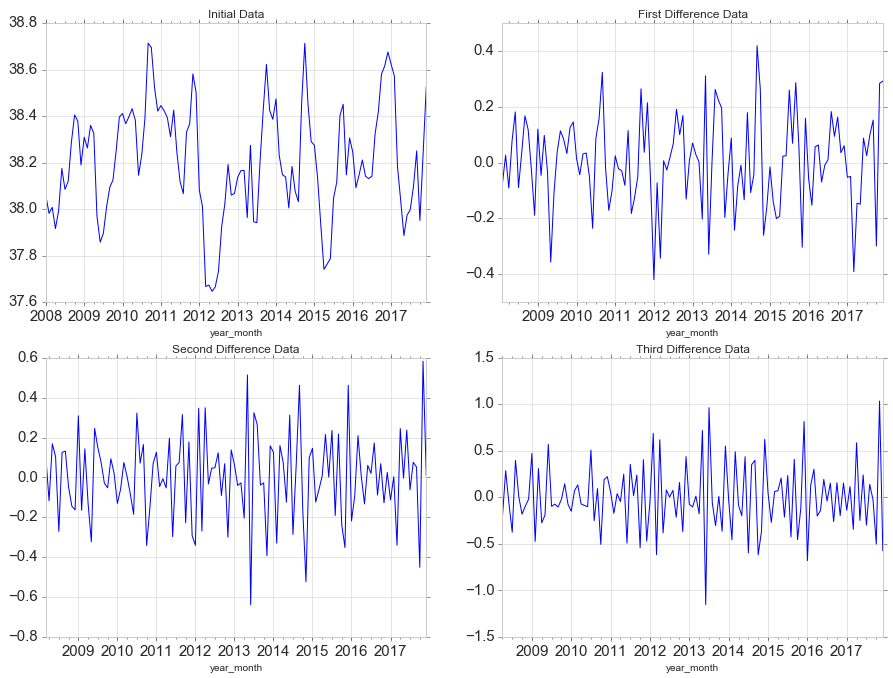
A stationary time series is one whose properties do not depend on the time at which the series is observed. Thus, time series with trends, or with seasonality, are not stationary — the trend and seasonality will affect the value of the time series at different times. On the other hand, a white noise series is stationary — it does not matter when you observe it, it should look much the same at any point in time. A time series with cyclic behavior (but with no trend or seasonality) is stationary. This is because the cycles are not of a fixed length, so before we observe the series we cannot be sure where the peaks and troughs of the cycles will be. That is, the statistical properties of the series like mean, variance and autocorrelation are constant over time.

Most statistical forecasting methods are designed to work on a stationary time series. The first step in the forecasting process is typically to do some transformation to convert a non-stationary series to stationary. Forecasting a stationary series is relatively easy and the forecasts are more reliable. We know that linear regression works best if the predictors (X variables) are not correlated against each other. So, stationarizing the series solves this problem since it removes any persistent autocorrelation, thereby making the predictors (lags of the series) in the forecasting models nearly independent.

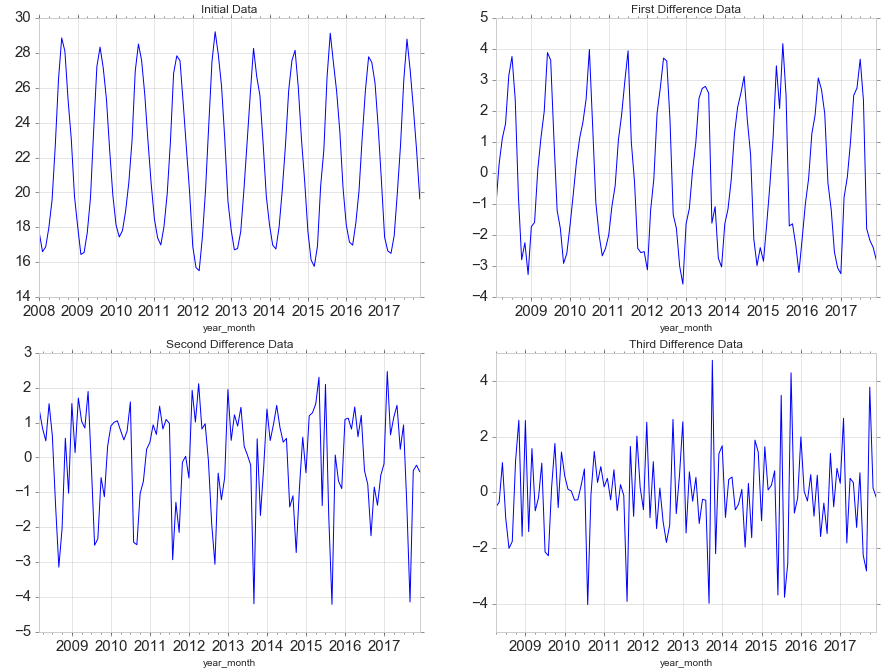
In order to **make** a time series **stationary** we can:

* Differencing the series once or more times (subtracting the next value by the current value)
* Take the log of the series (helps to stabilize the variance of a time series.)
* Take the nth root of the series
* Combinations of the above

**Apply differencing for Salinity**:



**Apply differencing for Temperature:**



We can see that temperature which is strongly seasonal needs at least 3rd order differencing to look like stationary.

To **test if** a time series **is stationary** we can:

* Look at the time plot.
* Split the series into 2 parts and compute descriptive statistics. If they differ, then it is not stationary.
* Perform statistical tests called **Unit Root Tests** like **Augmented Dickey Fuller test** (ADF Test), Kwiatkowski-Phillips-Schmidt-Shin – KPSS test (trend stationary), and Philips Perron test (PP Test).

The most commonly used is the **ADF** **test**, where the null hypothesis is that the time series possesses a unit root (or random walk with drift) and is non-stationary. So, if the P-Value in ADF test is less than the significance level (0.05), you reject the null hypothesis.

The **KPSS test**, on the other hand, is used to test for trend stationarity. The null hypothesis and the P-Value interpretation is just the opposite of ADH test.

**White Noise** is a stationary series that shows no autocorrelation. Its mean and variance does not change over time and its mean is 0. In white noise there is no pattern whatsoever.

**Granger Causality Tests**

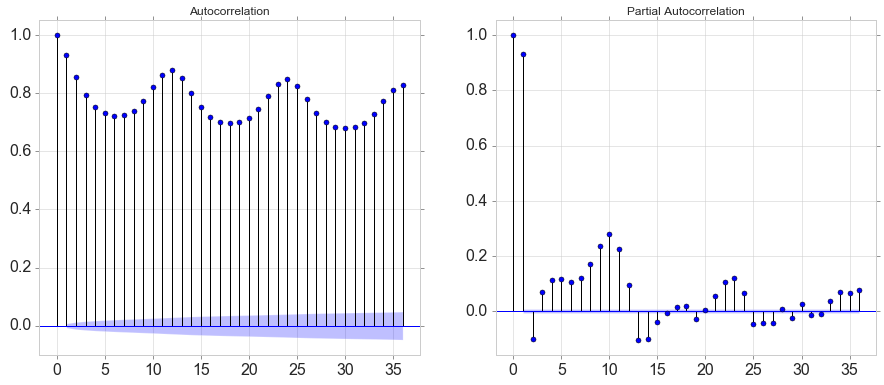
The Granger causality test is a hypothesis test to determine if one time series is useful in forecasting another. While it is fairly easy to measure correlations between series - when one goes up the other goes up, and vice versa - it's another thing to observe changes in one series correlated to changes in another after a consistent amount of time. This *may* indicate the presence of causality, that changes in the first series influenced the behavior of the second. However, it may also be that both series are affected by some third factor, just at different rates. Still, it can be useful if changes in one series can predict upcoming changes in another, whether there is causality or not. In this case we say that one series "Granger-causes" another.

In the case of two series, *y* and *x*, the null hypothesis is that lagged values of *x* do *not* explain variations in *y*. In other words, it assumes that *xt* doesn’t Granger-cause *yt*.

**6. Autocorrelation**

Just as correlation measures the extent of a linear relationship between two variables, **autocorrelation** measures the linear relationship between *lagged values* of a time series, for example between yt and yt-1. If a series is significantly autocorrelated, that means, the previous values of the series (lags) may be helpful in predicting the current value.

**Partial autocorrelations** measure the linear dependence of one variable after removing the effect of other variable(s) that affect both variables. That is, the partial autocorrelation at lag *k* is the autocorrelation between *yt* and *yt*+yt+k that is not accounted for by lags 1 through *k*−1. We essentially plot out the relationship between the previous day’s/month’s residuals versus the real values of the current day. In general we expect the partial autocorrelation to drop off quite quickly.

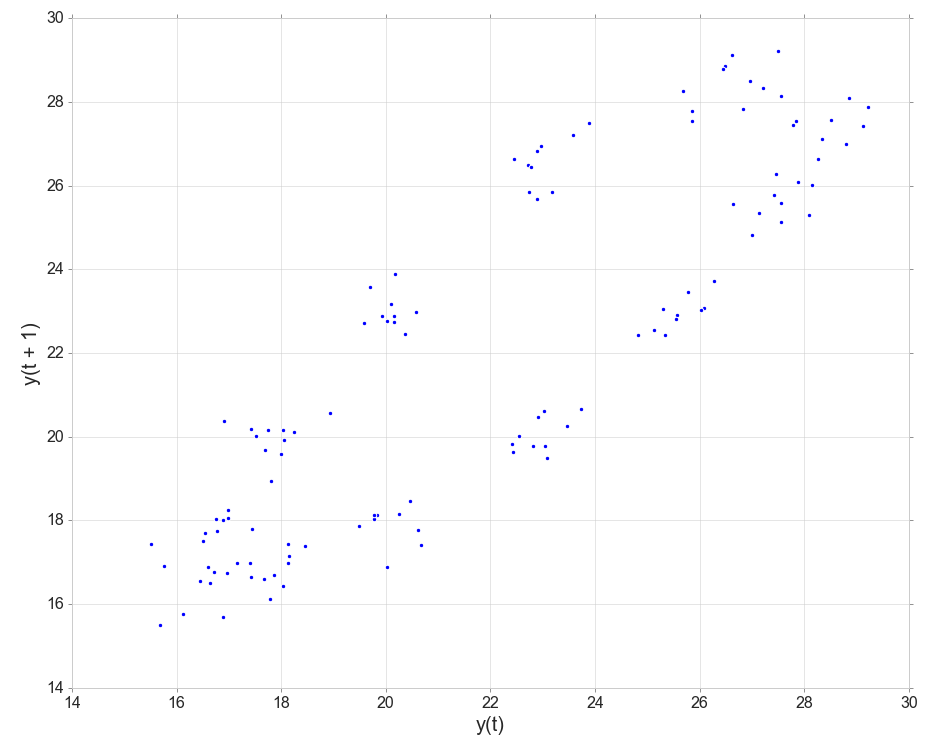


**7. Lag Scatter Plots**

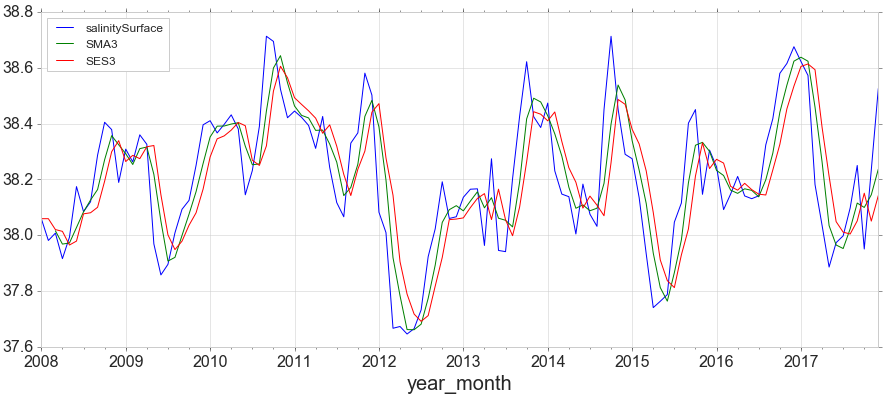
A useful type of plot to explore the relationship between each observation and a lag of that observation is called the scatter plot.

Pandas has a built-in function for exactly this called the lag plot. It plots the observation at time t on the x-axis and the lag1 observation (t-1) on the y-axis.

* If the points cluster along a diagonal line from the bottom-left to the top-right of the plot, it suggests a positive correlation relationship.
* If the points cluster along a diagonal line from the top-left to the bottom-right, it suggests a negative correlation relationship.
* Either relationship is good as they can be modeled.
* More points tighter in to the diagonal line suggests a stronger relationship and more spread from the line suggests a weaker relationship.
* A ball in the middle or a spread across the plot suggests a weak or no relationship.



**8. Smoothing - Moving Averages, Exponentially Weighted Moving Averages, Double Exponential Smoothing and Triple Exponential Smoothing**



**Moving average** is the estimation of the trend-cycle at time t, and is obtained by averaging the values of the time series within k periods of t. Observations that are nearby in time are also likely to be close in value. Therefore, the average eliminates some of the randomness in the data, leaving a smooth trend-cycle component.

The basic SMA has some weaknesses:

* Smaller windows will lead to more noise, rather than signal.
* It will always lag by the size of the window.
* It will never reach to full peak or valley of the data due to the averaging.
* Does not really inform you about possible future behavior, all it really does is describe trends in your data.
* Extreme historical values can skew SMA significantly.

**Exponentially Weighted Moving Averages** (EWMA) will allow us to reduce the lag effect from SMA and it will put more weight on values that occurred more recently (by applying more weight to the more recent values, thus the name). The amount of weight applied to the most recent values will depend on the actual parameters used in the EWMA and the number of periods given a window size.

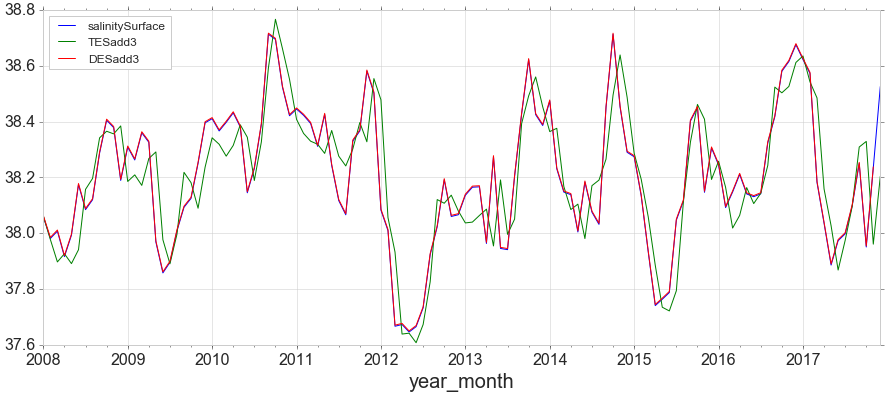
**Double Exponential Smoothing (DES)**



Where Simple Exponential Smoothing employs just one smoothing factor α (alpha), Double Exponential Smoothing adds a second smoothing factor *β* (beta) that addresses trends in the data. Like the alpha factor, values for the beta factor fall between zero and one (0<*β*≤10<β≤1). The benefit here is that the model can anticipate future increases or decreases where the level model would only work from recent calculations.

We can also address different types of change (growth/decay) in the trend. If a time series displays a straight-line sloped trend, you would use an **additive** adjustment. If the time series displays an exponential (curved) trend, you would use a **multiplicative** adjustment.

**Triple Exponential Smoothing**



Triple Exponential Smoothing, the method most closely associated with Holt-Winters, adds support for both trends and seasonality in the data.

**9. Forecasting**

Forecasting procedure:

1. Choose a model
2. Split data into train and test sets
3. Fit model on training test
4. Evaluate model on test set
5. Refit model on entire data set
6. Forecast future unknown data

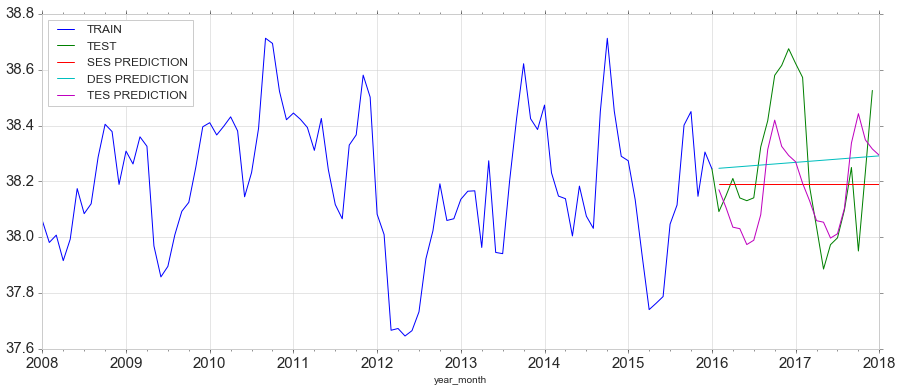
There are two types of forecasting. If you use only the previous values of the time series to predict its future values, it is called **Univariate Time Series Forecasting**. If you use predictors other than the series (a.k.a exogenous variables) to forecast it is called **Multi Variate Time Series Forecasting**.

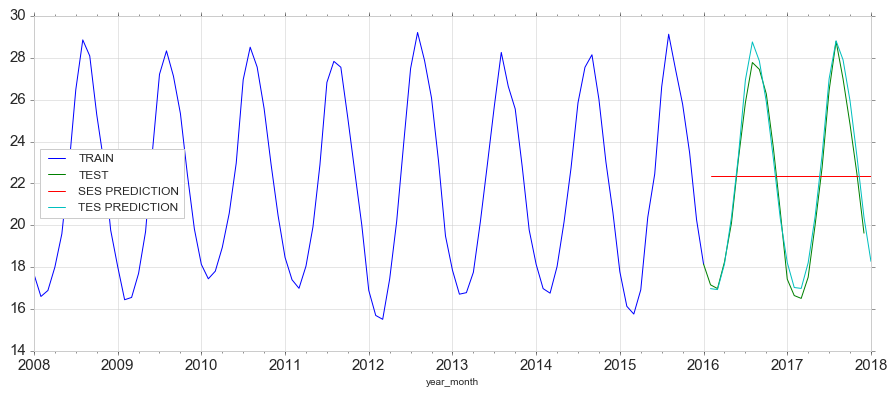
**Split data into train and test**



* There is no default value, but about 20% of total data for test is a good value.
* The test set should ideally be at least as large as the maximum forecast horizon required.
* The longer the horizon, the less accurate the prediction.

**We can use Simple, Double and Triple Exponential Smoothing to forecast.**





And **evaluate** with Root Mean Square Error. For example, for temperature we have 4.069 error for SES and 2.294 for TES. DES was way off and I chose to not plot it.

**10. ARIMA**

ARIMA, short for ‘Auto Regressive Integrated Moving Average’ is actually a class of models that ‘explains’ a given time series based on its own past values, that is, its own lags and the lagged forecast errors, so that equation can be used to forecast future values. Any ‘non-seasonal’ time series that exhibits patterns and is not a random white noise can be modeled with ARIMA models.

* ARIMA performs very well when working with a time series where the data is directly related to the time stamp.
* ARIMA is not able to understand any outside factors, like extra features.

An ARIMA model is characterized by 3 components: p, d, q

* **p** is the order of the **AR** term. It refers to the number of lags of Y to be used as predictors.
* **q** is the order of the **MA** term (size of the moving window). It refers to the number of lagged forecast errors that should go into the ARIMA Model. If a time series, has seasonal patterns, then you need to add seasonal terms and it becomes SARIMA, short for ‘Seasonal ARIMA’.
* **d** is the number of **differencing** required to make the time series stationary

**AR(p) Model**

A pure **Auto Regressive (AR only) model** is one where Yt depends only on its own lags. That is, Yt is a function of the ‘lags of Yt’. An autoregression is run against a set of *lagged values* of order *p.* The autoregressive model specifies that the output variable depends linearly on its own previous values and on a stochastic term (noise term).

**MA(q) Model**

It refers to the number of lagged forecast errors that should go into the ARIMA Model. If a time series, has seasonal patterns, then you need to add seasonal terms and it becomes SARIMA, short for ‘Seasonal ARIMA’. We essentially set up another regression model that focuses on this residual term between a moving average and the real values.

**I - Integrated Model**

Indicates that the data values have been replaced with the difference between their values and the previous values. This means how many times we have to difference the data to get it stationary so the AR and MA components could work.

If your series is slightly under differenced, adding one or more additional AR terms usually makes it up. Likewise, if it is slightly over-differenced, try adding an additional MA term.

**ARMA Model (p,0,q)**

The ARMA model has no I term, thus it can be used on already stationary datasets. It does not uses seasonality. It captures the general trend and the predicted values are very close to the mean of the test set.

**ARIMA Model (p,d,q)**

Uses all components (p,d,q). It can be used on not already stationary dataset, which will become stationary with the d term. It does not uses seasonality. It captures the general trend and the predicted values are very close to the mean of the test set.

**SARIMA Model (p,d,q)(P,D,Q)m**

It is an extension of ARIMA that accepts additional set of parameters that specifically describe the seasonal components of the model.

**SARIMAX Model (p,d,q)(P,D,Q)m**

Using the previous approaches the only data we can use are the previous historical data. The Seasonal AutoRegressive Integrated Moving Average with eXogenous regressors introduce the idea that external features can influence a time series.

**VAR Model (p)**

There are some cases where variables affect each other. That is the case we can use Vector Autoregression. The system of equations for a 2-dimensional (2 variables) VAR(1) model is:

where the coefficient *ϕii*,*l* captures the influence of the *l*th lag of variable *yi* on itself,  
the coefficient *ϕij*,*l* captures the influence of the *l*th lag of variable *yj* on *yi*,  
and *ε*1,*t* and *ε*2,*t* are white noise processes that may be correlated.

**VARMA Model (p,q)**

Same as the ARMA model, but adds extra variables that affect each other. The system of equations for a 2-dimensional (2 variables) VARMA(1) model is:

**11. Feature Engineering for Time Series**

Alright, model needs features and all we have is a 1-dimentional time series to work with. What features can we extract?

* **Trend.** It is common for time series data to be trending.
* **Window statistics.** Max, min, mean, variance value of series in a window.
* **Date and time features.** Minute, hour, week, month etc.
* **Dummy variables.** For example, when forecasting daily sales and you want to take account of whether the day is a public holiday or not. So the predictor takes value “yes” on a **public holiday**, and “no” otherwise. A dummy variable can also be used to account for an **outlier** in the data. Rather than omit the outlier, a dummy variable removes its effect. In this case, the dummy variable takes value 1 for that observation and 0 everywhere else. Also, a dummy variable for each **day of the week** can be created.
* **Spike dummy variables.** This is a dummy variable that takes value one in the period of the intervention and zero elsewhere. A spike variable is equivalent to a dummy variable for handling an outlier. It is often necessary to model interventions that may have affected the variable to be forecast. For example, competitor activity, advertising expenditure, industrial action, and so on, can all have an effect.
* **Trading days.** The number of trading days in a month can vary considerably and can have a substantial effect on sales data. To allow for this, the number of trading days in each month can be included as a predictor. For example, the number of Mondays in month, the number of Tuesdays in month etc.
* **Distributed lags.** Predictors like advertising for previous month, advertising for two months previously etc. Shifting the series ***n*** steps back we get a feature column where the current value of time series is aligned with its value at the time ***t−n***. If we make a 1 lag shift and train a model on that feature, the model will be able to forecast 1 step ahead having observed current state of the series. Increasing the lag, say, up to 6 will allow the model to make predictions 6 steps ahead, however it will use data, observed 6 steps back. If something fundamentally changes the series during that unobserved period, the model will not catch the changes and will return forecasts with big error. So, during the initial lag selection one has to find a balance between the optimal prediction quality and the length of forecasting horizon.
* **Fourier series.** If we have monthly seasonality, and we use the first 11 Fourier terms as predictor variables. Then we will get exactly the same forecasts as using 11 dummy variables. With Fourier terms, we often need fewer predictors than with dummy variables, especially when m is large. This makes them useful for weekly data, for example, where m≈52.
* If using XGBoost, we can first detrend the data, because it handles poorly trends compared to linear models. Ideally make the series stationary and then use XGBoost. For example, we can forecast trend separately with a linear model and then blend with XGBoost.

**References**:

[1] <https://www.machinelearningplus.com/time-series/time-series-analysis-python/>

[2] <https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/>

[3] <https://otexts.com/fpp2/>

[4] <https://www.udemy.com/python-for-time-series-data-analysis/>