DSTA

Chapter V - Financial networks.

This demonstration (no exercise) notebook is taken from the notebook for Ch. 5 of Caldarelli-Cheesa's textbook (CC).

Please see the class repository for the datasets.

For local usage, it is recommended to install yfinance from within Anaconda, e.g.

```
C:\WINDOWS\system32>conda install -c ranaroussi yfinance
```

or from the Navigator tool, e.g. by subscribing to the "ranaroussi" channel. Without Anaconda, you may install the package directly:

```
C:\WINDOWS\system32>pip install yfinance
```

In any case these commands are repeated below, please comment out as needed.

Changelog

- June 2024 version by A. Matuozzo updates the code to the current availability of modules. Notice: recent Networkx versiona are deprecating Graphviz.pydot
- March 2023 version by P. Lagias runs on a reduced dataset of tickers to avoid issues with delisted/defunct stocks.

```
!pip install graphviz
```

```
!pip install yfinance
```

Discontinued:

the original yahooFinacials module is no longer used as it is falling out of maintenance. It used to be installed with `!pip install yahoofinancials`

Now, let us make sure that we have the `seaborn` module for visualisation

Notice: this notebook can be run without it, `matplotlib` suffices.

```
!pip install seaborn
```

```
import sys
import time
import math

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns


import networkx as nx

from networkx.drawing.nx_pydot import graphviz_layout

from collections import Counter

import yfinance as yf

#from yahoofinancials import YahooFinancials

#from yahoofinancials.etl import ManagedException
```

Connecting to the Yahoo! Finance service

What with Apple Inc. on May 19th 2014?

```
# Download data for 1 day as an example
data = yf.download("AAPL", start = "2014-05-19", end = "2014-05-20")

print(data)
```

```
#this data is structured as a pandas dataframe to benefits from its built in methods
data.head()
```

```
#setting the Date as index gives access to pandas functions to sample time series data
data.index
```

Compute and plot transaction volumes

How was Microsoft traded in 2019?

```
# Now download the data for the entire 2019 year.
d = yf.download("MSFT", start = "2019-01-01", end = "2019-12-31")

d['Transaction_Volume'] = d['Volume'] * d['Adj Close']

print(d)
```

```
# Downloaded data are saved as a dataframe
type(d)
```

```
# Use seaborn style defaults and set the default figure size
sns.set_style(rc = {'figure.figsize': (12, 6)})

d['Transaction_Volume'].plot(linewidth = 0.5);
```

The NYSE tickers

From the relative nasdaq.com page we download all the information related to the market capitalization, sector and industry…

Discontinued: the top-cap companies have been downloaded in the `data` section, use `!head companylist.csv` to visualise, OR SIMPLY USE THE Pandas version below.

Get Stock Labels, Sector and Industries

```
DATAFILE = "./data/list_stocks_50B_6_may_2016.txt"
```

```
# Get stock data from the text file
f = open(DATAFILE, 'r')

list_stocks = []

while True:
    next_line = f.readline()
    if not next_line: break
    # print(next_line.split('\t'))
    list_stocks.append(tuple(next_line.split('\t')[:-1]))

f.close()

# a huge dump of all stocks, uncomment only if needed
for stock in list_stocks:
    print(stock)
```

```
# Alternatively, you could use an easier to read DataFrame structure
stocks = pd.read_csv(DATAFILE, sep = '\t', names = ['Ticker', 'Name', 'Sector', 'Industry',
```

```
stocks.head()
```

```
# This code must stay commented as yahoo financials has some api issues.
# We have used the file with the required data above.

# # get values
# # May 6th 2016
# # greater than 50B$
# cap_threshold = 50_000_000_000
# hfile = open("companylist.csv", 'r')

# list_stocks = []
# nextline = hfile.readline()

# while True:
#     nextline = hfile.readline()
#     if not nextline:
#         break

#     line = nextline.split(',')
#     sym = line[0][1:-1]
```

```
#     # Skip entries with "^" in stock name
#     if sym.find("^") != -1:
#         continue

#     share = YahooFinancials(sym) # this cause an issue
#     y_market_cap = None

#     try:
#         y_market_cap = share.get_market_cap()
#     except:
#         y_market_cap = None
#         print(f"No link for {sym}")
#     # y_market_cap1=y_m

#     if not y_market_cap:
#         print(f"No market cap found for {sym}")
#         continue

#     # We will exclude stocks with char '^' that will
#     # give errors in the query process
#     if y_market_cap > cap_threshold:
#         print(sym, y_market_cap, line)
#         stock_data = (line[0][1:-1], line[1][1:-1], line[5][1:-1], line[6][1:-1])
#         list_stocks.append(stock_data)
#     time.sleep(1)

# hfile.close()
# print(list_stocks[0])
```

Generate dictionaries for companies, sectors and colors

```
dict_sectors = {}

for s in list_stocks:
    # print(s)
    dict_sectors[s[0]] = s[2]

list_ranking = []

for s in set(dict_sectors.values()):
    count = 0
```

```
    for key in dict_sectors:
        if s in dict_sectors[key]:
            count += 1

    list_ranking.append((count,s))

list_ranking.sort(reverse = True)

# list_colors=['red','green','blue','black''cyan','magenta','yellow']
list_colors = ['0.0', '0.2', '0.4', '0.6', '0.7', '0.8', '0.9']

# 'white' is an extra color for 'n/a' and 'other' sectors
dict_colors = {}

# association color and more represented sectors
for s in list_ranking:
    if s[1] == 'n/a':
        dict_colors[s[1]] = 'white'
        continue

    if list_colors == []:
        dict_colors[s[1]] = 'white'
        continue

    dict_colors[s[1]] = list_colors.pop(0)

print(list_ranking)
```

```
# Here you could refactor the sector dictionary like this:
sd = {k:v for k,v in zip(stocks.Ticker, stocks.Sector)}

sd
```

Retrieving historical data

Ticker by ticker, we download the historical data from Yahoo! Finance. The cell below will take time to run.

Example: `AEK` is not listed anymore, while `BABA` is in the NSYE top-cap only since Sep. 2014.

```
dict_comp = {}

for s in list_stocks:
    print(s[0])

    #stock = yf.Ticker(s[0])
    #diz_comp[s[0]]=stock.history("1mo")
    dict_comp[s[0]] = yf.download((s[0]), start = "2013-05-01", end = "2014-05-31")

# create dictionaries of time series for each company
```

```
# this is a dict of DataFrames: let's look inside, for example the well-known 3M company

dict_comp['MMM'][0:5]
```

```
dict_historical = {} # this is a dict of tickers: Pd.Series of Closing Prices (how many days

for k in list(dict_comp.keys()):
    tmp = dict_comp[k]
    dict_historical[k] = tmp['Close']
    """
    for e in diz_comp[k]:
        print(e)
        # string indices must be integers
        #diz_historical[k][e['Date']]=e['Close']
        #print(e)
        #diz_historical[k][e[0]]=e[4]
    """

for k in list(dict_historical.keys()):
    print(k, len(dict_historical[k]))
```

```
dict_historical['MMM']
```

Calculation of the logreturns

Let's visualise a company

```
REF_COMPANY = 'MMM'
```

```
dict_returns = {}

d = dict_historical[REF_COMPANY].keys()

# d.sort()
# print(len(d),d)

for c in dict_historical:

    # check if the company has the whole set of dates
    if len(dict_historical[c].keys()) < len(d):
        continue

    dict_returns[c] = {}
    for i in range(1, len(d)):
        # price returns
        return_t = math.log(float(dict_historical[c][d[i]]))

        return_t_1 = math.log(float(dict_historical[c][d[i-1]]))

        dict_returns[c][d[i]] = return_t - return_t_1

print(dict_returns[REF_COMPANY])
```

Basic Statistics and the Correlation Coefficient

For fun's sake, we define our own aggretated stats, including Pearson's correlation coefficient.

$$\rho = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

```
# mean
def mean(X):
    m = 0.0
    for i in X:
        m += i
    return m / len(X)

# covariance
def covariance(X, Y):
```

```
    c = 0.0
    m_X = mean(X)
    m_Y = mean(Y)
    for i in range(len(X)):
        c = c + (X[i] - m_X) * (Y[i] - m_Y)
    return c / len(X)

# pearson correlation coefficient
def pearson(X, Y):
    return covariance(X,Y) / (covariance(X,X)**0.5 * covariance(Y,Y)**0.5)
```

Correlation of returns

For an example, let's explore the correlation between the returns of two companies: 3M and Union Pacific.

```
REF_COMPANY2 = "UNP"
```

```
def stocks_corr_coeff(h1, h2):

    l1 = []
    l2 = []

    intersec_dates = set(h1.keys()).intersection(set(h2.keys()))

    for d in intersec_dates:
        l1.append(float(h1[d]))
        l2.append(float(h2[d]))

    # correlation with the same company has to be 1!
    return pearson(l1, l2)
```

```
correl = stocks_corr_coeff(
    dict_returns[REF_COMPANY],
    dict_returns[REF_COMPANY2]
    )

print(correl)
```

Build the correlation Network

```
corr_network = nx.Graph()

list_of_comp = [keys for keys in dict_returns]

print(list_of_comp)

num_companies = len(dict_returns.keys())

print(num_companies)

for i1 in range(num_companies - 1):

    for i2 in range(i1 + 1, num_companies):

        stock1 = list_of_comp[i1]

        stock2 = list_of_comp[i2]

        # metric distance
        corr = stocks_corr_coeff(dict_returns[stock1], dict_returns[stock2])

        metric_distance = math.sqrt(2*(1.0 - corr))
        # building the network
        corr_network.add_edge(stock1, stock2, weight = metric_distance)

print("number of nodes:", corr_network.number_of_nodes())
print("number of edges:", corr_network.number_of_edges())

nx.draw(corr_network, with_labels = True)
```

Extract the Minimum Spanning Tree with Prim's algorithm

We arbitrarily root the MST in `MMM`.

```
tree_seed = REF_COMPANY
```

```
N_new = []
```

```
E_new = []

N_new.append(tree_seed)

while len(N_new) < corr_network.number_of_nodes():

    min_weight = 10_000_000.0

    for n in N_new:
        for n_adj in corr_network.neighbors(n):
            if not n_adj in N_new:
                if corr_network[n][n_adj]['weight'] < min_weight:

                    min_weight = corr_network[n][n_adj]['weight']

                    min_weight_edge = (n,n_adj)

                    n_adj_ext = n_adj

    E_new.append(min_weight_edge)
    N_new.append(n_adj_ext)

# generate the tree from the edge list
tree_graph = nx.Graph()

tree_graph.add_edges_from(E_new)

# setting the color attributes for the network nodes
for n in tree_graph.nodes():
    tree_graph.nodes[n]['color'] = dict_colors[dict_sectors[n]]
```

```
#this is a simpler representation if you have issues with graphviz

nx.draw(tree_graph, with_labels = True)
```

Printing the Financial MST with Graphviz

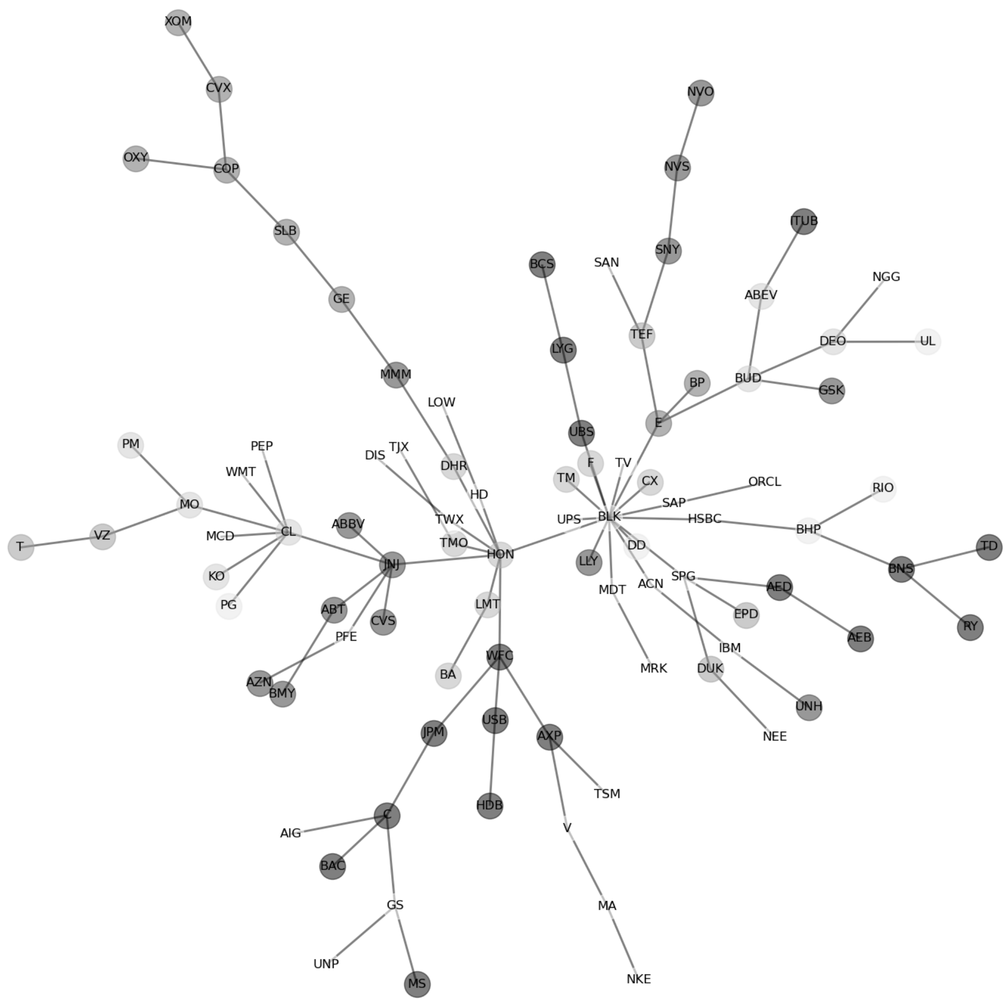
The cell below spans the full force of `Graphviz`; it might require further module installations such as `pydot` and `neato`.

To avoid issues with the *very tricky* `pygraphviz` installation, the static output for the period studied in the textbook, i.e., the years running up to 31 May 2014.

In the MST representation below we notice the emergence of a few hubs which are easy to interpret. For instance, one is around Honewell (`HON`), which is an industry/defense company, another is Wells Fargo (`WFC`) which is where a branch stems. Finally, BlackRock (`BLK`) is a big hub of financial tickers.

The most intresting insights are *across industries:* for example Johnson & Johnson (`JNJ`) is connected both to pharma, e.g., Pfizer (`PFE`), and to retail, e.g., Colgate (`CL`).

```
# Pygraphviz solution:
# pos = nx.nx_agraph.graphviz_layout(tree_graph)

# graphviz solution:
pos = graphviz_layout(
    tree_graph,
    prog = 'dot'
    )


plt.figure(figsize = (20, 20))

nx.draw_networkx_edges(
    tree_graph,
    pos,
    width = 2,
    edge_color = 'black',
    alpha = 0.5,
    style = 'solid'
    )

nx.draw_networkx_labels(tree_graph, pos)

for n in tree_graph.nodes():
    nx.draw_networkx_nodes(
        tree_graph, pos, [n], node_size = 600, alpha = 0.5,
        node_color=tree_graph.nodes[n]['color']
        )

plt.axis('off')

#plt.savefig('MST_50B_new.png', dpi=600)
```


XOM
CVX
OXY
COP
SLB
GE
MMM
LOW
DHR
HD
TJX
DIS
TWX
TMO
HON
PM
PEP
WMT
MO
CL
MCD
T
VZ
KO
PG
ABBV
JNJ
ABT
CVS
PFE
AZN
BMY
LMT
BA
WFC
USB
JPM
AXP
HDB
TSM
C
AIG
BAC
GS
UNP
MS
V
MA
NKE
NVO
NVS
SNY
SAN
BCS
TEF
LYG
ITUB
ABEV
NGG
DEO
UL
BP
BUD
GSK
E
UBS
F
TV
TM
CX
ORCL
SAP
UPS
BLK
HSBC
BHP
RIO
TD
BNS
RY
DD
LLY
MDT
ACN
SPG
AED
EPD
AEB
IBM
MRK
DUK
UNH
NEE