中国股票市场的网络演化建模与投资

摘要

股票市场是一个具有强相互作用的高度复杂系统，我们可以利用**复杂网络理论**研究股票市场的波动，并据此预测股票涨落的内在机理、推断出最优投资组合。在问题一中，我们基于股票的相关性建立了股票网络模型，并借助模型具体研究了网络的小世界特征和无标度特征。在问题二中，我们首先确定了新上市股票数量随时间的关系，并从（起点年份年）开始，每一年按顺序加入顶点进行演化，并与实际情况进行对比。在问题三和四中，我们基于 Markowitz 投资组合理论，依据之前的模型确定了股票的（方差），用股票收益率的均值和方差来代表股票的收益和风险。建立了关于股票投资的数学模型。

在问题一中，通过分析股票间的相关系数并与给定的阈值进行比较，我们可以确定网络的结构，并能计算得到网络的平均路径长度等参数。在网络特性确定方面，关于网络是否具有小世界特性，我们可以求解不同网络规模（节点数）下网络的平均路径长度，通过判断二者是否符合对数关系来确定。而关于网络的无标度特性，考虑到幂次分布具有长尾性且实际数据可能存在一定的随机因素，采取了一种基于 K-S 检验和最大似然估计的**非参数检验**方法来判断网络的无标度特性。最终证明了股票网络具有无标度特性，并且在网络相连的局部具有小世界特性。

在问题二中，我们选取了从 2010 年开始的数据，并整理得到之后各个年份内新上市股票数量随时间的关系。我们可以使用**综合演化模型**来模拟网络的变化情况。选取 1990 年作为演化起点，每一年演化一次，演化时顺序加入当年新增的股票作为顶点，并且依据择优连接方式来分配新节点连接边的概率。在演化结束时与实际股价对比来评价演化模型。通过演化得到的模型的聚类系数为 0.003，平均路径长度为 2.912，模型能够较好的反应新上市股票对股票网络的影响情况。

在问题三中，我们对各支股票进行分类，从而得到股票的选取方式。同时给出了各种投资组合对应的价值和方差。根据 Markowitz **投资组合理论**，建立了改进的投资组合模型，确定了在获得一定收益率的前提下风险最小的投资方案。

在问题四中，我们基于问题三的部分结论以及股票投资的实际情况给出了包括合理选择股票分类，购买平均成交量较大的股票等控制投资风险的合理化建议。模型在投资实际中有一定的实用价值。

关键字：复杂网络模型　非参数检验　投资组合理论　综合演化模型

一、 问题背景与问题重述

股票市场可视为一个具有强相互作用的高度复杂系统，我们可以利用复杂网络理论研究股票市场的波动，并据此预测股票涨落的内在机理、推断出最优投资组合。我们可以将每支股票视为一个节点，根据股票之间的某种相关性来建立股票网络，从而实现利用复杂网络理论建设和分析股票网络。股票之间相关性指标的选择会影响股票网络建立的准确度，进而影响网络的拓扑特性分析。

在题目附录中给出了中国股票市场截止 2021 年 8 月 10 日的后复权数据，包括最为重要的开盘价、最高价、最低价、收盘价，以及体现当天该支股票的活跃程度的成交量与成交额数据。在问题 1 中，首先在合理的股票数据清洗的基础上，研究各支股票之间的相关性建立合适的股票网络模型，进而分析股票网络的基本拓扑特征，验证其是否具有明显的小世界特性和无标度特性。

股票上市代表者股票网络中新节点的加入，但它与哪些股票具有紧密联系，还需要根据股票之间的相关性来确定。当获得股票网络的演化数据后，可建立股票网络节点度或其他拓扑特征的演化模型，并分析网络是否具有线性增长或加速增长的特性。在问题 2 中，我们需要根据每支股票的上市日期，结合每支新加入的股票与其他股票的相关性，建立股票网络演化模型，求解此演化模型并检验与现实股票网络演化的吻合度。判断网络是否具有加速增长的趋势。如果有，修改你们团队建立的演化模型并获得加速演化的最优参数。

人们投资股票总是希望获得投资收益，但投资风险与收益是并存的。最佳的投资方式是进行多个资产的组合投资，在降低投资风险水平的条件下获得最佳的投资收益。选择合适的拓扑特征来代表股票的风险与收益，是建立股票网络投资组合模型的关键。结合股票网络的特征与投资组合思想，可设计出实用有效的个人投资风险控制策略。在问题 3 中，我们需要基于 Markowitz 投资组合理论，选择合适的网络拓扑特征或参数，建立基于股票网络的投资组合模型，选择几支股票数据进行最优投资。比较新投资组合模型与 Markowitz 投资组合模型中最优投资组合的现实含义。最后在问题四中，基于股票网络建模与投资组合理论，建立数学模型给出个人投资风险控制的综合建议。

二、 问题分析

在问题 1 中，我们参考了 [1] 的部分相关方法，用股票前后两天价格之比来代表股票价格的增长率。通过分析股票间的相关系数并与给定的阈值进行比较，我们可以确定网络的结构，并能计算得到网络的平均路径长度等参数。在网络特性确定方面，网络是否具有小世界特性我们可以求解不同网络规模下网络的平均路径长度，通过判断二者是

否符合对数关系来确定。而关于网络的无标度特性，考虑到幂次分布具有长尾性且实际数据可能存在一定的随机因素，不适合用最小二乘法判断分布情况。我们依据论文 [2]，采取一种基于 K-S 检验和最大似然估计的非参数检验方法来判断网络的无标度特性。

在问题 2 中，我们选取了从 2010 年开始的数据，并整理了各年新上市股票数量随时间的关系，之后选取 2010 年作为演化起点，每一年演化一次。考虑到实际股票市场中往往存在同行业股票间有较强相关性的情况，在演化过程中采用**择优连接**方式来分配新节点连接边的概率，建立股票网络演化模型。通过求解网络演化模型并且与现实中的股票情况对比从而得到模型与现实股票网络演化的吻合度。

在问题三中，我们首先通过股票名称对各支股票进行分类，从而得到股票的选取方式。之后我们可以基于 Markowitz 投资组合理论，用用股票收益率的均值和方差来代表股票的收益和风险。最终选择在达到一定的投资收益的前提下使其总投资风险最小的投资方案。之后在问题四中，我们基于问题三的复合模型的求解结果，针对降低风险给出了合理化建议。

三、 模型假设及符号说明

3.1 模型假设

- 忽略数据之外的股票的退市对于市场的影响
- 忽略非市场因素（例如国家政策）对于股票价格的影响
- 认为各支上市股票对市场的潜在影响能力相同

3.2 符号说明

符号	含义
$p_i(t)$	t 时间的股票 i 的收盘价格
$r_i(t)$	t 时间的股票 i 的对数收益率
R_{ij}	股票 i 和 j 的相关系数
n	投资证券总数
$V_{(p,t)}$	投资组合 p 在 t 时刻的价值
$N_{(j,p)}$	投资组合 p 中 j 的持有量
$x_{(j,p)}$	证券 j 在投资组合 p 中的占比

四、 模型准备

4.1 数据清洗

数据包含目前上证 A 股和深证 A 股在去除所有股票在 2021 年 8 月 11 日之前未退市的所有股票在不同日期的开盘价、最高价、最低价、收盘价、成交量与成交额数据。考虑到数据中没有给出在 8 月 11 日的股票的成交量数据，我们排除了各支股票的在 8 月 11 日的收益情况。另外，由于股票在上市初期由于没有涨跌停机制，市场预期不确定等原因，往往有比较明显的价格波动，考虑这些股票可能会对我们的网络构造产生一定影响。因此，我们排除了最近 1 年新上市的股票。

由于我们在关于相关系数的计算中使用了股票的对数收益率作为参考指标。考虑到对数运算的性质，我们排除了收盘价含有 0 和负数的数据。

4.2 复杂网络简介

在数学上，任何网络都可以看成是由网络中的节点按某种方式连接组成的一个系统, 通常我们可以把网络中的节点看成是图的顶点，它是网络最基本的功能单元（在我们的例子中代表各支股票）；节点之间的连接看成是点与点之间的连边，它代表了网络节点间相互关系（这里代表股票的相互关系）。网络的问题可以这样抽象为图的问题。

常见的复杂网络可分为三类, 分别是随机图模型, 小世界网络, 无标度网络。

在随机图模型中，给定网络节点总数 N 和连线总数 V , 从所有可能的连线 $\frac{N(N-1)}{2}$ 中随机选取, 则生成的网络集合记为 $G(N,V)$, 由此构成一个概率空间。ER 随机图的节点度分布服从二项分布, 可以用泊松分布逼近。

在小世界网络中，构造过程由随机化重连概率 p 控制，$p=0$ 对应着完全规则网络，$p=1$ 对应着完全随机网络, $0<p<1$ 即为小世界网络存在的范围。网络中节点的平均最短路长度 150 非常小，通常与节点数的对数同阶，即平均最短路长度较小，而聚类系数比随机图模型要大的多。

在无标度网络中，网络具有择优连接性。当选择某个节点 i 连接一条新边时，被选中的概率与该点现有的度 k_i 有关。在这种模型中，节点的度与节点的秩之间、hop 数小于或等于某个正数的节点的个数与这个正数之间、连接矩阵的特征值与其序号之间等都满足幂律分布 [2]。

根据在股票市场上的实际情况，股票的相关关系往往并不是完全随机的，许多相关领域的股票价格间往往有着较强的相关性，这说明随机图模型不能用来表示股票间的相关情况，而现实中的股票模型可能具有一定的小世界特性和无标度特性。

五、 模型一

5.1 模型建立

5.1.1 网络模型建立

我们可以使用 $p_i(t)$ 来表示 t 时间的股票 i 的收盘价格，t 时间的股票 i 的对数收益率 $r_i(t)$ 表示为:

$$r_i(t) = \ln(\frac{p_i(t)}{p_i(t-1)}) \tag{1}$$

我们使用 Pearson 相关系数来表示股票 i 和 j 的相关关系，相关系数可以表示为:

$$R_{ij} = \frac{E(R_iR_j) - E(R_i)E(R_j)}{\sqrt{Var(R_i)Var(R_j)}} \tag{2}$$

我们可以依据股票间的相关系数来建立股票网络模型：对于任意节点对 i 和 j，若股票 i 和 j 的收益率相关系数 R_{ij} 大于或者等于所指定的阈值 θ , 就认为节点对 i 和 j 有边相连。我们可以选取不同的相关度阈值来反映不同情况下网络的连接情况。

5.1.2 小世界特性

根据文献 [4]，小世界网络的主要特性是网络的平均最短路长度与节点数的对数同阶，且网络的聚类系数 C 较大。我们可以从论证网络的聚类系数较大和网络的平均最短路长度符合对数关系来证明网络的小世界特性。

关于网络的聚类系数，在网络中若某节点 i 存在与其相连的 k_i 条边, 那么这些连边相应的节点定义为该节点的邻居节点。显然, 在这 k_i 个节点之间最多只可能有 $\frac{ki(ki-1)}{2}$ 条边, 而这 k_i 个节点之间实际存在的边数 E_i 与总的可能的边数的比值定义为节点 i 的聚类系数 C_i 。

$$C_i = \frac{2E_i}{k_i(k_i-1)} \tag{3}$$

我们可以通过对所有点的聚类系数取平均值来得到整个网络的聚类系数 C 来描述网络的聚类情况。

根据前面的分析，复杂网络的分析在数学上可以看作一个图论问题。网络中的最短路径即为在可以连通节点 i,j 的所有通路中，途径别的顶点数目最少的若干条路径。平均路径长度 L 就定义为给定的网络中任取两个节点之间的最短路径长度的平均，用 d_{ij} 表示为:

$$L = \frac{2\sum d_{ij}}{N(N-1)} \tag{4}$$

在求解得到各顶点间的最短路径长度后，我们就可以得到网络的平均路径长度 L。

5.1.3 无标度特性

无标度网络需要满足节点的度近似服从幂律分布。根据论文 [2]，在数学上如果一个随机变量 X 的的密度函数满足下面的幂次关系，则它服从幂律分布。

$$p(x) \propto x^{-\alpha} \quad (\text{通常 } 2 < \alpha < 3) \tag{5}$$

上面的密度函数中的 α 是分布的参数，通常满足 $2 < \alpha < 3$。

可以推到得到幂律分布在离散情况下的分布函数为：

$$F(x) = 1 - \frac{\xi(\alpha, x)}{\xi(\alpha, x_{min})} = 1 - \frac{\sum_{n=0}^{\infty}(n+x)^{-\alpha}}{\sum_{n=0}^{\infty}(n+x_{min})^{-\alpha}} \tag{6}$$

根据论文 [2]，我们可以通过极大似然估计方法，借助 KS 统计量估计 X_{min} 和 α 的值，并计算数据与幂律分布之间的拟合优度，从而判断节点的度的分布是否符合幂律分布。

Kolmogorov-Smirnov 检验（K-S 检验），是一种基于假设检验的，用来比较一个频率分布 f(x) 与理论分布 g(x) 或者两个观测值分布的是一种非参数检验方法。其原假设 H0: 两个数据分布一致或者数据符合理论分布。K-S 检验的统计量可以表示为：

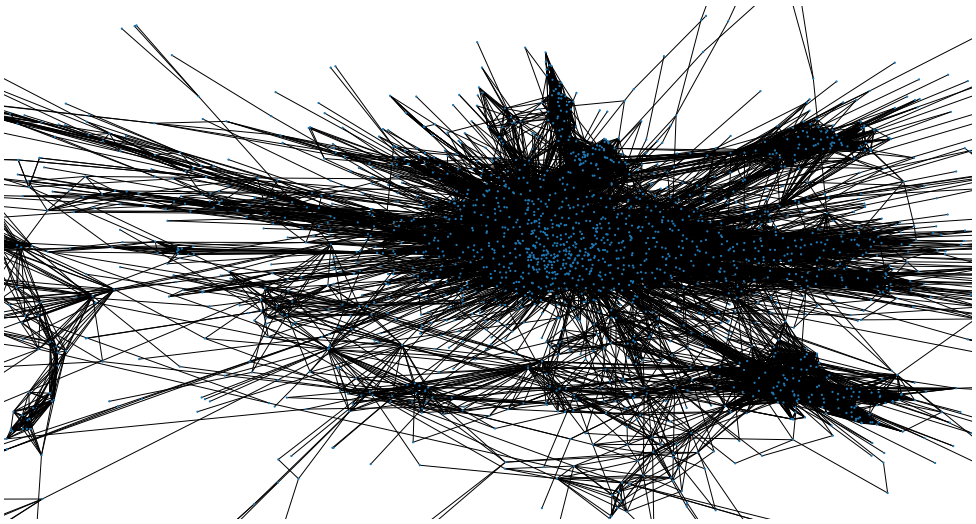
$$D = \max |f(x) - g(x)|$$

当实际观测值 D 大于参考值 $D(n, \alpha)$ 则拒绝 H_0，否则则接受 H_0 假设。

5.2 模型求解

5.2.1 网络模型建立

我们首先按照模型准备部分的的方法对数据进行初步的预处理，排除了部分近期上市的股票以及部分收盘价格为 0 或负数的股票。根据股票的对数收益率的变化趋势，并依据公式2求解得到各支股票间的 pearson 相关系数。之后我们选取了不同的相关度阈值，在相关度阈值为 0.5 的情况下，得到的网络局部和全局的连接情况如下：



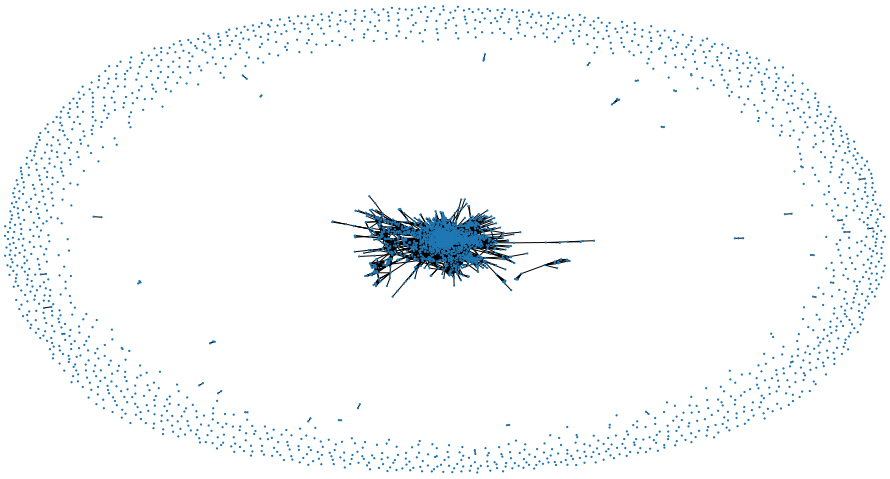


图 1　网络的局部与全局连接情况

观察上图可以发现，网络中的大部分顶点没有与其它顶点相连或者只与少量的其它顶点相连，而数量较少的部分顶点之间有密切的连接情况。

5.2.2 小世界特性

根据前面对于小世界网络的分析，我们需要求解网络的平均最短路径长度和网络的聚类系数来证明网络的小世界特性。根据上面的连接关系，我们发现网络中的大部分顶点没有和其它的顶点相连，如果认为这些节点与未相连的节点的路径长度为无穷大，由于网络的节点数量有限，此时的网络整体显然不满足小世界特性。因此，在以下的分析中我们忽略度为 0 的节点，以探究网络中的相连部分的相关特性。

首先，我们可以依据公式3来计算网络模型的聚类系数，得到的聚类系数为 0.002726。这个聚类系数相对随机图模型的聚类系数 (接近于 0) 较大。说明网络可能具有一定的小世界特性。

在求解最短路径的过程中，我们采取了弗洛伊德算法，这一算法可以求解网络中各个顶点间的距离。算法的基本流程如下：

Algorithm 1: 求解多原点最短路径的弗洛伊德算法

Data: A: 图的邻接矩阵

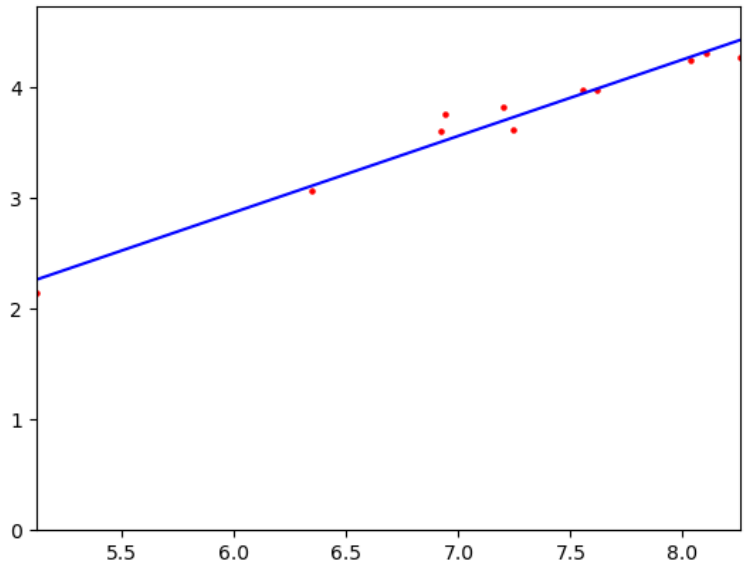
Result: 包含图中各原点间的最短路径的矩阵

```
根据 A 进行初始化;
for k=1 to n do
    for i=1 to n do
        for j=1 to n do
            if D[i,j]>D[i,k]+D[k,j] then
                D[i,j]=D[i,k]+D[k,j];
            end
        end
    end
end
```

在得到网络各个顶点间的路径长度后，我们可以计算得到网络的平均最短路径长度。由于我们需要在不同的节点数的情况下确定平均最短路径长度的变化趋势，这要求我们在不同网络规模下求解路径长度，得到在网络节点数 N 不同的情况下网络中的平均最短路径长度 L 如下表：

$\ln N$	5.118	6.346	6.920	6.938	7.197	7.244	7.555	7.619	8.035	8.107	8.256
L	2.143	3.065	3.605	3.765	3.823	3.623	3.977	3.986	4.251	4.311	4.274

为了判断网络的平均最短路长度与网络节点的对数是否符合线性关系，我们可以使用线性模型来拟合平均最短路长度与网络节点的对数间的关系（如下图）：



使用线性模型拟合上述关系得到的 R^2 为 0.965，这说明线性模型可以较好的反映网络的平均最短路长度与节点数的对数的关系，可以认为二者同阶。忽略没有与其他顶点相连的顶点后形成的网络具有小世界特性。

5.2.3 无标度特性

根据之前的分析，在描述股票相关性的网络中，各个顶点的度的频率分布和概率密度估计值如下图所示：

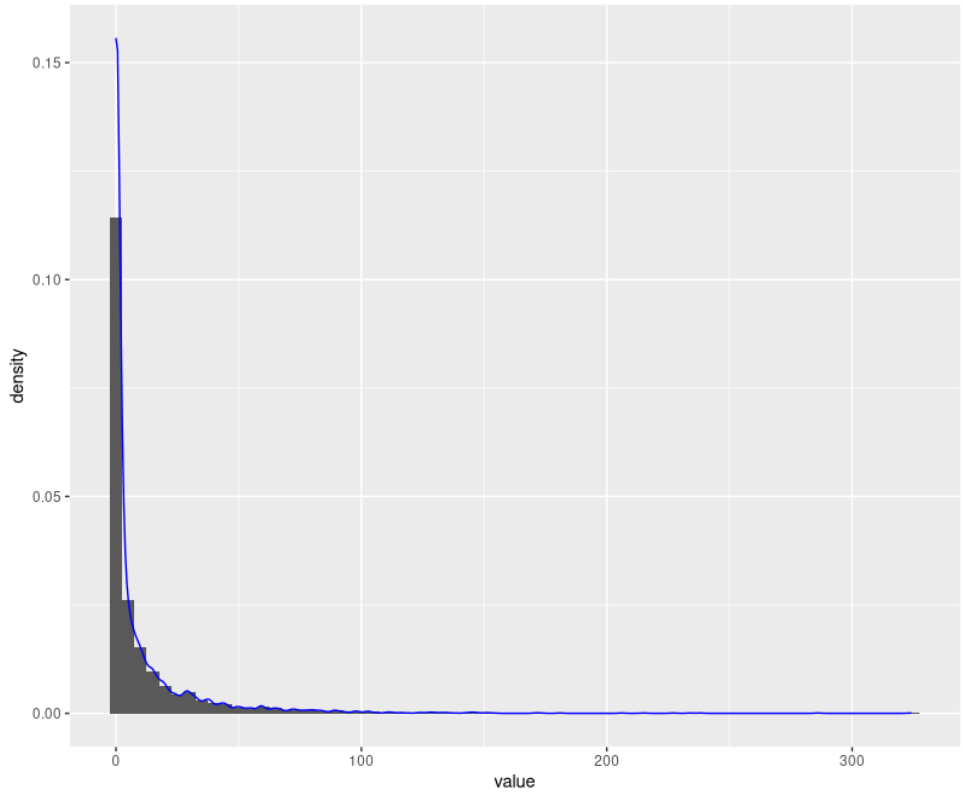


图 2　顶点的度的频率分布直方图

考虑到模型包括较长的概率密度较低的长尾部分，由于长尾部分的波动，直接使用最小二乘法估计参数的范围可能不够准确。根据论文 [2] 我们可以基于 KS 统计量与最大似然比采用 bootstrap 方法进行多次随机来避免上述波动和 X_{min} 的改变对模型的影响。算法的基本思想如下：

Algorithm 2: Bootstrap 方法

Data: ntail：各次估计大于 xmin 的样本数 gof：各次估计对应的 KS 统计量

Result: 用于判断是否能拒绝“服从幂律分布”这一假说的拟合优度 p

通过极大似然估计和 KS 统计量确定 Xmin 和 alp 两个参数;
for k=1 to n **do**
    从二项分布 B(n,ntail/n) 中抽取一个样 n1;
    从小于 xmin 的数中抽取一个样;
    从指数为 alp 的的幂律分布取 n1 个样;
    计算 KS 统计量;
    if ks>gof **then**
        P=P+1
    end
end
p=P/n

令初始的 $x_{min}=1$。为了更精确的判断数据服从的分布，使用 R 语言按照上述方法编程求解得到的拟合优度 p 值为 0.492。由于 $p>0.05$，没有充分的理由证明数据不服从幂律分布。此时对应的参数 α 的均值为 3.721，x_{min} 的均值为 70。

综上，模型具有无标度特性，并且在局部具有小世界特性。

六、 问题二

6.1 模型准备

经过问题一分析，我们发现网络具有局部上的小世界特性和总体上的无标度特性。我们结合了小世界网络和无标度网络的演化特性，建立了混合演化模型，用来模拟股票网络的演化过程。

参考 [5]，小世界网络的演化算法可以表示如下：

1. 网络的构造从规则图开始：给定一个含有 N 个点的环状最近邻耦合网络，其中的每个节点与左右相邻的 $K/2$ 个节点相连
2. 随机化重连：以概率 p 随机重连网络已有的每条边，每条边的一个节点不变，另一条边在没有重复边和自环的前提下随机选择节点进行连接。

具体演化过程如下图所示：

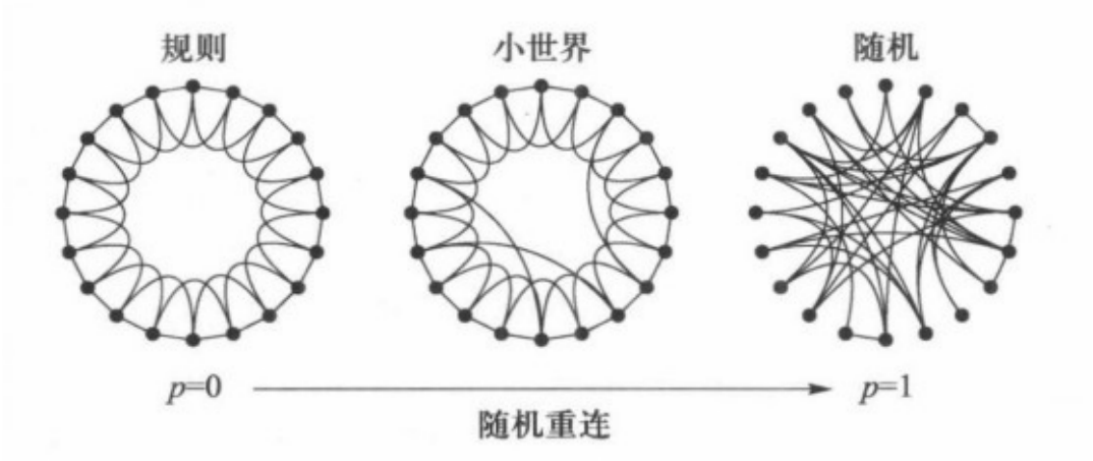


图 3　小世界网络的演化过程（参考自文献 [5]）

无标度网络的演化模型如下：

1. 增长：从一个具有 m_0 个节点的联通网络开始，每次引入一个新节点并且连接到 $m(m \leqslant m_0)$ 个已经存在的节点上。
2. 优先连接性：新节点与已经存在的节点相连的概率与节点的度相关。具体满足

$$\Pi_i = \frac{k_i}{\sum_j k_j} \tag{7}$$

具体演化过程如下图所示：

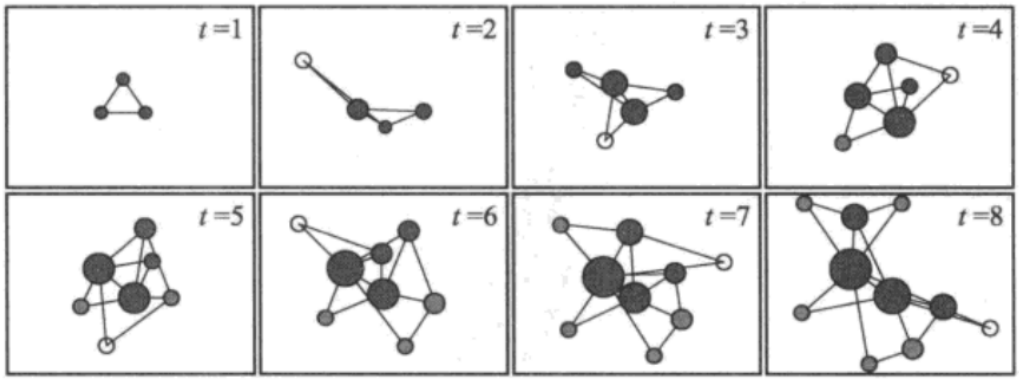


图 4　无标度网络的演化过程（参考自文献 [5]）

由上面的算法构造可知，小世界网络模型的演化不会增加节点个数，也不会增加边的个数，而无标度网络模型的演化不仅增加了节点的个数，而且增加了边的个数。根据问题一的分析，股票的节点度分布得到其基本符合幂率分布，股票网络模型的演化基本符合无标度网络模型的演化，而其在局部上又具有小世界演化的特性，所以使用综合这两者的混合演化模型来模拟股票网络的演化。

考虑到模型计算的复杂性，我们从 2010 年开始进行模型演化，每一年演化一次，直到 2020 年，然后使用实际 2020 年的网络模型和由我们的模型演化得到的网络模型进行比较，从而修正我们的模型，更新模型参数。

6.2 模型建立

6.2.1 网络节点变化模型

结点增加模型 节点的增加可以模拟新股票的入市，由于新股票入市一段时间之后才能得到一段其价格变化的时间序列，从而具体计算和其他的股票的相关关系，所以我们把节点加入网络的时间相对于股票入市延迟一年。

在去除部分错误以及不适用的数据后，统计了每年新上市的股票的数量如下表：

年份	上市股票数	年份	上市股票数
2010	342	2015	222
2011	277	2016	227
2012	154	2017	438
2013	2	2018	105
2014	122	2019	203

由于受到政策等多方面因素的影响，每年的股票上市数没有呈现出明显的规律性，可以看到，由于创业板政策的改变，2010 年之后每年股票上市率相比较于从前在总体上明显增加，但是也有像 2013 年这样的上市股票数量较之前明显减少的偶然情况，从而给预测带来了极大的困难。

我们假设 t 年新增节点为 $m(t)$，为了避免预测不准确带来的误差，我们使用 2011 年到 2020 年每年上市股票的平均值作为每年新增的节点，用 m_i 表示 i 年新上市的股票数，$N = 10$，则：

$$m(t) = \bar{m} = \frac{1}{N} \sum_{i=2010}^{2019} m_i \tag{8}$$

节点删除模型 在实际情况中，股票网络模型中节点的删除表示股票的退市。在删除节点之后，还应该删除节点所连接的边。经过检查，所给的数据中没有出现股票退市的情况，因此我们不考虑这种情况。

6.2.2 网络边变化模型

边的增加 我们可以根据问题一得到 2019 年节点平均度为 10.5，其中 57% 的节点的度大于 0。

在时间距离 2019 年较近的时候，新增节点对网络的影响较小，因而我们可以用 2019 年平均节点的度来代表其它年份的情况，把这一数值作为新增节点的度。我们假设新增节点的度为 d，已存在的第 i 个节点的度为 k_i，则一个新节点与节点 i 相连接的概率 Π 为:

$$\Pi_i = \frac{k_i}{\sum_j k_j} \tag{9}$$

其中 $j = (1, 2, \cdots, n)$，n 为节点的总数。

在算法实现上，因为 $\sum_i \Pi_i = 1$，我们可以把 (0,1) 划分为 n 个区间，每个区间属于一个节点，然后生成 d 个 0 到 1 之间的随机数，根据随机数在 n 个区间的范围，决定与新节点相连的节点。

边的删除 在实际意义上，两个节点之间边的删除表示两个节点所代表的股票之间的联系小于所设置的阈值，视为这两只股票之间没有联系。为了简化模型，我们不考虑这种情况。

边的重连 由于股票在局部范围内具有小世界特性，小世界网络是由规则网络随机重连边而生成的。由于股票网络演化模型图具有多个联通分量，因而可以视为多个小世界网络的组合，我们在对于总体网络图的中每一个小世界网络的边设置一个概率 p，p 是一个可调的参数。即每条边有 p 的概率重连。

6.3 模型求解

我们可以把数据的开始时间（1990 年）作为演化起点，每一年按照下面的方法演化一次：

1. 演化时按顺序加入顶点
2. 依据择优连接方式来分配新节点连接边的概率，度较大的节点连接边的概率较大
3. 在演化到时间终点时与实际股价对比

我们的演化模型的目标是使得通过演化得到的模拟股票相互作用的网络模型的各项拓扑参数与过程中真实网络的拓扑参数尽量接近。用 C_M 表示模型的聚类系数，α_M

表示模型中度分布的参数，L_M 表示模型的平均路径长度，这一优化目标可以表示为：

$$\begin{cases} \min C_M - C \\ \min \alpha_M - \alpha \\ \min L_M - L \end{cases} \tag{10}$$

按照上面的优化条件，我们得到的网络模型的全局的连接情况如下：

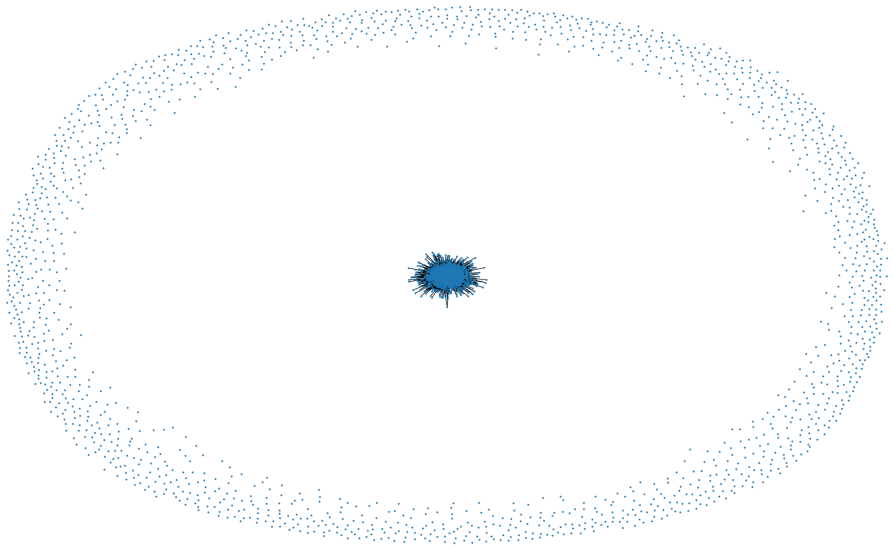
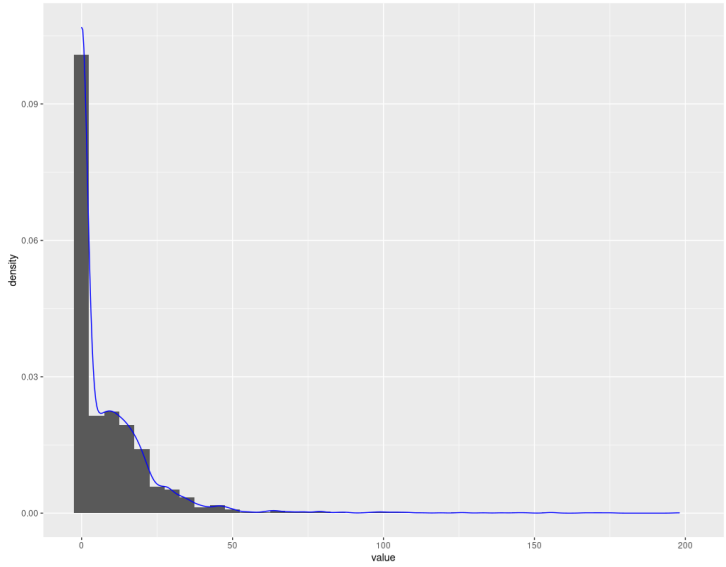


图 5　网络的全局连接情况

网络的平均路径长度 $L_M = 2.912$，网络的聚类系数 $C_M = 0.003$，使用幂律分布得到的拟合优度为 0.347，网络的度的分布情况如下图：



七、 问题三

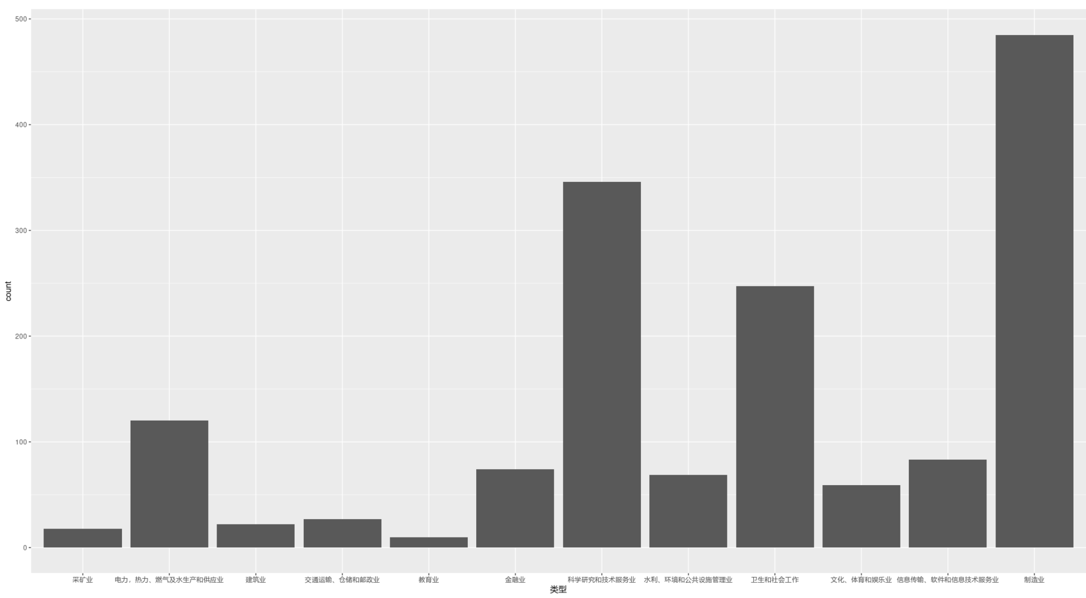
7.1 模型准备

7.1.1 股票数据的分类与选取

在问题三中，我们建立基于股票网络的投资组合模型，并选择几支股票数据进行最优投资。考虑到在股票投资中的实际情况，我们需要把股票按照一定的类型进行划分，以便于进行投资选择。考虑到股票市场中实际的股票板块划分情况以及我国的行业分类情况，我们把股票分类为金融业、教育业、制造业等 12 大类。

我们可以从各支股票的股票名称中提取关键词来判断各支股票的所属分类。对所有股票提取关键词后构造的词云图如下：

我们根据股票具有的关键词对股票进行分类（例如包括“银行”关键词的分类为金融业）。最终把约 1700 支股票划分进 12 个类别，具体分类情况如下图：



我们可以通过从各个大类中选择特定数量的股票的方法来实现股票数据的随机选择。

7.1.2 投资组合理论

根据文献 [6] 人们进行投资,本质上是在不确定性的收益和风险中进行选择。Markowits 投资组合理论用均值—方差来刻画这两个关键因素。所谓均值，是指投资组合的期望收益率，它是单只证券的期望收益率的加权平均，权重为相应的投资比例。即:

$$\bar{p} = \frac{1}{n}\sum_{i=1}^{N} p_i \tag{11}$$

所谓方差，是指投资组合的收益率的方差。我们把收益率的标准差称为波动率，它刻画了投资组合的风险。记 X 为 n 项资产的收益向量，V 为协方差矩阵，则投资组合的方差可表示为:

$$\sigma^2 = X^T V X \tag{12}$$

投资者可以预先确定一个期望收益, 通过 Markowits 投资组合理论模型来确定投资者在每个投资项目上的投资比例, 使其总投资风险最小。

7.2 模型建立

针对选取的的不同投资组合，我们可以首先计算对应各个投资组合的价值情况。在 $t-1$ 时刻，我们选取的投资组合的价值为

$$V_{(p,t-1)} \sum_{i=1}^{n} N_{(i,p)} p_i(t-1) \tag{13}$$

在 t 时刻，我们选取的投资组合的价值为:

$$V_{(p,t)} \sum_{i=1}^{n} N_{(i,p)} p_i(t) \tag{14}$$

投资组合的回报率可表示为（它也可以表示为各支股票的收益率的加权平均）:

$$r_{(p,t)} = \frac{V_{(p,t)}}{V_{(p,t-1)}} - 1 \tag{15}$$

$$= \sum_{i=1}^{n} x_{(i,p)} r_{(i,t)} \tag{16}$$

其中在 $t-1$ 时刻证券 i 在投资组合中的的占比 $x_{(i,p)}$ 为

$$x_{(i,p)} = \frac{N_{(i,p)} P_{(j,t-1)}}{V_{(p,t-1)}} \tag{17}$$

为了便于之后对于投资组合方差的分析，我们使用矩阵表示投资组合 p，以及投资的收益率 Rw, 同时用 r_i 来表示股票 i 的**收益率**而非对数收益率。则：

$$X_p^T = [x_{(1,p)}, x_{(2,p)}, \ldots, x_{(n,p)}]$$
$$Rw^T = [E(r_1), E(r_2), \ldots, E(r_n)]$$

用 V 表示证券市场的协方差矩阵，则投资组合 p 的方差为：

$$Var(p) = \sum_{j=1}^{n}\sum_{i=1}^{n} x_{(j,p)}x_{(i,p)}Cov(r_j, r_i) = X_p^T V X_p \tag{18}$$

投资组合 p 和 q 的协方差为：

$$Cov(p,q) = \sum_{j=1}^{n}\sum_{i=1}^{n} x_{(j,p)}x_{(i,q)}Cov(r_j, r_i) = X_p^T V X_q \tag{19}$$

我们可以在能够取得一定的收益率的方案中选取方差最小的方案，从而实现选择在达到既定收益前提下风险最小的方案。

7.3 模型求解

我们首先根据模型准备部分的方法，对股票数据进行了最基本的分类。并且在每个分类中选取 1 只股票来反映不同分类的情况。

之后，用 V 代表既定收益，我们可以把上面的模型表述为一个形式如下的优化模型：

$$\begin{aligned} \arg\min Var &= \sum_{j=1}^{n}\sum_{i=1}^{n} x_{(j,p)}x_{(i,p)}Cov(r_j, r_i) \\ \text{s.t.} \quad & V_{(p,t)} > V \end{aligned} \tag{20}$$

求解过程的算法步骤如下：

1. 将股票数据集按照相关程度进行划分，保证相关程度大的股票被分到同一类中。
2. 从划分得到的每一类股票中各随机选取一种，组成股票投资组合方案。
3. 通过方差来计算该股票投资组合方案的风险度。
4. 多次重复步骤（2）（3）进行模拟，取风险度最小的方案作为所求的目标方案。

八、 问题四

在问题四中，我们需要基于股票网络建模与投资组合理论建立数学模型来给出个人投资风险控制的综合建议。我们在问题三中我们已经建立了关于股票网络建模与投资组合理论的符复合模型，我们可以借鉴模型三中“使用网络建模中的参数来确定投资组合理论的均值”这一思想来对已有模型进行改进。

由于在投资组合理论中，方差代表了投资风险，因而控制个人的投资风险可以转化为控制投资方案的方差。

用 V 表示证券市场的协方差矩阵，则投资组合 p 的方差为：

$$Var(p) = \sum_{j=1}^{n}\sum_{i=1}^{n} x_{(j,p)}x_{(i,p)}Cov(r_j, r_i) = X_p^T V X_p \tag{21}$$

我们可以通过限制方差的范围来实现对于风险的控制，从而给出个人投资风险控制的综合建议。

在问题 4 的求解过程中，我们考虑了综合了网络模型与投资组合理论模型的问题三的部分求解结果。从而得到了在控制风险在一定范围时的投资方案。

根据控制风险后的投资方案，关于在实际的股票投资中进行风险控制，我们在这里给出以下建议：

- 由于股票的涨跌具有一定的相关性，且这种相关性在同行业的股票中较强，应当尽可能综合搭配各个分类的股票
- 在多数情况下，“金融业”和“电力，热力、燃气及水生产和供应业”的股票的价格比较稳定，风险因素较小，
- 在多数情况下，成交额大，股本较多的股票的价格比较稳定，风险因素较小

九、 模型评价与推广

9.1 模型优点

- 模型通过预测不同年度的股票节点增长，较好的模拟了股票网络的演化过程
- 一句各种投资组合的期望收益率及其方差，建立了投资组合模型，能够较好的确定使得总投资风险最小的投资方案

9.2 模型缺点

- 模型没有考虑到实际证券交易中**拆股**等因素对于股票间相关性的影响，对于部分有过拆股行为的股票可能无法找到实际相关性最大的股票
- 问题二的预测模型没有较好的考虑到近年来股票市场的部分政策带来的影响（例如创业板开放导致上市股票数增加）
- 问题二的预测以年为单位，预测的精度较差
- 问题三的选择中忽略了同类股票内部的差异性

9.3 模型改进

在问题二中，我们可以求解在不同时间下节点的度，并用这些数据预测不同年份节点度的新增情况，从而使得每一年的网络的边变化情况更接近于真实值。

在问题三中，根据文献 [7] 和 [8]，很多学者认为 Markowitz 投资组合理论中用方差来刻画风险的方法并不恰当，因为均值—方差模型把高于均值的那部分超额收益也当成风险，而实际上这部分收益是投资者所喜好的。

根据论文 [8]，在收益部分的确定中，我们可以考虑使用局部积分均值法和移动平均法来提高对模型对收益率变化的灵敏度。在风险部分的确定中，可以考虑使用半方差测量或在险价值来衡量风险因素。这些方法相对原方法能够较好的评价高于均值的部分超额收益。

参考文献

[1] Chi K T, Liu J, Lau F C M. A network perspective of the stock market[J]. Journal of Empirical Finance, 2010, 17(4): 659-667.

[2] Clauset A, Shalizi C R, Newman M E J. Power-law distributions in empirical data[J]. SIAM review, 2009, 51(4): 661-703.

[3] Liu W, Ma Q, Liu X. Research on the dynamic evolution and its influencing factors of stock correlation network in the Chinese new energy market[J]. Finance Research Letters, 2021: 102138.

[4] 李本田. 基于复杂网络的股票网络建模与结构分析 [D]. 兰州理工大学,2016.

[5] M.E.J.Newman. 网络科学引论 [M]. 电子工业出版社, 2014:272-276.

[6] 李惟进. 证券投资组合理论综述 [J]. 改革与开放,2010(06):52.

[7] 郑振龙, 陈志英. 现代投资组合理论最新进展评述 [J]. 厦门大学学报 (哲学社会科学版),2012(02):17-24.

[8] 蔡冰晶. 马克维茨均值方差模型在中国股票市场的应用 [D]. 复旦大学,2012.

附录 A　用于建立网络并绘制网络图的 Python 代码

```
import math
import numpy as np
import pandas as pd
import glob
import os
import networkx as nx
import urllib
import matplotlib.pyplot as plt
import random


def calLogProfit(arr):
    l=len(arr)
    arr=np.log(arr)
    list=[]
    for i in range(l-1):
        list.append(arr[i+1]-arr[i])
    return list

list=glob.glob(r'./2021年8月11日后复权数据CSV/*.csv')
dataArr=[]
namelist=[]
totnum=100000000

for item in list:
    name=os.path.split(item)[1].split('.')[0]
    namelist.append(name)
    data=pd.read_csv(item,encoding='gbk')
    #date=data['日期']
    #opendisc=np.array(data['开盘'])
    #maxn=np.array(data['最高'])
    #minn=np.array(data['最低'])
    closedisc=np.array(data['收盘'])
    #dealnum=np.array(data['成交量'])
    #dealmoney=np.array(data['成交额'])
    res=calLogProfit(closedisc)
    # totnum=min(totnum,len(res))
    dataArr.append(res[-225:])

i1=random.randint(0,len(namelist)-1)
i2=random.randint(0,len(namelist)-1)
while i1==i2:
    i2=random.randint(0,len(namelist)-1)
```

```
inde1=min(i1,i2)
inde2=max(i1,i2)

# dataArr=np.array(dataArr[inde1:inde2])
# namelist=np.array(namelist[inde1:inde2])
dataArr=np.array(dataArr)
namelist=np.array(namelist)
corref=np.corrcoef(dataArr)
g=nx.Graph()
totedge=0
totnode=0
singdegree={}
degloc={}
for item in namelist:
    g.add_node(item)
    totnode+=1
    singdegree.update({item:0})

for i in range(totnode-1):
    degloc.update({i+1:0})

for i in range(namelist.size-1):
    for j in range(i+1,namelist.size):
        if corref[i][j]>0.5:
            g.add_edge(namelist[i],namelist[j])
            g.nodes[namelist[i]]['size']=0.01
            totedge+=1
            singdegree[namelist[i]] += 1;degloc[singdegree[namelist[i]]] += 1
            singdegree[namelist[j]] += 1;degloc[singdegree[namelist[j]]] += 1
            if singdegree[namelist[i]] > 1: degloc[singdegree[namelist[i]]-1] -= 1
            if singdegree[namelist[j]] > 1: degloc[singdegree[namelist[j]]-1] -= 1

for it in degloc.copy().items():
    if it[1]==0:
        degloc.pop(it[0])

totdist=0
totpath=0

for i in range(namelist.size-1):
    for j in range(i+1,namelist.size):
        if nx.has_path(g,namelist[i],namelist[j]):
            totdist+=nx.shortest_path_length(g,source=namelist[i],target=namelist[j])
            totpath+=1

#singorder=sorted(degloc.items(),key=lambda x:x[1],reverse=True)
#x=[it[0] for it in degloc.items()]
```

```
#y=[it[1]/totnode for it in degloc.items()]
#plt.plot(x,y,linewidth=2.5)
#plt.ylim([0,1.1*max(y)])
#plt.xlim([1,max(x)])
nx.draw(g,node_size=1,width=1,with_labels=False,font_size=10)
plt.show()
print(corref)
print(namelist)
print("Ccoe",2*totedge/(totnode*(totnode-1)))
print(totdist)
print(totdist/totpath)
print(math.log(totnode))
print(degloc.keys())
print(degloc.values())
ls1=[]
for itm in singdegree.items():
    ls1.append(itm)
df=pd.DataFrame(ls1)
df.to_csv("deg.csv")
#print(totnum)
```

附录 B　验证小世界特性的 Python 语言代码

```
import scipy.optimize as op
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score

def func(x,a,b):
    return a*x+b

y=np.array([2.142857,3.065280,3.605354,3.765330,3.823375,3.622547,3.9770449311959775
,3.986068402433029,4.250885080353088,4.310753734670081,4.274056336224455])
x=np.array([5.117993812416755,6.345636360828596,6.919683849847411,6.93828448401696,7.197435354096591,
7.24422751560335,7.554858521040676,7.619233416226805,8.034955024502159,8.106816038947052,8.256347772918016])
popt,_=op.curve_fit(func,x,y,maxfev=2000)
yt=func(x,popt[0],popt[1])
plt.plot(x,yt,color='blue')
print(r2_score(y,yt))

plt.scatter(x,y,s=5,c='red')
plt.ylim([0,1.1*max(y)])
plt.xlim([min(x),max(x)])
plt.show()
print(popt)
```

附录 C　验证无标度特性的 R 语言代码

```r
library(poweRlaw)
library(ggplot2)
#deg.csv存储各个节点的度
data<-read.csv("deg.csv")
d<- as.numeric(as.character(data$value))
for (i in 1:length(d)) {
   d[i]=d[i]+1
}

p<- ggplot(data,mapping = aes(x=value))
p+geom_histogram(binwidth = 5,mapping = aes(y=stat(density)))+geom_density(color="blue")
#p+geom_area(stat="density",fill="blue")

m=displ$new(d)
est=estimate_xmin(m)
m$setXmin(est)

b<-bootstrap_p(m,threads = 11,no_of_sims = 1000)

print(data,b)
print(mean(b$bootstraps$pars))
print(mean(b$bootstraps$xmin))
```

附录 D　画词云图的 Python 代码

```python
import glob
import os
import pandas as pd
import jieba
import wordcloud
from imageio import imread
import matplotlib.pyplot as plt

def typecheck(name,checklist):
   for itm in checklist:
      if itm in name:
         return True
   return False
```

```
type1=['电力','能源','新能','燃气','煤电','水利','电气','水务','铁建','西电','电建','核电','天然气']
type2=['建设','路桥','建工','建筑']
type3=['物流','高速','交通','客车','海运']
type4=['矿业','矿','稀土']
type5=['汽车','电器','电子','化工','化学','机械','光电','食品','水泥','纸业','装备','数控','机电','重工','航空','船
      '石化','钢','船舶','铝','铜','糖','醋','玻璃','盐','合金','轮胎','酵母','铂','车轴','纺织','锌','锗','铅','
      '半导','光学','照明','精工','卫浴','家居','气体','硅','新材','永磁','激光','动力','传感','复材','电材','眼镜
      '精密','材料','工具','晶体','精机','红外','仪器','光电','线缆','应材','高材','磁材','电源']
type6=['科技','技术','数据']
type7=['环境','生态','园林','节能','机场','环保','环卫']
type8=['信息','软件','通信','网络','通讯','数据']
type9=['证券','银行','资本','投资']
type10=['医疗','制药','生物','医药','医学','药','健康']
type11=['教育']
type12=['文化','传媒','影视','出版','旅游','电影','影业','游乐','超媒','演艺']

list=glob.glob(r'./2021年8月11日后复权数据/*.txt')
checklist=glob.glob(r'./2021年8月11日后复权数据CSV/*.csv')
checklist=[os.path.split(item)[1].split('.')[0] for item in checklist]

numlist=[]
namedict={}
numdict={}
for item in list:
    f=open(item,'r')
    namelist=os.path.split(item)
    name=namelist[1].split('.')[0]
    text=f.read()
    text=text.replace(' ',',')
    text=text.replace('\t',' ')
    st=0
    ed=0
    flag=0
    for i in range(len(text)):
        if text[i]==',' and flag==0:
            flag=1
            st=i+1
        elif text[i]==',' and flag==1:
            flag=2
            ed=i
        elif flag==2:
            break
    tar=text[st:ed]
    if name in checklist:
        if typecheck(tar,type1):
            namedict.update({name: (tar,'电力，热力、燃气及水生产和供应业')})
        elif typecheck(tar,type2):
```

```
            namedict.update({name: (tar,'建筑业')})
        elif typecheck(tar,type3):
            namedict.update({name: (tar,'交通运输、仓储和邮政业')})
        elif typecheck(tar,type4):
            namedict.update({name: (tar,'采矿业')})
        elif typecheck(tar,type5):
            namedict.update({name: (tar,'制造业')})
        elif typecheck(tar,type6):
            namedict.update({name: (tar,'科学研究和技术服务业')})
        elif typecheck(tar,type7):
            namedict.update({name: (tar,'水利、环境和公共设施管理业')})
        elif typecheck(tar,type8):
            namedict.update({name: (tar,'信息传输、软件和信息技术服务业')})
        elif typecheck(tar,type9):
            namedict.update({name: (tar,'金融业')})
        elif typecheck(tar,type10):
            namedict.update({name: (tar,'卫生和社会工作')})
        elif typecheck(tar,type11):
            namedict.update({name: (tar,'教育')})
        elif typecheck(tar,type12):
            namedict.update({name: (tar,'文化、体育和娱乐业')})
        else:
            namedict.update({name: (tar,'其他')})

nameframe=pd.DataFrame({'文件名':namedict.keys(),'企业名':[itm[0] for itm in
    namedict.values()],'企业类型':[itm[1] for itm in namedict.values()]})
nameframe.to_csv('企业名+企业类型.csv',index=False)
    #
    # li=jieba.lcut_for_search(tar)
    # li=[itm for itm in li if len(itm)>=2]
    # if 'ST' in li:
    #     li.remove('ST')
    # if '股份' in li:
    #     li.remove('股份')
    # if '集团' in li:
    #     li.remove('集团')
    # if '东方' in li:
    #     li.remove('东方')
    # if '浙江' in li:
    #     li.remove('浙江')
    # if '上海' in li:
    #     li.remove('上海')
    # if '江苏' in li:
    #     li.remove('江苏')
    # if '中国' in li:
    #     li.remove('中国')
    # li=' '.join(li)
```

```
    # numlist.append(li)
    # for itm in li:
    #     if itm in numdict:
    #         numdict[itm]+=1
    #     else:
    #         numdict[itm]=1

# for item in numdict.copy().keys():
#     if numdict[item]<3:
#         numdict.pop(item)

# wname=' '.join(numlist)
# w=wordcloud.WordCloud(background_color='white',scale=10,font_path='msyh.ttc').generate(wname)
# #plt.show()
# w.to_file('字云图.jpg')
    # if name in checklist:
    #     namedict.update({name:tar})

# nameframe=pd.DataFrame({'文件名':namedict.keys(),'企业名':namedict.values()})
# nameframe.to_csv('企业名.csv',index=False)
```

附录 E　用于描述网络演化模型的 Python 代码

```
    #股票网络演化模型
import matplotlib.pyplot as plt
import networkx as nx
import numpy as np
from random import random
import pandas as pd

class stock_network:
    def __init__(self, init_node_nums = 20, P_zero_degree = 0.43):
        self.G = nx.Graph()
        self.node_id = 1                        #节点编号
        self.init_node_nums = init_node_nums  #初始放入图中的节点数
        self.total_evo_times = 0                #网络总演化次数，1990年开始，演化一次算一年
        self.P_zero_degree = P_zero_degree    #节点度为0的概率

    def init_nodes(self):                               #初始化一些节点
        for i in range(self.init_node_nums):
            self.G.add_node(self.node_id)
            self.node_id += 1
        self.G.add_edge(1, 5)
        self.G.add_edge(4, 14)
        self.G.add_edge(9, 20)
```

```
def network_evolution(self, evo_times):          #股票网络演化，输入演化次数，每次演化表示一年
    for i in range(evo_times):
        self.total_evo_times += 1
        new_node_nums = self.num_of_new_nodes(self.total_evo_times)
        for j in range(new_node_nums):
            self.G.add_node(self.node_id)         #增加一个新节点
            self.node_id += 1
            if self.if_zero_degree():             #如果节点没有边，跳过
                continue
            connected_node_list = self.connected_node()
            for i in connected_node_list:
                self.G.add_edge(self.node_id-1, i)


def connected_node(self):                         #返回某一个节点连接的其他节点的编号
    total_degree = 0
    for i in range(1, self.G.number_of_nodes() + 1): #求当前图节点的总度数
        total_degree += self.G.degree[i]
    range_interval = np.zeros(self.G.number_of_nodes() + 1)
    for i in range(1, self.G.number_of_nodes() + 1):
        range_interval[i] = range_interval[i - 1] + self.G.degree[i] / total_degree
    new_edge_nums = self.num_of_new_edges(self.total_evo_times)
    connected_node_list = []
    for i in range(new_edge_nums):
        ran_number = random()
        for i in range(len(range_interval)):
            if range_interval[i] > ran_number:
                connected_node_list.append(i)
                break
    return connected_node_list

def num_of_new_nodes(self, total_evo_times):     #每次演化新增加节点的个数，是一个时间的函数
    return int(380 * random())

def num_of_new_edges(self, total_evo_times):     #每个新增加节点的边数，时间的函数，和度有关
    return int(18 * random())

def if_zero_degree(self):
    ran_number = random()
    if ran_number < self.P_zero_degree:
        return True
    else:
        return False

def draw_network(self):                           #画图
    plt.figure()
```

```
        nx.draw(self.G, node_size=4, width=1, with_labels=False, font_size=10)
        plt.show()

    def graph_info(self):                               #计算平均最短路，聚类系数等信息
        #print(1)
        # cluster_coeffi = nx.clustering(self.G)       #聚类系数
        # print('clustering coefficient:', cluster_coeffi)
        #
        # tot_dist = 0                                  #计算平均最短路
        # tot_path = 0
        # for i in range(1, self.G.number_of_nodes()-1):
        #     for j in range(i + 1, self.G.number_of_nodes()-1):
        #         if nx.has_path(self.G, i, j):
        #             tot_dist += nx.shortest_path_length(self.G, source = i, target = j)
        #             tot_path += 1
        # print('average path distance:', tot_dist/tot_path)

        max_degree = 0                                  #计算度分布
        for i in range(1, self.G.number_of_nodes() + 1):
            if self.G.degree[i] > max_degree:
                max_degree = self.G.degree[i]
        degree_distribute = np.zeros(max_degree+1)
        for i in range(1, self.G.number_of_nodes() + 1):
            degree_distribute[self.G.degree[i]] += 1
        lis=[i for i in range(max_degree+1)]

        datdf=pd.DataFrame({'degree':lis,'num':degree_distribute})
        datdf.to_csv('abc.csv')
        print(degree_distribute)

if __name__ == '__main__':                              #程序执行起点
    a = stock_network()
    a.init_nodes()
    a.network_evolution(20)
    #a.draw_network()
    a.graph_info()
```