**Week 8**

1. Pandas
2. Seaborn

**Pandas**

Pandas is the de facto library for data manipulation and data frames in Python and contains all of the methods you would need to do typical data analysis. We will cover the most commonly used methods in this section but you may reference the API documentation for a full list of available methods. [API reference — pandas 1.4.3 documentation (pydata.org)](https://pandas.pydata.org/docs/reference/index.html). When importing pandas into python it is customary to use: import pandas as pd and reference any methods with the pd alias.

**Reading Data into Python with Pandas**

Pandas provides a very robust API for reading data into Pandas data frames from outside sources like CSV, text files, and other flat file formats. It also provides direct reads from many database technologies given a connection engine, but that is outside of the scope for this class.

The full documentation for read\_csv can be found [here](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html). As you can see Pandas allows you to easily change the column separator from comma to any separator you would like.

A screenshot of a computer

Description automatically generated with medium confidence

**Pandas Data Types**

Pandas supports a variety of different data types. Below is a table of pandas data types mapped to their base Python equivalents.

| **Pandas dtype** | **Python type** | **NumPy type** | **Usage** |
| --- | --- | --- | --- |
| object | str or mixed | string\_, unicode\_, mixed types | Text or mixed numeric and non-numeric values |
| int64 | int | int\_, int8, int16, int32, int64, uint8, uint16, uint32, uint64 | Integer numbers |
| float64 | float | float\_, float16, float32, float64 | Floating point numbers |
| bool | bool | bool\_ | True/False values |
| datetime64 | NA | datetime64[ns] | Date and time values |
| timedelta[ns] | NA | NA | Differences between two datetimes |
| category | NA | NA | Finite list of text values |

To extract the data types from your dataframe you can use the method .dtypes which will print each column and the associated data type.

Text

Description automatically generated with medium confidence

**Changing Data Types**

Pandas allows you to change the data types of your columns relatively easily by using the method .astype() with the corresponding data type. This method supports str, int, and Boolean types. Examples on how to use this functionality can be found [here](https://www.geeksforgeeks.org/change-the-data-type-of-a-column-or-a-pandas-series/). To convert strings (objects) into date time in Pandas you will need to use the to\_datetime method as shown in the below example.

Text

Description automatically generated

**Subsetting**

Pandas provides several different methods for subsetting rows and columns from dataframes.

Selecting columns from a dataframe:

Text

Description automatically generated

Subsetting rows:

This is relatively straight forward. Notice that unlike in R, you do not have to put a comma after the conditionals to specify you are subsetting rows. If you have multiple conditionals make sure that you have each individual conditional statement enclosed in parenthesis. One **very important** thing to understand is that when you subset rows in python the row names, or **index,** of the dataframe is subset with it. This can be problematic in any future statements you write against the subsetted dataframe. Pandas allows you to reset the index to avoid complications using the .reset\_index() method.

Text

Description automatically generatedText

Description automatically generated

Text

Description automatically generated

Subsetting rows & columns simultaneously:

In this case, a subset of both rows and columns is made in one go and just using selection brackets [] is not sufficient anymore. The loc/iloc operators are required in front of the selection brackets []. When using loc/iloc, the part before the comma is the rows you want, and the part after the comma is the columns you want to select.

Text

Description automatically generated

Text

Description automatically generated

**Creating new columns**

Creating columns in Pandas is relatively straightforward. In the below example we are creating the columns: StateAbbreviation, Year, and OrderID by splitting the Order ID column by the dash.

Text

Description automatically generated

**Removing Columns**

You can remove columns by providing a list of column names or a single column name to the method .drop() as shown. This same method can be used to drop rows by changing the axis.



**Reshaping data**

As we discussed in the R section of the course sometimes, we will need to reshape our data from wide to long or long to wide for various reasons. Pandas provides methods to do this in a very straight forward fashion.

Pivot:

Using pivot we can pivot long data into wide format. Notice that when doing this we have created a multi-index where you have the values columns in the first column name position, and the categories in the second. Most of the time we will want a single column name. To resolve this we can use the to\_flat\_index() method to coerce the columns names to a single index.

Text

Description automatically generated

Text

Description automatically generated

Melt:

Unpivot a DataFrame from wide to long format, optionally leaving identifiers set.

This function is useful to massage a DataFrame into a format where one or more columns are identifier variables (id\_vars), while all other columns, considered measured variables (value\_vars), are “unpivoted” to the row axis, leaving just two non-identifier columns, ‘variable’ and ‘value’.

Text

Description automatically generated

**Aggregation**

Aggregation in pandas provides various functions that perform a mathematical or logical operation on our dataset and returns a summary of that function. Aggregation can be used to get a summary of columns in our dataset like getting sum, minimum, maximum, etc. from a particular column of our dataset. The function used for aggregation is agg(), the parameter is the function we want to perform.

Some functions used in the aggregation are:

Function Description:

sum() :Compute sum of column values

min() :Compute min of column values

max() :Compute max of column values

mean() :Compute mean of column

size() :Compute column sizes

describe() :Generates descriptive statistics

first() :Compute first of group values

last() :Compute last of group values

count() :Compute count of column values

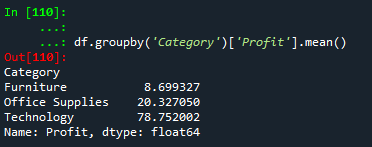
std() :Standard deviation of column

var() :Compute variance of column

sem() :Standard error of the mean of column

Example 1

We can perform a simple aggregation by grouping by a single variable and just calculate some aggregate value as shown below.



Example 2

By using the agg() method you can calculate more than one statistic on a group as shown. Simple assign them in a list.

Text

Description automatically generated

Example 3

Its possible to create custom aggregation functions and use them within a group by using the apply method.

Text

Description automatically generated

Example 4

You can use dictionaries to perform different calculations for different variables.

Text

Description automatically generated

Example 5

Its also possible to group by multiple variables and name the output columns. This can be useful for multiple calculated variables to avoid multi-indexing along the columns. Keep in mind when you group by more than one variable you will get multiple indexes on the rows. The syntax is the new variable name equal to a tuple of the original variable and the method to apply.

Graphical user interface, text

Description automatically generated

**Seaborn**

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. The reason it is typically used is because of how well it works with pandas pipelines. Since it is built on top of matplotlib you will usually import matplotlib along with seaborn as the labeling of the plots are still done through matplotlib. The best place to find professional examples is at their [gallery.](https://seaborn.pydata.org/examples/index.html)

Text

Description automatically generated

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

Adding custom titles and axis labels:

Again, this is done with the matplotlib library. You can change axis labels, axis formats, titles, etc. with matplotlib. For this to take effect, you must run this in the **same block of code.**

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

Formatting axis labels and axis label rotation

Using matplotlib we can create a custom formatter for the y axis and rotate the x axis labels so they are legible.

Text

Description automatically generated

Chart

Description automatically generated