Riginos Samaras

CEI 524– Network Science (Επιστήμη Δικτύων)
Assignment 2 : Report / Readme file, due by November 19th

Dr. Fragkiskos Papadopoulos

Cyprus University of Technology
Department of Electrical Engineering, Computer Engineering and Informatics.
MSc in Data Science and Engineering

Table of Contents

Introduction

My main focus was to create a program that is functional and has a very small execution time, even though the data that it manipulates is big. The output data is a 32Gb file, which has by far exceeded any other data given to me so far to process. So I decided to implement that program in C++ which is one of the fastest languages for computation. I also considered to implement every algorithm in a way that its complexity does not exceed O(NlogN). Using the most common fastest algorithms like merge join, binary search and by indexing my data I managed to run the core program in less than 20 seconds on a 4-core,16Gb RAM MacBook Pro. The raw data that the main program produces is processed by 3 small python programs to create my plots. I chose Python for the plotting because the produced raw data is very small and does not need much computational power to be processed.

1. Purpose

The purpose of the assignment was to get familiar with:

- Big Data
- Network concepts and attributes
- Network Visualization Tools
- Plots on our raw data by using plot libraries

2. Objectives:

1) First we had to read the paper “Scale-Free Functional Networks” which would help us understand the basic concept of the Brain Networks and give us some feedback on how our output should be.

2) Use a data file containing a dataset and code from the paper to create our data. After following the instructions on how to compile that data we got an output of 32gb data(output.data). That file contained 3 columns: Voxel v_1, voxel v_2 and r, which is correlation coefficient. If that r exceeded a given threshold r_c then v_1 and v_2 are connected.

3) Based on that data we should estimate the percolation transition value in order to have a Giant component which is sized about 50% of the whole Network.

4) Calculate the size, the average degree and the average clustering of that GC.

5) The raw data for the degree distribution P(k), average degree of neighbors k_nn k of nodes of degree k, and average clustering c(k) of nodes of degree k in this network.

6) The log-log plots of those statistics

7) The visualization of network

3. How I accomplished my Objectives:

For the objective (2) I installed the Fortran compiler and followed the instructions to create the "output.dat" file. I used gfortran instead of g77.

- That output.dat file contained many unnecessary data like negative correlation and was very big to keep calling it from my main program.
- So I separate that file in 1000 smaller files based on the correlation *r*. I created files named: 0.001, 0.002 … 0.999.

For the objective (3). I created a Graph of Nodes in my main program by reading each line of my data.

- In order to insert only unique nodes in my graph I used the data type <Set>. The big O of insert in a set is O(logN) so the creation of the graph was much more efficient.
- Each node had attributes like name and ListOfNeighbors which contained all the edges to his neighbors.
- To find the Giant Component each time I used the BFS algorithm.
- I manually changed the value of threshold rc and found out that the value that gives me a Giant Connected Component is **0.699.** That value gives me a GCC which contains the **50.96%** of the Network nodes. After finding the GCC I erased every other Node in the Network and started calculations

For the objective (4):

- To calculate the size of the Network I gone through the whole data once and increased my counter each time I found a unique node. Size of the total Network is: 34921 nodes (ignoring negative nodes).
- The size of the GCC is: 17796 nodes.
- The average degree is calculated by using the list: LinkedNodes in each node. So passing the graph once was enough to calculate the average degree Average degree is: **7.785**
- The average clustering is calculated by using the clustering algorithm for each node separately: $\frac{2N}{k(k-1)}$ Average Clustering is: **0.1423**

For the objective (5):

- I used 3 hash-maps with key:degree, to store the values of degree distribution, average degree of neighbors . So each time a node with the same degree was calculating its attributes I added it to the last inserted key-value pair.
- After that I only had to pass from each hash-map once and print my raw data.

For the objective (6):

- I created 3 small python program that use the "plot/*.dat" files to illustrate the plots. Each of them I uses methods in the matplotlib library for the plots. The degree distribution, average clustering and Neighboring node's degree outputs can be found in the How to run / Outputs section.

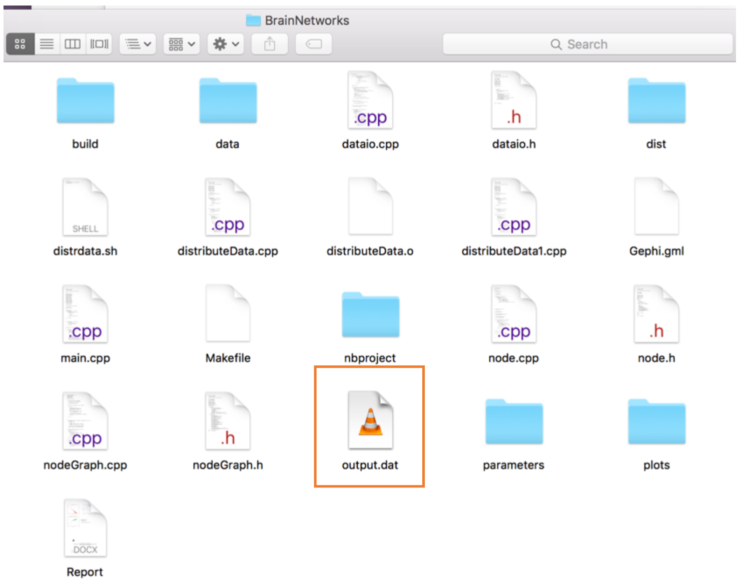
For the objective (7):

- I used I created a ".gml" file from my main program, which contained every node and every edge in the GCC. Then I opened that file in Gephi network visualization tool and manipulated the graph to create a more attractive output which can be found in the How to run / Outputs section.

4. How to run / Outputs:

4.1 Distribute data:

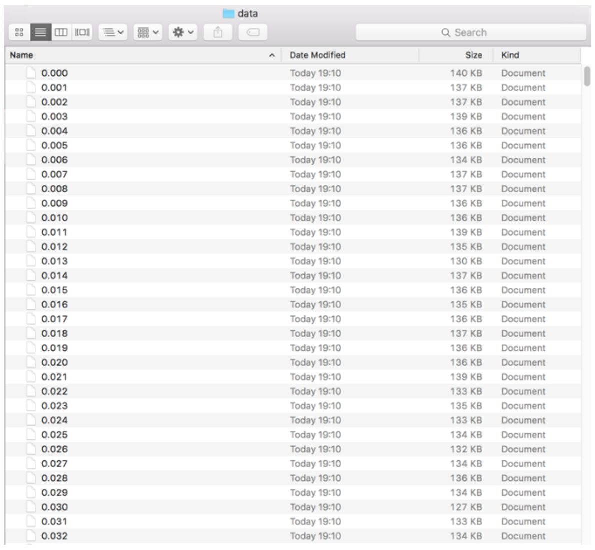
I separate my data based on the correlation coefficient *r* for faster execution.
-Move the “output.dat” file in the BrainNetwork folder:



-Use terminal to run the script “distrdata.sh”:

```
Riginos at Riginoss-MacBook-Pro in ~/NetBeansProjects/BrainNetworks
[$ sh distrdata.sh
```

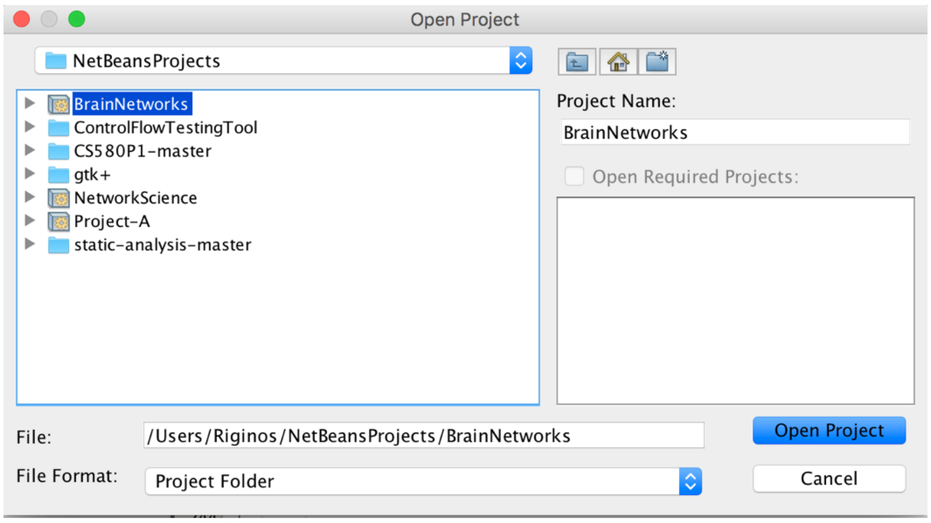
-The “output.data” file will be separated to smaller files based on the correlation coefficient *r*. Negative lines are not taken into consideration. (This might take some hours to finish)
The Output in the “data” folder should look like this:



Name	Date Modified	Size	Kind
0.000	Today 19:10	140 KB	Document
0.001	Today 19:10	137 KB	Document
0.002	Today 19:10	137 KB	Document
0.003	Today 19:10	139 KB	Document
0.004	Today 19:10	136 KB	Document
0.005	Today 19:10	136 KB	Document
0.006	Today 19:10	134 KB	Document
0.007	Today 19:10	137 KB	Document
0.008	Today 19:10	137 KB	Document
0.009	Today 19:10	136 KB	Document
0.010	Today 19:10	136 KB	Document
0.011	Today 19:10	139 KB	Document
0.012	Today 19:10	135 KB	Document
0.013	Today 19:10	130 KB	Document
0.014	Today 19:10	137 KB	Document
0.015	Today 19:10	136 KB	Document
0.016	Today 19:10	135 KB	Document
0.017	Today 19:10	136 KB	Document
0.018	Today 19:10	137 KB	Document
0.019	Today 19:10	136 KB	Document
0.020	Today 19:10	136 KB	Document
0.021	Today 19:10	139 KB	Document
0.022	Today 19:10	133 KB	Document
0.023	Today 19:10	135 KB	Document
0.024	Today 19:10	133 KB	Document
0.025	Today 19:10	134 KB	Document
0.026	Today 19:10	132 KB	Document
0.027	Today 19:10	134 KB	Document
0.028	Today 19:10	136 KB	Document
0.029	Today 19:10	134 KB	Document
0.030	Today 19:10	127 KB	Document
0.031	Today 19:10	133 KB	Document
0.032	Today 19:10	134 KB	Document

4.2 BrainNetwork Application:

The given folder “BrainNetwork” is a project created in Netbeans so its preferable to run the program by using NetBeans.
-Open your preferred IDE for C++ and open the BrainNetworks project:



Project Name: BrainNetworks

File: /Users/Riginos/NetBeansProjects/BrainNetworks

File Format: Project Folder

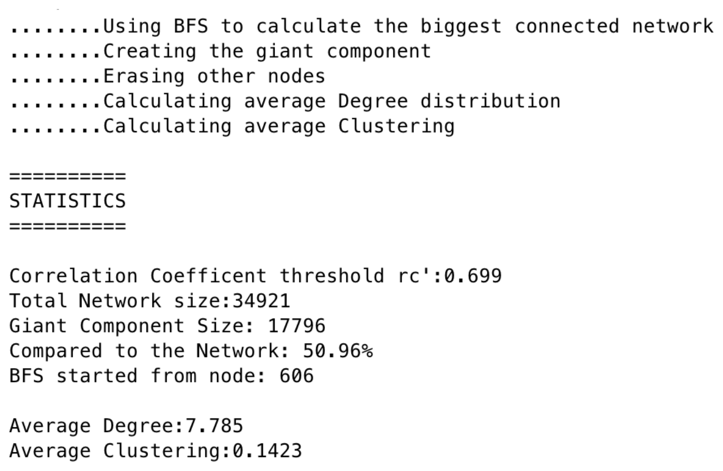
-Compile and run:
-You will be asked to give an rc. You can have a Giant Connected Component when u give an rc which is less or equal to 0.699

```
........Checking if size of the whole network is calculated:
Already calculated:34921
Give correlation threshold:
>0.699
```

-The program will start to read only the files with that exceed the above rc:

```
data/0.978:                              data/0.709:
data/0.976:                              data/0.708:
data/0.974:                              data/0.707:
data/0.973:                              data/0.706:
data/0.972:                              data/0.705:
data/0.971:          •     •     •       data/0.704:
data/0.970:                              data/0.703:
data/0.969:                              data/0.702:
data/0.968:                              data/0.701:
data/0.967:                              data/0.700:
data/0.965:                              data/0.699:
```

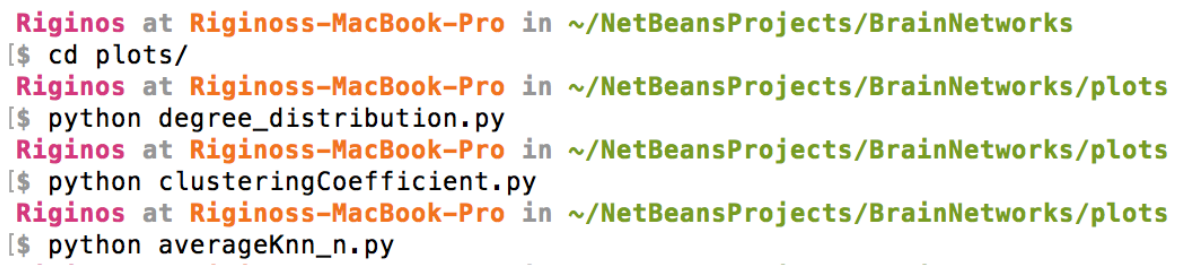
The output should look like this:



```
........Using BFS to calculate the biggest connected network
........Creating the giant component
........Erasing other nodes
........Calculating average Degree distribution
........Calculating average Clustering

==========
STATISTICS
==========

Correlation Coefficent threshold rc':0.699
Total Network size:34921
Giant Component Size: 17796
Compared to the Network: 50.96%
BFS started from node: 606

Average Degree:7.785
Average Clustering:0.1423
```

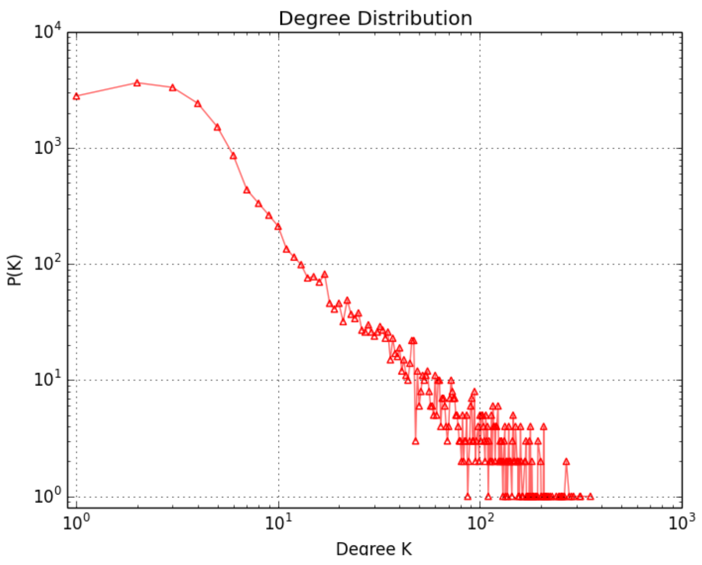
4.3 Plots:

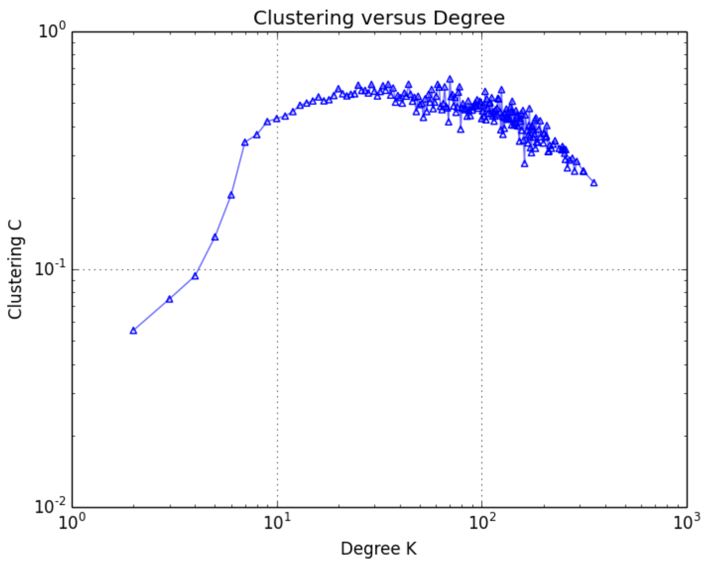
- Use the terminal to go to the plot folder, which is located in the BrainNetwork program and run each python program separately. Those programs use the raw data to create log plots.

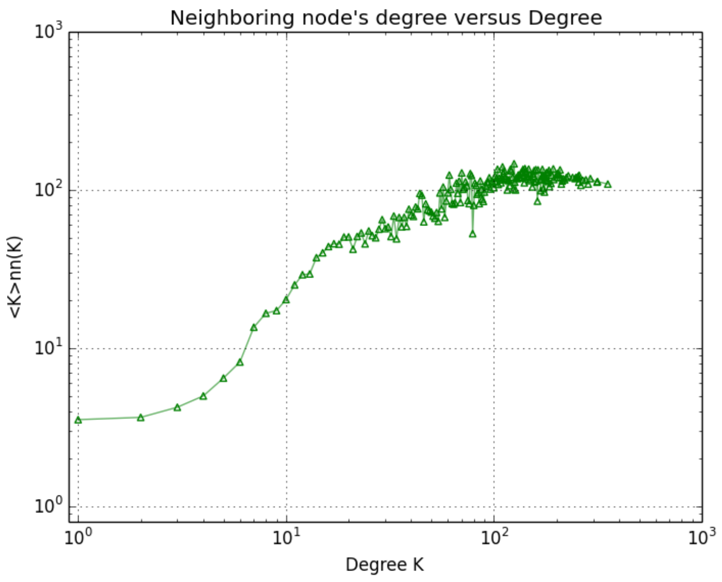


```
 Riginos at Riginoss-MacBook-Pro in ~/NetBeansProjects/BrainNetworks
[$ cd plots/
 Riginos at Riginoss-MacBook-Pro in ~/NetBeansProjects/BrainNetworks/plots
[$ python degree_distribution.py
 Riginos at Riginoss-MacBook-Pro in ~/NetBeansProjects/BrainNetworks/plots
[$ python clusteringCoefficient.py
 Riginos at Riginoss-MacBook-Pro in ~/NetBeansProjects/BrainNetworks/plots
[$ python averageKnn_n.py
```

The output should look like this:



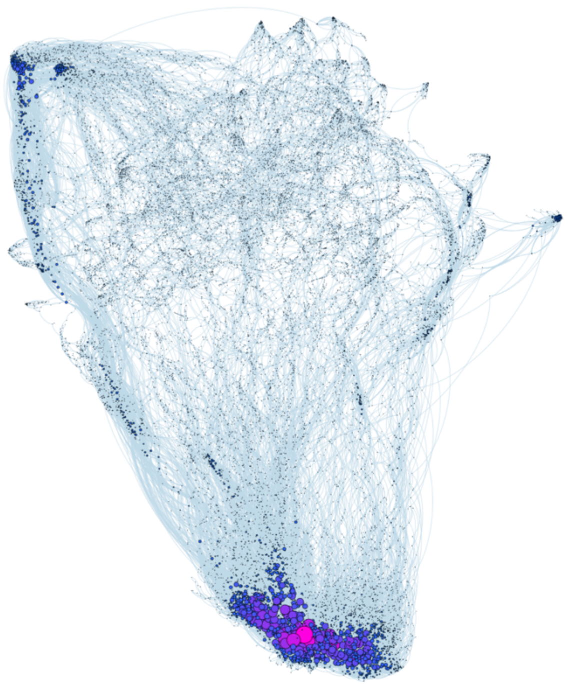




Gephi – Network Visualization:

-Open the network.gephi file which is located in the “plots/” folder with the Gephi application.

Output:



-I also attached the PDF file that I exported from Gephi which is also in the “plots/” folder.

Implementation

A C++ Application program called “BrainNetwork” to:

- Read the whole data.
- Create a Graph that has nodes which contain neighbors and weights(r) on their edges.
- Run a BFS to find the Giant Connected component.
- Find a Giant Component that contains 50% of all the nodes Network.
- Calculate every data of objectives (4) and (5) and write the raw data to files.

4 smaller Python programs in plot folder for the creation of the log-log plots.

Requirements

- C and C++ development: Netbeans 8.0.2 IDE.
- C++ compiler: Xcode 7.1.1.
- Python version: Python 2.7.10
- Plotting library: matplotlib

*This project is build on MacOSX El Capitan version 10.11.1 and has not been tested on Windows.