Python for finance and optimization
Classical tasks of buy-side finance

The goal of this part of the course is twofold: (i) learn how to deal with datasets and carry out some data visualization tasks, and (ii) learn how to build simple financial strategies, backtest them and assess their performance.

We shall use classical Python libraries: `pandas`, `numpy`, `matplotlib` and `seaborn`.

First lecture

From a raw dataset to a meaningful one

- Download the Excel file `https://www.oliviergueant.com/uploads/4/3/0/9/4309511/sbf120_as_of_end_2018.xlsx` that contains prices of SBF 120 components (as of end 2018) over the period 2011-Sept 2021.
- Discover what is in the file using `pd.read_excel`. Warning: there are several sheets.
- Understand by yourself the format of dates in Excel and how to use `pd.to_datetime` to carry out the conversion.
- Build a fast procedure to obtain a dataframe indexed by dates, with tickers as columns, and prices as values.
- Are there missing values? Why? How to deal with them?

Plots

- Using the dataset, draw by yourself (using `matplotlib`) the following graph (Figure 1) that represents the prices of BNP and Société Générale from January 2019 (with two axes). You should at least annonate similarly as I did, set the legends at the same places, rotate dates and draw a vertical grid.

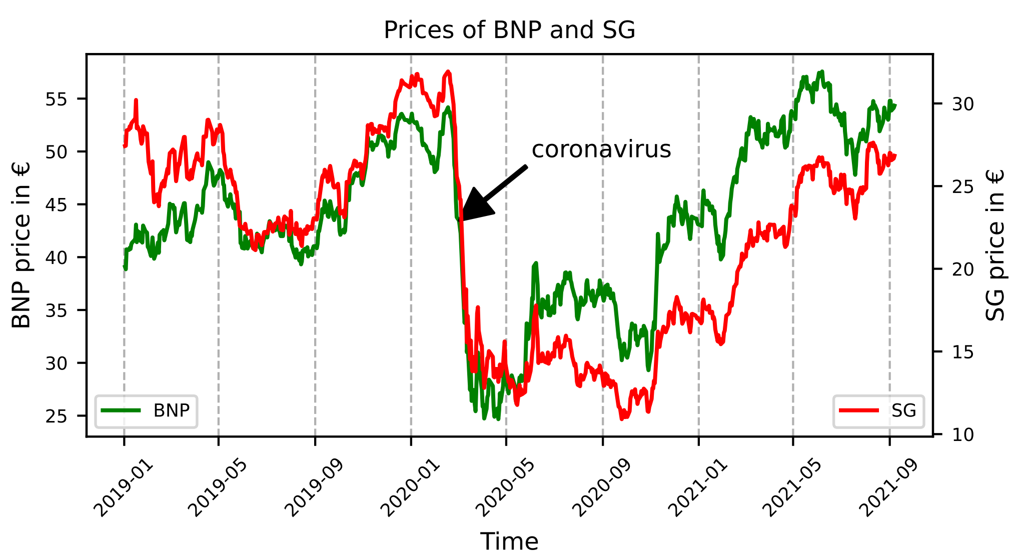

Figure 1: Prices

- Using the dataset, draw by yourself (using `seaborn`) graphs similar to Figures 2 and 3 that compare the one-dimensional and two-dimensional distributions of returns with a year-by-year analysis.

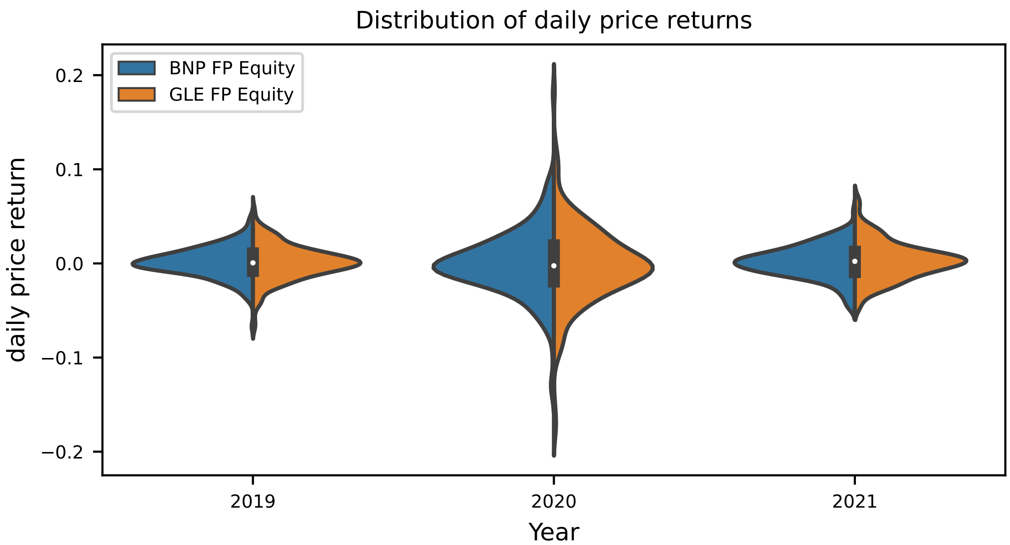

Figure 2: One-dimensional analysis

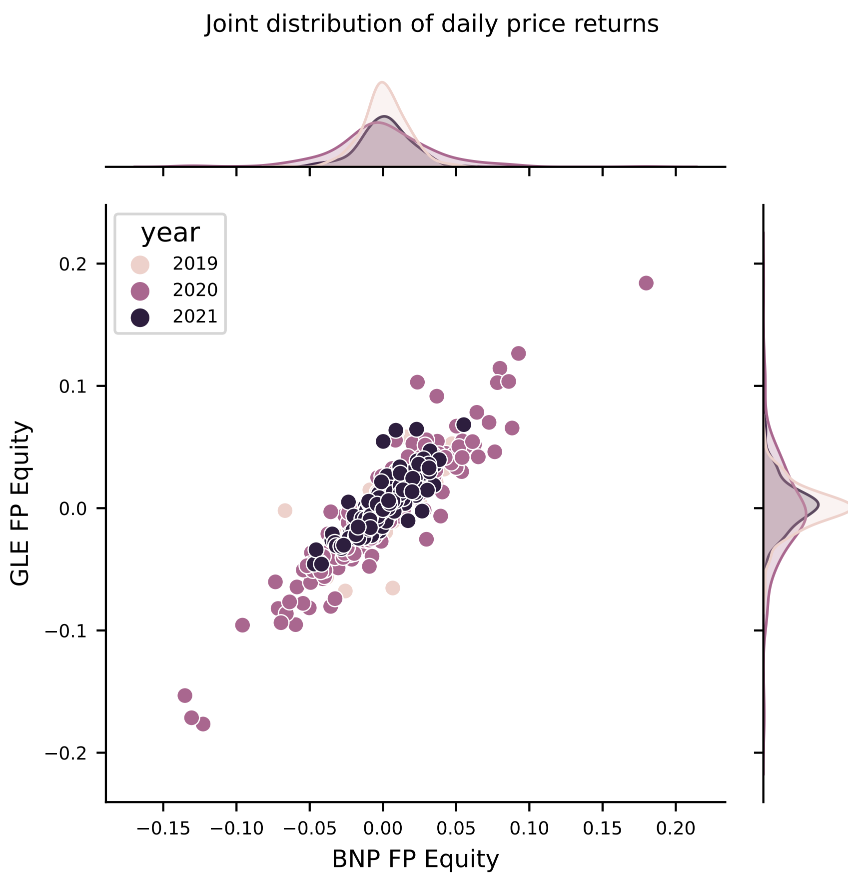

Figure 3: Two-dimensional analysis

End of the first lecture

Second lecture

Warning: from now on we focus on the period 2019-2020.

Datasets

- Clean your notebook from last week and get a clean dataframe for the daily returns of the 120 stocks over the period 2019-2020.
- Using the same dataset as in the first lecture, build a procedure to obtain a dataframe indexed by dates, with tickers as columns, and market capitalization as values.
- Clean the latter dataset to obtain a meaningful dataset over the period 2019-2020.

A class for analyzing strategies

- Create a class `Strategy` with the following characteristics:
  - a constructor taking a string (for the name of the strategy) and a Series of returns as inputs.
  - methods computing annual volatility, Sharpe ratio and maximum drawdown.
    Remark: for the maximum drawdown, please do not use loops.
  - a method to illustrate graphically the strategy (the way you like).

Reminder: If $(S_n)_{0 \leq n \leq N}$ *is the sequence of values of a strategy over a given period, the maximum drawdown is defined as*

$$MDD = -\min_{0 \leq n \leq N} \min_{m \geq n} \frac{S_m - S_n}{S_n}.$$

Remark: you can test your class on a long-only strategy on a single stock (for instance GLE FP Equity).

Capitalization-weigthed strategy

- Create a class `CapiWeighted` that inherits from the class `Strategy` and that allows you to analyze a capitalization-weighted portolio of a given list of stocks over a given period of time. The constructor should take dataframes as inputs.
- Use the previous class to study the performance of a capitalization-weighted portfolio in the 120 assets of the database over the period 2019-2020 and over each year of the period. Plot at least the PnL associated with the portfolio and document its main characteristics on the graph.