5
Financial Networks

5.1 Introduction

Activity in the stock markets, has always attracted a great deal of interest from not only investors, but also scientists. Both for different (or maybe the same) reasons, have wanted to discover regularity in price fluctuations. The theory of random walks (Bachelier, 1900) and fractals theory (Mandelbrot, 1963) both originated from studies on pricing of commodities. Continuous storage of a variety of data i.e. numbers of transactions, pricing, numbers of bids and asks for all traded stocks worldwide constantly produces one of the largest datasets available to researchers. As mentioned, this discipline has attracted over time the interest of investors convinced that it would in principle be possible to predict the future behaviour by inspecting the past history. It is noteworthy that since the market is not totally isolated, if many believe that the price will go up, then the price will effectively go up (self-fulfilling prophecy). This creates an interesting feedback between observer and system observed. What can be considered to be certain is that we have clear evidence of correlations in the form of self-affinities in price history, with the presence of characteristic roughness exponents. The study of time series is just one of the ways in which we can study quantitatively economic and financial networks. Another approach that is particularly fruitful is to describe the various connections between financial institutions in the form of a network. The structure obtained is particularly complex, since an edge (or various kinds of edges) can represent lending, exposure, insurance, credit default swaps (CDS), ownership, interlock in the board etc. The aim of this chapter is to provide the reader with the main quantitative instruments to describe these systems. Codes, data and/or links for this chapter are available from **http://book.complexnetworks.net**.

5.2 Data from Yahoo! Finance

Financial data are very difficult to collect, essentially due to disclosure problems, but also because of the absence of specific policy regulations on certain kinds of transactions; also most of the data are not available in an aggregated form. Nevertheless, after the financial crisis which started with sub-prime mortgages in 2008, it became clear to a variety of policy regulators and control organisations, that the complexity of the financial structure and our poor knowledge of it had been one of the causes of the turndown in the economy. From that moment a series of international organisations and companies started collecting and making available various data, unfortunately not always accessible to scientists. The set of data we present here has been downloaded from the Yahoo! Finance web service, which offers daily historical data for the closure

Data Science and Complex Networks. First Edition. Guido Caldarelli and Alessandro Chessa.

prices of stock traded in various markets. In the following we present how to interact with the service in order to get the relevant data we need to explore the correlations between stocks for companies present in the NYSE (New York Stock Exchange) index.

Connecting with the Yahoo! Finance service

```
import yahoo_finance as yf

yahoo = yf.Share('YHOO')
d=yahoo.get_historical('2014-05-19', '2014-05-20')
print "A week of stock daily quotations:"
for e in d:
    print e
print "Info about the company:",yahoo.get_info()
print "Market capitalization in dollars:",yahoo.get_market_cap()

OUTPUT
A week of stock daily quotations:
{'Volume': '18596700', 'Symbol': 'YHOO', 'Adj_Close': '33.869999',
'High': '34.470001', 'Low': '33.669998', 'Date': '2014-05-20',
'Close': '33.869999', 'Open': '33.990002'}
{'Volume': '14845700', 'Symbol': 'YHOO', 'Adj_Close': '33.889999',
'High': '33.990002', 'Low': '33.279999', 'Date': '2014-05-19',
'Close': '33.889999', 'Open': '33.41'}

Info about the company: {'start': '1996-04-12', 'symbol': 'YHOO',
'end': '2015-06-17', 'CompanyName': None}

Market capitalization in dollars: 38.13B
```

The historical data from Yahoo! Finance presents information about the volume of stocks transacted, the highest, the lowest, the opening, and the closing values, as well as an adjusted closing value that provides the closing price (on the requested day, week, or month for any stock) adjusted for all applicable splits and dividend distributions. Starting from this data we can easily compute the total transaction volume for the day as the product of the number of shares exchanged and the adjusted closing value.

Transaction volumes computation and plotting (see Fig. 5.1)

```
d=yahoo.get_historical('2014-01-01', '2014-12-31')
V = []
for s in d:
    print s['Date'],float(s['Volume'])*float(s['Adj_Close'])
```

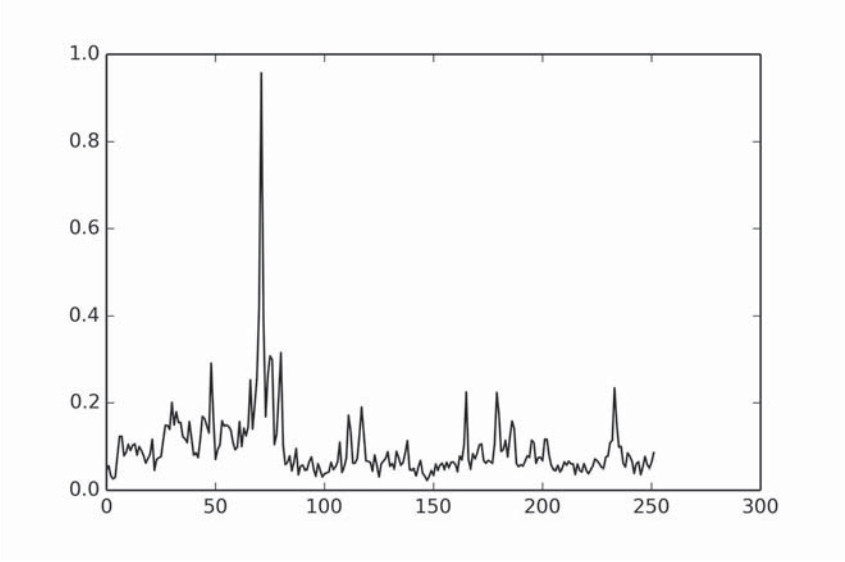

Fig. 5.1 One year of stock quotations for the Yahoo share from January 01, 2014, up to December 31, 2014. The values are in $10^{10}$$ (US Dollars).

```
    V.append(float(s['Volume'])*float(s['Adj_Close']))

plot(V)
savefig('yahoo_volume.png')

#OUTPUT
2014-12-31 469995531.39
2014-12-30 548233280.704
2014-12-29 334735978.375
2014-12-26 262930947.17
2014-12-24 301970246.924
2014-12-23 776010280.0
2014-12-22 1228679313.04
2014-12-19 1226727000.11
```

We could also get all information related to shares present in the NYSE index querying the Yahoo! Finance service, but here we will follow a mixed and hopefully simpler approach. We will retrieve the sector and industry from a web page where it is possible to download a CSV file with all of this information (http://www.nasdaq.com/screening/company-list.aspx), while the actual market capitalisation will be obtained from the Yahoo service. Only companies with a capitalisation greater than 50 billion dollars will be considered in our analysis.

Get stock labels, sector, and industries

```
#this code will take approximative 1 hour to retrieve the data
#depending on the internet connection
#if you want to skip this procedure just uncomment
#the following lines
#import sys
#f=open("./data/list_stocks_50B_6_may_2016.txt",'r')
#list_stocks=[]
#while True:
#    next_line=f.readline()
#    if not next_line: break
#    list_stocks.append(tuple(next_line.split('\t')[:-1]))
#f.close()
#sys.exit()

import time

hfile=open("./data/companylist.csv",'r')
#we choose to get only companies with a market capitalisation
#greater than 50B$
cap_threshold=50.0

list_stocks=[]
nextline=hfile.readline()
while True:
    nextline=hfile.readline()
    if not nextline:
        break
    line=nextline.split(',')
    sym=line[0][1:-1]
    share = yf.Share(sym)
    y_market_cap=share.get_market_cap()
    if not y_market_cap: continue
    #we will exclude stocks with char ^ that will
    #give errors in the query process
    if y_market_cap[-1]=='B' and float(y_market_cap \
        [:-1])>cap_threshold and line[0].find('^')==-1:
        print sym,y_market_cap
        list_stocks.append((line[0][1:-1],line[1][1:-1],\
                            line[5][1:-1],line[6][1:-1]))
    time.sleep(1)
```

```
hfile.close()

print list_stocks[0]

OUTPUT
MMM 99.27B
ABB 50.54B
ABT 72.17B
ABBV 106.37B
ACN 60.56B
AEB 50.61B
..........

('MMM', '3M Company', 'Health Care', 'Medical/Dental Instruments')
```

When we need to plot using specific colour codes for companies in the plot, and we need specific dictionaries to handle companies, colours, and sectors.

Generate dictionaries for companies, sectors, and colours

```
diz_sectors={}
for s in list_stocks:
    diz_sectors[s[0]]=s[2]

list_ranking=[]
for s in set(diz_sectors.values()):
    list_ranking.append((diz_sectors.values().count(s),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

diz_colors={}

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

```
        diz_colors[s[1]]='white'
        continue
    diz_colors[s[1]]=list_colors.pop(0)
```

5.3 Prices time series

The time series of a stock price is a typical quantity that investors (right or wrong) use when considering their investments (we do not comment here whether they are right or not in doing so). This field of finance is particularly awkward to study since phenomena observed are heavily affected by our actions. As already mentioned, if all investors of a stock suddenly believe that a signal in a stock price time series is indicating impending bankruptcy, all of them will sell the stock causing the bankruptcy for real (a typical case of a self-fulfilling prophecy).

Retrieving historical data

```
start_period='2013-05-01'
end_period='2014-05-31'
diz_comp={}
for s in list_stocks:
    print s[0]
    stock = yf.Share(s[0])
    diz_comp[s[0]]=stock.get_historical(start_period, end_period)

#create dictionaries of time series for each company
diz_historical={}
for k in diz_comp.keys():
    if diz_comp[k]==[]: continue
    diz_historical[k]={}
    for e in diz_comp[k]:
        diz_historical[k][e['Date']]=e['Close']

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

In the (strong) hypothesis that, in calm periods, various psychological effects cancel out, investors study the statistical properties of the time series, trying to spot regularities that could anticipate the future behaviour of the price. While the link between past and future performance has never been demonstrated, there is nevertheless a certain consensus that "on average" this information is valuable to the investors. In particular the return and the volatility are considered the most important indicators. Given a time interval Δt, let us consider an asset price at the beginning $p(t_0)$ and at the end $p(t_0 + \Delta t)$. We define the proportional return of the investment in the period

Δt as

$$r(\Delta t) = \frac{p(t_0 + \Delta t) - p(t_0)}{p(t_0)}. \tag{5.1}$$

Here we assumed investment in only a certain number of one type of stock, so that we can use the price to determine costs and gains. The above equation in the limit $(\Delta t \to 0)$ can be written as $r(t) \simeq \frac{d \ln (p(t))}{dt}$.

This expression passing to discrete time steps takes the following form:

$$r = \ln p(t_0 + \Delta t) - \ln p(t_0). \tag{5.2}$$

Return of prices

```
reference_company='ABEV'
diz_returns={}
d=diz_historical[reference_company].keys()
d.sort()
print len(d),d

for c in diz_historical.keys():
    #check if the company has the whole set of dates
    if len(diz_historical[c].keys())<len(d): continue
    diz_returns[c]={}
    for i in range(1,len(d)):
        #price returns
        diz_returns[c][d[i]]=math.log( \
        float(diz_historical[c][d[i]])) \
        -math.log(float(diz_historical[c][d[i-1]]))

print diz_returns[reference_company]
```

Among the various definitions of volatility σ, the simplest is the standard deviation of the value of prices $p(t)$.

Basic statistics and the correlation coefficient

```
#mean
def mean(X):
    m=0.0
    for i in X:
        m=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)
```

5.4 Correlation of prices

In the same spirit, correlations in time series (or more simply comovements) are also considered to be extremely valuable. The idea is that every investor has precise knowledge of the market (highly unrealistic (Greenwald, Bruce and Stiglitz, 1993)) and since (s)he is perfectly rational (another strong assumption), (s)he wants to maximise the return and at the same time minimise the risk of their investments. This is obtained by choosing the proportion of the investments among all the assets present in the market (considered "complete") and by essentially building a portfolio of all the different assets. All these concepts have been formalised in the "Theory of portfolio" (Markowitz, 1952) and constitute the basis of operation for professional investors.

Coming back to the real market, if two or more assets have a past history of common behaviour (i.e. they both go up or down at the same time) we can measure a correlation between their price evolution as given by these "comovements". Of course there is no proof that the presence of such a correlation in the past is also a good proxy of its presence in the future. On the other hand, it may very well be that two assets belonging to the same industrial sector have similar behaviour. For example when few firms producing high technology objects (such as computers) go up in the market, it is true that all the assets linked to that technology go up as well. Another example could be when the first asset owns a part of the second, so that a movement (up or down) of one causes the same movement in the second. In these exempla if the correlation in the price is caused by a "hidden link" between the assets, it is fair to assume that we shall also have price correlation in the future. This is important to know when we build our portfolio. Having many assets all behaving the same is like putting all our eggs in a single basket, thereby reducing the risk protection. In a market of say 1000 assets, correlations are of the order of millions (since for N assets, the independent correlations we need to check are of the order of N^2) a number typically too large to allow eye-inspection analysis. Therefore filtering of information is necessary before proceeding to any choice of investment.

The crucial variable is the daily closure price $r_i(t)$ of company i on day t. From that, one can consider all the possible pairs of companies and compute the correlation

between the respective price returns. Two price stocks are correlated if they vary in a similar way. In other words companies i and j are correlated when the price of stock i increases if the price of stock j also increases. To quantify such a relation, we compute the correlation $\rho_{ij}(\Delta t)$ between the price returns over a time Δt. Correlation is computed by means of

$$\rho_{ij}(\Delta t) = \frac{\langle r_i r_j \rangle - \langle r_i \rangle \langle r_j \rangle}{\sqrt{(\langle r_i^2 \rangle - \langle r_i \rangle^2)(\langle r_j^2 \rangle - \langle r_j \rangle^2)}}. \tag{5.3}$$

By definition, $\rho_{ij}(\Delta t)$ can vary from -1 (when stocks i and j are completely anti-correlated) to 1 (when stocks i and j are completely correlated). In between there is another important situation: when $\rho_{ij}(\Delta t) = 0$ the two stocks i and j are uncorrelated. Given its meaning, the matrix of the correlation coefficient is symmetric with a value of $\rho_{ij}(\Delta t) = 1$ along the main diagonal (autocorrelation).

Correlation of price returns

```
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]))
    return pearson(l1,l2)

#correlation with the same company has to be 1!
print stocks_corr_coeff(diz_returns[reference_company], \
                        diz_returns[reference_company])

OUTPUT
1.0
```

5.5 Minimal spanning trees

We have already seen that both Traceroute paths and food webs can be represented in the form of trees. Trees are economical graphs in the sense that they connect a fixed number of vertices through the minimal number of edges. One can further reduce the number of links in a tree, by dividing it into two parts and creating more, smaller sub-clusters. A set of disjoint (sub-)trees is called (intuitively) a forest.

Given this "economical" feature, it is hardly a surprise that they are very frequently used in complex systems. For similar reasons, trees are also often used to investigate network structure, as in the case of the breadth first search algorithms and/or as in this case, to filter the information present in a complete graph. More generally, trees are perfect for classifying information. In the case of botany or zoology, this is very

easy and is the basis of taxonomic trees. We start from species and we cluster them according to their morphology. Classes of species can be clustered in the same way. Step by step we form a tree composed of different layers.

Using the correlation values previously defined we obtain a set of $n \times (n-1)/2$ numbers characterising the similarity of any of the n stocks with respect to all the other $n-1$ stocks. This set of numbers forms a complete graph with different edge strengths given by the correlation value. At this point we use trees to filter the information reducing the density of the graph. To every entry of the above-defined correlation matrix we can associate a metric distance between any pair of stocks by defining

$$d_{i,j}(\Delta t) = \sqrt{2(1 - \rho_{ij}(\Delta t))}. \tag{5.4}$$

With this choice, $d_{i,j}(\Delta t)$ fulfils the three axioms of a metric distance:

- $d_{i,j}(\Delta t) = 0$ if and only if $i = j$;
- $d_{i,j}(\Delta t) = d_{j,i}(\Delta t) \forall i, j$;
- $d_{i,j}(\Delta t) \leq d_{i,k}(\Delta t) + d_{k,j}(\Delta t) \forall i, j, k$.

The distance matrix $D(\Delta t)$ is then used to determine the MST connecting the n stocks (Gower, 1966; Mantegna, 1999).

Building the network with the metric distance

```
import math
import networkx as nx

corr_network=nx.Graph()

num_companies=len(diz_returns.keys())
for i1 in range(num_companies-1):
    for i2 in range(i1+1,num_companies):
        stock1=diz_returns.keys()[i1]
        stock2=diz_returns.keys()[i2]
        #metric distance
        metric_distance=math.sqrt(2*(1.0-stocks_corr_coeff\
                (diz_returns[stock1],diz_returns[stock2])))
        #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()
```

The method for constructing the MST linking N objects is known in multivariate analysis as the "nearest neighbour single linkage cluster algorithm" (Mardia *et al.*, 1979). The idea is to consider the above-defined distance (5.4) between two vertices as the weight of the link connecting them. At this point we keep only the strongest

correlations or the shortest distances. To filter among the $\simeq n^2$ links we first rank all the edges, then we start from the vertices which are nearest and we keep adding new vertices by following the rank of the edges, discarding all the links that would form a cycle (in this way, by construction, the graph is acyclic, i.e. a tree). Finally, we stop when all the vertices are drawn (in this way the tree is spanning). Schematising:

1. rank a couple of vertices (stocks) from the nearest to the farthest
2. draw the first edge from this rank
3. continue in the rank
4. if the new edge does not close a cycle draw it
5. go to point 3
6. stop when all the vertices have been drawn.

Minimal spanning tree (Prim's algorithm)

```
tree_seed=reference_company
N_new=[]
E_new=[]
N_new.append(tree_seed)
while len(N_new)<corr_network.number_of_nodes():
    min_weight=10000000.0
    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.node[n]['color']=diz_colors[diz_sectors[n]]
```

Printing the financial minimum spanning tree (see Fig. 5.2)

```
pos=nx.graphviz_layout(tree_graph,prog='neato', \
```

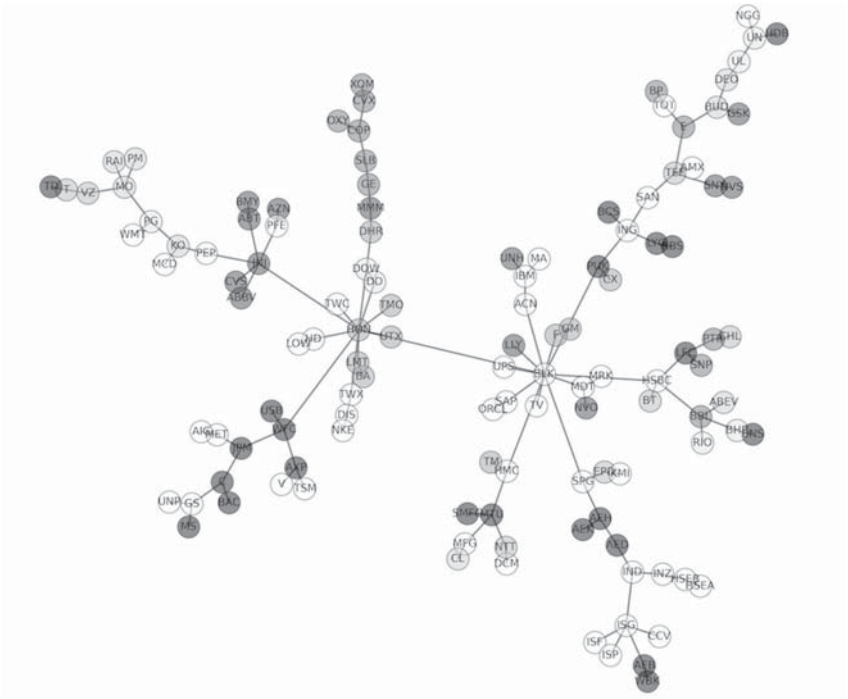

Fig. 5.2 Minimum spanning tree of 141 highly capitalised stocks traded in the US equity markets (NYSE). The filtering procedure has been obtained by considering the correlation coefficient of stock returns time series computed at a one trading day time horizon (6 h and 30 min). Each circle represents a stock labelled by its tick symbol. The minimum spanning tree presents a large amount of stocks having a single link and some stocks having several links. Some of these stocks act as the "hub" of a local cluster.

```
                        args='-Gmodel=subset -Gratio=fill')

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.node[n]['color'], \
    with_labels=True)

axis('off')

savefig('./data/MST_50B_new.png',dpi=600)
```