Workshop 10: Aggregation and merging

FIE463: Numerical Methods in Macroeconomics and Finance using Python

Richard Foltyn
NHH Norwegian School of Economics

March 20, 2025 (self-study)

There is no workshop this week due to the Student Symposium

See GitHub repository for notebooks and data:
`https://github.com/richardfoltyn/FIE463-V25`

Exercise 1: Daily returns of the magnificent seven

In this exercise, you are asked to analyze the weekly stockmarket returns of the so-called magnificent 7 which are some of the most successful tech companies of the last decades years: Apple (AAPL), Amazon (AMZN), Google (GOOG), Meta (META), Microsoft (MSFT), Nvidia (NVDA), and Tesla (TSLA).

1. Load the CSV data from `../../data/stockmarket/magnificent7.csv`. Inspect to first few rows to familiarize yourself with the columns present in the `DataFrame`.

   Keep only the columns `Date`, `Ticker`, `Open`, and `Close`.

2. You want to compute weekly returns for each of the 7 stocks. To this end, you need to reshape the `DataFrame` so that `Date` is the index and the remaining dimensions are in (hierarchical) columns.

   One way to achieve this is to use the `pivot()` functions. Call this function with the arguments `index='Date'` and `columns='Ticker'` and inspect the result.

   This should generate a hierarchical column index with `Open` and `Close` and the top level.

   Drop all rows with any missing values which arise because these stocks have been listed at different points in time.

3. Your data is now in a format that can be resampled to weekly frequency. Use `resample()` to convert the data to weekly observations.

   Compute the weekly returns as the relative difference between the *first* `Open` quote and the *last* `Close` quote for each ticker in each week.

   Hint: You should use `resample('W-MON')` so that the resampled weeks begin on Mondays (as opposed to the default Sundays).

   Hint: For example, to select the first `Open` value in each week, you should use `resample('W-MON')['Open'].first()`.

4. Create a 3-by-3 figure and plot the weekly returns you computed for each ticker as a histogram, using 25 bins (i.e., `bins=25` should be passed to the `hist()` function).

   Since you have only 7 tickers but 9 subplots in the figure, the last two remaining subplots should remain empty.

   Hint: You can either use `DataFrame.hist()` to plot the histogram, or Matplotlib's `hist()` function. In either case, you should add `density=True` such that the histogram is appropriately rescaled and comparable to the normal density.

5. **[Advanced]** Compare the histograms you created to the normal (Gaussian) probability density function (PDF) to get an idea how much weekly returns differ from a normal distribution.

   First, compute mean and standard deviation for each ticker and tabulate these.

   Then add a line showing the normal PDF to each of the return histograms you created previously, using the mean and standard deviation for each ticker.

   Hint: Use the `pdf()` method of the `scipy.stats.norm` class to compute the normal density.

6. Finally, you are interested in how the weekly returns are correlated across the 7 stocks.

   Create a figure with 7-by-7 subplots showing the pairwise correlations for each combination of stocks.

   You can do this either with the `scatter_matrix()` function contained in `pandas.plotting`, or build the figure using Matplotlib functions.

   [Advanced] Additionally, use the DataFrame.corr() method to compute the pairwise correlation matrix. Extract these values and add them as text to each of the 7-by-7 subplots (e.g., the correlation between returns on AAPL and AMZN is about 0.43, so this text should be added to the subplot showing the scatter plot of AAPL vs. AMZN).

Exercise 2: Business cycle correlations

Use the macroeconomic data from the folder `../../data/FRED` to solve the following tasks:

1. There are seven decade-specific files named `FRED_monthly_19X0.csv` where `X` identifies the decade (`X` takes on the values 5, 6, 7, 8, 9, 0, 1). Write a loop that reads in all seven files as DataFrames and store them in a list.

   Hint: Recall that you can use `pd.read_csv(..., index_col='DATE', parse_dates=['DATE'])` to automatically parse strings stored in the `DATE` column as dates.

2. Use `pd.concat()` to concatenate these data sets into a single `DataFrame` and make sure that `DATE` is set as the index.

3. You realize that your data does not include GDP since this variable is only reported at quarterly frequency. Load the GDP data from the file `GDP.csv` and merge it with your monthly data using an *inner join*.

4. You want to compute how (percent) changes of the variables in your data correlate with percent changes in GDP.

   1. Create a *new* `DataFrame` which contains the percent changes in CPI and GDP (using `pct_change()`), and the absolute changes for the remaining variables (using `diff()`).
   2. Compute the correlation of the percent changes in GDP with the (percent) changes of all other variables (using `corr()`). What does the sign and magnitude of the correlation coefficient tell you?

Exercise 3: Okun's law

In this exercise, we investigate Okun's law based on quarterly US data for each of the last seven decades.

Okun's law relates unemployment to the output gap. One version (see Jones: Macroeconomics, 2019) is stated as follows:

$$\underbrace{u_t - \overline{u}_t}_{\text{cyclical unempl.}} = \alpha + \beta \underbrace{\left(\frac{Y_t - \overline{Y}_t}{\overline{Y}_t}\right)}_{\text{output gap}} \tag{3.1}$$

where u_t is the unemployment rate, $\overline{u}_t$ is the natural rate of unemployment, Y_t is output (GDP) and $\overline{Y}_t$ is potential output. We refer to $u_t - \overline{u}_t$ as "cyclical unemployment" and to the term in parenthesis on the right-hand side as the "output gap." Okun's law says that the coefficient β is negative, i.e., cyclical unemployment is higher when the output gap is low (negative) because the economy is in a recession.

Use the FRED data in the `../../data/FRED` folder and perform the following tasks:

1. Load the time series stored in `GDP.csv` (real GDP), `GDPPOT.csv` (real potential GDP), `UNRATE.csv` (unemployment rate) and `NROU.csv` (noncyclical rate of unemployment), where the last series corresponds to the natural rate of unemployment mentioned above.

   Combine these series into a single `DataFrame` so that each represents a column, and keep only observations from from 1950-2019. The resulting data should be at quarterly frequency since GDP is only observed at these intervals.

   Hint: Use `pd.read_csv(..., index_col='DATE', parse_dates=['DATE'])` to automatically parse strings stored in the `DATE` column as dates and set it as the index.

2. Compute the output gap and cyclical unemployment rate as defined above and add them as columns to the `DataFrame`.

   Plot these variables in a scatter plot with the output gap on the x-axis and the cyclical unemployment on the y-axis. Does Okun's law hold over the sample period?

3. You wonder if the relationship has changed over the last decades. To answer this question, create a new column `Decade` which stores the decade of each observation, e.g., 1950, 1960, etc. Verify that each decade has 40 quarterly observations in your data.

   Hint: Since you have a date index, the calendar year can be retrieved from the attribute `df.index.year`.

4. Create a figure with 3-by-3 subplots showing the same scatter plot as above, but separately for each decade. Since we have data for only 7 decades, the last two subplots should remain empty.

5. **[Advanced]** Write a function `regress_okun()` which accepts a `DataFrame` containing a decade-spefic sub-sample as the only argument, and estimates the coefficients α (the intercept) and β (the slope) of the above regression equation (3.1).

   This function should return a `Series` with two elements which store the intercept and slope.

   To run the regression by decade, group the data by `Decade` and call the `apply()` method, passing `regress_okun` you wrote as the argument.

   Hint: Use NumPy's `lstsq()` to perform the regression. To regress the dependent variable `y` on regressors `X`, you need to call `lstsq(X, y)`. To include the intercept, you manually have to create `X` such that the first column contains only ones.

6. **[Advanced]** Plot your results: for each decade, create a scatter plot of the raw data and overlay it with the regression line you estimated.