2. Climate Data and Financial Analysis

2.1 What is Climate Data?

Climate data is the information collected on Earth's land, seas, and the surrounding atmosphere (EPRI). As shown in Figure 1, we can classify climate data into five categories: temperature, wind, hydrologic, precipitation, and solar.

Figure 1. Climate Data Categories and Examples

Category	Example
Temperature	Air temperature, extreme temperature, humidity
Wind	Wind Speed, Wind Direction
Hidrologic	Riverflow, Runoff
Precipitation	Rainfall, Snowfall
Solar	Cloud Cover, Radiation

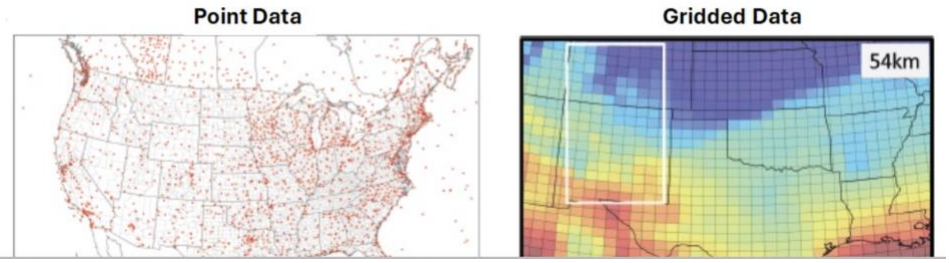
Source: EPRI

3. Characteristics of Climate Data

3.1 Data Coverage ¶

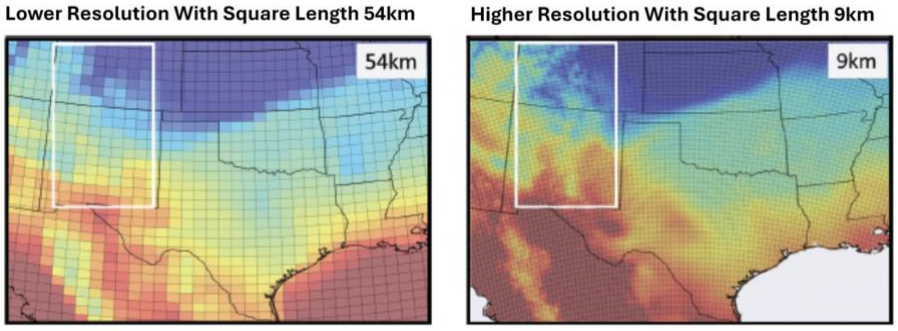
There are two ways to present climate data: **point data** and **gridded data**.
Point data is the climate data representative of a specific location and is collected from surface stations scattered around the globe. Since a station is located at a specific point, the data collected will represent the climate conditions for that specific location. Surface stations are distributed unevenly; as such, point data may be unavailable for certain parts of the globe. We will talk about this issue later.
Gridded data is the climate data representative of a specific area on Earth. In order to collect Earth data systematically, scientists divide the globe into many squares to create a grid system, like CRS, which we introduced in the lesson on geospatial data. Then, climate data is collected for each squared area. Figure 2 demonstrates the concept of data covering areas of point data and gridded data.

Figure 2. Data Covering Areas of Point Data and Gridded Data



The grid system varies according to the size of the square. The length of the square can range from 9km to 111km. The shorter the length of the square, the more squares can cover the globe and the more data can be collected. This results in higher resolution of the data. We can also say this data has higher spatial resolution. Figure 3 demonstrates examples of different spatial resolutions.

Figure 3. Low Spatial Resolution vs. High Spatial Resolution



Source: EPRI

Both point data and gridded data can be collected with different time frequencies. The common frequencies are hourly, daily, seasonal, and annual data. The more frequently data is collected, the more data is available. We usually call the data collected with higher frequencies as having higher **temporal resolution**.

3.2 Data Sources

There are two sources for climate data: **direct observation** and **reanalysis**.
Direct observation involves collecting climate data through surface station networks and satellite observation. A station records the precipitation, air temperature, and other climate data at a specific location. Direct observation data from a surface station is also called point data as described above. The earliest historical direct observation data can go back to around the 1850s. There can be some issues with direct observation data. The first one is a data gap. If a station's monitoring sensor breaks down, there will be missing data. The second issue is unevenly distributed observations. In the last section, we mentioned that surface stations are not evenly distributed around the globe. Currently, there are more stations set up in Europe and North America. Observation data outside of these two regions are sparse. Also, the data frequency or temporal range can be different by data type. Hence, direct observation data alone cannot provide us with a comprehensive view of the global climate system.
Reanalysis is a method to combine historical direct observation data with short-range forecasted data from previous observations to fill in data gaps and get the best view of the current Earth system. With reanalysis, we can get complete actual-simulated-combined historical global climate data, and the data will also be consistent in time.

4. Climate Projection Data

In the last section, we talked about the data sources for historical climate data. Earth scientists have also been working closely to create **projected** climate data into the next 30 years and even further out. Projected climate data will provide us with a forward-looking view for climate change and help us plan to adapt to future changes to our living environment.
There are two important components for generating climate projection data: **climate models** and **climate scenarios**.

4.1 Climate Models

A climate model contains a collection of physics principles and mathematical equations to represent the Earth system. The model usually contains thousands of lines of codes and requires a supercomputer to run it. There are climate models with different coverages: some climate models only focus on specific regions of the Earth, while some climate models cover the whole planet. One group of popular climate models is the general circulation models or global climate models (GCMs). **Global climate models (GCM)** are 3D climate models that cover the interactions between the atmosphere, land, and ocean on the whole Earth. 3D means the model includes latitude, longitude, and height (pressure). GCMs divide the Earth into grid cells and model on each cell and its interactions with neighboring grid cells. Figure 4 demonstrates the modeling structure of GCM.

Figure 4. Modeling Structure of Global Climate Models

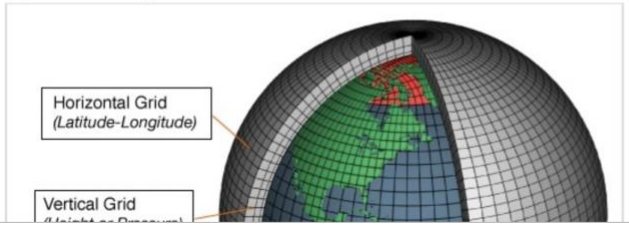
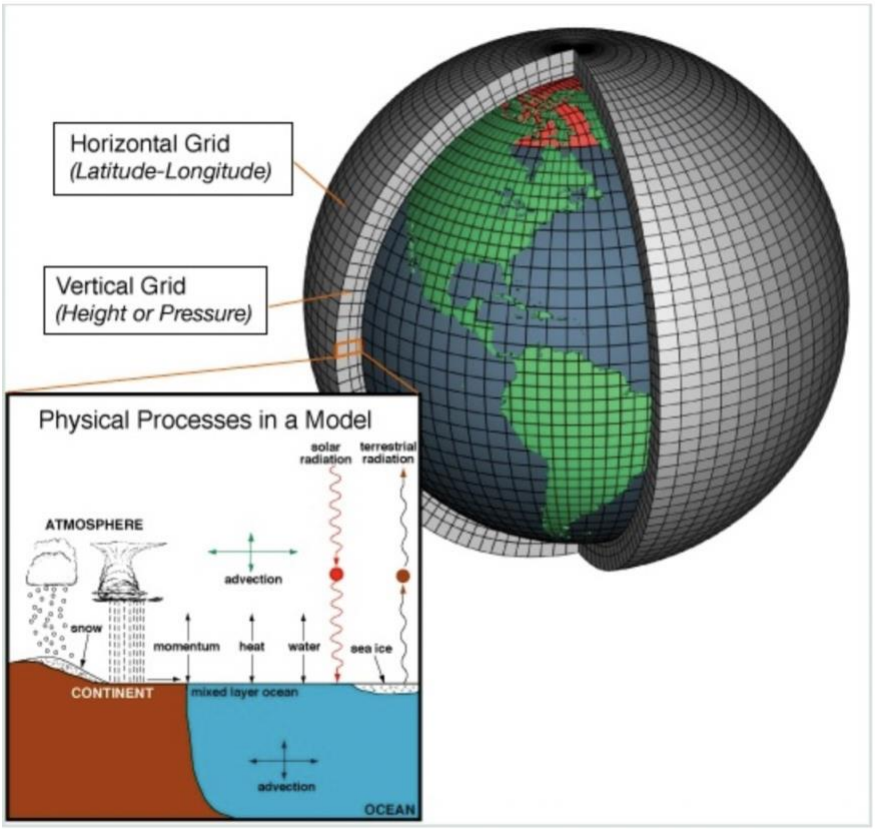


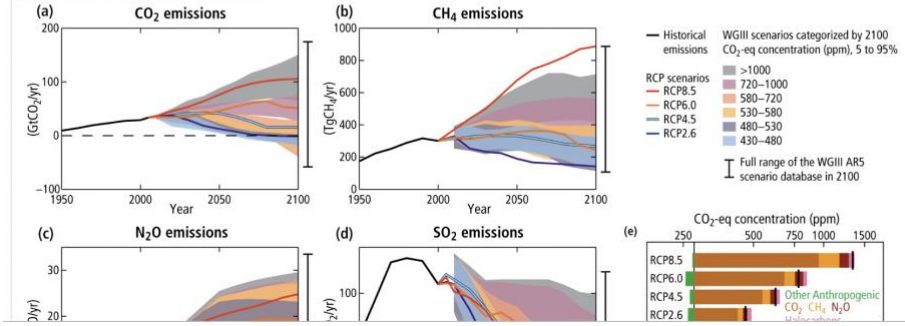
Figure 4. Modeling Structure of Global Climate Models



Source: NOAA

The key input variables to a GCM are external factors that can change the amount of energy from the Sun that is absorbed on Earth or will stay in the atmosphere. Some of the input variables are changes in the Sun's energy emissions and changes in greenhouse emissions. The model can output thousands of variables in different timeframes (hourly, daily, seasonal, and annual) to provide a complete view of the Earth climate system. ##**4.2 Climate Scenarios** Once a GCM is built, we can use the model to simulate the future Earth climate system and obtain projected climate data. To conduct a simulation, we need to provide climate scenarios as inputs to the model. There are several GCMs built by different institutions around the globe. In order for the projected outputs from different GCMs to be comparable, the Intergovernmental Panel of Climate Change (IPCC) from the United Nations has created standardized climate scenarios for climate projection. The most commonly used scenarios are **representative concentration pathways (RCP)**. There are four pathways showing different levels of greenhouse emissions, air pollution, and other external factors in the 21st century. They are RCP8.5, RCP6.5, RCP4, and RCP2.6. RCP8.5 has the highest level of greenhouse emissions and RCP2.6 has the lowest.

Figure 5. RCP Emission Scenarios



They are RCP8.5, RCP6.5, RCP4, and RCP2.6. RCP8.5 has the highest level of greenhouse emissions and RCP2.6 has the lowest.

5. Application with Python

In this section, we're going to demonstrate how to retrieve climate data from the **NOAA Physical Science Laboratory (PSL)** using Python. We will also introduce a data format called NetCDF4 that is commonly used to store climate data. Then, we will use Python to process and clean the data to be ready for analysis.

5.1 Climate Data from NOAA PSL

NOAA is short for **National Oceanic and Atmospheric Administration**, a U.S. governmental organization in charge of environmental intelligence. The Physical Science Laboratory (PSL) is a division in NOAA responsible for weather, climate, and hydrologic research. PSL houses a large amount of gridded climate data that is open to public scientists. You can find more information about available data from NOAA PSL here.

5.2 NetCDF Data Format

Network Common Data Form (NetCDF) is a popular data format to store gridded climate data. NetCDF is a multidimensional and array-based data format. NetCDF also contains its metadata and data attribute information. Hence, it is also called a "self-describing" data format. There are several versions of NetCDF data formats, but the latest version is NetCDF4. This is the data format currently used for NOAA PSL gridded climate data. We will discuss more properties of the NetCDF data format in the following example.

5.3 U.S. Daily Precipitation Data - Data Retrieve

We will use U.S. daily precipitation data for this demonstration. There has been tremendous progress made by scientists for global precipitation analysis for the last two decades (Xie et al.). This section will give you an overview of how to collect the data and analyze it. You can read the description of the dataset here. We are going to pull the data from 2011 to 2014. The Python example is based on Ali Ahmadalipour's Google Colab notebook example.

[1]
```python
# Import libraries for this demonstration
import glob
import matplotlib.pyplot as plt
import urllib.request
import xarray as xr
import pandas as pd
```

[2]
```python
# Retrieve data from NOAA PSL. Use for loop to pull data from 2011 to 2014
for yr in range(2011,2015):
    url = f'https://downloads.psl.noaa.gov/Datasets/cpc_us_precip/RT/precip.V1.0.{yr}.nc'
    savename = url.split('/')[-1]
    urllib.request.urlretrieve(url,savename)
```

5.4 Data Inspection

[3]
```python
# use open_dataset function from xarray to open data files for 2011 and assign file names
ds2011 = xr.open_dataset('precip.V1.0.2011.nc')
ds2011
```

xarray.Dataset

▶ Dimensions: (**lat**: 120, **lon**: 300, **time**: 365)

▼ Coordinates:

lat	(lat)	float32	20.12 20.38 20.62 ... 49.62 49.88
lon	(lon)	float32	230.1 230.4 230.6 ... 304.6 304.9
time	(time)	datetime64[ns]	2011-01-01 ... 2011-12-31

▼ Data variables:

precip	(time, lat, lon)	float32	...

▼ Indexes:

lat	PandasIndex
lon	PandasIndex
time	PandasIndex

▼ Attributes:

5.5 Data Slice

Now, let's try to pull some data points from the 2011 dataset and draw a graph. Since the whole dataset is for the U.S., let's focus on a city and use Boston as an example. The latitude of Boston is 42.36 and the longitude is -71.05. Let's first put this coordinate on a map to make sure this is the right one.

```
[5] # Import a mapping library
    import folium
```

```
[6] # Create a map with the coordinates for Boston
    latitude = 42.361145
    longitude = -71.057083
    map = folium.Map(location=[latitude, longitude], zoom_start=14, control_scale=True)
    folium.Marker(location=[latitude, longitude]).add_to(map)
    map
```

We presented the metadata for 2011 data in the above code. From the above metadata, we can see there are 3 dimensions: latitude, longitude, and time. There is one variable, precip, in the dataset.
The NetCDF data format is a hierarchical data structure. The longitude and latitude dimensions identify the location of the daily precipitation on the grid. The time dimension identifies the time location of the array. Let's check out some data from a 2011 dataset.

```
[ ]: # Check out the latitude values of first 4 observations from 2011 dataset
ds2011["precip"]["lat"].values[:4]
```

```
[5]: array([20.125, 20.375, 20.625, 20.875], dtype=float32)
```

```
[ ]: # Check out the longitude values of first 4 observations from 2011 dataset
ds2011["precip"]["lon"].values[:4]
```

```
[6]: array([230.125, 230.375, 230.625, 230.875], dtype=float32)
```

```
[ ]: # Check out the time values of first 4 observations from 2011 dataset
ds2011["precip"]["time"].values[:4]
```

```
[7]: array(['2011-01-01T00:00:00.000000000', '2011-01-02T00:00:00.000000000',
       '2011-01-03T00:00:00.000000000', '2011-01-04T00:00:00.000000000'],
      dtype='datetime64[ns]')
```

```
[ ]: # Check attribution of 2011 dataset
ds2011.attrs
```

```
[8]: {'title': 'CPC Unified Gauge-Based Analysis of Daily Precipitation over CONUS RT at PSD',
 'Conventions': 'COARDS',
 'description': 'Gridded daily Precipitation',
 'platform': 'Observations',
 'Comments': 'Preciptation is accumulated from 12z of previous day to 12z of day stored',
 'history': 'originally created RT starting 04/2010 by CAS from data obtained from NCEP/CPC\n converted to unpacked chunked netCDF4 Au
g 2014',
 'dataset_title': 'CPC Unified Gauge-Based Analysis of Daily Precipitation over CONUS',
 'References': 'http://www.psl.noaa.gov/data/gridded/data.unified.daily.conus.rt.html'}
```

```
# Slice the data for Boston. In this dataset, the longitude coordinate is represented by the absolute value of data value + 180.
latitude = 42.361145
longitude = 71.057083 + 180
boston_precip = ds2011.precip.sel(lat=latitude, lon=longitude, method = 'nearest')
boston_precip
```

xarray.DataArray 'precip' (**time**: 365)

[365 values with dtype=float32]

▼ Coordinates:

lat	()	float32	42.38
lon	()	float32	251.1
time	(time)	datetime64[ns]	2011-01-01 ... 2011-12-31

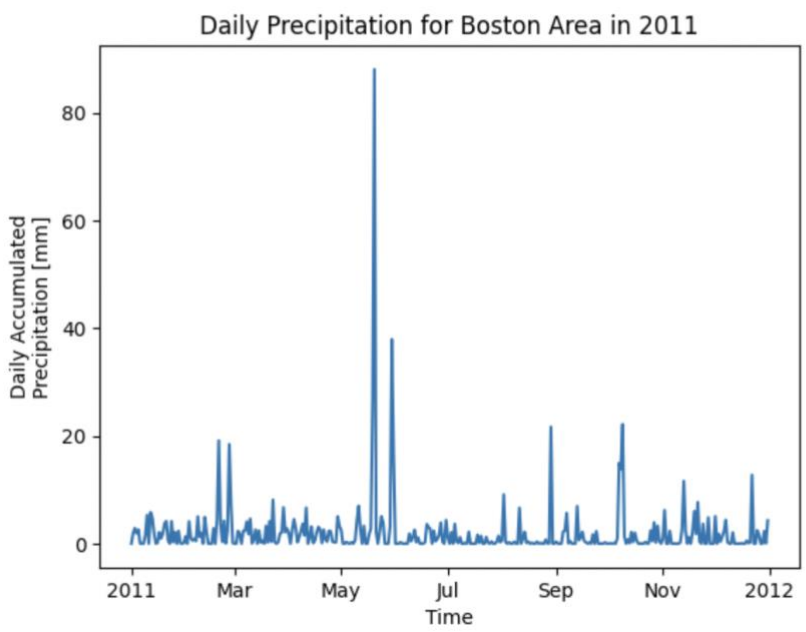
▼ Indexes:

time	PandasIndex

▼ Attributes:

long_name :	Daily Accumulated Precipitation
valid_range :	[0. 700.]
units :	mm
precision :	2
least_significant...	2
var_desc :	Precipitation
dataset :	CPC Unified Gauge-Based Analysis of Daily Precipitation over CONUS RT
level_desc :	Surface
statistic :	Daily Accumulation
parent_stat :	Observation
actual_range :	[0. 260.58218]

```python
# Let's draw a line graph
boston_precip.plot.line()
plt.title("Daily Precipitation for Boston Area in 2011")
plt.show()
```

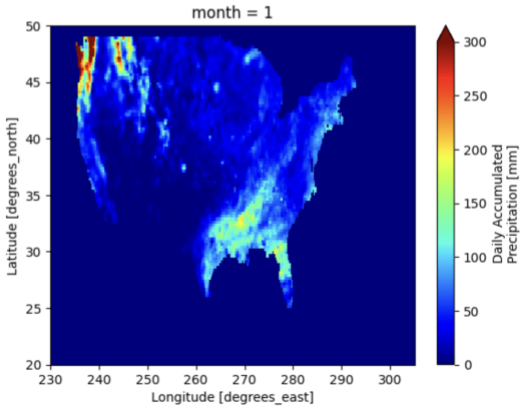


5.6 Data Aggregation

In this section, let's do some data aggregation. First, we'll use the 2011 dataset. It's a daily precipitation dataset. Let's aggregate to a monthly precipitation dataset and visualize it. The following Python code demonstrates how this is done.

```python
# Aggregate 2011 dataset from daily to monthly and visualize it.
ds2011_precip_mon = ds2011.groupby('time.month').sum()
ds2011_precip_mon.precip[0].plot(cmap='jet', vmax=300)
```

```
<matplotlib.collections.QuadMesh at 0x7c7bcbf174d0>
```



Expand/collapse section

In the above data aggregation example, we rolled up daily precipitation data into monthly precipitation data and showed the amount of precipitation in January 2011 on a map. The next aggregation we would like to demonstrate is combining several files into one. At the start of this lesson, we downloaded data from 2011–2014 from NOAA PSL. The following code shows how to combine them together.

```
# Combine several datasets
ds2011_2014 = xr.open_mfdataset('precip.V1.0.*.nc', concat_dim='time', combine='nested')
ds2011_2014
```

xarray.Dataset

▸ Dimensions: (**time**: 1461, **lat**: 120, **lon**: 300)

▾ Coordinates:

lat	(lat)	float32	20.12 20.38 20.62 ... 49.62 49.88	
lon	(lon)	float32	230.1 230.4 230.6 ... 304.6 304.9	
time	(time)	datetime64[ns]	2011-01-01 ... 2014-12-31	

▾ Data variables:

precip	(time, lat, lon)	float32	dask.array<chunksize=(1, 120, 300), meta=np…

▸ Indexes: (3)

▾ Attributes:

title :	CPC Unified Gauge-Based Analysis of Daily Precipitation over CONUS RT at PSD
Conventions :	COARDS
description :	Gridded daily Precipitation
platform :	Observations
Comments :	Preciptation is accumulated from 12z of previous day to 12z of day stored
history :	originally created RT starting 04/2010 by CAS from data obtained from NCEP/CPC converted to unpacked chunked netCDF4 Aug 2014
dataset_title :	CPC Unified Gauge-Based Analysis of Daily Precipitation over CONUS
References :	http://www.psl.noaa.gov/data/gridded/data.unified.daily.conus.rt.html


```
# Combine several datasets
ds2011_2014 = xr.open_mfdataset('precip.V1.0.*.nc', concat_dim='time', combine='nested')
ds2011_2014
```

xarray.Dataset

▸ Dimensions: (**time**: 1461, **lat**: 120, **lon**: 300)

▾ Coordinates:

lat	(lat)	float32	20.12 20.38 20.62 ... 49.62 49.88	
lon	(lon)	float32	230.1 230.4 230.6 ... 304.6 304.9	
time	(time)	datetime64[ns]	2011-01-01 ... 2014-12-31	

▾ Data variables:

precip	(time, lat, lon)	float32	dask.array<chunksize=(1, 120, 300), meta=np…

▸ Indexes: (3)

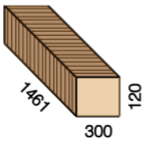
▾ Attributes:

title :	CPC Unified Gauge-Based Analysis of Daily Precipitation over CONUS RT at PSD
Conventions :	COARDS
description :	Gridded daily Precipitation
platform :	Observations
Comments :	Preciptation is accumulated from 12z of previous day to 12z of day stored
history :	originally created RT starting 04/2010 by CAS from data obtained from NCEP/CPC converted to unpacked chunked netCDF4 Aug 2014
dataset_title :	CPC Unified Gauge-Based Analysis of Daily Precipitation over CONUS
References :	http://www.psl.noaa.gov/data/gridded/data.unified.daily.conus.rt.html

From the above metadata, we can see the data variable is precipitation (precip) and the coordinates are latitude, longitude, and time. The time dimension is daily frequency. The following chart from Python can give us an idea of what the metadata structure looks like.

```
[ ] ds2011_2014.precip.data
```

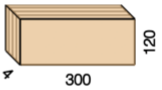
	Array	Chunk
Bytes	200.64 MiB	140.62 kiB
Shape	(1461, 120, 300)	(1, 120, 300)
Dask graph	1461 chunks in 9 graph layers	
Data type	float32 numpy.ndarray	



From the above 3D chart, one rectangular shape consists of 36,000 (300 X 120) precipitation data points for a day. There are 1,461 days from 2011 to 2014. On the time dimension, we can roll up by month or year. The following example shows how to roll up the daily data into annual data.

```
[ ] data_temp = ds2011_2014.groupby("time.year").sum()
    data_temp.precip.data
```

	Array	Chunk
Bytes	562.50 kiB	140.62 kiB
Shape	(4, 120, 300)	(1, 120, 300)
Dask graph	4 chunks in 52 graph layers	
Data type	float32 numpy.ndarray	



With a better understanding of the data structure, we can easily manipulate the dataset to get the different time aggregations we would like to have. In the next section, we'll show you how to turn this dataset into a pandas dataframe for future analysis work.

5.7 Convert NetCDF data to dataframe and save as a .csv file.

In this section, we are going to show you how to convert a NetCDF data into a data frame. Here is the code.

```
# Convert NetCDF to dataframe
ds2011_2014_df = ds2011_2014.to_dataframe()
ds2011_2014_df = ds2011_2014_df.dropna()
ds2011_2014_df.head(10)
```

			precip
time	**lat**	**lon**	
2011-01-01	**25.125**	**278.875**	0.0
		279.125	0.0
		279.375	0.0
		279.625	0.0
	25.375	**278.875**	0.0
		279.125	0.0
		279.375	0.0
		279.625	0.0
	25.625	**278.625**	0.0
		278.875	0.0

```
type(ds2011_2014_df)
```

```
pandas.core.frame.DataFrame
def __init__(data=None, index: Axes | None=None, columns: Axes | None=None, dtype: Dtype |
None=None, copy: bool | None=None) -> None

/usr/local/lib/python3.11/dist-packages/pandas/core/frame.py
Two-dimensional, size-mutable, potentially heterogeneous tabular data.

Data structure also contains labeled axes (rows and columns).
Arithmetic operations align on both row and column labels. Can be
thought of as a dict-like container for Series objects. The primary
```

From the above code, we learned how to convert the NetCDF4 dataset to a pandas dataframe. This will help us in future analysis.

6. Conclusion

In this lesson, we first introduced what climate data is and then described its attributes. In the second half of the lesson, we demonstrated how to pull climate data from a government agency and process climate data for future analysis. This lesson only covered the most basic concepts and methods for exploring climate data. For those who are interested in this topic, there is more information online.

L3: Numerical Methods for data preparation

1. Data Cleaning

In this first section, we review two groups of numerical methods: those for imputing and those for excluding.

1.2 Imputing: Filling in Missing Values

When we are given data, we may notice that some of the values are missing. Imputation is the name given to the action of providing missing values. We will learn various methods to fill in missing data, some of which are quite simple while others will instead involve a fair amount of complexity.

On the simple side, we have methods that simply use the last known value (e.g., fill forward). We could also replace a missing value by looking at its neighbors and performing some degree of interpolation: linear, quadratic, cubic, quartic, splines, etc. We'll also use resampling methods to handle missing or censored data. As we build our stochastic modeling skills, we'll look at the expectation maximization algorithm and the Brownian bridge approach.
One data-imputing application we learned in this course (Module 1, Lesson 4) is to use a cubic spline equation to fit a yield curve. In the section Cubic Spline Fitting of Yield Curve, we use a cubic spline equation to estimate missing yield information from the yield dataset. With this method, we are able to fit a smooth and piecewise yield curve for the following yield curve analysis.

1.2 Excluding: Outlier Detection

The previous subsection discussed how to handle missing values. We will also develop skills in the opposite direction: taking existing data values and excluding them, a process known as outlier detection. We will learn various methods for labeling and discarding outliers. Outliers are observations that cause the model's results to be extremely sensitive to those points. To understand outliers, we'll need to use both exploratory data analysis (prior to modeling) and diagnostic plots (post modeling). Exploratory data analysis is early visualization of raw data. Diagnostic plots show the sensitivity of data points to model parameter values and estimation.

Under the same umbrella of excluding, we'll look at dropping not just rows but also entire variables. Indeed, some of the methods we'll use take care of this automatically. For example, the Lasso method drops variables based on a parameter. Discarding the most egregious outliers could result in getting different parameters, which we can interpret and apply very differently. However, in our taxonomy, we'll consider feature extraction a separate category.

2. Data Engineering

2.1 Transforming and Normalizing

Sometimes, only a simple transformation is needed. A transform could be as simple as a conversion (e.g., degrees Fahrenheit to degrees Celsius), where the conversion factor never changes. Other times, the conversion factor does change: for example, converting Japanese yen to Nigerian naira.

What are some of the transforms we'll do? You've already transformed financial data series using logs. You may transform prices into volume-weighted average prices or VWAPs. We'll do this in the next lesson. You can transform prices to returns. You can transform financial time series using lags so that a series can be regressed on itself or lagged versions of other series. Some transformations are more complex, like a Fourier transformation, which will prove to be useful in option pricing.

Sometimes, we need to transform data to make it more usable through better-behaved measures such as having a zero-mean or a unit variance. Another set of transformations, called normalization, can have multiple meanings that depend on the context. For example, this includes the transformations that relate to making data more Gaussian: symmetric and bell shaped. In other cases, it is used interchangeably with standardization, where we subtract the mean and divide by the standard deviation. Lastly, it is used when we want to eliminate the effects of external factors, which is what we do when we detrend a time series. The purpose of these normalizations is to facilitate the analysis of the features of a dataset. Almost every example we'll do has one type of transformation. Over time, these will become second nature.

In this course, you have already transformed daily closing prices into logarithmic or percent returns. You have also seen that daily closing prices themselves are a transformation of raw tick-by-tick data. Furthermore, we computed the adjusted-close price by transforming the closing prices using data for splits and dividends. So, one can see that even the financial data readily available from multiple vendors are already heavily cleaned and transformed. In the rest of the Financial Engineering course, you will dive deep into multiple transformations such as standardization for comparing different assets on the same scale, Fourier transformations for option pricing, and many more. You'll learn not just how to apply these transformations but also understand their mathematical foundations and, most importantly, when and why to use them in real-world scenarios.

2.2 Factoring and Extracting

Factoring and feature extraction are another way of reorganizing data. Factoring is a difficult process because it involves both identification of factors and their measurement. There are a bewildering number of variations of forming factors. Principal components can be a first step, but these components are often not very intuitive. So sometimes, we simplify the problem by using classical features engineering to shape data that can be used in many ML algorithms. Less straightforward methods require a serious amount of subject matter expertise to define relevant factors. Within the professional world, you will encounter factor models, such as the FICO scores in credit risk. On the portfolio management and risk management sides, you will encounter commercially available factor models like the MSCI Barra Factor Index. There are different versions for different regions of the world.

Incidentally, the British refer to factor modeling as latent variable analysis: a good name to indicate that the variables need to be extracted. An example of a latent variable comes from the book (and Hollywood movie) *Moneyball*. For many years, it was thought that the most important metric of a baseball player was the number of hits or homeruns. However, a less obvious but more valuable factor is the player's percentage of getting on base. Aside from getting hits, there are two other ways to get on base. One way is to walk. The other is to get struck by a pitch. Whether the batter gets a hit, a walk, or hit by a pitch, the batter winds up on base and can potentially score, which is what wins games. The percent-on-base metric proved to be more important for winning games than the more easily observed number of hits. The "percent-on-base" factor is an example of a latent variable. One has to form a ratio: total times the batter gets on base from any of the three methods divided by the total number of times the batter has an official plate appearance.

One of the complexities of finding factors is overcoming biases. After writing *Moneyball*, Michael Lewis wrote *The Undoing Project*. In that book, he explains why walks were not considered as valuable as hits. In order for a batter to draw a walk, he must avoid swinging his bat four times when the pitches are not in the strike zone. From one perspective, it appears that the batter is taking no action. On the other hand, when the batter swings and connects the ball, there is the crack of the bat, running by both the batter and the players in the field, and applause by the fans—all reactions that give the illusion that the action is more valuable than the motionless stance of a walk, which typically draws less applause. However, both a single and a walk have the same outcome: the batter proceeds to first base. The cognitive bias is one of discounting a result because it was achieved by little action—not swinging. But the batter is exhibiting excellent skill with a good eye on a fast-moving ball and restraint from chasing bad pitches.

3. Statistical Analysis

3.1 Inferencing and Estimating: Parameter Evaluation

Flush with data, we will be well equipped to work on inference problems. Given samples of data, how do we best estimate the parameters of the population? What is a population's mean? Median? Standard deviation? First percentile? Skewness? Kurtosis? Correlation to another variable?

Suppose you assume the yield curve follows a polynomial model given by Nelson Siegel. Then, you can estimate the NS parameters from the data. This exercise is one you performed back in Module 1. Thanks to advances in computational power and contributions from fields outside statistics, there are many specific methods with which to perform inference. These methods typically fall into one of two major classes: frequentist approaches and Bayesian approaches. Within these are many subcategories, like nonparametric and resampling. Of course, regression problems involve an estimation of parameters, but we'll consider this category as the estimation of parameters outside of regressions. For example, if we fit a Student's t-copula to model the returns of two corporate bonds in a portfolio, how can we infer their correlation?

3.2 Simulating and Randomizing

This is one example where we do not need any data. Simulations provide an opportunity to create randomized data according to a known distribution. Generating the data is useful when we have new securities or histories that are too short or when we would like to test how our model works under known distributions. Monte Carlo (MC) simulations are the pre-eminent simulation in finance. We will use MC simulation to generate stock prices, returns, volatilities, interest rates, and many other variables. Depending on the complexity of the distribution, these simulations can provide a deep understanding of how models work with the specified type of data. Many methods in derivatives rely on MC simulations whenever simple closed-form analytic solutions do not exist. Recall that in Module 3, Lesson 2, we performed a Monte Carlo simulation to compute the Value at Risk from a Gaussian distribution.

If we want to impose a distribution on a system, then MC simulation tends to work very well. Other times, we may have a complex system in which it is implausible to run an MC simulation. In these cases, instead of working top down, we may work bottom up. One example of this approach is an agent-based model (ABM). Agent-based simulations are a different type of simulation that can model complex systems. Rather than fitting a distribution to a complex system with many interacting parts, an agent-based model (ABM) can be used to model the behavior of individual agents

If we want to impose a distribution on a system, then MC simulation tends to work very well. Other times, we may have a complex system in which it is implausible to run an MC simulation. In these cases, instead of working top down, we may work bottom up. One example of this approach is an agent-based model (ABM). Agent-based simulations are a different type of simulation that can model complex systems. Rather than fitting a distribution to a complex system with many interacting parts, an agent-based model (ABM) can be used to model the behavior of individual agents that then interact in complex ways. From this system, properties emerge that can create outcomes like those observed in the real world. Emergence is one of the key properties of ABM. We'll look at an ABM of a marketplace that replicates the Flash Crash.

In general, we'll use simulations for back-testing models; pricing and hedging derivatives; optimizing portfolios and trading strategies; and stress-testing risk models. In some cases, we'll generate multiple series of random variables, which preserve a given correlation to them. The ability to create random numbers is key throughout the entire program.

4. Conclusion

In this lesson and the next one, we are gathering and organizing the numerical methods you have encountered in this course and will encounter throughout the program. In this lesson, we focused on data cleaning, data engineering, and statistical analysis. Data cleaning identified imputing missing values and excluding outliers. Data engineering focused on transforming and normalizing data, as well as methods for factoring and conditioning data. Statistical analysis covered inference and estimation, as well as simulating and randomizing. In the next lesson, we will complete the second half of numerical methods, including core modeling, model refinement, and finding optimal solutions.

L4: Numerical Methods for Core Modeling

1. Core Modeling

In this first section, we review two large groups of core models: regression/classification models and models that reduce the search space and/or condition of the data.

1.1 Regressing/Classifying

The most common technical skill is regressing one variable on various predicators. Those regressions can take many different forms, which we explore in Econometrics and Machine Learning. If the target variable is continuous, we refer to the model as a regression problem. If the target variable is categorical, we refer to the model as a classification problem. We'll run linear and nonlinear regressions; cross-sectional and time series regressions; and univariate and multivariate regression. In some regressions, we'll easily observe the predictors; in others, we'll use factoring (another technical skill) to identify what the features are. Regressions can appear on the news channel, where you hear stories like "studies show that eating chocolate leads to a happier life" (undoubtedly sponsored by the chocolate companies).

From a software point of view, regressions are "black boxes" to which we supply inputs and crunch the numbers to produce the model outputs. However, like all the other areas, we will want to understand the details of the algorithm: its assumptions, derivation, advantages and disadvantages, and appropriate interpretations and applications of the results. In this program, black boxes should become white boxes, with a clear understanding of the mechanics to facilitate their best use.

In Financial Econometrics, you will encounter a very interesting regression problem for the variance rather than the level of a series. In a model known as GARCH, you will regress today's volatility using yesterday's volatility and yesterday's squared return. In Module 5, we cover generalized autoregressive conditional heteroskedasticity models, known as GARCH models, that are autoregression models for a time-varying volatility.

As you will see in Machine Learning, we will run a classification model using logistic regression. The difference between regression and classification mirrors the difference between continuous and categorical variables.

1.2 Reducing/Conditioning

The data we are given may be too big to handle. Learning how to divide the data is key. One method is using principal components: a data rotation technique that orders the data in orthogonal components ranked in order of decreasing variance. From this new set, we may choose only a selection of the components that captures the bulk of the variability of the data. Clustering is another way we may reduce the dimensionality of our data. Even when you perform conditional studies, you reduce the data by using only the subset that fulfills the condition. Classification trees and Naïve Bayes classifiers are two examples.

Recall from Module 1, Lesson 4 that a principal component analysis was done on n columns, and by keeping 3 principal components, we explained a vast majority of the variation of the data.

2. Model Refinement

In this section, we review two large groups of refining models: filtering and detecting change.

2.1 Filtering

Technically, filtering can happen anywhere in the data processing pipeline. For example, once we have imputed missing values and excluded outliers, we may want to filter data. We'll use filtering as a process to prepare data for analysis. At times, we may not need to do any filtering. End-of-day adjusted closing prices of an equity are likely ready to be converted to returns and modeled. However, an intraday version of the same stock series, with bid prices, ask prices, and trade prices, may require filtering to make the data more manageable.

Within a model, one way to think of filtering is applying weights. (Assigning a weight of 0 would mean we are excluding.) Other times, we will use filters to smooth data due to high noise content. One example of filtering will be the Kalman filter for improved estimation of signals. We'll see these mathematical topics in Stochastic Modeling. Another type of filtering is regularization, where we effectively assign penalties to complexity.

For example, in the Financial Econometrics course, we explore regularization techniques such as L1 (Lasso) and L2 (Ridge) when working with Linear Regression models. These methods help us "filter out" less important variables (in the case of Lasso) or normalize the distribution of regression coefficients by assigning weights in a more robust way (in the case of Ridge). These powerful techniques are just a glimpse of what

For example, in the Financial Econometrics course, we explore regularization techniques such as L1 (Lasso) and L2 (Ridge) when working with Linear Regression models. These methods help us "filter out" less important variables (in the case of Lasso) or normalize the distribution of regression coefficients by assigning weights in a more robust way (in the case of Ridge). These powerful techniques are just a glimpse of what you'll learn in Financial Econometrics, where we dive deeper into sophisticated statistical methods for analyzing financial data.

2.2 Detecting Change

One of the most useful technical skills in statistics is the ability to detect meaningful changes in the system. Detecting change may trigger a trading model to start trading or a risk model to no longer be effective due to exceeding a volatility breach. Changes can occur statistically when we look at a process that transitions from stationary to non-stationary or from mean-reverting to trending or that switches signs from positive to negative. We'll explore change detection when we run tests for unit roots and stationarity. We'll explore these in regime shifting models. We'll look at error correction models within this group to determine how a temporary disequilibrium may revert to their equilibrium.

Furthermore, we can think of detecting change as a rate of change, which brings us to derivatives. Classical calculus teaches us that the first derivative represents the rate of change. We'll learn methods of stochastic calculus and learn how to take derivatives, including partial derivatives, of stochastic functions. These derivatives will turn out to be key sensitivities of how an option price varies with the underlying variables and parameters.

In our Stochastic Modeling course, you'll discover fascinating tools like Hidden Markov Models (HMM) that help us understand how financial markets switch between different states or "regimes." You will learn how to detect when the market transitions from a low-volatility "calm" state to a high-volatility "turbulent" state, or from a bullish to a bearish regime. This powerful technique allows analysts to adapt their strategies based on these changing market conditions. HMMs are just one example of the sophisticated yet practical tools you'll master in Stochastic Modeling.

3. Finding Optimal Solutions

In this section, we review two large groups of refining models: calibrating/optimizing and combining models.

3.1 Calibrating/Optimizing: Finding Solutions

Sometimes, we assume values for the constant parameters of models and then use them to price securities or estimate random variables. Other times, we will want to work in the opposite direction: input the prices and value from the current market and imply what parameter values cause those prices. This process is known as calibration. Optimization is related because it seeks to find a minimum (if maximum, we take the negative) of an objective function by changing the value of key parameters subject to specified constraints. Optimizations are a type of constrained regression if you will. These tools will require us to be relentlessly methodical and meticulous in our ability to hunt for the global minimum. Thanks to an ensemble of topics in stochastic modeling, machine learning, and deep learning, you will encounter many examples of both calibration and optimization. One of the great quants, the late Peter Carr, said that the top skill the industry looks for from FE graduates is their ability to calibrate and optimize models they use. The skills require range across mathematics, statistics, and programming so that deficiencies in any one area limit the optimizer in performing their job. For example, we'll estimate the coefficients of a GARCH(1,1) model using Bayesian techniques.

Optimizations appear in many places. Every regression problem has an associated optimization problem (minimize the sum of least squares; maximize the likelihood, etc.). Many imputations will involve an optimization. When we train and test a model, we will want to optimize the bias-variance tradeoff by minimizing a mean squared error or similar metric.
In Module 5 of the following Financial Econometrics course, we will learn the Bayesian technique to estimate a GARCH model. Also, in Module 2 of the Stochastic Modeling course, we will learn how to use various methods to calibrate a credit risk model.

3.2 Combining: Strength in Numbers

Sometimes, the "best" model is not a single model but a combination of models. Indeed, we will be examining how models that were developed independently can actually be combined to work better. For example, in a future course, you will see how an econometric model GARCH can be combined with a deep learning model to improve its performance. Instead of always looking for the best model, we may combine models. For instance, in Machine Learning, you will see two linear regression methods, one called LASSO and the other called Ridge regression. Which works better? Well, there's a third method, called elastic nets, that uses both in the same regression. So oftentimes, you may use elastic nets and set the weights to include some amount of each model. In addition, we'll learn how to combine results from different types of models through ensemble learning. Specifically, we'll discuss bumping, bagging, and especially boosting. As we increase our modeling ability, we'll even be able to combine classical models like GARCH with machine learning algorithms. For example, we can combine GARCH models and neural networks to get better results than with either one individually.
In the following Machine Learning course, you will learn how to combine results from several models to produce better predictions than a single model in Module 4. This module will provide us more information on this topic.

4. Building Your Technical Foundation

4.1 Mathematics

The Quantitative Proficiency Test addressed many of the mathematical areas. Math provides much needed rigor. Often, engineering students may feel that the code or solution is enough: a problem of the ends justifying the means. The problem is that mathematics is the language that is needed to get to the heart of understanding. It is not essential to master the math as if you were a mathematician. Indeed, math is rigorous in that the mathematician is tasked with proofs of theorem, lemmas, and many details. The financial engineer (FE) needs to learn enough math to be rigorous when needed but not too much at the expense of other areas. Math is almost never learned by reading the book but by reading, rereading, getting used to notation, studying examples, solving problems, getting stuck, figuring out where the disconnect occurs, remedying that, and repeating. Math skills also include the ability to read and create equations in LaTeX so that you can post or reply in forums and complete your assignments correctly.

4.2 Modeling: Statistics/Machine Learning

Continuing our discussion of the quantitative area, let's call modeling skills a separate skill set. Under the umbrella of mathematics, we have linear algebra, calculus, probability, and discrete mathematics. In these areas, questions are more objective. There are more theorems, transformations, and processes than there are models, p-values, and test scores. Linear regression requires an understanding of mathematics (e.g., expressing the equation in matrix notation or implementing a maximum likelihood estimation). But linear regression is also an art. Two different statisticians working with the same data may come up with two different looking models that perform equally well. Some models may favor an easier interpretation, others a good prediction rate, and others could help by making costly errors less likely. We'll develop metrics for all these models starting in Econometrics and continuing in Machine Learning. There are often many ways to derive factors, use them in regressions, and combine the results. Statistics and ML are empirical and require an understanding of the assumption (often expressed mathematically) behind the models and this use.

4.3 Coding Skills

Being skilled in the mathematical and modeling areas, one may think the rest of the work is easy. This assumption is not the case because of implementation. Unfortunately, many students do not get enough hands-on experience coding. Coding involves multiple skills. Let's divide these into four areas.

The first skill is problem solving. Can students tackle a project by writing pseudocode that solves the problem devoid of any language, architecture, or framework? It is stunning that some students cannot outline clear instructions for an algorithm when taking a quiz, completing a capstone project, or answering an interview question. To experienced people, this step of an ordered, comprehensive set of instructions is second nature and is not explicitly separate from the coding. Nevertheless, as you learn this material, make sure you understand an algorithm, transformation, or model by outlining its steps in a spoken language (e.g., English) rather than Python.

The second skill is design. How will students write functions, design classes, and apply design patterns to architect a solution? For a simple project, writing a handful of functions may work well enough. But as a project's complexity and scope increase, it is prudent to design how the solution looks. While a certificate in software design or architecture is certainly not needed, an FE engineers a solution in software, so that solution should be organized to solve the problem efficiently, with room to make changes and additions. Practical experience shows that more time can be spent maintaining code than writing it in the first place. So solid code should allow the FE to maintain and update code as needed.

The third skill is the actual coding itself. Our official coding language is Python. That means you might learn how to do things in Python the same way you would have done them in C# or Java. Alternatively, you might learn how to do things the Pythonic way. These skills run the gamut of importing and exporting data, working with the right data structures, managing memory, and optimizing your use of existing packages. By the end of the program, Python should become as familiar as typing. You will literally spend hundreds of hours coding throughout the curriculum, including running existing examples, modifying and commenting code, and writing your own functions, classes, and even a package.

A fourth skill is debugging. It's amazing how much code is written without any checks. Students have submitted assignments with negative stock prices, negative probabilities, and negative volatilities. Some have submitted simulations with stock prices that don't change or where they climb to 10 to the 300th power! Debugging is essential because even if you can write code perfectly the first time, you are likely going to review a colleague's code at some point. Comments may be outdated if they even exist at all. There are often no help files, or the original author may be inaccessible (e.g., for model validation teams) or completely unavailable if they are not with the firm anymore. Debugging is more than throwing print statements in the code. It is a systematic way to find and fix errors as efficiently as possible. While the goal of this program is NOT to make you full-time software engineers, as an FE, you engineer models in code, so bugs in the code reflect on your skill as an FE. Remember that a weakness in coding due to a bug shows up as an FE who has not mastered their model.

5. Focus on Finance

Think of a stool with 3 legs: the legs are mathematics, statistics/machine learning, and computation. Strengthening each leg ensures the stool is sturdy. With a balanced, strong set of legs, the stool provides a place to sit at the table. On top of the table is a finance problem. Focus on understanding the problem from a financial point of view. When you enjoy and are successful at technical problems, it is tempting to focus on the technicalities, be attracted to learning new material, and take pride in the craftsmanship. These are noble attributes, and to be sure, the details are important.

But rather than focus on the tools, the emphasis should be on the value your tools bring to solving a problem. For example, suppose volatility is being measured with a metric X. You are excited to learn a new area, so you spend a few weeks using tools from that area to build a new metric Y. In the end, your new metric Y measures volatility very much the same as before. It is not more insightful, not quicker, not cheaper, not better—just the same. What is the value in such a project? There are liberties to undertake new research areas, but imagine project after project where the emphasis is on the tools and not the problems. If the project were motivated by a need to produce the metric more quickly, more accurately, or with greater insight, then the focus is finance driven rather than tool driven. Perhaps a metric is needed that works in sideways markets better. Perhaps a metric is needed on days where the trading volume is well above a certain threshold. This focus is key because it reminds us that we are engineering solutions to problems and not merely coding for coding's sake or doing math for math's sake. Maintain an element of practicality.

6. Building Your Non-Technical Foundation

Technical skills alone will not allow you to collaborate and communicate effectively with others. These non-technical skills are essential for success in working with others.

6.1 Communication Skills

Communication skills encompass a broad set of areas: visualizations, presentations, speaking, and written skills. Equipped with a trisection of math, modeling, and computational skills, students may think they are well prepared to enter the industry. However, there is another array of skills that is essential for professional success. Collectively, these are sometimes known as soft skills. For some people they are easier, and for others, they are much more difficult. The first is communication.

Our program builds communication skills through assignments, projects, and occasional presentations. The first of these areas is in visualization. How effectively can you communicate the results or application of your model? For example, some students print two pages of numerical output in an assignment thinking that is thorough. Why print out hundreds of numbers when a time series graph, histogram, or summary table will convey the same ideas in a cleaner, clearer, and more concise manner?

Although FE is a technical area, professionals are expected to communicate ideas to both technical and non-technical audiences. This includes your ability to write documents in which you keep your target audience in mind. Presenting to an audience is another important skill, whether it's an in-person or virtual presentation. Preparing a slide deck is a communication skill that finds a balance between the details in a project and the limited time and space to emphasize key points, some of which can be technical. Presentations can vary from code reviews to conferences, weekly team meetings, or even a milestone presentation given to senior management. Being able to use language, tone, and details effectively is an important skill. Managers tend to prefer employees who give effective presentations because they trust these people will be able to represent the group in a positive light to both internal colleagues and external clients. Fortunately, you will take a workshop on giving presentations. Interestingly, your technical skills can certainly assist in your ability to give effective presentations. Likewise, mastering a technical area should mean that you are able to explain the ideas and applications to both general audiences and technical audiences.

Communication also involves reading and listening. This includes being able to read forum posts, instructions for an assignment, and different types of required readings in your classes. You will also practice communicating with your fellow group members on more than two dozen group

7. Conclusion

If you only have the mindset of a mathematician, you simply write equations, derivations, and solutions on paper. If you just have the mindset of a programmer, you write the code, compile it, and feel a sense of accomplishment when it runs. If you only have the mindset of a statistician, you run the data through the model and call it quits when there's output. An FE needs to build the mindset of all three of those while also connecting the dots: how does this metric, model, or methodology solve a problem that was yet unsolved, improve a solution we already have, or give insight to a process that did not exist? Focus on the financial problem.