Anomaly detection - Part 5

One should look for what is and not what he thinks should be. -Albert Einstein

Welcome back!

- In the last module we learned about two anomaly detection techniques. **Who remembers what they were called?**

- In this module we will revisit time series modeling before discussing how ARIMA models can be used in anomaly detection. **Who remembers the two variables that time series data requires?**

DATASOCIETY: © 2023

Recap: time series data

- The simplest forms of time series data consist of 2 variables:
  - Some numeric **quantity**
  - **Time step** at which that quantity has occurred
- Observations are **not necessarily independent**
- They **may be correlated** and the degree of correlation may depend on their positions in the sequence
- Only **stable series can be analyzed**

DATASOCIETY: © 2023

Module completion checklist

Objective	Complete
Preprocess time series data	
Review time series modeling basics and key components of time series	
Explain the concept of stationarity and differencing	
Explain how to measure linear relationships within time series	
Describe an autoregressive (AR) model and moving average (MA) model	

DATASOCIETY: © 2023

Import packages

- Let’s import the libraries we will be using today

```
import os
from pathlib import Path
import pandas as pd
from pandas.plotting import lag_plot
import numpy as np
import pickle
from math import sqrt
import datetime

import matplotlib.pyplot as plt
import matplotlib.dates as mdates

from statsmodels.tsa.stattools import acf
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima_model import ARIMA
from sklearn.metrics import mean_squared_error
```


DATASOCIETY: © 2023

Directory settings

- In order to maximize the efficiency of your workflow, you should encode your directory structure into `variables`
- We will use the `pathlib` library
- Let the `main_dir` be the variable corresponding to your course materials folder
- `data_dir` be the variable corresponding to your `data` folder

```
# Set 'main_dir' to location of the project folder
from pathlib import Path
home_dir = Path(".").resolve()
main_dir = home_dir.parent.parent
print(main_dir)
```

```
data_dir = str(main_dir) + "/data"
print(data_dir)
```


DATASOCIETY: © 2023

Load PJM Energy dataset

- Let's read in the `PJME_hourly.csv` file from the data folder

```
pjm_energy = pd.read_csv(str(data_dir) + "/PJME_hourly.csv")
print(pjm_energy.head())
```

```
                 Datetime  PJME_MW
0  2002-12-31 01:00:00  26498.0
1  2002-12-31 02:00:00  25147.0
2  2002-12-31 03:00:00  24574.0
3  2002-12-31 04:00:00  24393.0
4  2002-12-31 05:00:00  24860.0
```

- Time series data analysis requires dates and times to be formatted properly
- Currently the `date` column is of type `object`. We will convert it into `datetime` format

```
pjm_energy['Datetime'] = pd.to_datetime(pjm_energy['Datetime'])
pjm_energy.dtypes
```

```
Datetime    datetime64[ns]
PJME_MW            float64
dtype: object
```


DATASOCIETY: © 2023

Extract Date from Datetime

- To reduce the complexity and better understand the working of the model we implement today, we will convert the time step quantity from **hourly** to **daily**.
- We will sort the dates and filter the dataset to include only for year 2018.

```
pjm_energy = pjm_energy.sort_values(by=['Datetime'])
pjm_energy = pjm_energy[pjm_energy['Datetime'] > '2018-01-01 00:00:00']
pjm_energy.shape
```

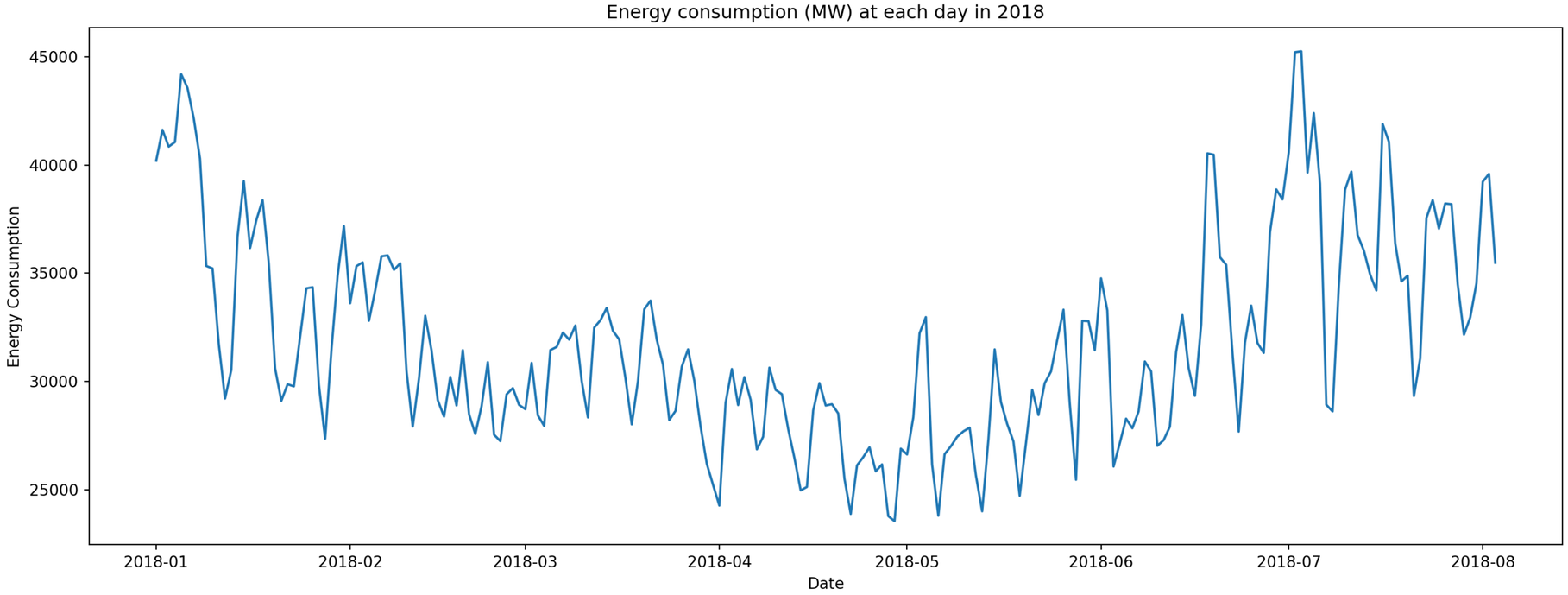
```
(5135, 2)
```

- Then, we will extract the **dates** from the datetime variable and the calculate the mean of the energy consumption for each day

```
pjm_energy['Date'] = pjm_energy['Datetime'].dt.date
pjm_energy_datewise = pjm_energy.groupby(by=['Date'])['PJME_MW'].mean()
```

Visualize the Energy consumption for each day

```
pjm_energy_datewise.plot(x='Date', y='PJME_MW', figsize=(17,6))
plt.xlabel('Date')
plt.ylabel('Energy Consumption')
plt.title('Energy consumption (MW) at each day in 2018')
plt.show()
```



DATASOCIETY: © 2023

Module completion checklist

Objective	Complete
Preprocess time series data	✔
Review time series modeling basics and key components of time series	
Explain the concept of stationarity and differencing	
Explain how to measure linear relationships within time series	
Describe an autoregressive (AR) model and moving average (MA) model	

DATASOCIETY: © 2023

Model of a line

- **Simple linear regression**

$$Y = \beta_0 + \beta_1 X + \epsilon$$

  - Y: quantity we would like to predict (i.e., target variable)
  - X: quantity we use as input to predict Y (i.e., predictor variable)
  - β_0: intercept of a line with `y-axis` (a.k.a. b)
  - β_1: slope of a line (a.k.a. m)
  - ϵ: error between the predicted value and the actual value of Y

DATASOCIETY: © 2023

Model of a line for time series

- **Time series**

$$Y_t = \beta_0 + \beta_1 X_t + \epsilon_t$$

  - Y_t: quantity at time t we would like to predict (i.e., target variable)
  - X_t: ...
  - The key difference between the **linear regression** model and the **time series** model is the added time component t

DATASOCIETY: © 2023

In search of conditional mean

- Most time series models are based on estimating the **conditional mean** of the time series Y_t
- The conditional mean of Y_t is also called an *expected value* of Y_t
- It's called *conditional*, because the value of Y_t depends on some quantity X_t and is represented by the following notation:

$$E(Y_t|X_t) = \mu_t$$

- If we set the error term to zero ($\epsilon_t = 0$) in our time series model equation, we will get the formula for the conditional mean of Y_t

$$E(Y_t|X_t) = \mu_t = \beta_0 + \beta_1 X_t$$

DATASOCIETY: © 2023

In search of conditional mean: generalized

- In general, X_t can really be any information known to us
- Historic data for Y (i.e., previously known values of Y) is often used as a predictor variable and often denoted as H_{t-1}
- The definition of the **conditional mean** of Y_t given historic data at time $t-1$ is

$$E(Y_t|H_{t-1}) = \mu_t$$

- Then, any time series model will boil down to an observation at time t as a sum of the **conditional mean** for time t and the **error** at time t

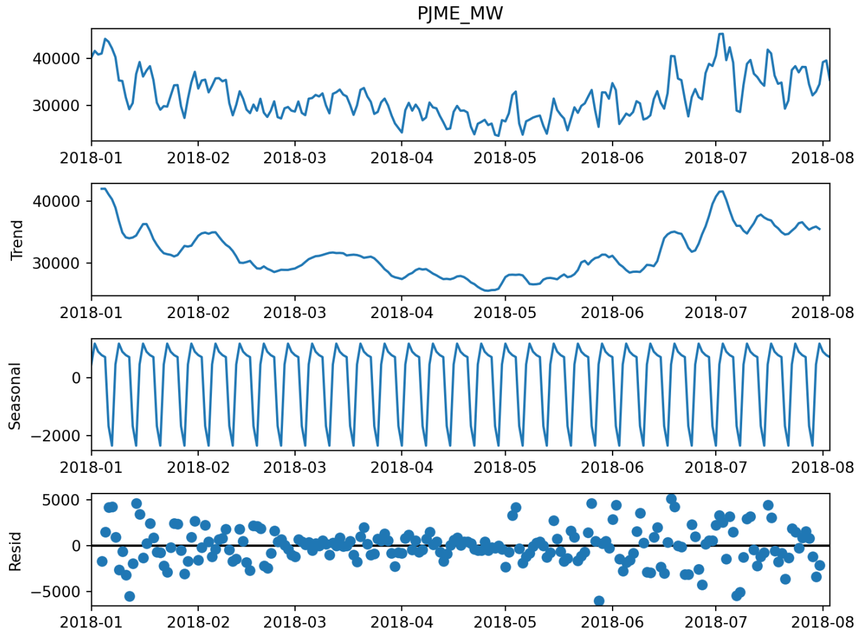
$$Y_t = \mu_t + \epsilon_t$$

DATASOCIETY: © 2023

What's the trend this season?

- Time series data comes with some *deterministic components* that don't occur in other types of data
  - The **trend** component (T_t)
  - The **seasonal** component (S_t)
  - The **residual** component (ϵ_t)
- When decomposed into the above components, the *additive* time series model now looks like this:

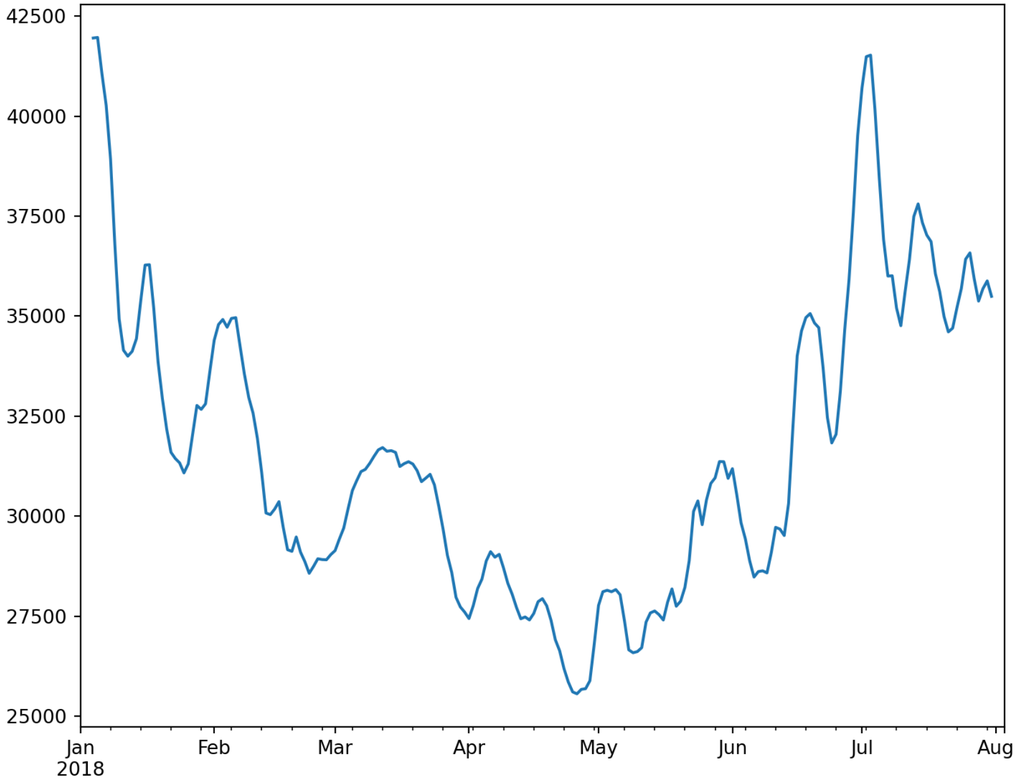
$$Y_t = T_t + S_t + \epsilon_t$$



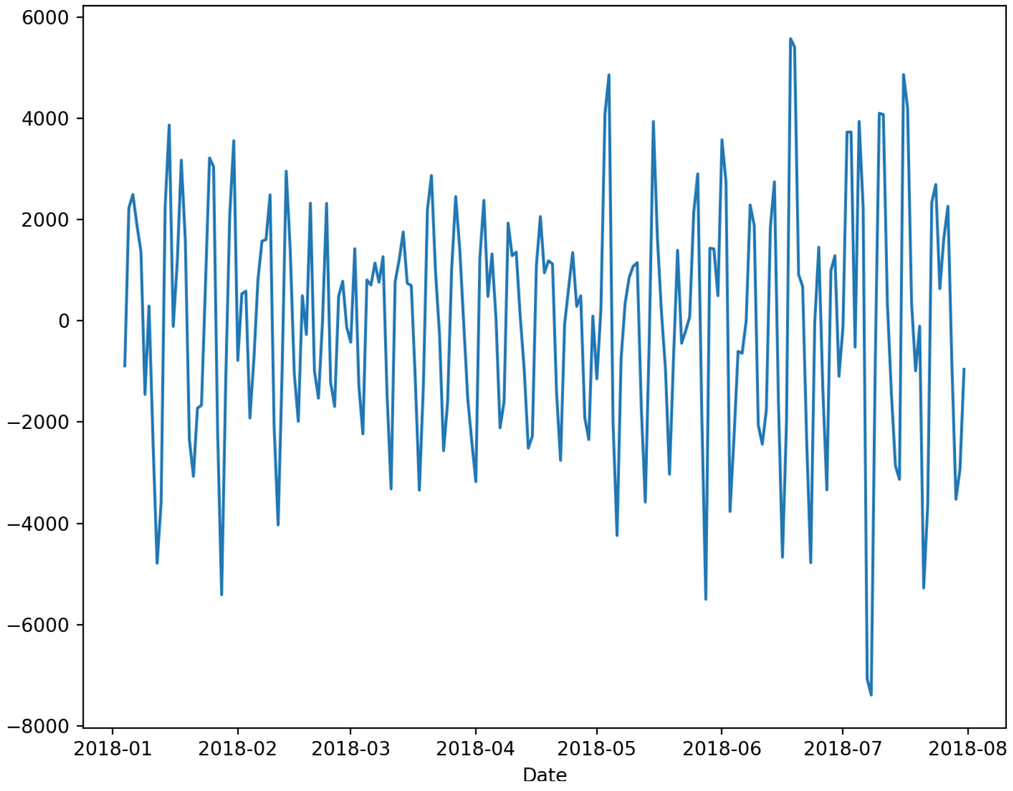
DATASOCIETY: © 2023

The trend component

- T_t: the **trend** component at time t



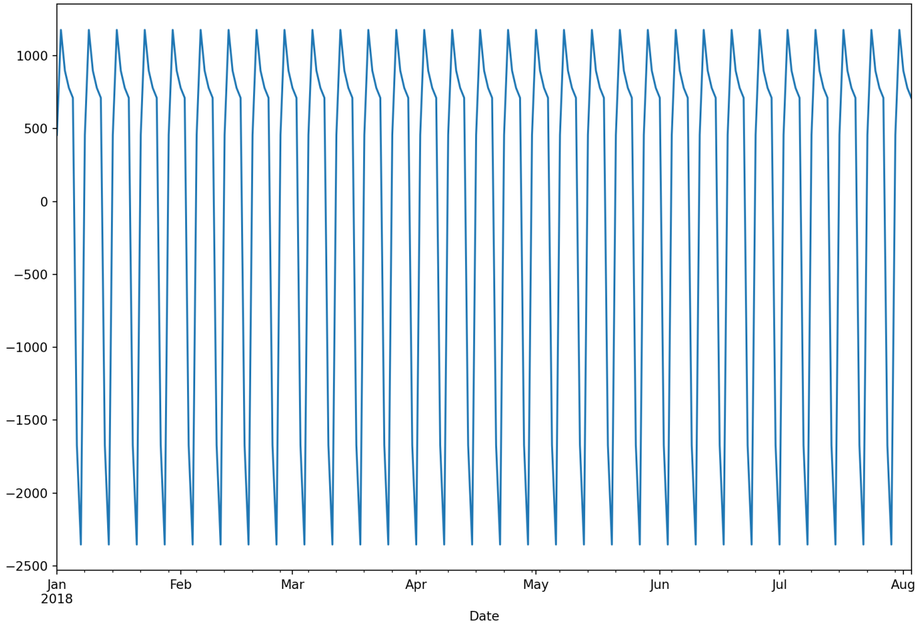
- $(Y_t - T_t)$: series without trend, or **detrended series**



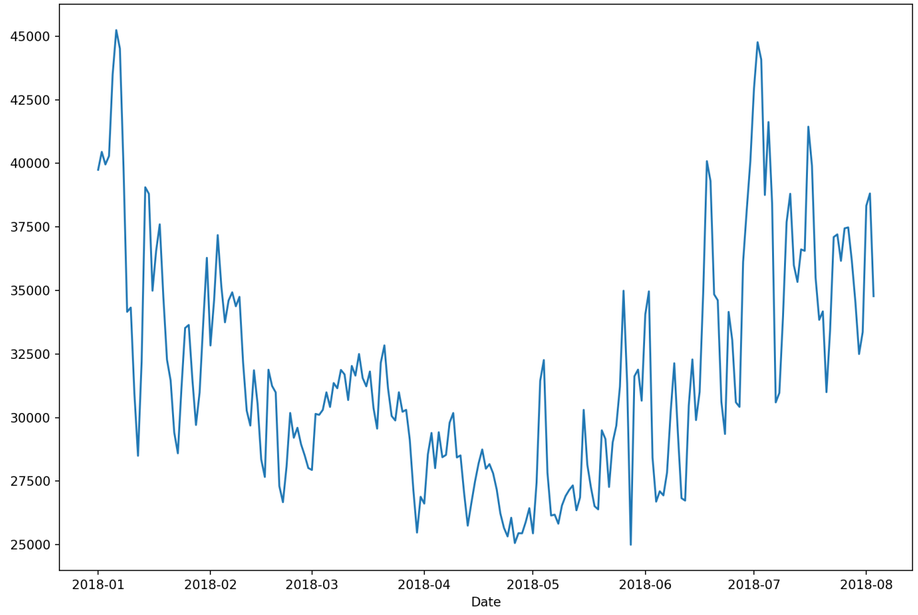
DATASOCIETY: © 2023

The seasonal component

- S_t: the **seasonal** component



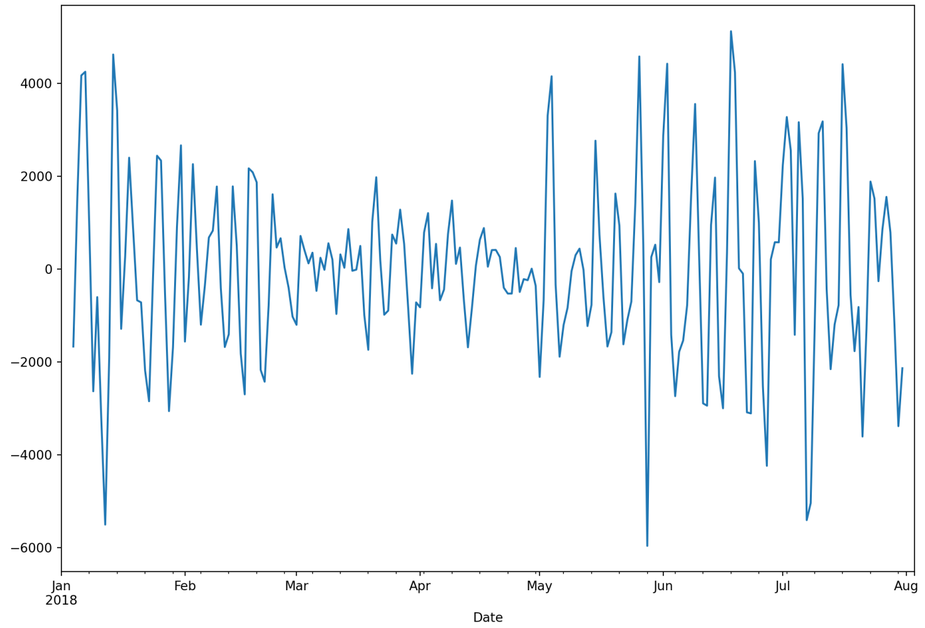
- $(Y_t - S_t)$: series without the seasonal component, or **deseasonalized series**



DATASOCIETY: © 2023

Detrended and deseasonalized series

- $(Y_t - S_t - T_t)$: series without the seasonal and trend components, or **deseasonalized** and **detrended** series
- Let's take out the trend and seasonal components from this equation $Y_t = T_t + S_t + \epsilon_t$
- What's left?
  - **noise**
  - **residuals**
  - **error**



DATASOCIETY: © 2023

Residuals, noise, and error

- **Residuals** in a time series model are what is left over after fitting a model. For many (but not all) time series models, the residuals are equal to the difference between the observations and the corresponding fitted values.
- **Noise** simply refers to random fluctuations in the time series about its typical pattern. It is that part of the residual which is not feasible to model by any other means other than a purely statistical description.
- **Error** is a component of the residual that remains after accounting for the noise.

DATASOCIETY: © 2023

What is all that noise?

- If we re-define our **conditional mean** through trend and seasonal components we get the following:

$$E(Y_t|H_{t-1}) = \mu_t = T_t + S_t$$

- This quantity is our **forecast** for Y_t at time $t-1$
- The residual term ϵ_t is a simple difference between the **forecast** and the actual value of Y_t

$$\epsilon_t = Y_t - \mu_t = Y_t - (T_t + S_t) = Y_t - T_t - S_t$$

- When we remove the trend and the seasonality components of the time series, all we have left is the residual $Y_t = \epsilon_t$

DATASOCIETY: © 2023

Knowledge check 1

DATASOCIETY: © 2023

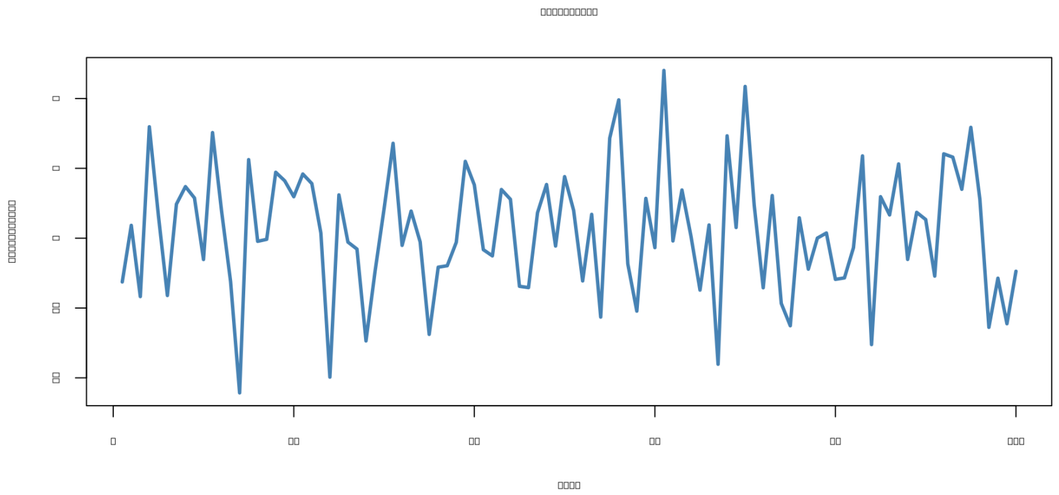
Module completion checklist

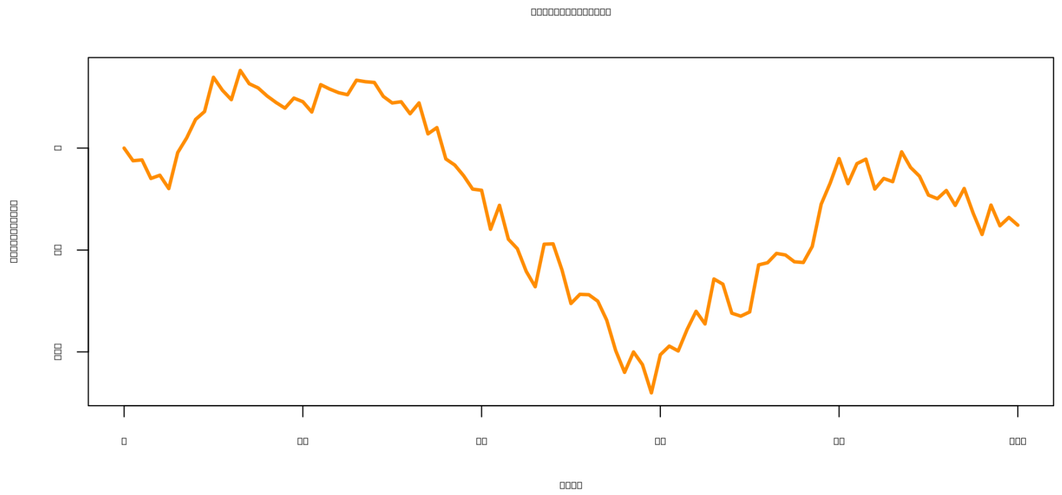
Objective	Complete
Preprocess time series data	✔
Review time series modeling basics and key components of time series	✔
Explain the concept of stationarity and differencing	
Explain how to measure linear relationships within time series	
Describe an autoregressive (AR) model and moving average (MA) model	

DATASOCIETY: © 2023

Stationarity

- A **stationary** time series is one **whose properties do not depend on the time** at which the series is observed
- A *white noise* series is **stationary**; it does not matter when you observe it, it should look the same at any point in time
- A *random walk* series is **not stationary**, but it can be made stationary by using **differencing**





DATASOCIETY: © 2023

Differencing

- **Differencing** is a method used to make non-stationary time series stationary
  - It is done by computing differences between consecutive observations
- A random walk model is stabilized and made stationary by taking first differences
  - First difference formula: $diff(Y_t) = Y_t - Y_{t-1}$
  - It is also known as lag-1 difference

DATASOCIETY: © 2023

Differencing: other types

- Sometimes the data is still non-stationary after taking first differences and it may be necessary to repeat the process again for the second time
  - This process is called **second order differencing:** $diff2(Y_t) = diff(Y_t) - diff(Y_{t-1})$
- To make seasonal data stationary, **seasonal differences** are taken
  - It is the difference between an observation and the previous observation from the same season: $diffS(Y_t) = Y_t - Y_{t-m}$, where m is the number of seasons
  - It is also known as lag-m difference
- Sometimes differencing techniques are combined (e.g., if seasonal difference doesn't make data stationary, then first difference can be applied)

DATASOCIETY: © 2023

Model decomposition steps

- What do we do if we have a model that contains a *trend* and *seasonality*? Those two components most definitely make the series **non-stationary**
  - We can decompose the series and then detrend and deseasonalize it
  - We can use one of differencing methods

1. Compute the trend component T_t ✔
2. Calculate the detrended series: $Y_t - T_t$ ✔
3. Estimate the seasonal component S_t ✔
4. Calculate deseasonalized and detrended series (i.e., estimate the error term):

   $\epsilon_t = Y_t - T_t - S_t$ ✔

DATASOCIETY: © 2023

Importance of stationarity

- A **stationary** time series has properties such as the **constant mean**, **variance**, and **autocorrelation**, which do not change with time
- This means that its statistical properties are **easy to predict**, since they will be the same in the future as they have been in the past
- The **transformations** on predictions for the stationarized series **can then be reversed** to obtain predictions for the original series

DATASOCIETY: © 2023

Importance of stationarity (cont'd)

- Most forecasting methods are based on the **assumption that the time series can be made stationary** through a host of mathematical transformations (e.g., differencing)
- Finding the **sequence of transformations needed to stationarize a time series** often provides important clues **in the search for an appropriate forecasting model**
- Stationarizing a time series through differencing (where needed) is an important part of the process of fitting an ARIMA model

DATASOCIETY: © 2023

Stationarity test

- The Augmented Dickey-Fuller test (ADF) test will test for stationarity by looking for a ***unit root***, where:

$$H_0 : \text{there is a unit root}$$
$$H_a : \text{the time series is stationary}$$

- Unit roots, at a high level, can be a signal of randomness in a model and result in spurious regressions and errant behavior

```
# Perform ADF test on original series.
result_pm = adfuller(pjm_energy_datewise)
```

- The ADF statistic is a negative number. The more negative it is, the stronger the rejection of the hypothesis that there is a unit root at some level of confidence

```
print('ADF Statistic: %f' %
result_pm[0])
```

```
ADF Statistic: -1.625863
```

```
print('p-value: %f' % result_pm[1])
```

```
p-value: 0.469657
```

- The `p-value` is very high and we fail to reject H_0
- Our original data is **not stationary**

DATASOCIETY: © 2023

Module completion checklist

Objective	Complete
Preprocess time series data	✔
Review time series modeling basics and key components of time series	✔
Explain the concept of stationarity and differencing	✔
Explain how to measure linear relationships within time series	
Describe an autoregressive (AR) model and moving average (MA) model	

DATASOCIETY: © 2023

Measuring linear relationships in a time series

- If we want to compare the *strength of linear relationships* in time series, we also use **covariance** and **correlation**
- We just need to replace x_i with X_t and y_i by Y_t

$$Corr(X_t, Y_t) = \frac{Cov(X_t, Y_t)}{\sqrt{var(X_t)var(Y_t)}}$$

- This way, we compare a sample of paired observations for every point in time

DATASOCIETY: © 2023

Autocovariance and autocorrelation

- Since time series is a collection of random variables $\{Y_t\}$ for $t = 1, 2, \ldots, n$, we only observe just one sample at each point in time
- So how do we estimate a *mean* or a *variance* at time t based on just one observation?
- We make use of **autocorrelation** and **autocovariance**
- Both are based on comparing variable at time t to itself at some different point $t - k$ in the past
  - $Cov(X_t, Y_t)$ becomes $Cov(Y_t, Y_{t-k})$
  - $Corr(X_t, Y_t)$ becomes $Corr(Y_t, Y_{t-k})$, or $r_{Y_t, Y_{t-k}}$

Autocovariance and autocorrelation (cont'd)

- **Autocovariance** and **autocorrelation** are always measured for an observation at time t with respect **some point in past** $t - k$
- The number k is known as a **lag**, because we are always looking at the same variable, just k time steps *lagging behind*
  - $Cov(Y_t, Y_{t-k}) = \gamma_k$ is autocovariance at lag k and measures the linear relationship between the original series and k-period lagged series
  - $Corr(Y_t, Y_{t-k}) = \rho_k$ is autocorrelation at lag k and is the standardized form of autocovariance at lag k

DATASOCIETY: © 2023

Measuring autocorrelation

- The plot of autocorrelation r_k vs the lag k is known as the **autocorrelation function** of the time series
- It is computed by measuring autocorrelation coefficients for all possible values of lag k a given dataset

$$r_k = \frac{\sum_{t=k+1}^{n} (Y_t - mean_Y)(Y_{t-k} - mean_Y)}{\sum_{t=1}^{n} (Y_t - mean_Y)^2}$$

DATASOCIETY: © 2023

Autocorrelation pattern

- Recall one of the key features of time series: *observations may be correlated* and the degree of correlation may depend on their positions in the sequence
- *Closer ones might be more related to each other*. Below is an example of some autocorrelation values
  - $r_0 = 1$ (always, we don't need to compute it!)
  - $r_1 = -0.3762887$
  - $r_2 = -0.04123711$
  - $r_3 = 0.02061856$
- Notice that as we are **increasing the lag**, the absolute value (i.e., the magnitude) of **autocorrelation is decreasing**

Visualize autocorrelation function

- To plot ACF, we can also use a module from `statsmodels` library called `statsmodels.graphics.tsaplots`
- The function we will use is called `plot_acf` and takes a time series as its main argument
- For more information, review the ***function documentation***

statsmodels.graphics.tsaplots.plot_acf

`statsmodels.graphics.tsaplots.`**`plot_acf`**(*x*, *ax=None*, *lags=None*, *alpha=0.05*, *use_vlines=True*, *unbiased=False*, *fft=False*, *title='Autocorrelation'*, *zero=True*, *vlines_kwargs=None*, ***kwargs*) [source]

Plot the autocorrelation function

Plots lags on the horizontal and the correlations on vertical axis.

Parameters:

x : *array_like*

Array of time-series values

ax : *Matplotlib AxesSubplot instance, optional*

If given, this subplot is used to plot in instead of a new figure being created.

lags : *int or array_like, optional*

int or Array of lag values, used on horizontal axis. Uses np.arange(lags) when lags is an int. If not provided, `lags=np.arange(len(corr))` is used.

DATASOCIETY: © 2023

Why bother?

- Goal of the time series analysis is to reduce the given time series to **noise**
  - Recall that an ideal time series without its trend and seasonal components is indeed just plain noise: $Y_t - T_t - S_t = \epsilon_t$
- In reality, we are given a time series **sample** and we would like to determine whether the given sample is coming from a **noise** process in order **to be able to predict future values of the series**
- Since *we can estimate autocorrelations and their sampling distributions*, we can *develop tests* that allow us to determine whether the time series we are analyzing is indeed originating from **noise**

DATASOCIETY: © 2023

Knowledge check 2

DATASOCIETY: © 2023

Module completion checklist

Objective	Complete
Preprocess time series data	✔
Review time series modeling basics and key components of time series	✔
Explain the concept of stationarity and differencing	✔
Explain how to measure linear relationships within time series	✔
Describe an autoregressive (AR) model and moving average (MA) model	

DATASOCIETY: © 2023

Autoregression model

- An autoregression model is used for time series where **signal** in the data can be explained by simply adding in the autoregressive components p to the modeling equation
- It does not need to have the other two parts of the $ARIMA$

DATASOCIETY: © 2023

Multiple regression vs autoregression

- An autoregressive model looks very similar to a **multiple regression** model, where X_i is a predictor variable, C is some constant term, ϕ_i is some function applied to the corresponding predictor, and ϵ is the error term

$$Y = C + \phi_1 X_1 + \phi_2 X_2 + \ldots + \phi_p X_p + \epsilon$$

- The only difference between an **autoregression** model and the above model is the **auto** nature of the regression:
  - Instead of X_i is a predictor variable, we use Y_i - past values of the Y variable itself

$$Y_t = C + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + \epsilon_t$$

DATASOCIETY: © 2023

AR(p)

- The equation below represents an **autoregressive model of order** p, because it contains p **lagged values** of Y

$$Y_t = C + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + \epsilon_t$$

- The shorthand notation is simply $AR(p)$
- In ARIMA terms, it would be $ARIMA(p, 0, 0)$

DATASOCIETY: © 2023

AR(p): interpretation

- The interpretation of $AR(p)$ is fairly simple:
  - The model takes into account only p previous values of Y to predict Y_t
  - $AR(1)$ simply uses a multiple of its own previous value Y_{t-1}, plus a constant to predict Y_t

DATASOCIETY: © 2023

AR(1): broken down

- This is an equation of $AR(1)$ model

$$Y_t = C + \phi_1 Y_{t-1} + \epsilon_t$$

- Autoregressive models of order 1 are surprisingly flexible at handling a wide range of different time series patterns and all thanks to the ϕ_1!
- Changing the parameter ϕ_1 results in different time series patterns
  - If $\phi_1 = 0$, we have our white noise model
  - If $\phi_1 = 1$ and $C = 0$, we have our random walk model
  - If $\phi_1 < 0$, Y_t oscillates around the mean
  - The range of ϕ_1 is usually restricted to be between -1 and 1
- The error term ϵ_t will only change the scale of the series, but not the general patterns

DATASOCIETY: © 2023

AR(1): assumptions and properties

- In order to see whether this time series can be modeled with $AR(1)$, it needs to fit a set of **assumptions** (just like a linear model)
  - The errors ϵ_t are independent - they form a normal distribution with $\mu = 0$ and constant $\sigma^2_{\epsilon_t}$
  - Properties of the errors ϵ_t are independent of Y_t
  - The series is stationary and ϕ_1 falls within $(-1, 1)$ range
- If the above assumptions are fulfilled, the series will have some nice **properties**
  - The slope of the $AR(1)$ model is just ϕ_1 and it is also autocorrelation of `lag = 1`
- The `ACF` plot has **distinct pattern**:
  - For a positive value of ϕ_1, the `ACF` exponentially decreases to 0 as the lag increases
  - For negative ϕ_1, the magnitude of `ACF` also exponentially decreases to 0 as the lag increases, but the signs alternate

Moving average model

- A moving average model (MA) is also a special case of an $ARIMA(0, 0, q)$
- It's used for time series where the **signal** in the data can be explained by simply adding in the moving average components q to the modeling equation
- It does not need to have the other two parts of the $ARIMA$

DATASOCIETY: © 2023

AR vs MA

- A moving average model looks very similar to an **autoregressive** model, where Y_i - past values of the Y variable itself, C is some constant term, ϕ_i is some function applied to the corresponding predictor, and ϵ is the error term

$$Y_t = C + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + \epsilon_t$$

- The only difference between **moving average** model and the above model is the ϵ_t:
  - Instead of past values of Y_i is predictor variables, we use past forecast errors

$$Y_t = C + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q}$$

DATASOCIETY: © 2023

MA(q)

- The equation below represents an **moving average model of order** q, because it contains q **past values of** of ϵ

$$Y_t = C + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q}$$

- The shorthand notation is simply $MA(q)$
- In ARIMA terms, it would be $ARIMA(0, 0, q)$

DATASOCIETY: © 2023

MA(q): interpretation

- The interpretation of $MA(q)$ is not as intuitive as the $AR(p)$ model interpretation, since we are using the **errors of previous forecast values for prediction** and those errors technically were not observed yet
- We look at **every value of Y_t** as **a weighted moving average of the past few forecast errors**
  - Calculate your forecast for period 1
  - Subtract the forecast value from the actual value for that period to generate the error for period 1
  - Use that error for period 1 to calculate the forecast for period 2, and so on
- *Note: MA model is NOT the same as `MA smoothing`*, the $MA(q)$ is used for predictions and `MA` smoothing is used to estimate the trend-cycle component of the series!

DATASOCIETY: © 2023

MA(1): assumptions and properties

- In order to see whether a time series can be modeled with $MA(1)$, it needs to fit a set of **assumptions** (just like a linear model!)
  - The errors ϵ_t are independent, they form a normal distribution with $\mu = 0$ and constant $\sigma^2_{\epsilon_t}$
  - A moving average term in a time series model is a past error multiplied by a coefficient
  - The series is stationary and θ_1 falls within $(-1, 1)$ range
- If the above assumptions are fulfilled, the series will have some nice **properties**
  - The mean of the Y_t model is just C
  - The variance of Y_t is $\sigma^2 = \sigma^2_{\epsilon_t}(1 + \theta^2_1)$
- The `ACF` plot has **distinct pattern**:
  - The only nonzero value in the `ACF` is for `lag = 1`, all other autocorrelations should be `0`

Knowledge check 3

Exercise 1

DATASOCIETY: © 2023

Module completion checklist

Objective	Complete
Preprocess time series data	✔
Review time series modeling basics and key components of time series	✔
Explain the concept of stationarity and differencing	✔
Explain how to measure linear relationships within time series	✔
Describe an autoregressive (AR) model and moving average (MA) model	✔

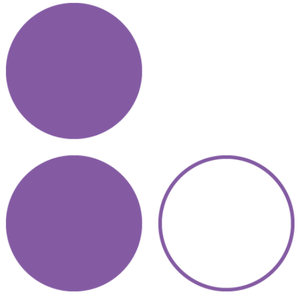
DATASOCIETY: © 2023

Summary

- In this course we learned:
  - Different models like DBSCAN, Decision trees, and Isolation forests and how they are used to perform anomaly detection
- Refer to this ***link*** to explore how ARIMA models can be used in anomaly detection

DATASOCIETY: © 2023

Congratulations on completing this module!



DATASOCIETY: © 2023