DATA SOCIETY®

Time series day 1

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

Module completion checklist

Objective	Complete
Introduce forecasting and time series analysis use cases	
Review key quantitative forecasting methods	
Introduce data used in time series analysis	
Review important features of time series data	
Visualize time series	
Explain time series modeling basics and key components of time series	
Explain how to measure linear relationships within time series	
Estimate and visualize autocorrelation of time series	

What is forecasting?

forecast verb

fore·cast | \ fȯr-ˌkast; fȯr-kast\

forecast *also* **forecasted**; **forecasting**

Definition of *forecast* (Entry 1 of 2)

transitive verb

1 a : to calculate or predict (some future event or condition) usually as a result of study and analysis of available pertinent data

// The company is *forecasting* reduced profits.

Source: ***Merriam-Webster Online Dictionary***

Why forecasting?

- To provide basis for decision making and planning
- To make decisions that depend on **future** uncertainties
- To reduce risk associated with **future** uncertainties
- …

Forecasting use cases: marketing

- Analyzing and predicting:
  - Demand
  - Market share
  - Prices and trends in new product development

Forecasting use cases: production

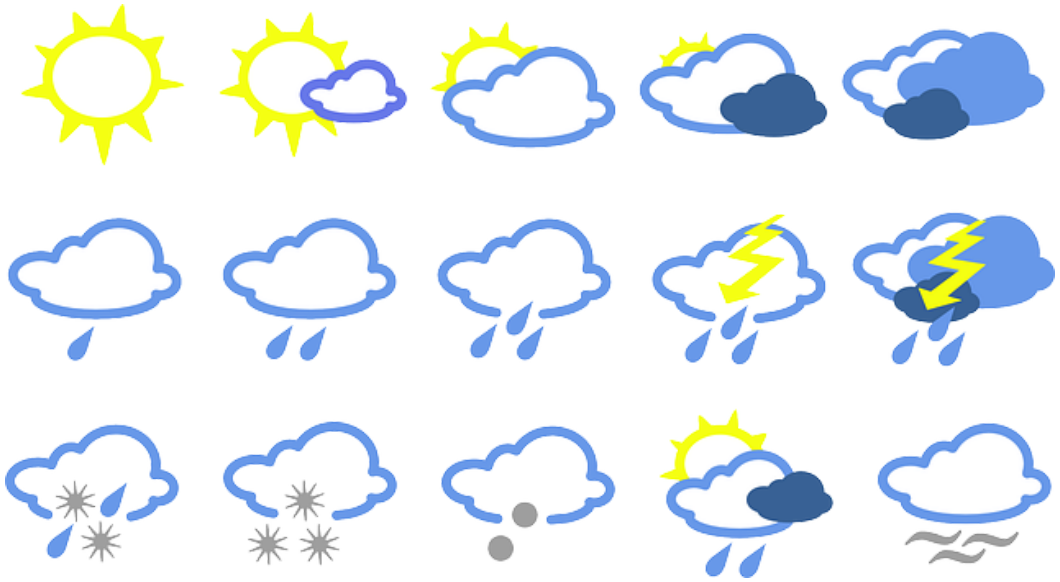
- Analyzing and predicting:
  - Product demand
  - Material requirements
  - Maintenance requirements

Forecasting use cases: finance

- Analyzing and predicting:
  - Cash flows
  - Stock returns
  - Bond rates
  - Interest rates
  - Exchange rates

Forecasting use cases: weather

- Analyzing and predicting:
  - Temperature
  - Humidity
  - Precipitation

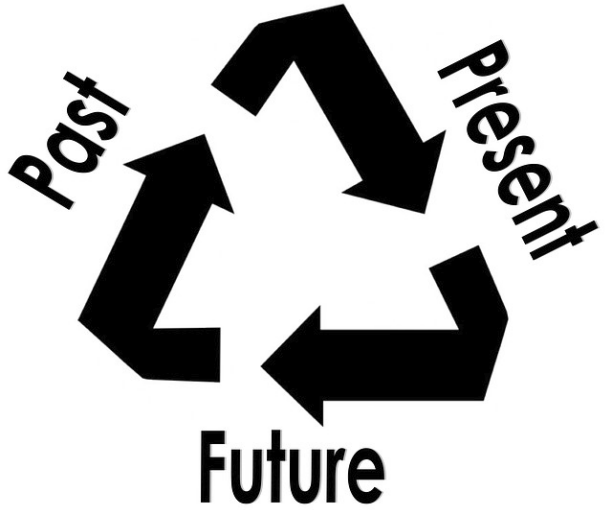


Module completion checklist

Objective	Complete
Introduce forecasting and time series analysis use cases	✔
Review key quantitative forecasting methods	
Introduce data used in time series analysis	
Review important features of time series data	
Visualize time series	
Explain time series modeling basics and key components of time series	
Explain how to measure linear relationships within time series	
Estimate and visualize autocorrelation of time series	

How do we forecast?

- Collect historic data
- Detect patterns on previous occurrences
- Make an **assumption** that **there is a relation of the past to the future**



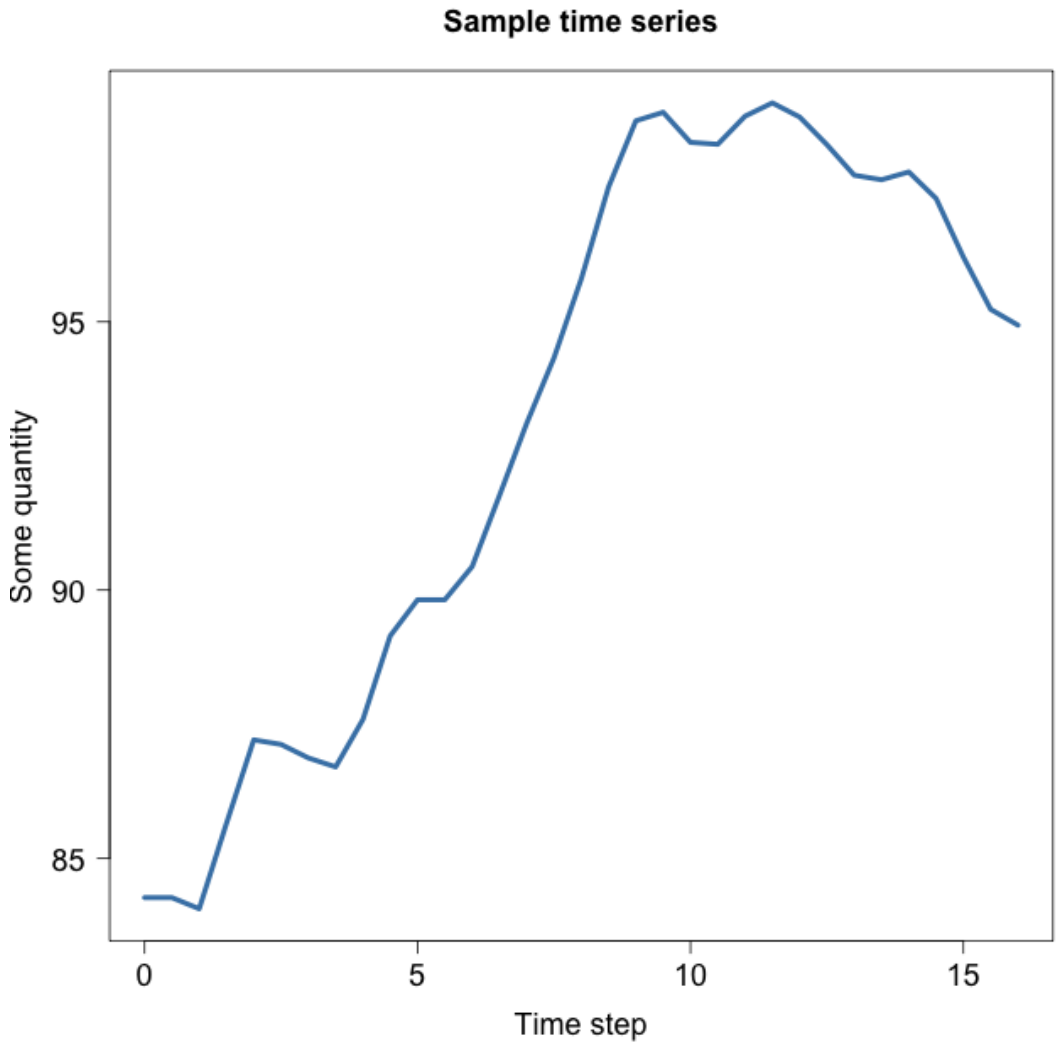
What are the quantitative forecasting methods?

- **Deterministic**
  - Trend models (a.k.a. seasonal models)
  - Smoothing models (a.k.a. exponential models)
- **Stochastic**
  - ARIMA Models
  - Box-Jenkins Methods

Module completion checklist

Objective	Complete
Introduce forecasting and time series analysis use cases	✔
Review key quantitative forecasting methods	✔
Introduce data used in time series analysis	
Review important features of time series data	
Visualize time series	
Explain time series modeling basics and key components of time series	
Explain how to measure linear relationships within time series	
Estimate and visualize autocorrelation of time series	

What does time series data look like?



	quantity	time_step
1	84.2697	0.0
2	84.2697	0.5
3	84.0619	1.0
4	85.6542	1.5
5	87.2093	2.0
6	87.1246	2.5
7	86.8726	3.0
8	86.7052	3.5
9	87.5899	4.0
10	89.1475	4.5
11	89.8204	5.0
12	89.8204	5.5
13	90.4375	6.0
14	91.7605	6.5
15	93.1081	7.0
16	94.3291	7.5
17	95.8003	8.0
18	97.5119	8.5
19	98.7457	9.0
20	98.9040	9.5
21	98.3437	10.0
22	98.3075	10.5
23	98.8313	11.0
24	99.0789	11.5
25	98.8157	12.0
26	98.2998	12.5
27	97.7311	13.0
28	97.6471	13.5
29	97.7922	14.0
30	97.2974	14.5

What does time series data look like?

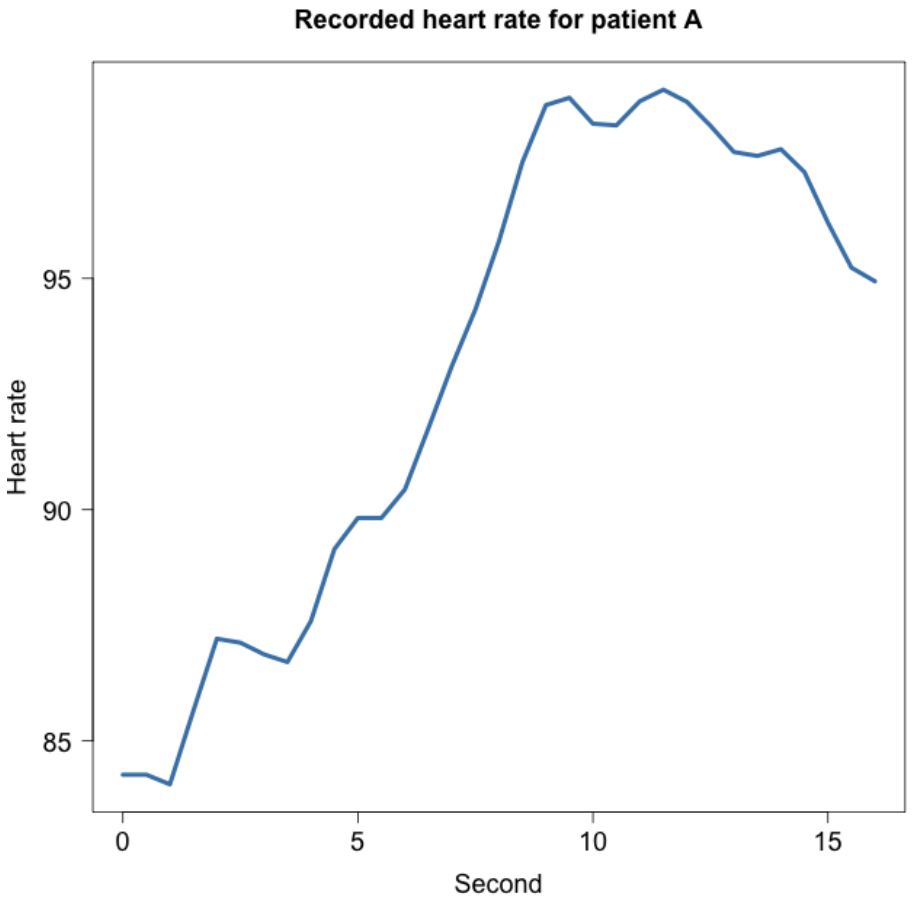
The simplest forms of time series data consists of 2 variables:

1. Some numeric **quantity**
2. **Time step** at which that quantity has occurred

What do you think this dataset may represent?

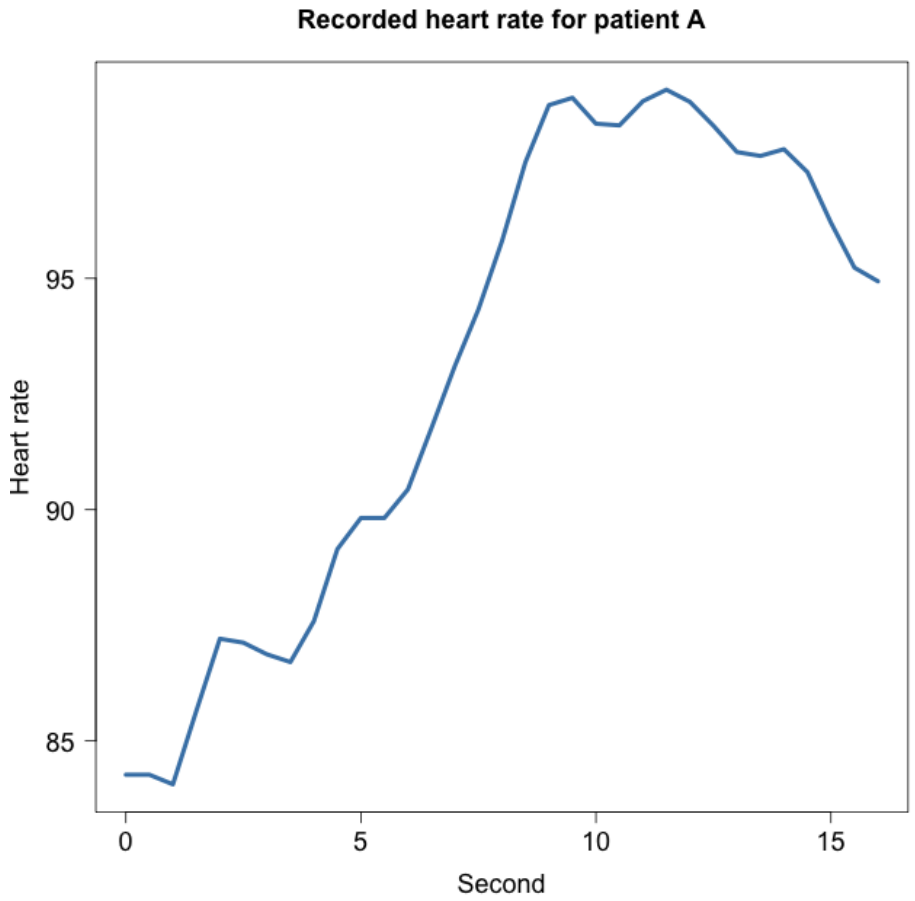
What does time series data look like?

- This dataset is a set of recorded heart rates for a patient during some clinical trial
- The **time step** in this case is equal to **0.5 seconds** and is recorded **on the `x-axis`**
- The **quantity** is the **heart rate** and changes for every time step and is recorded **on the `y-axis`**
- What can we say about this data based on just looking at the plot?

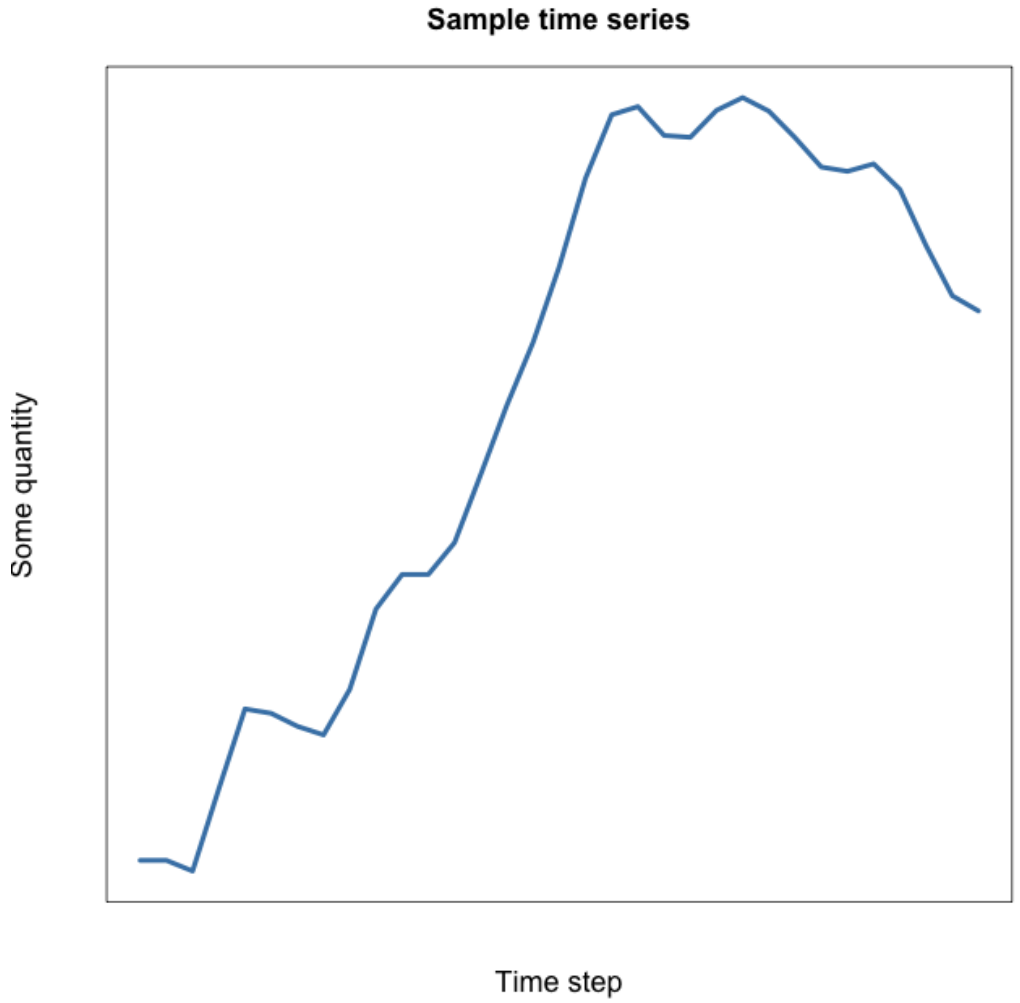


What can we say about this data based on plot?

- The heart rate was taken for a total of `16` seconds
- The interval between each time step is equal to `0.5` seconds
- The heart rate was increasing until about the `12th` second, before it started to decrease again
- **Conclusion:** this patient could have been performing a physical exercise or given a stimulant that caused the heart rate to increase



What else could this plot represent?



- If you remove the values off both axis, what other data may look like this?

Data: use case in civil aviation

- Air passenger mobility can be studied by analyzing seat and passenger miles
- We are going to work with **passenger mile data** that contains time series of three variables **from January 1979 to April 2002**
- This study was used in the paper ***"Estimated Impacts of September 11th on US Travel"*** published by U.S. Department of Transportation's Bureau of Transportation Statistics in 2006

Data: use case in civil aviation

- Our dataset contains the following variables
  - `Revenue passenger mile` is a measure of the volume of air passenger transportation; a revenue passenger-mile is equal to one paying passenger carried one mile
  - `Available seat mile` is the number of seats multiplied by the distance traveled per flight and can represent the overall capacity of the aircraft
  - `Unused mile` is the difference between the above two variables and is used to measure the airline capacity utilization
- We will be working with `Revenue passenger mile` variable to **predict the overall volume of air passenger transportation** and detect things that fall out of the usual patterns **in class slides**
- We will be working with `Available seat mile` variable to **predict the overall capacity of the aircraft** and detect things that fall out of the usual patterns **in class exercises**

Source: Bureau of Transportation Statistics

Data: similar use case relevant to your work

- By looking at the type of data, what would be an applicable use case in your work?
  - For instance, the effects of an external event on air traffic at a certain location
- If you have a dataset in mind you would like to use during your workshop, feel free to show it to the instructor to verify if it could be used

Knowledge check 1

Module completion checklist

Objective	Complete
Introduce forecasting and time series analysis use cases	✔
Review key quantitative forecasting methods	✔
Introduce data used in time series analysis	✔
Review important features of time series data	
Visualize time series	
Explain time series modeling basics and key components of time series	
Explain how to measure linear relationships within time series	
Estimate and visualize autocorrelation of time series	

Import packages

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

```
import os
import pandas as pd
from pandas.plotting import lag_plot
import numpy as np
import pickle
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from statsmodels.tsa.stattools import acf
from statsmodels.graphics.tsaplots import plot_acf
```

Directory settings

- In order to maximize the efficiency of your workflow, you should encode your directory structure into `variables`
- Let the `main_dir` be the variable corresponding to your `af-werx` folder

```
# Set `main_dir` to the location of your `af-werx` folder (for Linux).
main_dir = "/home/[username]/Desktop/af-werx"
```

```
# Set `main_dir` to the location of your `af-werx` folder (for Mac).
main_dir = '/Users/[username]/Desktop/af-werx'
```

```
# Set `main_dir` to the location of your `af-werx` folder (for Windows).
main_dir = "C:\\Users\\[username]\\Desktop\\af-werx"
```

```
# Make `data_dir` from the `main_dir` and
# remainder of the path to data directory.
data_dir = main_dir + "/data"
```

Working directory

- Set working directory to the `data_dir` variable we set
- We do this using the `os.chdir` function, change directory
- We can then check the working directory using `.getcwd()`
- For complete documentation of the `os` package, ***click here***

```
# Set working directory.
os.chdir(data_dir)
```

```
# Check working directory.
print(os.getcwd())
```

```
/home/[user-name]/af-werx/data
```

Load passenger miles dataset

```
# Read csv file into `passenger_miles` variable.
passenger_miles = pd.read_csv(data_dir + "/passenger_miles.csv")
print(passenger_miles.head())
```

```
     Date  Revenue Passenger Miles  Available Seat Miles  Unused Seat Miles
0  1979-1                    15.50                 26.64              11.15
1  1979-2                    16.58                 27.20              10.62
2  1979-3                    18.85                 27.87               9.02
3  1979-4                    17.23                 23.22               5.99
4  1979-5                    16.04                 23.27               7.23
```

```
# Rename columns for convenience.
passenger_miles.columns = ['date', 'revenue_passenger_miles',
                           'available_seat_miles', 'unused_seat_miles']
print(passenger_miles.head())
```

```
     date  revenue_passenger_miles  available_seat_miles  unused_seat_miles
0  1979-1                    15.50                 26.64              11.15
1  1979-2                    16.58                 27.20              10.62
2  1979-3                    18.85                 27.87               9.02
3  1979-4                    17.23                 23.22               5.99
4  1979-5                    16.04                 23.27               7.23
```

Format the dates

- Time series data analysis requires dates and times to be formatted properly
- This allows us to be able to manipulate date-time components, extract them, and visualize them easily
- Currently the `date` column is of type `object` in `YYYY-M` format

```
print(passenger_miles.head())
```

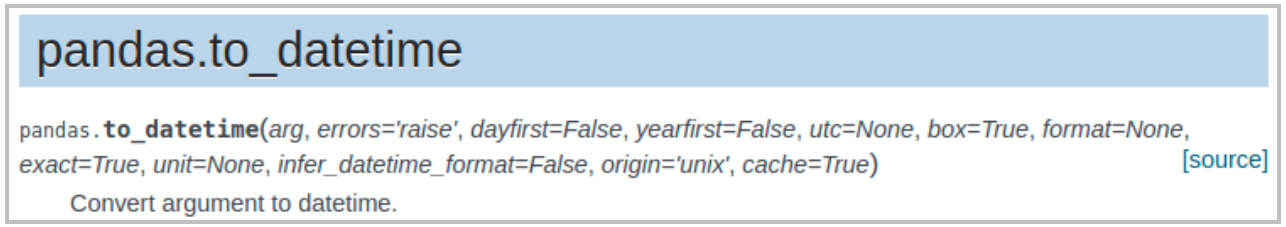
```
     date  revenue_passenger_miles  available_seat_miles  unused_seat_miles
0  1979-1                    15.50                 26.64              11.15
1  1979-2                    16.58                 27.20              10.62
2  1979-3                    18.85                 27.87               9.02
3  1979-4                    17.23                 23.22               5.99
4  1979-5                    16.04                 23.27               7.23
```

```
print(passenger_miles.dtypes)
```

```
date                        object
revenue_passenger_miles    float64
available_seat_miles       float64
unused_seat_miles          float64
dtype: object
```

Format the dates (cont'd)

- We will use `.to_datetime()` function from `pandas` to convert it to date-time format suitable for time series analysis (***see documentation here***)



- The one argument we need (other than the `date` column itself) to convert the column to a proper `datetime` type is `format`
  - The `format` argument allows us to tell the function in which format the current date is written: `YYYY-M` in our case
  - It uses special notation that allows the function to recognize the format: `%Y` stands for a four-digit year and `%m` for a month
  - For more information about date time formatting, you can review a table with ***a list of all date time directives***

Format the dates (cont'd)

- Since this is monthly data in the form of `YYYY-M`, we will cast it by using `%Y-%m` notation

```
# Convert `date` column to datetime format: YYYY-MM.
passenger_miles['date'] = pd.to_datetime(passenger_miles['date'], format = "%Y-%m")
print(passenger_miles.dtypes)
```

```
date                          datetime64[ns]
revenue_passenger_miles              float64
available_seat_miles                 float64
unused_seat_miles                    float64
dtype: object
```

```
print(passenger_miles.head())
```

```
        date  revenue_passenger_miles  available_seat_miles  unused_seat_miles
0 1979-01-01                    15.50                 26.64              11.15
1 1979-02-01                    16.58                 27.20              10.62
2 1979-03-01                    18.85                 27.87               9.02
3 1979-04-01                    17.23                 23.22               5.99
4 1979-05-01                    16.04                 23.27               7.23
```

Key features of time series data

- Let's take a closer look at our dataset
- Observations arrive according to some **order**: *starting on Jan 01, 1979 and ending on Apr 01, 2002*
- Measurements are made in discrete **equally spaced time intervals**: *every month*

```
        date  revenue_passenger_miles  available_seat_miles  unused_seat_miles
0 1979-01-01                    15.50                 26.64              11.15
1 1979-02-01                    16.58                 27.20              10.62
2 1979-03-01                    18.85                 27.87               9.02
3 1979-04-01                    17.23                 23.22               5.99
```

...

```
          date revenue_passenger_miles available_seat_miles
278 2002-02-01                   36.01                53.40
279 2002-03-01                   41.21                54.85
280 2002-04-01                   39.50                55.54
    unused_seat_miles
278             17.39
279             13.64
280             16.04
```

Key features of time series data (cont'd)

- Measurements can be **discrete** or **continuous**: *passenger miles are assumed as* **continuous**

```
print(passenger_miles.head(20))
```

```
         date  revenue_passenger_miles  available_seat_miles  unused_seat_miles
0  1979-01-01                    15.50                 26.64              11.15
1  1979-02-01                    16.58                 27.20              10.62
2  1979-03-01                    18.85                 27.87               9.02
3  1979-04-01                    17.23                 23.22               5.99
4  1979-05-01                    16.04                 23.27               7.23
5  1979-06-01                    19.11                 27.30               8.19
6  1979-07-01                    19.88                 29.28               9.40
7  1979-08-01                    21.47                 31.23               9.76
8  1979-09-01                    16.55                 28.55              12.00
9  1979-10-01                    16.55                 27.84              11.28
10 1979-11-01                    16.42                 27.56              11.14
11 1979-12-01                    16.31                 28.87              12.56
12 1980-01-01                    15.68                 28.70              13.02
13 1980-02-01                    16.07                 28.38              12.31
14 1980-03-01                    17.94                 28.66              10.72
15 1980-04-01                    17.05                 28.76              11.71
16 1980-05-01                    16.15                 27.96              11.82
17 1980-06-01                    18.67                 29.92              11.25
18 1980-07-01                    18.50                 30.24              11.74
19 1980-08-01                    19.60                 30.21              10.61
```

Key features of time series data (cont'd)

- Observations are **not necessarily independent**
- They **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!
- Only **stable series can be analyzed**

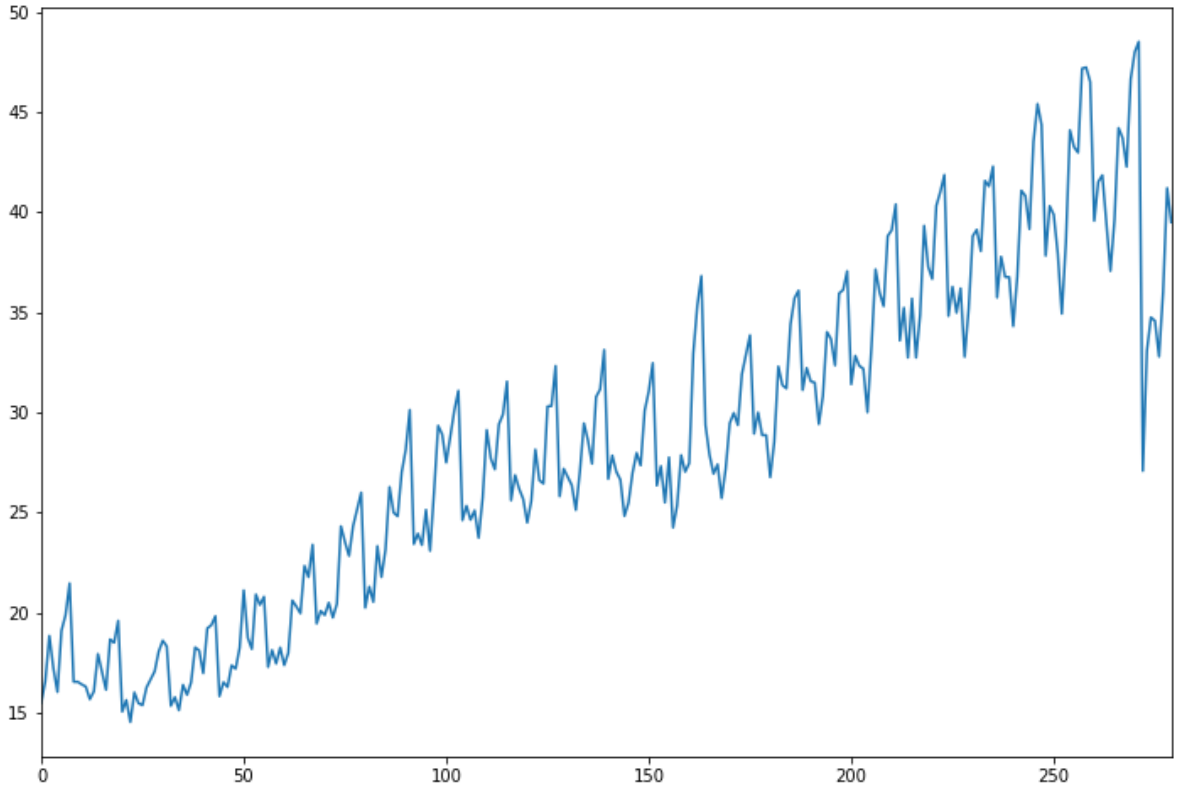
Note: the above features are not evident from the plot or the data itself and require performing time series analysis before they can be detected. We will discuss these features in more detail in materials to follow as they play crucial role in time series model performance!

Module completion checklist

Objective	Complete
Introduce forecasting and time series analysis use cases	✔
Review key quantitative forecasting methods	✔
Introduce data used in time series analysis	✔
Review important features of time series data	✔
Visualize time series	
Explain time series modeling basics and key components of time series	
Explain how to measure linear relationships within time series	
Estimate and visualize autocorrelation of time series	

Visualize time series: line plot

```
# Plot series.
passenger_miles['revenue_passenger_miles'].plot()
plt.show()
```



Visualize time series: set index to date

- Setting the index to a `datetime` object will allow us to select, subset and manipulate the observations using date-time references with ease

```
# Set index of the dataframe to `date` column.
passenger_miles.set_index('date', inplace = True)
print(passenger_miles.head())
```

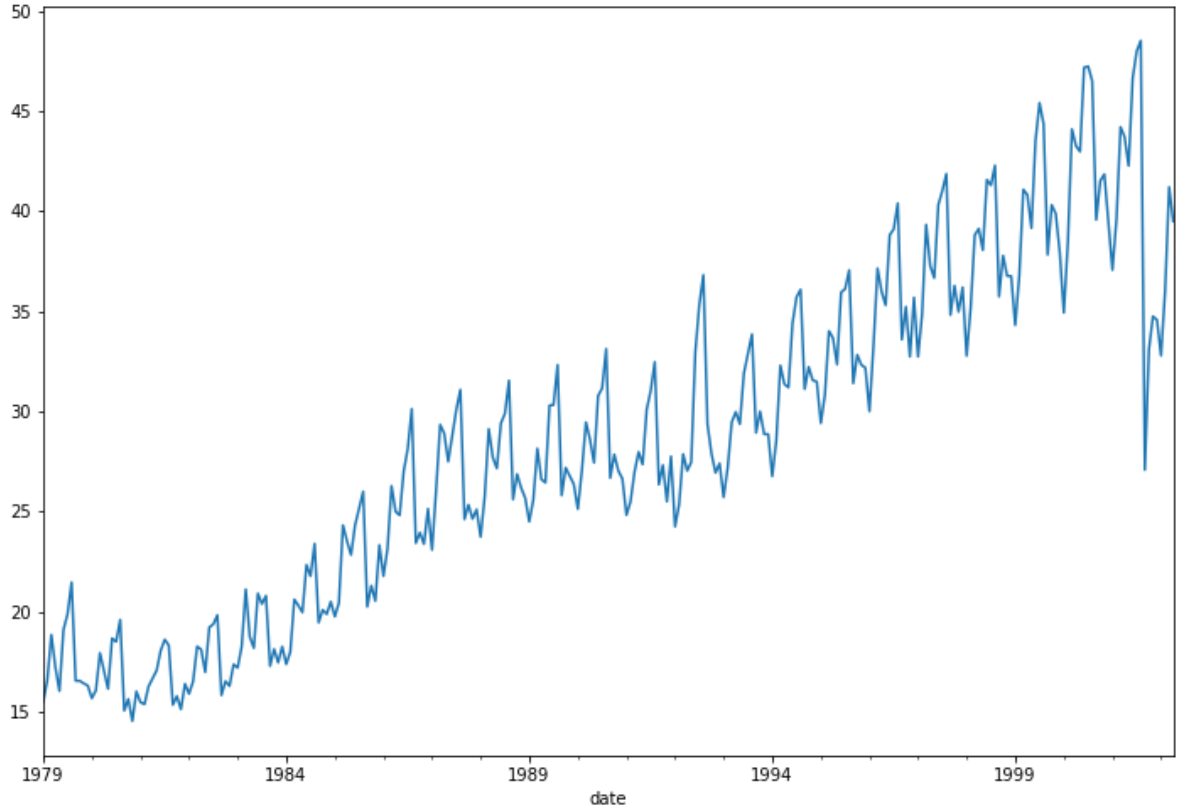
```
            revenue_passenger_miles  available_seat_miles  unused_seat_miles
date
1979-01-01                    15.50                 26.64              11.15
1979-02-01                    16.58                 27.20              10.62
1979-03-01                    18.85                 27.87               9.02
1979-04-01                    17.23                 23.22               5.99
1979-05-01                    16.04                 23.27               7.23
```

- Let's save the resulting dataframe to a pickle for easy loading in our upcoming modules

```
# Pickle and save the file.
pickle.dump(passenger_miles, open(data_dir + "/passenger_miles.sav","wb" ))
```

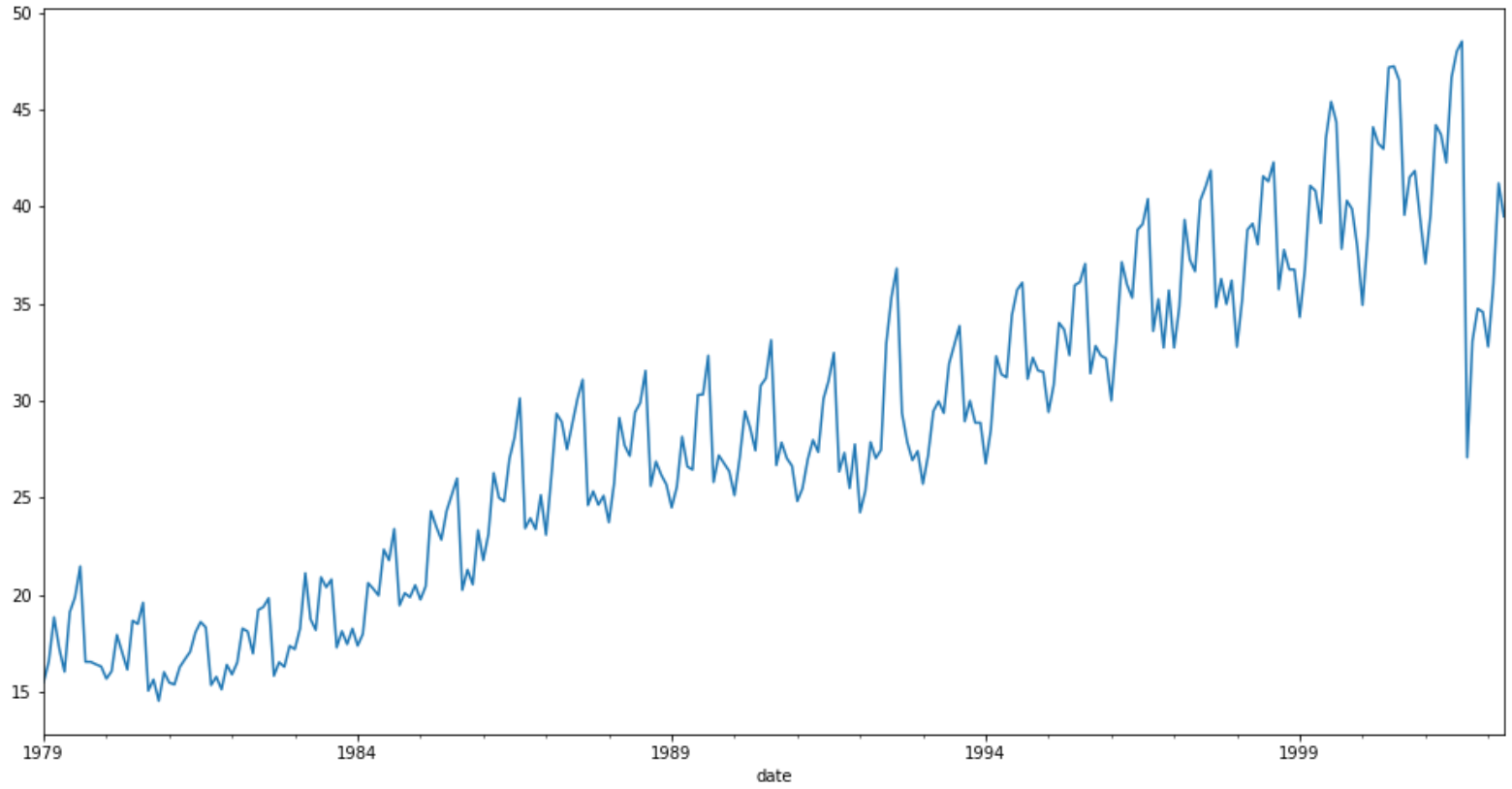
Visualize time series: line plot (cont'd)

```
# Plot series.
# Notice that the x-axis automatically changed to a date!
passenger_miles['revenue_passenger_miles'].plot()
plt.show()
```



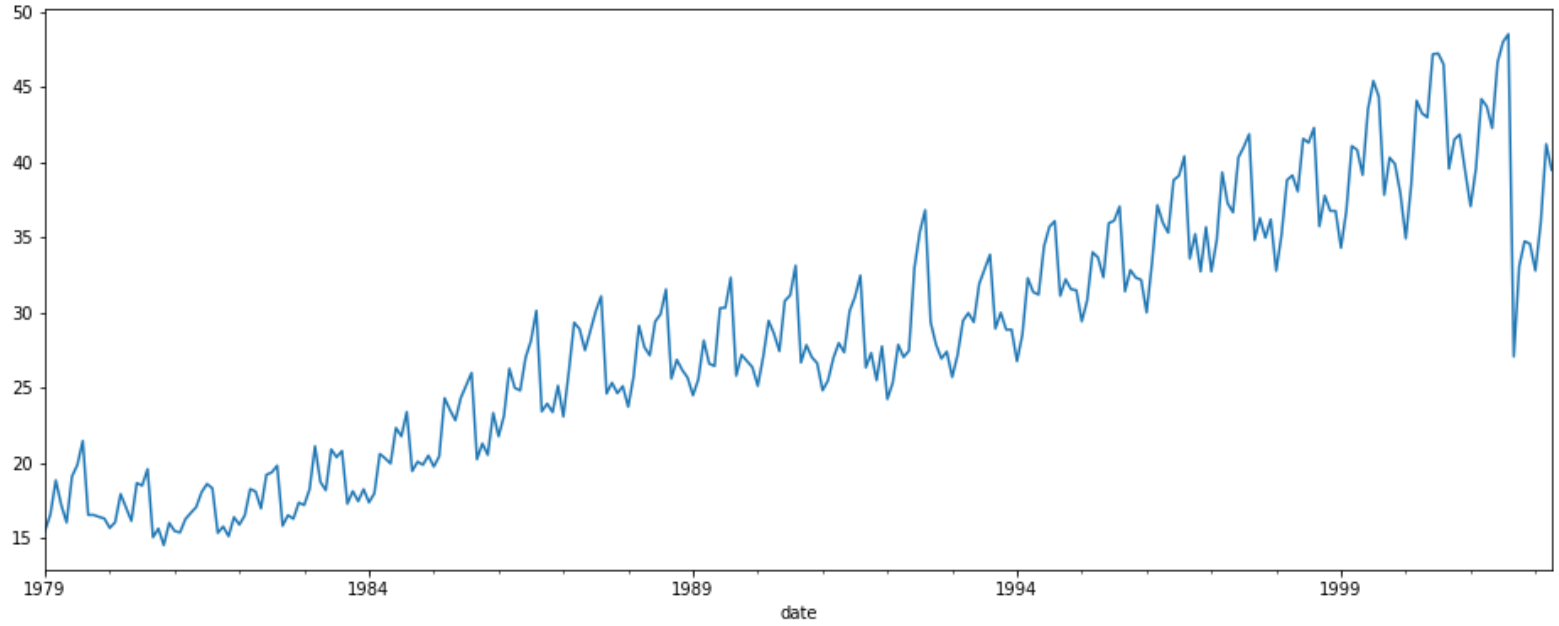
Visualize time series: adjust figure size

```
fig = plt.subplots(figsize = (16, 8))
passenger_miles['revenue_passenger_miles'].plot()
plt.show()
```



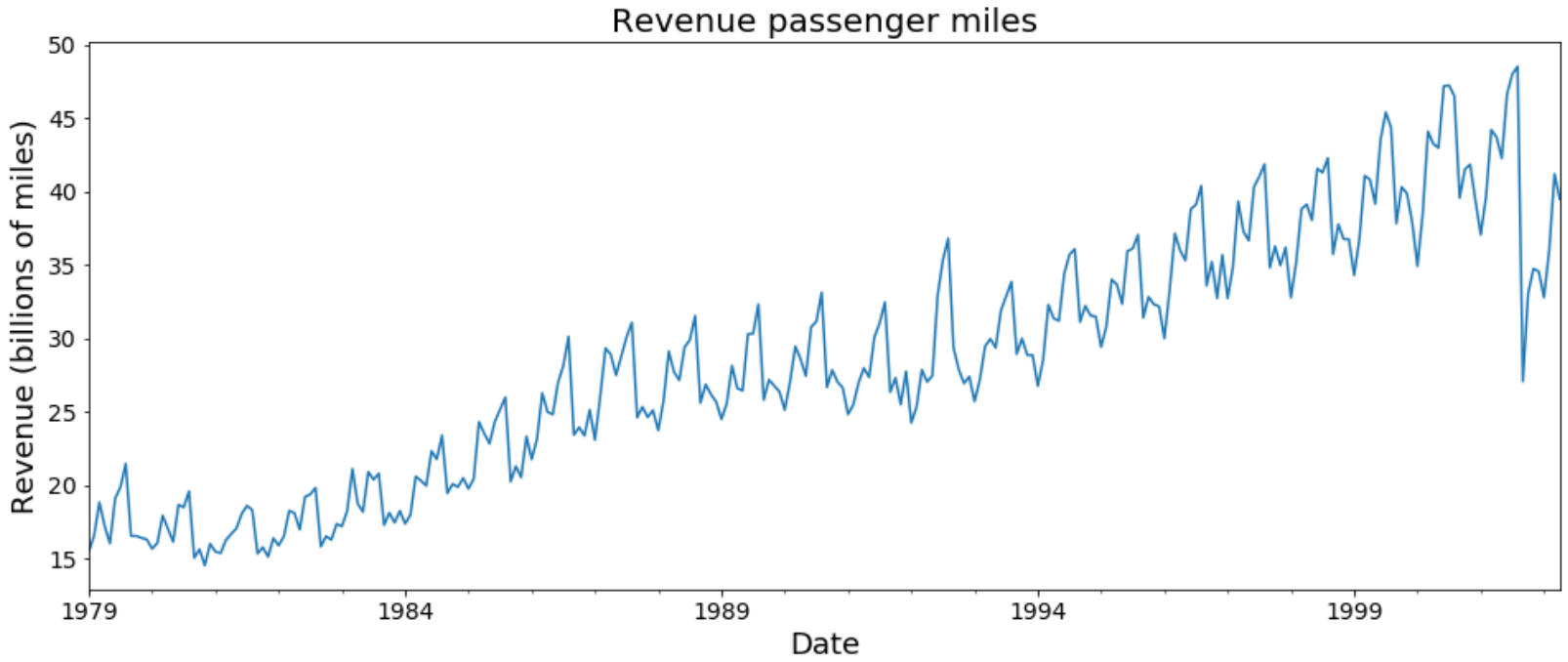
Visualize time series: adjust figure size

```
fig = plt.subplots(figsize = (16, 6))
passenger_miles['revenue_passenger_miles'].plot()
plt.show()
```



Visualize time series: adjust figure labels

```
fig, ax = plt.subplots(figsize = (16, 6))
passenger_miles['revenue_passenger_miles'].plot()
plt.title('Revenue passenger miles', fontsize = 20)
plt.xlabel('Date', fontsize = 18)
plt.ylabel('Revenue (billions of miles)', fontsize = 18)
ax.tick_params(labelsize = 14)
plt.show()
```



- What can you tell by looking at this plot?

Knowledge check 2

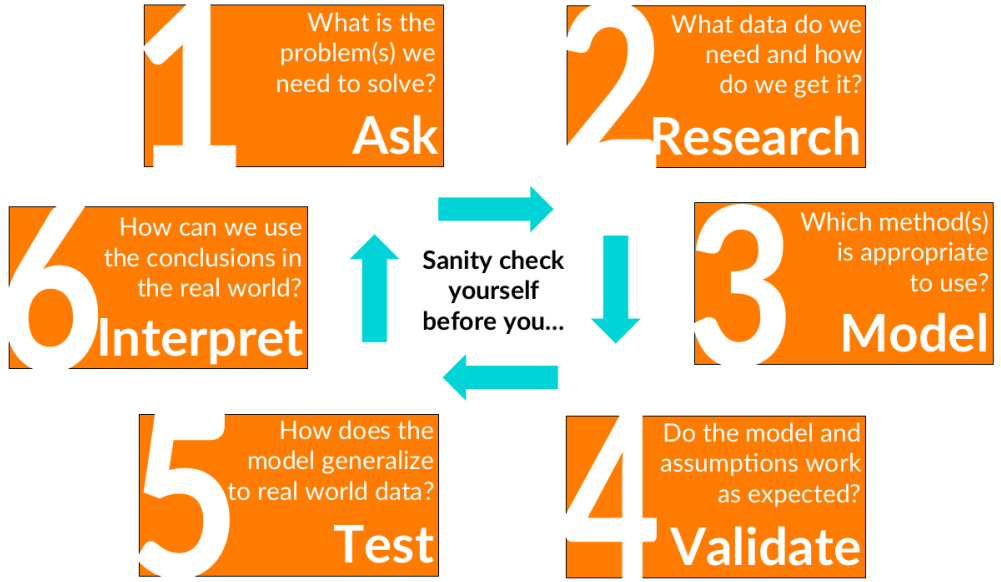
Exercise 1

Module completion checklist

Objective	Complete
Introduce forecasting and time series analysis use cases	✔
Review key quantitative forecasting methods	✔
Introduce data used in time series analysis	✔
Review important features of time series data	✔
Visualize time series	✔
Explain time series modeling basics and key components of time series	
Explain how to measure linear relationships within time series	
Estimate and visualize autocorrelation of time series	

Data science control cycle for time series

- First stage
  - Model **specification** (*Step 3*)
  - Model **estimation** (*Step 3*)
  - Model **validation** (*Step 4*)
- Second stage
  - **Forecasting** based on the model (*Step 5*)
  - **Evaluation of** forecast **performance** (*Step 5*)
  - Model **updating** (return to *Step 3* upon model improvement if necessary)



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

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

In search for conditional mean

- Most time series models are based on estimating the **conditional mean** of the time series Y_t
- **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

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

In search for 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

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

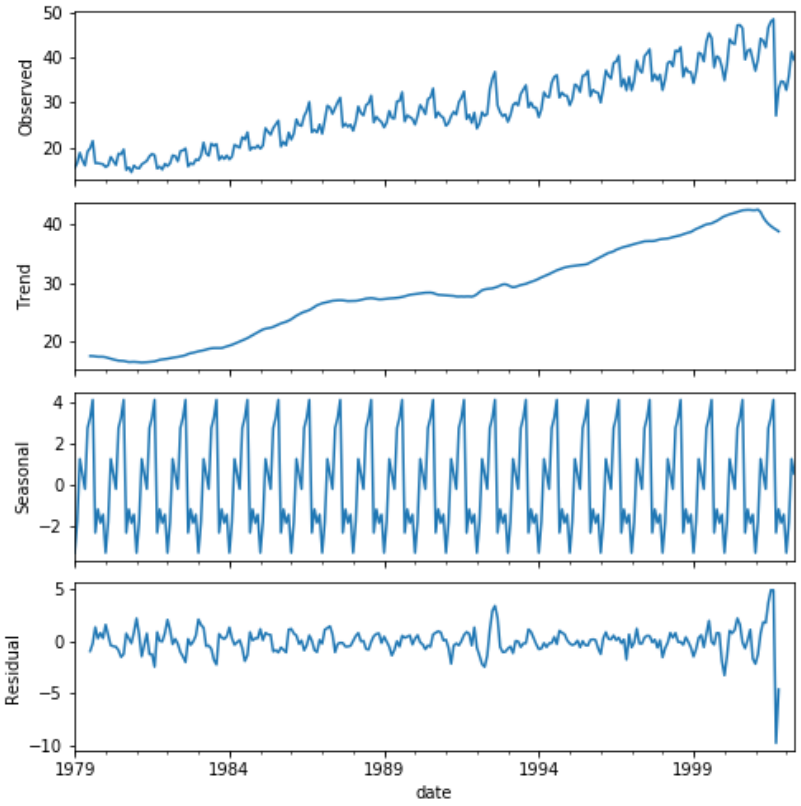
- Then, any time series model will boil down to an observation at time t as a sum of the **conditional mean** for time t and the **error** at time t

$$Y_t = \mu_t + \epsilon_t$$

What's the trend this season?

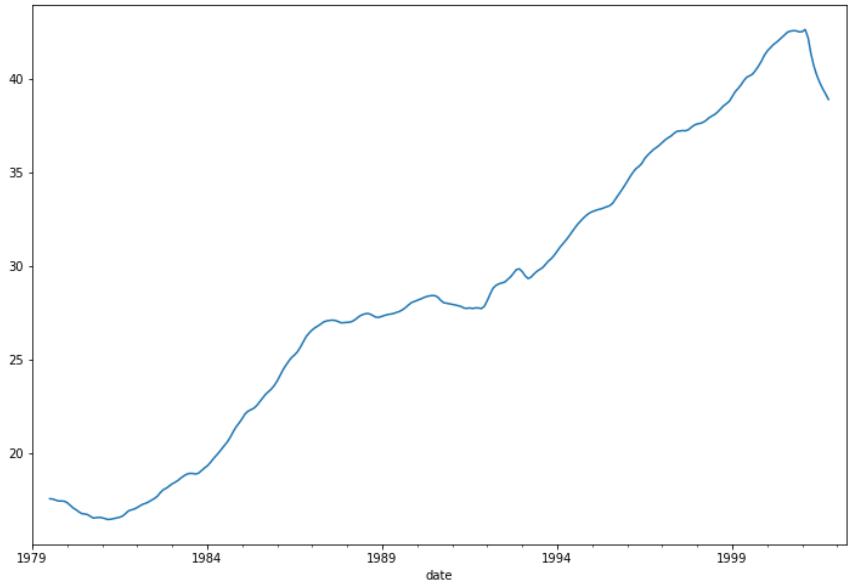
- No, we are not in the fashion business, but we ARE in the modeling business, *time series modeling* to be precise!
- Time series data comes with some *deterministic components* that don't occur in other types of data
  - The **trend** component (T_t)
  - The **seasonal** component (S_t)
  - The **random error** component (ϵ_t)
- When decomposed into the above components, the *additive* time series model now looks like this:

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

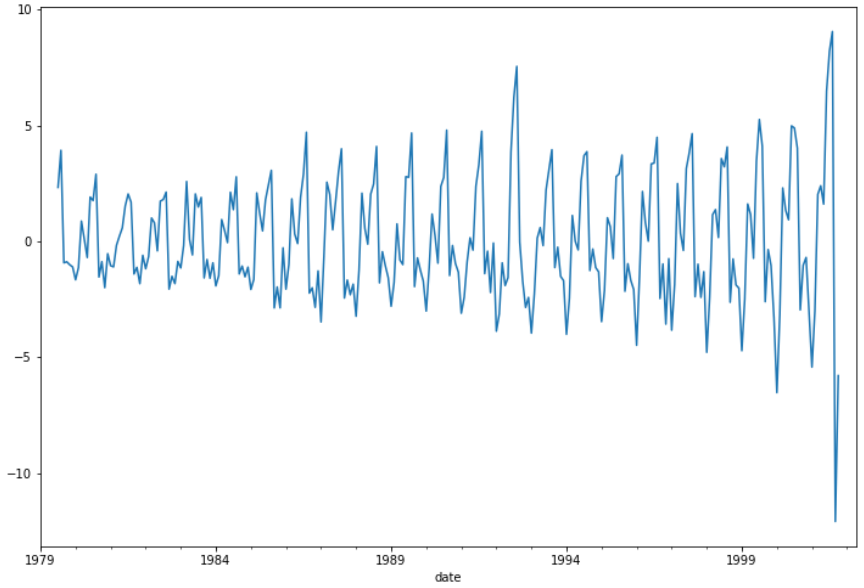


The trend component

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

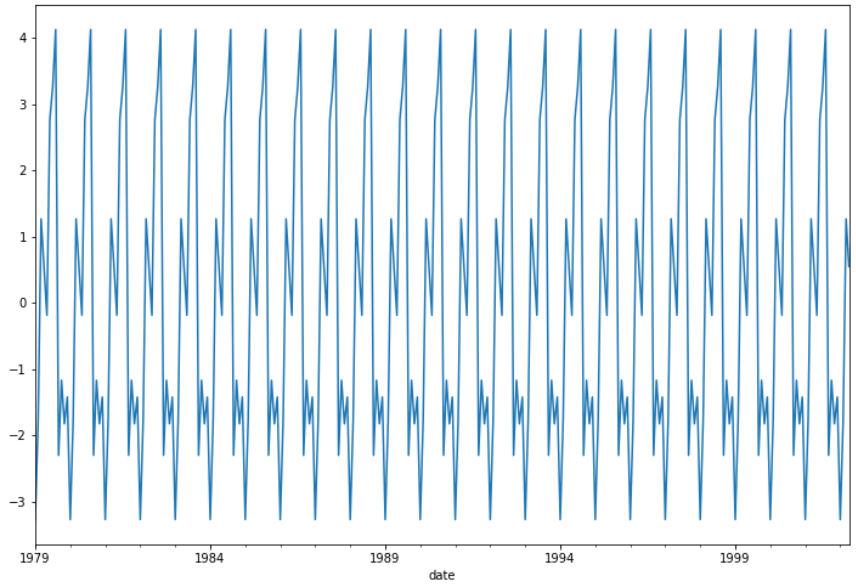


- $(Y_t - T_t)$: series without trend, or **detrended series**

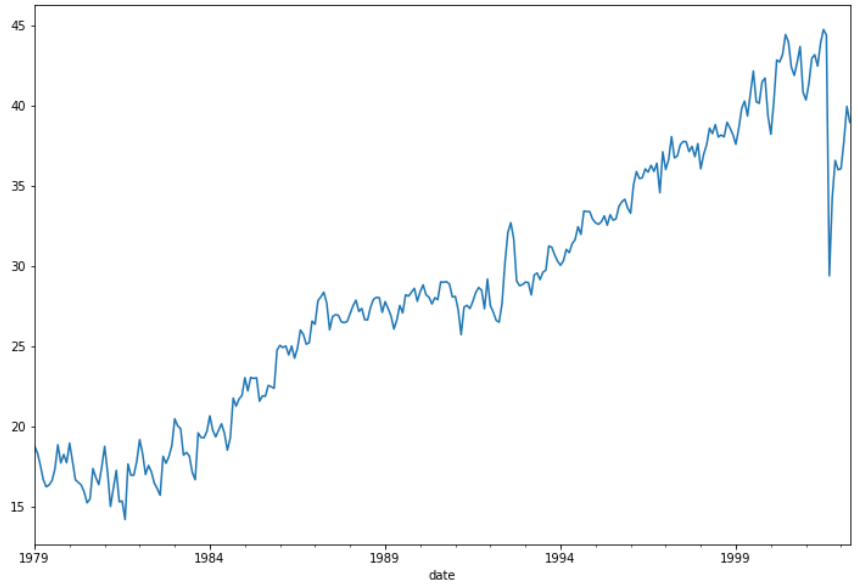


The seasonal component

- S_t: the **seasonal** component

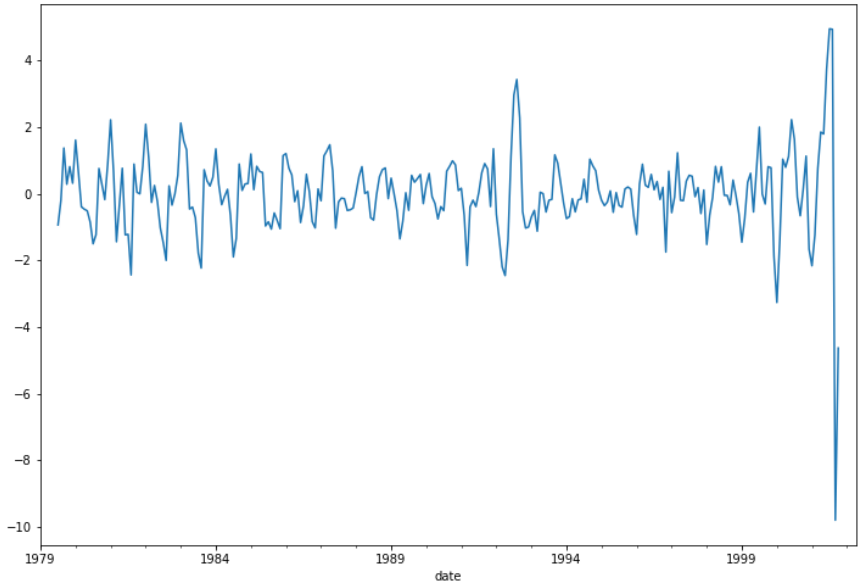


- $(Y_t - S_t)$: series without the seasonal component, or **deseasonalized series**



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**, also known as **residuals**, also known as **error**



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 error term ϵ_t, which is also known as the **noise** term 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 error $Y_t = \epsilon_t$

Knowledge check 3

Module completion checklist

Objective	Complete
Introduce forecasting and time series analysis use cases	✔
Review key quantitative forecasting methods	✔
Introduce data used in time series analysis	✔
Review important features of time series data	✔
Visualize time series	✔
Explain time series modeling basics and key components of time series	✔
Explain how to measure linear relationships within time series	
Estimate and visualize autocorrelation of time series	

Measuring linear relationships: linear regression

- We used **covariance** in our *linear regression model* to measure linear dependence between two variables X and Y

$$Cov(X, Y) = \frac{\sum_{i=1}^{N} (x_i - mean_x)(y_i - mean_y)}{(N - 1)}$$

- We also used **correlation coefficient** in our *linear regression model* to measure how well the change in variable X explains the change in variable Y

$$Corr(X, Y) = r_{xy} = \frac{Cov(X, Y)}{\sqrt{var(X)var(Y)}}$$

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

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

Autocovariance for lagged series

- To compute autocovariance at lag k, we adjust the formula by factoring k into the number of datapoints we are considering

$$Cov(Y_t, Y_{t-k}) = \gamma_k = \frac{1}{n-k-1} \sum_{t=k+1}^{n} (Y_t - mean_Y)(Y_{t-k} - mean_Y)$$

- **A few things to remember:**
  - Autocovariance at $k = 0$ is just variance of Y
  - Our sample size will effectively shrink by k datapoints
  - If n is small, the estimates of autocovariance become less reliable since we lose k observations in estimating γ_k
  - Rule of thumb: estimate autocovariances up to $k \leq n/5$ or $k \leq n/4$

Autocorrelation for lagged series

- Variance of Y at any point in time is assumed to be stationary and the same, so $var(Y_t) = var(Y_{t-1}) = var(Y)$
- To compute autocorrelation at lag k, we simply use autocovariance at lag k in the numerator and variance of Y in the denominator

$$Corr(Y_t, Y_{t-k}) = \rho_k = \frac{\gamma_k}{\sqrt{var(Y_t)var(Y_{t-1})}} = \frac{\gamma_k}{var(Y)}$$

- **Note:** autocorrelation at lag $k = 0$ is always equal to 1, why do you think that happens?

Module completion checklist

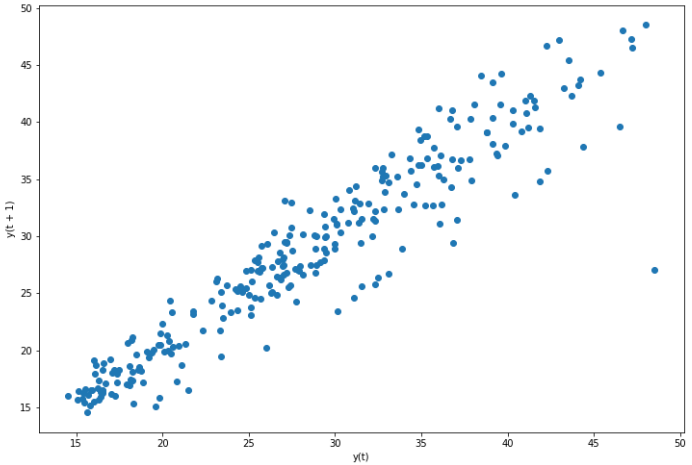
Objective	Complete
Introduce forecasting and time series analysis use cases	✔
Review key quantitative forecasting methods	✔
Introduce data used in time series analysis	✔
Review important features of time series data	✔
Visualize time series	✔
Explain time series modeling basics and key components of time series	✔
Explain how to measure linear relationships within time series	✔
Estimate and visualize autocorrelation of time series	

Exploring and visualizing the lag: lag_plot

- Scatterplots are a great way to detect an obvious pattern in relationship of 2 variables
- For time series, we can plot the value of a variable at time t vs the value of the same variable at time $t - k$ to explore how an observation at time t is related to an observation at $t - k$
- Such scatterplot is called a **lag plot**
- We will use `lag_plot` function from `pandas` library
  - It takes a time series as its single main argument
  - The `lag` argument is optional with default $k = 1$

Exploring and visualizing the lag

```
# Plot lag (default is 1).
lag_plot(passenger_miles['revenue_passenger_miles'])
plt.show()
```



- The **more the points follow a linear pattern**, the higher the correlation
- Since we are plotting the variable against itself at a different time step, the distinct linear pattern will indicate a **stronger autocorrelation**

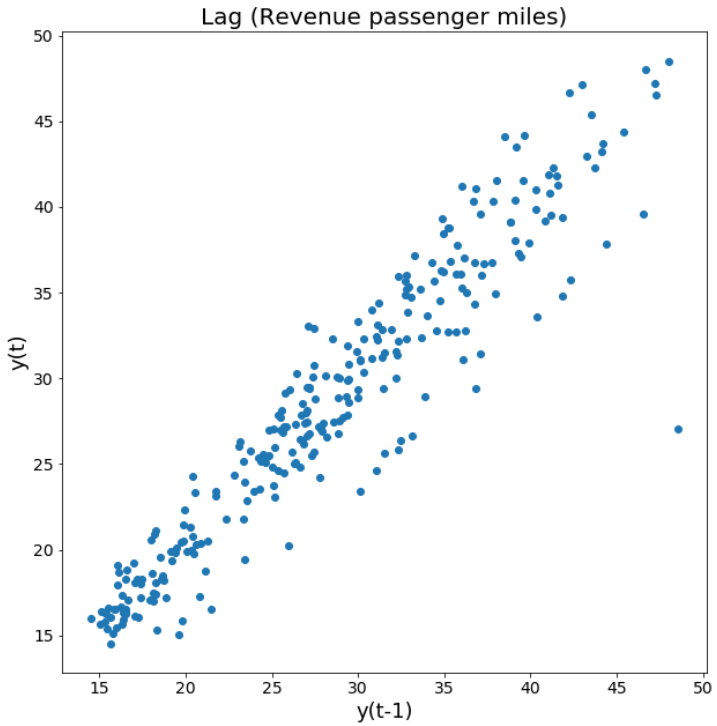
Note:

- Notice that the plot shows `y(t)` on `x-axis` and `y(t+1)` on the `y-axis` by default for `lag = 1`
- This is just an alternative notation for `y(t-1)` on `x-axis` and `y(t)` on the `y-axis` for `lag = 1`

Exploring and visualizing the lag (cont'd)

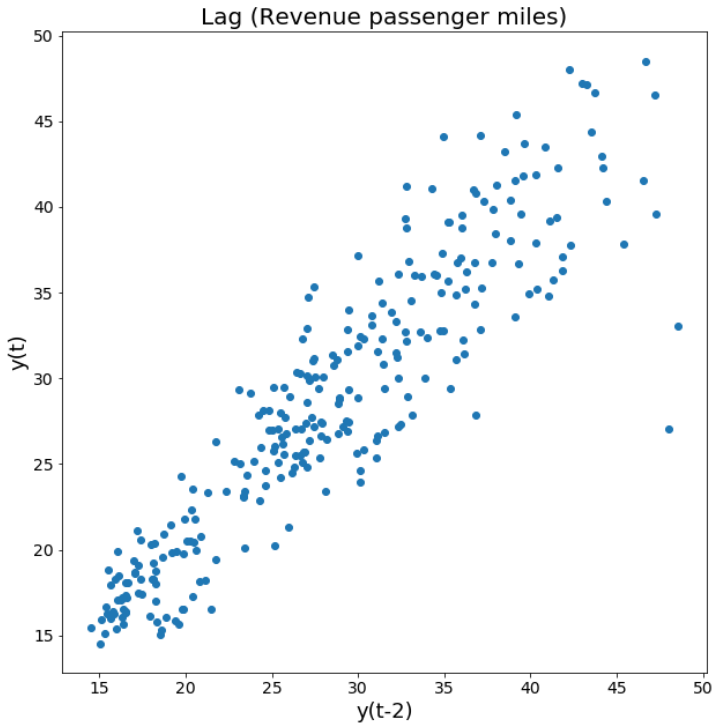
- Let's adjust the plot's labels and size for better readability

```
fig, ax = plt.subplots(figsize = (10, 10))
# Plot lag (default is 1).
lag_plot(passenger_miles['revenue_passenger_miles'])
plt.title('Lag (Revenue passenger miles)',
          fontsize = 20)
plt.xlabel('y(t-1)', fontsize = 18)
plt.ylabel('y(t)', fontsize = 18)
ax.tick_params(labelsize = 14)
plt.show()
```



Exploring and visualizing the lag: k = 2

```
fig, ax = plt.subplots(figsize = (10, 10))
# Plot lag.
lag_plot(passenger_miles['revenue_passenger_miles'],
         lag = 2)  #<- set lag to 2
plt.title('Lag (Revenue passenger miles)',
          fontsize = 20)
plt.xlabel('y(t-1)', fontsize = 18)
plt.ylabel('y(t)', fontsize = 18)
ax.tick_params(labelsize = 14)
plt.show()
```



- Notice that the points on the scatterplot still follow a linear pattern, but they are a lot more spread out
- This indicates a weakening linear relationship

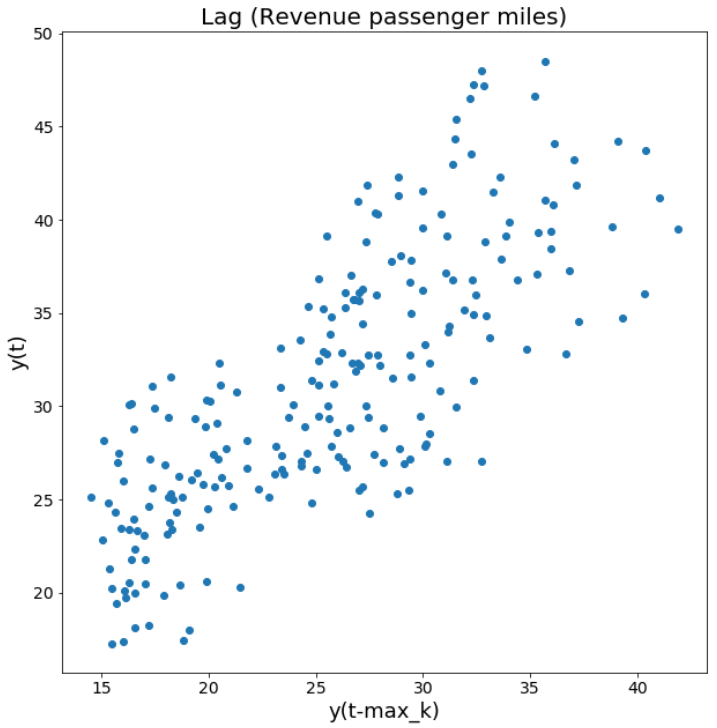
Exploring and visualizing the lag: k = max_lag

- Let's take a look at an even higher value of k
- Since the rule of thumb for an optimal value of k is about $1/5$ of the total number of observation, let's compute it and then plot it

```
# Let's get a maximum lag based on the rule of thumb.
max_k = passenger_miles.shape[0]//5
print(max_k)
```

```
56
```

```
# Plot get lag plot for max_k.
lag_plot(passenger_miles['revenue_passenger_miles'],
         lag = max_k)  #<- set lag to 2
plt.title('Lag (Revenue passenger miles)',
          fontsize = 20)
plt.xlabel('y(t-max_k)', fontsize = 18)
plt.ylabel('y(t)', fontsize = 18)
ax.tick_params(labelsize = 14)
plt.show()
```



- What is this plot telling us?

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

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

- Let's see if the values of r_k will follow the same pattern as lag plots!

Estimate autocorrelation: worked example (lag 1)

t	Y_t	Y_{t-1}
1	13	_
2	8	13
3	15	8
4	4	15
5	4	4
6	12	4
7	11	12
8	7	11
9	14	7
10	2	14

1. $n = 10$
2. $mean_Y = \frac{13+8+...+2}{10} = \frac{90}{10} = 9$
3. $k = 1$
4. $$r_1 = \frac{\sum_{t=2}^{10}(Y_t-9)(Y_{t-1}-9)}{\sum_{t=1}^{10}(Y_t-9)^2}$$
   i. $$= \frac{(8-9)(13-9)+(15-9)(8-9)+...+(2-9)(14-9)}{(13-9)^2+(8-9)^2+...+(2-9)^2}$$
   ii. $$= \frac{-73}{194} = -0.3762887$$

Estimate autocorrelation: worked example (lag 2)

t	Y_t	Y_{t-1}	Y_{t-2}
1	13	_	_
2	8	13	_
3	15	8	13
4	4	15	8
5	4	4	15
6	12	4	4
7	11	12	4
8	7	11	12
9	14	7	11
10	2	14	7

1. $n = 10$
2. $mean_Y = 9$
3. $k = 2$
4. $$r_1 = \frac{\sum_{t=3}^{10}(Y_t - 9)(Y_{t-1} - 9)}{\sum_{t=1}^{10}(Y_t - 9)^2}$$
   i. $$= \frac{(15-9)(13-9)+(4-9)(8-9)+\ldots+(2-9)(7-9)}{(13-9)^2+(8-9)^2+\ldots+(2-9)^2}$$
   ii. $$= \frac{-8}{194} = -0.04123711$$

Estimate autocorrelation: worked example (lag 3)

t	Y_t	Y_{t-1}	Y_{t-2}	Y_{t-3}
1	13	_	_	_
2	8	13	_	_
3	15	8	13	_
4	4	15	8	13
5	4	4	15	8
6	12	4	4	15
7	11	12	4	4
8	7	11	12	4
9	14	7	11	12
10	2	14	7	11

1. $n = 10$
2. $mean_Y = 9$
3. $k = 3$
4. $$r_1 = \frac{\sum_{t=4}^{10}(Y_t - 9)(Y_{t-1} - 9)}{\sum_{t=1}^{10}(Y_t - 9)^2}$$
   i. $$= \frac{(4-9)(13-9)+(4-9)(8-9)+...+(2-9)*(11-9)}{(13-9)^2+(8-9)^2+...+(2-9)^2}$$
   ii. $$= \frac{4}{194} = 0.02061856$$

Is there an emerging pattern?

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*, which is exactly what we have been observing in our worked example:
  - $r_0 = 1$ (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**

Estimate autocorrelation for our dataset

- The ***`statsmodels` library*** contains a `tsa.stattools` module which contains tools that allow us to analyze time series
- We will use its function `acf` to evaluate the **autocorrelation coefficient** for our revenue passenger miles
  - It takes the series itself as its main argument
  - It can also take the number of lags `nlags` as an optional argument, the default is `40`
- For more information, take a look at the `acf` function ***documentation***

statsmodels.tsa.stattools.acf

statsmodels.tsa.stattools.**acf**(*x, unbiased=False, nlags=40, qstat=False, fft=None, alpha=None, missing='none'*) [source]

Calculate the autocorrelation function.

Parameters:

x : **array_like**

The time series data.

unbiased : **bool**

If True, then denominators for autocovariance are n-k, otherwise n.

nlags : **int, optional**

Number of lags to return autocorrelation for.

Estimate autocorrelation for our dataset (cont'd)

```
# Let's compute autocorrelation values for the passenger miles data.
acf_values = acf(passenger_miles['revenue_passenger_miles'],
                 nlags = max_k) #<- set lag to max number of lags
print(acf_values)
```

```
[1.         0.93838611 0.89762249 0.87207465 0.86699336 0.82581884
 0.79743702 0.82080624 0.8573936  0.84592471 0.84824511 0.85951148
 0.88761677 0.83390407 0.79540573 0.76757301 0.75888645 0.71849394
 0.68837434 0.70165777 0.72731273 0.71495413 0.71622465 0.72249774
 0.74573339 0.6933294  0.65519676 0.62886933 0.62069378 0.5826795
 0.55427546 0.56491271 0.58467645 0.57059391 0.5724076  0.57869764
 0.60041912 0.55025001 0.51735987 0.49258093 0.48316294 0.44644257
 0.42133674 0.43176564 0.45078537 0.43559364 0.43859129 0.44469399
 0.46447814 0.41642748 0.3854394  0.36379539 0.35494062 0.32012856
 0.2950635  0.30487397 0.32171171]
```

Note:

- $r_0 = 1$ (autocorrelation coefficient at lag 0)
- As the value of lag increases, the autocorrelation weakens, which means that as our observations are more separated in time, there is less and less correlation between them

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.

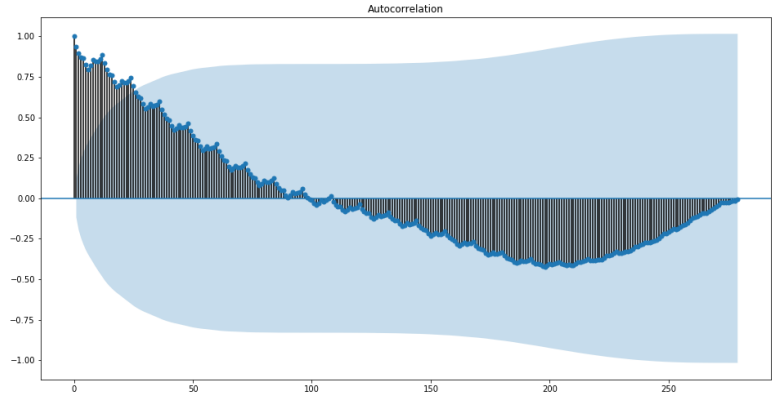
lags : *int or array_like, optional*

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.

Visualize autocorrelation function (cont'd)

```
# Autocorrelation function (ACF) plot.
plot_acf(passenger_miles['revenue_passenger_miles'])
plt.show()
```

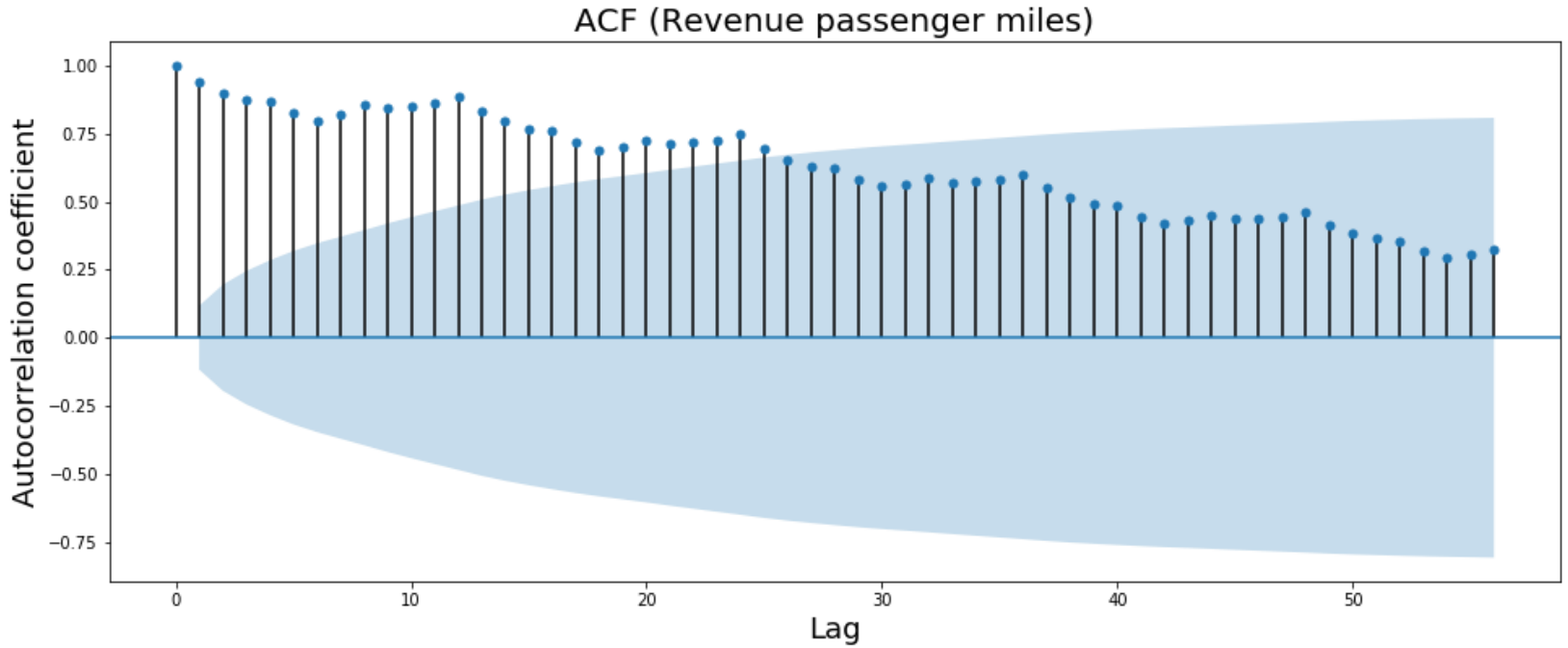
- Notice that this plot by default only includes ACF for lag 30



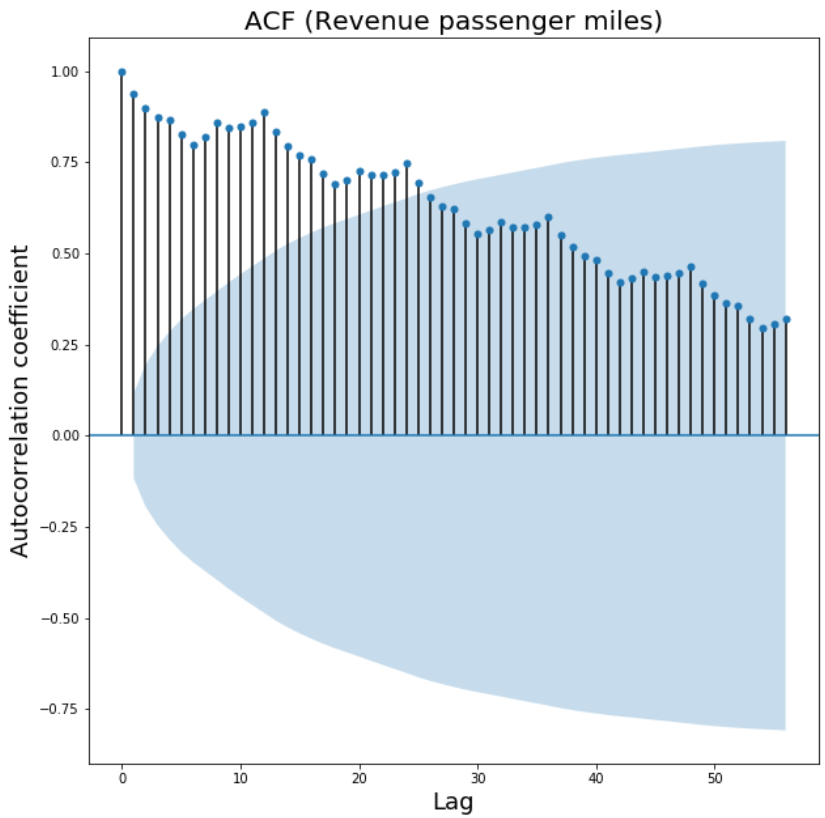
Visualize autocorrelation function (cont'd)

- Adjust lag and plot labels

```
fig, ax = plt.subplots(figsize = (16, 6))
plot_acf(passenger_miles['revenue_passenger_miles'], lags = max_k) #<- adjust k
plt.title('ACF (Revenue passenger miles)', fontsize = 20)
plt.xlabel('Lag', fontsize = 18)
plt.ylabel('Autocorrelation coefficient', fontsize = 18)
ax.tick_params(labelsize = 14)
plt.show()
```



Interpret autocorrelation function

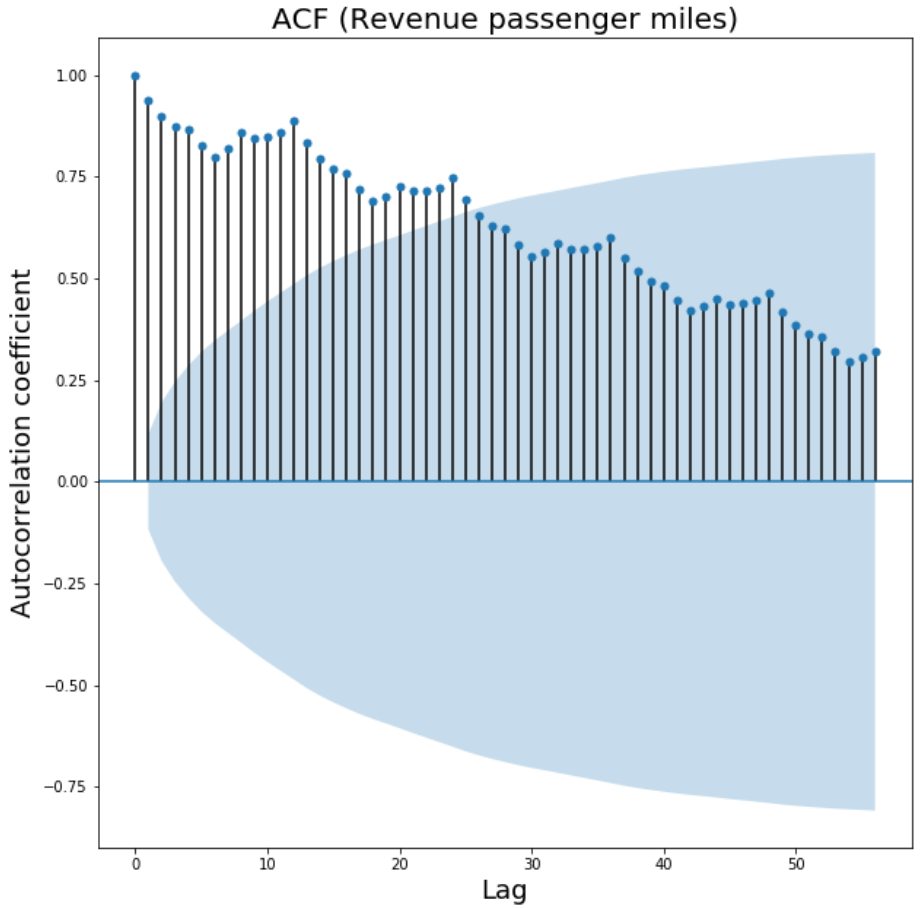


Interpretation:

- The lag value k is displayed on the `x-axis`
- The autocorrelation coefficient r_k is displayed on the `y-axis` (notice it is a value between $[-1, 1]$)
  - Vertical lines and points help us get a better perception of the magnitude of the autocorrelation
- The **shaded area represents confidence intervals** (95% by default), which suggest that correlation values outside of this cone are very likely to occur in real life and are not just a statistical artifact
- This plot is sometimes called a *correlogram* or an *autocorrelation plot*

Interpret autocorrelation function: our use case

- In our dataset, $k = 1$ means a lag of 1 month
- The plot shows autocorrelation coefficients outside of the blue shaded confidence interval up until about $k = 26$
- This means that our autocorrelation estimates become unreliable beyond that point
- In reality, how well do you think you could predict passenger mobility 2 years from now?



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**

Knowledge check 4

Exercise 2

Module completion checklist

Objective	Complete
Introduce forecasting and time series analysis use cases	✔
Review key quantitative forecasting methods	✔
Introduce data used in time series analysis	✔
Review important features of time series data	✔
Visualize time series	✔
Explain time series modeling basics and key components of time series	✔
Explain how to measure linear relationships within time series	✔
Estimate and visualize autocorrelation of time series	✔

Workshop: next steps!

- Workshops are to be completed in the afternoon either with a dataset for a capstone project or with another dataset of your choosing
- Make sure to annotate and comment your code
- This is an exploratory exercise to get you comfortable with the content we discussed today

Tasks for today:

- Read a NY Times Magazine article: ***The Weatherman is Not a Moron***
- Discuss with your peers
  - Which concept taught in today's class demonstrates the idea in the article about inaccuracies of long-term forecasts?
- Find a time series dataset to analyze
- Set an objective of what you would like to be able to do at the end of the time series course
- Explore and plot the data
- Experiment with various values of lag and create lag plots
- Compute ACF and review the results

Congratulations on completing this module!