Workshop 9: Introduction to pandas

FIE463: Numerical Methods in Macroeconomics and Finance using Python

Richard Foltyn
NHH Norwegian School of Economics

March 13, 2025

See GitHub repository for notebooks and data:

https://github.com/richardfoltyn/FIE463-V25

Exercise 1: Data cleaning

Before doing actual data analysis, we usually first need to clean the data. This might involve steps such as dealing with missing values and encoding categorical variables as integers.

Load the Titanic data set in `titanic.csv` and perform the following tasks:

1. Report the number of observations with missing `Age`, for example using `isna()`.
2. Compute the average age in the data set. Use the following approaches and compare your results:
    1. Use the `mean()` method.
    2. Convert the `Age` column to a NumPy array using `to_numpy()`. Experiment with NumPy's `np.mean()` and `np.nanmean()` to see if you obtain the same results.
3. Replace the all missing ages with the mean age you computed above, rounded to the nearest integer. Note that in "real" applications, replacing missing values with sample means is usually not a good idea.
4. Convert this updated `Age` column to integer type using `astype()`.
5. Generate a new column `Female` which takes on the value one if `Sex` is equal to `"female"` and zero otherwise. This is called an *indicator* or *dummy* variable, and is preferrable to storing such categorical data as strings. Delete the original column `Sex`.
6. Save your cleaned data set as `titanic-clean.csv` using `to_csv()` with `,` as the field separator. Tell `to_csv()` to *not* write the `DataFrame` index to the CSV file as it's not needed in this example.

Exercise 2: Daily returns of US stock market indices

In this exercise, we examine how the three major US stock market indices performed last year using data from Yahoo! Finance.

1. Use the `yfinance` library and its `download()` function to obtain the time series of daily observations for the S&P 500, the Dow Jones Industrial Average (DJIA) and the NASDAQ Composite indices. Restrict the sample to the period from 2024-01-01 to 2024-12-31 and keep only the closing price stored in column `Close`.

    Hint: The corresponding ticker symbols are `^GSPC`, `^DJI`, `^IXIC`, respectively.

2. Rename the DataFrame columns to `'SP500'`, `'Dow Jones'` and `'NASDAQ'` using the `rename()` method.

   Hint: `rename(columns=dict)` expects a dictionary as an argument which maps existing to new column names.

3. Plot the three time series (one for each index) in a single graph. Label all axes and make sure your graph contains a legend.

   Hint: You can directly use the `DataFrame.plot()` method implemented in `pandas`.

4. The graph you created in the previous sub-question is not well-suited to illustrate how each index developed in 2024 since the indices are reported on vastly different scales (the S&P500 appears to be an almost flat line).

   To get a better idea about how each index fared in 2023 relative to its value at the beginning of the year, normalize each index by its value on the first trading day in 2024 (which was 2024-01-02). Plot the resulting normalized indices.

5. For each index, compute the daily returns, i.e., the relative change vs. the previous closing price in percent. Create a plot of the daily returns for all indices.

   Hint: Use `pct_change()` to compute the change relative to the previous observation.