Homework 1: On REST APIs and Data Visualization

Proff. Alessio Martino & Irene Cannistraci

October 15, 2023

1 Scenario of Interest

Assume that, in order to build a dataset of securities' data, we want to retrieve the data related to the quotations of a given list of securities (from publicly available web resources):

- AAPL (Apple)
- GE (General Electric)
- GM (General Motors)
- ...

It is possible to retrieve historical stock data thanks to the following API:

https://financialmodelingprep.com/api/v3/historical-price-full/AAPL?serietype=line&apikey=[redacted]

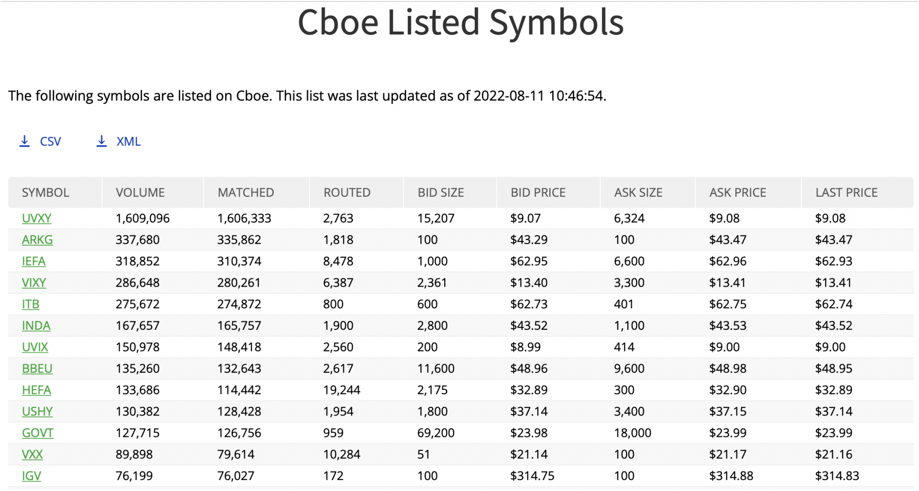
In this particular case, the URL features the AAPL symbol, which corresponds to Apple. By changing the symbol in the URL, you can retrieve data for a different security. A list of symbols can be found here (leftmost column in the table – see Figure 1 for an except):

http://markets.cboe.com/us/equities/market_statistics/listed_symbols/

Another aspect of the URL is the apikey parameter: the key already in the URL ([redacted]) should work. Otherwise, you can try the following ones:

- [redacted]
- [redacted]

If you query the API (i.e., by clicking the first link) you will get the usual response in JSON format (see Figure 2 for an except).

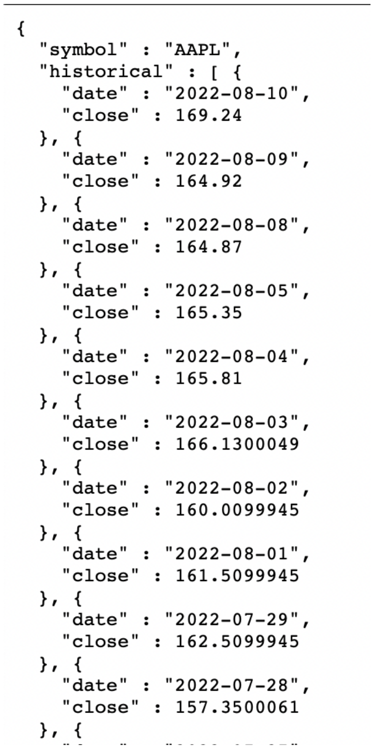


Cboe Listed Symbols

The following symbols are listed on Cboe. This list was last updated as of 2022-08-11 10:46:54.

SYMBOL	VOLUME	MATCHED	ROUTED	BID SIZE	BID PRICE	ASK SIZE	ASK PRICE	LAST PRICE
UVXY	1,609,096	1,606,333	2,763	15,207	$9.07	6,324	$9.08	$9.08
ARKG	337,680	335,862	1,818	100	$43.29	100	$43.47	$43.47
IEFA	318,852	310,374	8,478	1,000	$62.95	6,600	$62.96	$62.93
VIXY	286,648	280,261	6,387	2,361	$13.40	3,300	$13.41	$13.41
ITB	275,672	274,872	800	600	$62.73	401	$62.75	$62.74
INDA	167,657	165,757	1,900	2,800	$43.52	1,100	$43.53	$43.52
UVIX	150,978	148,418	2,560	200	$8.99	414	$9.00	$9.00
BBEU	135,260	132,643	2,617	11,600	$48.96	9,600	$48.98	$48.95
HEFA	133,686	114,442	19,244	2,175	$32.89	300	$32.90	$32.89
USHY	130,382	128,428	1,954	1,800	$37.14	3,400	$37.15	$37.14
GOVT	127,715	126,756	959	69,200	$23.98	18,000	$23.99	$23.99
VXX	89,898	79,614	10,284	51	$21.14	100	$21.17	$21.16
IGV	76,199	76,027	172	100	$314.75	100	$314.88	$314.83

Figure 1: Snippet of symbols table.



```
{
  "symbol" : "AAPL",
  "historical" : [ {
    "date" : "2022-08-10",
    "close" : 169.24
  }, {
    "date" : "2022-08-09",
    "close" : 164.92
  }, {
    "date" : "2022-08-08",
    "close" : 164.87
  }, {
    "date" : "2022-08-05",
    "close" : 165.35
  }, {
    "date" : "2022-08-04",
    "close" : 165.81
  }, {
    "date" : "2022-08-03",
    "close" : 166.1300049
  }, {
    "date" : "2022-08-02",
    "close" : 160.0099945
  }, {
    "date" : "2022-08-01",
    "close" : 161.5099945
  }, {
    "date" : "2022-07-29",
    "close" : 162.5099945
  }, {
    "date" : "2022-07-28",
    "close" : 157.3500061
  }, {
```

Figure 2: Snippet of JSON response for AAPL.

2 Goal #1: File Manipulation

Select three (or more) symbols amongst the ones in table (see second link and/or Figure 1) and write a program to store the results in different .json files (a file for each response).

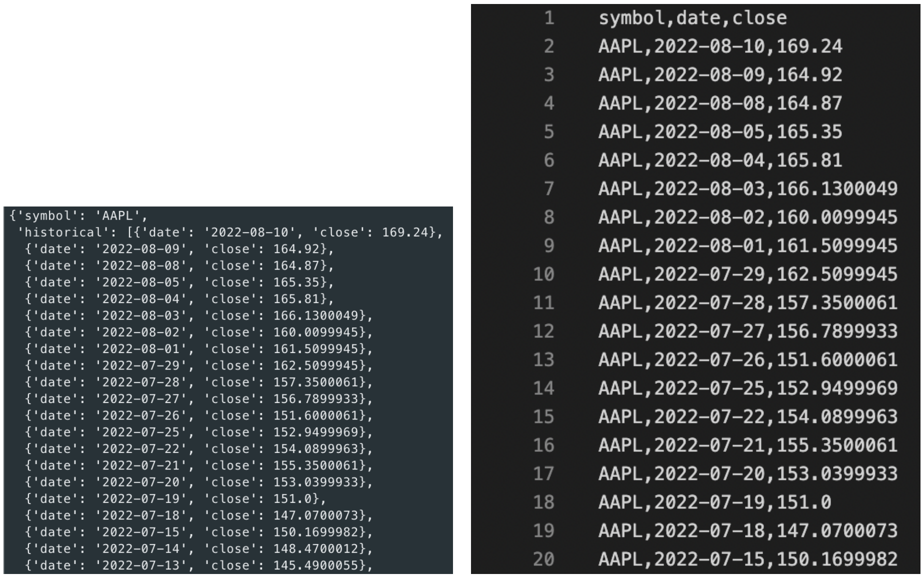
Hint. You can use the following lines of code to create a file, given a response variable `response` and a symbol stored in the variable `symbol`.

```
file = open("./"+symbol+".json", "w+")
file.writelines(response.text)
file.close()
```

Next, your Python program must be able to merge the contents of several .json files and convert it into a .csv (comma separated value) file.

Hint. Use `json.loads()` to load each .json content in Python objects. Then, perform the necessary manipulations to create a unique (merged) object containing all the data.

An example of conversion from .json to .csv is shown in Figure 3.



```
{'symbol': 'AAPL',
 'historical': [{'date': '2022-08-10', 'close': 169.24},
  {'date': '2022-08-09', 'close': 164.92},
  {'date': '2022-08-08', 'close': 164.87},
  {'date': '2022-08-05', 'close': 165.35},
  {'date': '2022-08-04', 'close': 165.81},
  {'date': '2022-08-03', 'close': 166.1300049},
  {'date': '2022-08-02', 'close': 160.0099945},
  {'date': '2022-08-01', 'close': 161.5099945},
  {'date': '2022-07-29', 'close': 162.5099945},
  {'date': '2022-07-28', 'close': 157.3500061},
  {'date': '2022-07-27', 'close': 156.7899933},
  {'date': '2022-07-26', 'close': 151.6000061},
  {'date': '2022-07-25', 'close': 152.9499969},
  {'date': '2022-07-22', 'close': 154.0899963},
  {'date': '2022-07-21', 'close': 155.3500061},
  {'date': '2022-07-20', 'close': 153.0399933},
  {'date': '2022-07-19', 'close': 151.0},
  {'date': '2022-07-18', 'close': 147.0700073},
  {'date': '2022-07-15', 'close': 150.1699982},
  {'date': '2022-07-14', 'close': 148.4700012},
  {'date': '2022-07-13', 'close': 145.4900055},
```

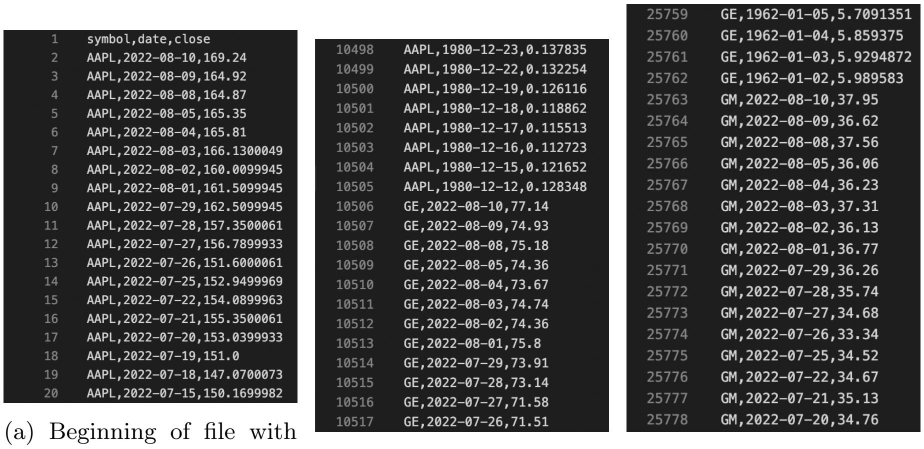
(a) .json snippet for AAPL

```
symbol,date,close
AAPL,2022-08-10,169.24
AAPL,2022-08-09,164.92
AAPL,2022-08-08,164.87
AAPL,2022-08-05,165.35
AAPL,2022-08-04,165.81
AAPL,2022-08-03,166.1300049
AAPL,2022-08-02,160.0099945
AAPL,2022-08-01,161.5099945
AAPL,2022-07-29,162.5099945
AAPL,2022-07-28,157.3500061
AAPL,2022-07-27,156.7899933
AAPL,2022-07-26,151.6000061
AAPL,2022-07-25,152.9499969
AAPL,2022-07-22,154.0899963
AAPL,2022-07-21,155.3500061
AAPL,2022-07-20,153.0399933
AAPL,2022-07-19,151.0
AAPL,2022-07-18,147.0700073
AAPL,2022-07-15,150.1699982
```

(b) .csv snippet for AAPL

Figure 3: Example of conversion from .json to .csv.

Figure 3 shows an example of conversion for AAPL to show the desired output. However, the .csv file must contain the data related to all of the three securities (stacked together) – see Figure 4.



```
symbol,date,close
AAPL,2022-08-10,169.24
AAPL,2022-08-09,164.92
AAPL,2022-08-08,164.87
AAPL,2022-08-05,165.35
AAPL,2022-08-04,165.81
AAPL,2022-08-03,166.1300049
AAPL,2022-08-02,160.0099945
AAPL,2022-08-01,161.5099945
AAPL,2022-07-29,162.5099945
AAPL,2022-07-28,157.3500061
AAPL,2022-07-27,156.7899933
AAPL,2022-07-26,151.6000061
AAPL,2022-07-25,152.9499969
AAPL,2022-07-22,154.0899963
AAPL,2022-07-21,155.3500061
AAPL,2022-07-20,153.0399933
AAPL,2022-07-19,151.0
AAPL,2022-07-18,147.0700073
AAPL,2022-07-15,150.1699982
```

(a) Beginning of file with AAPL

```
AAPL,1980-12-23,0.137835
AAPL,1980-12-22,0.132254
AAPL,1980-12-19,0.126116
AAPL,1980-12-18,0.118862
AAPL,1980-12-17,0.115513
AAPL,1980-12-16,0.112723
AAPL,1980-12-15,0.121652
AAPL,1980-12-12,0.128348
GE,2022-08-10,77.14
GE,2022-08-09,74.93
GE,2022-08-08,75.18
GE,2022-08-05,74.36
GE,2022-08-04,73.67
GE,2022-08-03,74.74
GE,2022-08-02,74.36
GE,2022-08-01,75.8
GE,2022-07-29,73.91
GE,2022-07-28,73.14
GE,2022-07-27,71.58
GE,2022-07-26,71.51
```

(b) After AAPL, I add GE

```
GE,1962-01-05,5.7091351
GE,1962-01-04,5.859375
GE,1962-01-03,5.9294872
GE,1962-01-02,5.989583
GM,2022-08-10,37.95
GM,2022-08-09,36.62
GM,2022-08-08,37.56
GM,2022-08-05,36.06
GM,2022-08-04,36.23
GM,2022-08-03,37.31
GM,2022-08-02,36.13
GM,2022-08-01,36.77
GM,2022-07-29,36.26
GM,2022-07-28,35.74
GM,2022-07-27,34.68
GM,2022-07-26,33.34
GM,2022-07-25,34.52
GM,2022-07-22,34.67
GM,2022-07-21,35.13
GM,2022-07-20,34.76
```

(c) After GE, I add GM

Figure 4: Final .csv file example.

Hint. If you properly manipulate the data and then you load the data in a Pandas DataFrame, you can use the to_csv() function to create a .csv file.

3 Goal #2: Time series plotting

As you can see from the .csv file, you have the evolution of the close over time (i.e., date) for each symbol. The next step is to plot the time series for each symbol.

To do so, you can import the .csv file back and consider each symbol separately. Then, make sure to sort the close values so that most recent ones appear first. Finally, use the Matplotlib library to plot the close evolution over time. An example for AAPL is given in Figure 5.

Hint. If you are familiar with Pandas DataFrames, you can import your .csv file thanks to the read_csv() function and then the sort_values() method to sort the DataFrame according to one of its columns.

Extra bit. As you can see, the plot in Figure 5 looks nice thanks to the titles on the x and y axes, the "AAPL" plot title and the grid. You can add those thanks to the plt.title(), plt.xlabel(), plt.ylabel() and plt.grid() functions. Go and read the docs to see how do they work:

- https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.title.html
- https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.xlabel.html

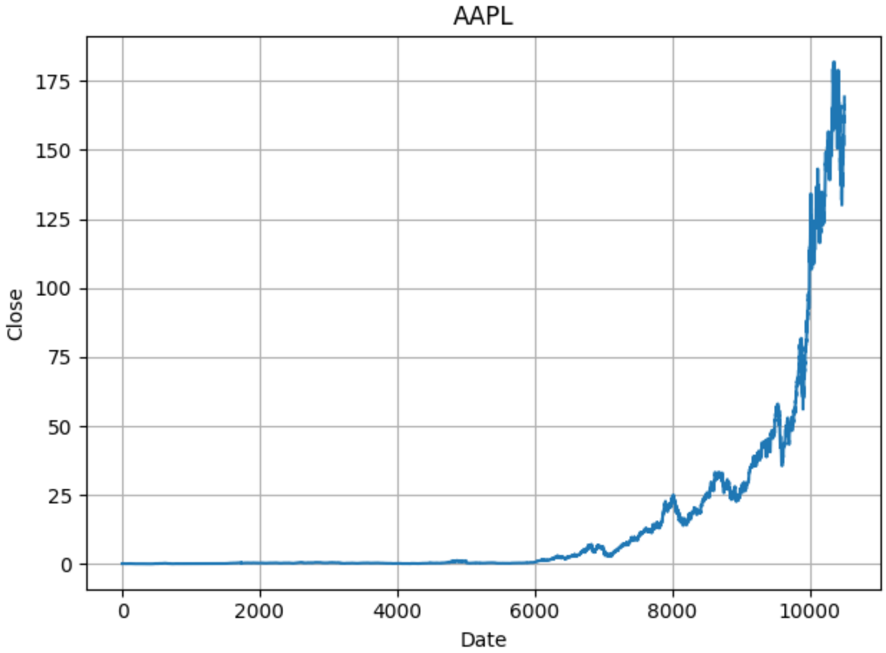


Figure 5: AAPL time series.

- https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.ylabel.html
- https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.grid.html

And yet, there is a very inelegant feature in the plot: the labels on the x axis. We plotted the `close` values as a function of the `date`, but there are no dates on the x axis! You can change the labels on the x axis by modifying the so-called x-ticks. Go and read the docs to see how do they work:

- https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.xticks.html

A nicer example is given in Figure 6.

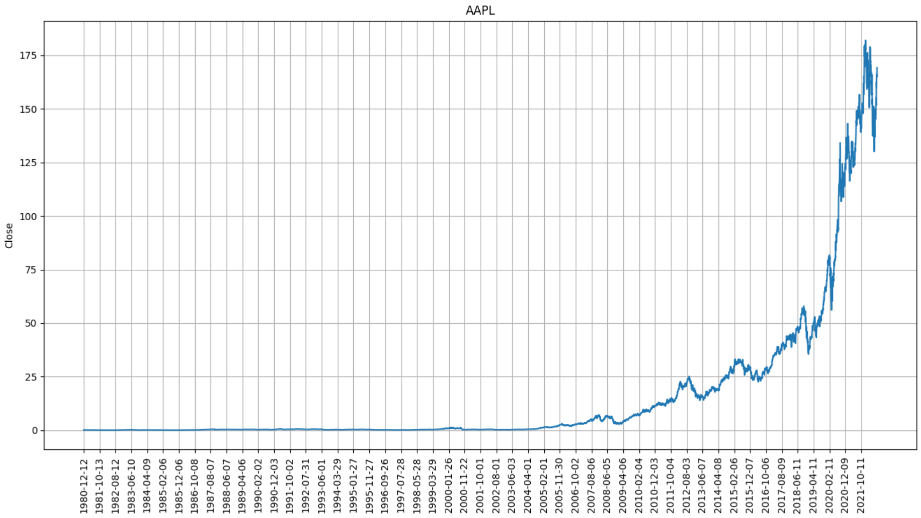


Figure 6: AAPL time series.