Python 1: Core Data Analysis - Assignments

Python 1 – Part 2

Data Manipulation Using *pandas*

Exercise 1

In this exercise you will practice importing data stored in a CSV file, viewing records and setting the index. Note: df is a common reference to a generic DataFrame.

1. Use pd.read_csv() to import the Boeing stock data stored in BA.csv and store the DataFrame in a variable named ‘ba’.
2. Print the DataFrame info.
3. Convert the ‘Date’ column to a DateTime type using pd.to_datetime.
    a. Specify the format using: format=r‘%Y-%m-%d’.
4. Set the index of the DataFrame to be the date column using df.set_index().
    a. Make sure to pass ‘Date’ instead of ba[‘Date’] when setting the index, the latter will create a copy of the column.
    b. Print the info and head to verify the index was set correctly and the column was dropped.

We will import Boeing stock data again and setting the index using the parameters of read_csv().

1. Use pd.read_csv() to import the Boeing stock data store in BA.csv and store the DataFrame in a variable named ‘ba’.
    a. Specify the index column by passing: index_col=0.
    b. Instruct pandas to interpret the date as a DateTime type by passing: parse_dates=True.
2. Print the info of the DataFrame to verify the index is a DateIndexRange type.
3. Print the following:
    a. First 10 records of the DataFrame using .head().
    b. Last 3 records of the DataFrame using .tail().
4. Repeat Step 1, this time setting parse_dates to False.
    a. Print the info of the DataFrame, what do you notice about the index?

Python 1: Core Data Analysis - Assignments

Exercise 2

In this exercise you will practice extracting data from DataFrames, creating new variables and merging DataFrames.

1. Import the S&P500 (sp500.csv) and Boeing (ba.csv) data and store as DataFrames named sp500 and ba respectively.
2. Extract the Open and Close data from both DataFrames and store as separate DataFrames named sp500oc and ba_oc.
    a. Remember to use .copy() if you are not using .loc().
3. Calculate the simple and log returns for S&P500 and Boeing using the closing price.
    a. Use .shift() to create a column (variable) of the previous day's closing price.
4. Merge the two DataFrames into a new DataFrame.
5. Print the mean and standard deviation of the log and simple returns, format the output professionally.

Exercise 3

In this exercise you will practice extracting data from DataFrames, creating new variables based on conditions and merging DataFrames.

1. Extract the Open and Close data from both DataFrames and store as separate DataFrames named sp500oc and ba_oc.
    a. Remember to use .copy() if you are not using .loc().
2. Create a variable named 'Pos_Day' where the value is true if the stock/index had a positive day.
3. Merge the two DataFrames into a new variable.
4. Create a new variable that is equal to 1 if Boeing had a positive day and S&P500 had a negative day.
    a. Use .apply with a lambda function to check the condition for each row (axis=1).
    b. Parenthesis around each condition is required, and use 'and' instead of '&'.
5. Print the number of times Boeing closed positive and the S&P closed negative using .value_counts().
    a. Create a new column that keeps track of the days, use 1 or True and 0 or False.
    b. Apply the .value_counts() method to this column inside a print function.

Python 1: Core Data Analysis - Assignments

Exercise 4

In this exercise you will practice joining data that has gaps.

1. Import the Blackberry stock data from NYSE and TSE into separate DataFrames.
2. Extract only the data for January 2018, you can reassign the extracted data to the same variables.
3. Create a new DataFrame that is the TSE data joined with the NYSE.
    a. Use .join(), and don't forget to specify suffixes '_tse' and '_nyse'.
4. Print the Slice that is between January 12th and January 17th. How was the gap handled?

Exercise 5

In this exercise you will practice renaming columns and creating lagged and moving average variables. Note: When performing calculations, especially simulations, using return data, it is common to scale returns by a factor of 100 (0.005 becomes 0.5) and variances by 100^2 to maximize precision, and minimize the chance of an underflow error.

1. Import the S&P500 data into a DataFrame.
2. Rename the columns so that they are all lowercase and spaces are replaced with '_'.
    a. This can be doing using a loop and the .lower() and .replace() methods
3. One measure of volatility is the weighted average of the squared return of the previous day close to current day open, and the squared return of the current day open to current day close.
    a. Create a variable that is the previous day close using the shift function.
    b. Calculate the log return of the previous day close to current day open, store this value in a variable named 'log_rtn_c_to_o'.
    c. Calculate the log return of the current day open to the current day close, store this value in a variable named 'log_rtn_o_to_c'.
    d. Create variables named 'vol_c_to_o' and 'vol_o_to_c' which are the squares of 'log_rtn_c_to_o' and 'log_rtn_o_to_c' respectively.
    e. Create a variable that represents the weight applied to the close to open volatility, set the variable to 0.5.
    f. Calculate the weighted average of 'vol_c_to_o' and 'vol_o_to_c' and store the value in a variable named 'vol'.
4. Create a lagged variable of 'vol' named 'vol_1'.
5. Create variables that are the 5 day and 22 day moving average of 'vol' named 'vol_ma_5' and 'vol_ma_22' respectively.
    a. Use .rolling() and the .mean() methods on 'vol_1'
6. Print the first 30 records using .head() and save the DataFrame to a file.

Python 1: Core Data Analysis - Assignments

Advanced Questions

Exercise 6

In this exercise you will practice merging data of different frequencies and creating variables using financial report data.

1. Import the Boeing stock data into a DataFrame.
    a. Convert the header to lowercase and replace the spaces with '_'.
    b. Convert the 'date' column to DateTime.
2. Import the financials data that is stored in 'fundamentals.csv' into a DataFrame named 'financials'.
    a. Convert the header to lowercase and replace the spaces with '_'.
    b. Convert the 'period_ending' column to DateTime.
3. Regulators allow firms three months to complete their financial reports. In order to avoid matching data that technically would not have been available, we need to create another date variable to match along by offsetting it by three months. There are two approaches to handle this:
    a. Create a date variable 'match_date' in the stock data that is the month end minus three months. For example, for return data from Sept 15th, we would use financial data from June 30th (three months prior), which is the latest financial data that can be used. This can be done using the *pd.offsets.MonthEnd(-3)* function and adding it to the 'date' variable.
    b. Create a date variable 'match_date' in the financial data that is the month end plus three months. For example, if the financial data is for the period ending September 30th, three months later is December 31st which is the earliest the new financial data can be used. This can be done using the *pd.offsets.MonthEnd(3)* function and adding it to the 'period_ending' column.
    c. Regardless of approach, the other DataFrame will also require a 'match_date' variable which is simply the current month end variable. This can be done using the *pd.offsets.MonthEnd(0)* function. This is needed because merging using DateTime types must be exact, down to the millisecond.
4. Create a variable in the 'financials' DataFrame named 'roa' which is calculated by dividing Net Income by Assets.
5. Perform a left merge on the DataFrame containing the Boeing data with the financial data only selecting the data where 'ticker_symbol' equals 'BA' and only selecting the 'match_date' and 'roa' columns. Store this in a new DataFrame variable called 'ba_fin'.
6. Fill in the NA's using the .fillna(method='ffill') method.
7. This would have overfilled the data going forwards, and we also have NAs where financial data is not available.
    a. Create a variable from the financial DataFrame that is the max of the 'match_date' column where the 'ticker_symbol' equals 'BA'. Add a 1 year offset to this value because we use the last available financial data for the next 12 months. Call this variable 'max_date'.
    b. Create a variable from the financial DataFrame that is the min of the 'match_date' column where the 'ticker_symbol' equals 'BA'. Call this variable 'min_date'.
    c. Filter the 'ba_fin' DataFrame where the 'match_date' is less than or equal to the 'max_date'. You can reassign the filtered DataFrame back to 'ba_fin'.

   d. Filter the ‘ba_fin’ DataFrame where the ‘match_date’ is greater than or equal to the ‘min_date’. You can reassign the filtered DataFrame back to ‘ba_fin’.

8. Print .info() to see if there are any NAs in the ‘roa’ column.

Challenge: Repeat the steps above but use the .drop_na() method instead of filtering using the ‘min_date’ variable.

Exercise 7

In this exercise you will practice resampling data frequency and merging factor data.

1. Load the Boeing stock data into a DataFrame.
   a. Convert the header to lowercase and replace the spaces with ‘_’.
   b. Convert the ‘date’ column to DateTime.
2. Load the Fama French 3-Factor data stored in FF3.csv in a DataFrame called ‘ff3’.
   a. Convert the ‘date’ column to DateTime and offset to the current month end.
3. Create a dictionary that contains the rules for resampling:
   ohlc_rule = {'open':'first', 'high':'max', 'low':'min', 'close':'last', 'volume':'sum', 'adj_close':'last'}
4. Resample the daily Boeing data to monthly using the ‘ohlc_rule’. Assign the resampled data to ‘ba_mon’.
5. Calculate log or simple returns for ‘ba_mon’ using the ‘adj_close’ variable.
6. Perform a merge on the ‘ba_mon’ DataFrame with the ‘ff3’ DataFrame.
7. Print the first five records using .head(), what do you noticed about the scaling?
   a. Correct the scale problem by multiplying the returns by 100.