4-29-2021

Using Spectral Graph Theory to Analyze Gene Expression Networks

Chuyen Nguyen
Mississippi State University

Using Spectral Graph Theory to Analyze Gene Expression Networks

By

Chuyen Nguyen

Andy Perkins	Bindu Nanduri
Professor	Associate Professor
(Director of Thesis)	(Committee Member)

Seth Oppenheimer

Professor

(Shackouls Honors College Representative)

Honors Thesis submitted to Shackouls Honors College, Mississippi State University, Spring 2021

Using Spectral Graph Theory to analyze gene expression networks

By

Chuyen Nguyen

Approved by:

Andy Perkins (Major Professor)
Bindu Nanduri
Seth Oppenheimer

Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Provost Scholars Research Project

Mississippi State, Mississippi

April 2021

Name: Chuyen Nguyen

Date of Degree: April 29, 2021

Institution: Mississippi State University

Major Field: Computer Software Engineering

Major Professor: Andy Perkins

Title of Study: Using Spectral Graph Theory to analyze gene expression networks

Pages in Study 24

Member of Provost Scholarship

This paper covers the potential applications of using the spectral analysis of a graph's Laplacian matrix to gene co-expression networks. The general idea is to take publically available genetic data from cancer studies, organize them into gene co-expression networks, and analyze them using Spectral Graph theory.

The publically available cancer study data includes files that show the occurrence of several different genes within a single person's genetic data (given by fragments per kilobase million). The research takes the data of several patients and organizes them into a table. From there, each gene's occurrence is measured against every other gene's occurrence using Pearson correlation values, resulting in another table of pairs of genes and their correlation values. For each line, a gene, another gene, and their Pearson correlation value are represented in three columns corresponding to each aforementioned piece of data. Only genes that have a Pearson correlation value above a certain threshold were added to this file so that the file represents only significant correlations between genes.

This file could be interpreted as an edge list for a gene co-expression network, which is a graph showing connections between genes. Each gene represents a single node of the graph, and every line of the file contained the connection between two genes, this connection being the edge

between two nodes. With the file able to be interpreted as a graph, Spectral Graph theory concepts applied to it very naturally, allowing us to extract the second-smallest eigenvalue of the graph's Laplacian matrix and the degree of zero eigenvalues, which gave us an understanding of the graph's structure.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The main goal of our research is to find out if the relationship between different genes within cancer patients can allow us to understand the way that cancer growth appears based on the presence of certain genes within a person. We try to achieve this understanding by performing a series of data transformations and analyzing the data which results from these transformations.

The research begins by collecting gene data gathered from several cancer patients through a publicly accessible database. The data shows the expression or activity of certain genes within a person. This gene data from each patient is compiled together into one large table, and this table is analyzed to find the relationship between each gene, namely how often one gene appears alongside another gene.

In order to quantify this comparison, we find a value called the correlation between two compared genes, which is a measure of how similar the two genes are. If the correlation value does not meet a certain limit, we do not add it to the list of genes that have a relationship with each other. The list that is compiled from these comparisons has three columns: one gene, another gene, and their correlation value. We take this table and interpret it in a way that allows us to form a graph from it.

A graph is a manner of representing data that consists of two major elements: nodes and edges. A node represents a single piece of data, while an edge connects two nodes to represent a

relationship between the nodes. In our graph, the nodes will be the genes, and the edges will be that relationship we found in the last step. As a result of us setting a limit to the correlation value required to represent a relationship, only genes with a strong relationship will be represented in this graph.

The purpose of organizing the data into a graph is to apply a method of analysis that gives us clues to how the graph is structured, which may allow us to understand the relationship between many genes. In this research, we derive from the graph structures called matrices, which are essentially tables, and find the eigenvalues of these matrices which tells us properties of this graph we constructed. Through this analysis and the properties we derive from the eigenvalues, we hope to understand the graph structure and eventually how relationships between genes result in cancer growth.

Spectral Graph Theory

Spectral Graph Theory (SGT) is a theory concerning graph analysis. It concerns the spectrum of graphs, which is "the multiset of eigenvalues, that is the set of eigenvalues repeated according to their multiplicity." (Nica, 2018) Many different matrices related to graphs are able to produce interesting spectrums; however, in the context of this work, the prevalent aspects of SGT are the study and analysis of a graph through its Laplacian matrix and the eigenvalues and eigenvectors of that matrix.

The Laplacian Matrix

The formal definition of the Laplacian matrix of a graph given by Nica Bogdan in his publication *A Brief Introduction to Spectral Graph Theory* is

$$L(u,v) = \begin{cases} \deg(v) & if\ u = v \\ -1 & if\ u \sim v \\ 0 & otherwise \end{cases}$$

An important belief of SGT and one that is highly relevant to this paper is that the analysis of a graph's Laplacian and its eigenvalues and eigenvectors can lead to an understanding of properties of the graph and thus the data that the graph was created from.

Eigenvalues and Eigenvectors

The definition of a spectrum of the graph, which SGT is greatly concerned with, derives solely from the existence of eigenvalues of a matrix representation of a graph. Eigenvalues in this context allow for an in-depth understanding of several graph properties, many of which would not be easy to find without the analysis of eigenvalues. Examples of which include the prominence of graph spectra in minimizing energies of Hamiltonian systems and the arising links in SGT and Riemannian geometry through analysis of spectrum. (F. Chung, 1985)

The table below demonstrates some of the potential usage of graph spectra derived from Laplacian matrices.

Table 1 *Table of Laplacian Eigenvalue Relationships to Graph Properties*

Characteristic	**Meaning**
Multiplicity of eigenvalues of 0	Represents the number of connected components of the graph. Every Laplacian matrix of a graph will have at least one eigenvalue of zero. (Das, 2004)
Second smallest eigenvalue (λ_1)	Represents the "algebraic connectivity" of the graph, which refers to how connected the graph is. This eigenvalue is bounded between zero and one. (Fiedler, 1989)
λ_1 is greater than 0	The graph of the Laplacian matrix is connected. (Fiedler, 1989)
λ_1 is equivalent to 1	The graph of the Laplacian matrix is complete. (Fiedler, 1989)

Lower bound on λ_1	Implies an upper bound on the diameter of the graph. (F. R. K. Chung, 1995)
An eigenvalue of 2 from a normalized Laplacian matrix	The upper bound of any eigenvalue of a normalized Laplacian matrix of a graph is two. If any of a graph's normalized Laplacian's eigenvalues is two, then there is a connected component of said graph that is bipartite and nontrivial. (F. R. K. Chung, 1995)
The sum of all eigenvalues in the Laplacian spectrum is less than the number of vertices in the graph.	The graph has no isolated vertices. (F. R. K. Chung, 1995)
Sum of Laplacian eigenvalues	Equal to the sum of the degrees of a graph. (K. CH. DAS)

RNA Sequencing Data

RNA sequencing data, or RNA-Seq for short is a method of genome data cataloging that enables researchers who desire the means to analyze genetic information great breadths of information to analyze. This is allowed through the usage of next-generation sequencing techniques that give researchers information on billions of individual bases (Chu & Corey, 2012). RNA-Seq represents a great advancement in gene expression transcription from the previous method of microarrays which were by comparison less cost and time effective than the next-generation sequencing techniques used to produce RNA-Seq data. RNA-Seq data also provides the benefit of being "particularly attractive for the quantitative analysis of transcript expression levels" (Marguerat & Bähler, 2010), allowing for thorough analysis and study of the gene expressions of the subject collected from. Particular advantages of RNA-Seq data that have been utilized in this research is its ability to generate expression data for every base genome and the data representation's ease of mRNA analysis.

Gene Co-Expression Networks

Gene co-expression networks (GCN) are a method of organizing gene expression data in a manner that elicits the understanding of the relationship between different genes. This is done through the construction of an undirected graph with each node corresponding to a gene, and edges between these nodes whenever a relationship exists among them. Analysis of a structure created through this manner allows for the studying of what genes often appear together and which ones do not, giving researchers the ability to determine how relationships between genes elicit certain developmental characteristics that may be brought about by these co-expressions. For the details of this research, the Pearson correlation coefficient formula was used to compute the co-expressions between gene pairs.

There are many terms relating to graph theory and using graph theory to analyze biological networks that apply to GCNs. These terms will be relevant throughout future portions of this paper. The sparsity and density of a graph does not have a concrete definition. For this paper, a sparse graph is one where the number of edges is close to the lowest number of edges it could have, and a dense one having a number of edges close to the maximum number of edges it could have. The diameter of a graph is represented as $D = \max_{i,j} \delta_{min}(i,j)$, otherwise known as the "longest shortest path of the graph." The "clustering coefficient" of a graph tells the graph's tendency to divide into clusters. A network is scale-free if it's degree distribution exhibits a power law distribution (Pavlopoulos et al., 2011).

Data Used

The data used in this research is mRNA gene expression data received from the RNA-Seq data collection methods discussed in the earlier section. This data has been provided in an open-access model from The Cancer Genome Atlas, or TCGA (Tomczak et al., 2015). TCGA is a government funded joint effort between the National Cancer Institute and National Human Genome Research Institute which provides publicly accessible genomic data of cancer patients.

Approach

Efforts in this research project are made to uncover the potential uses of Spectral Graph Theory in analyzing gene expression networks. In order to accomplish this, RNA-Seq mRNA transcription data collected from TCGA is processed and organized into gene expression network graphs, with nodes representing each mRNA gene expression, edges representing their relationships and edge weights representing the Pearson correlation values derived between each gene.

The main pilot of these research efforts is NetworkX, a Python package that allows for the extensive code analysis of graphs (Hagberg et al., 2008). Through NetworkX, the gene correlation data is processed and transformed into a gene co-expression network in the fashion mentioned in the prior section. After which, NetworkX's function to derive a Laplacian matrix from a graph will be used and its eigenvalues and eigenvectors extracted through the utilization of SciPy - a broad-covering science and math package for Python (Virtanen et al., 2020).

Literature

A Brief Introduction to Spectral Graph Theory by Bogdan Nica gives a great overview of the concepts integral to SGT. From this beginner resource, I established a solid foundation of understanding for the subject of this research. Many of the equations used in the Spectral Graph Theory section of this introduction were sourced from this book. *Spectral Graph Theory* by Fan Chung provides a more comprehensive coverage of SGT, where it can be applied and its usages and is considered to be the book for those who are interested in using SGT.

An overview of the process used to derive RNA Sequencing data from biological material come from the articles "RNA Sequencing: Platform Selection, Experimental Design, and Data Interpretation" written by Yongjun Chu and David Corey and Samuel Marguerat and Jurg Bahler's "RNA-seq: from technology to biology". Both articles also give insight into the advantages and disadvantages of gathering and analyzing RNA-Seq data. I found these works to be useful for learning about the way RNA-Seq data is extracted and reasons why it should be used for gene analysis.

Practical usages of gene data analyzed through the lens of gene co-expression networks can be found in the following articles. "A Gene-Coexpression Network for Global Discovery of Conserved Genetic Modules" gives an overview of the usage of GCNs and tries to answer the problem of co-expression of unrelated genes by analyzing co-expressed genes across several different organisms.

CHAPTER II

METHODS AND RESULTS

Making Gene Co-expression Networks

In order to produce GCNs from the mRNA data gathered from The Cancer Genome Atlas, first the correlation value for gene was computed. This was done by first compiling the data from different patients into one table with columns representing each patient and rows representing each gene present in the patient RNA-Seq data. An R script was then used to compute correlation data for every gene against every other gene, producing a table with three columns: the first being a gene, the second being another gene, and the third being their correlation value.

$$r = \frac{\sum(x - m_x)(y - m_y)}{\sqrt{\sum(x - m_x)^2 (y - m_y)^2}}$$

The Pearson correlation coefficient formula used by R to compute correlation values.(*Correlation Test Between Two Variables in R*, n.d.)

Through limiting the output of the R script, the table only contains compared genes with a Pearson correlation value greater than 0.75. NetworkX's readedgelist function allowed for this table to be read and used as an edge list to construct a graph object, allowing us to create the gene co-expression network by reading the R script's output and creating a weighted undirected graph with it. The weights of this graph are the correlation values in the third column of its input.

Deriving the Laplacian Matrix and its Eigenvalues and Eigenvectors

NetworkX comes well equipped for spectral analysis and comes with a function that derives the Laplacian matrix of a graph. The required input is the graph itself, and its return value is a SciPy sparse matrix representing the Laplacian matrix of the given graph. This graph being native to the SciPy language can now have scientific and mathematical operation ran on it through the SciPy package.

$$L = D - A$$

The formula used by NetworkX to construct a Laplacian matrix. L represents the Laplacian matrix of the graph, and D and A represent the degree matrix and adjacency matrix of the graph. Both D and A are square matrices with their rows and columns being the nodes of the graph. D gives the number of connected edges per node across its diagonal and A shows whether or not a node-pair has an edge (F. Chung et al., 2003).

Within the SciPy package are numerous amounts of linear algebra functions. Some of which are compatible with sparse matrix objects of the package, one of which being the eigs function in its sparse.linalg package. The eig function takes the input of an ndarray object, sparse matrix or Linear Operator object and outputs two ndarray objects, the first of which being an array containing the input's eigenvalues and the second being an array of more ndarray objects, each representing an eigenvector. In the eigenvector ndarray, each index of [:, i] represents the index [i] in the eigenvalue ndarray.

$$Ax_i = \lambda_i v_i$$

The equation SciPy solves to find eigenvalues and vectors from the eigs function. A represents the input matrix, λ_i represents the i^{th} eigenvalue and v_i represents the i^{th} eigenvector.

After some analysis of the output of the SciPy package, it was determined that the eigenvalues did not output as desired, and therefore a function made available by the NetworkX package was used: laplacian_spectrum. This function produces a sorted list of eigenvalues for the supplied graph and was used to output the eigenvalues in place of the SciPy function.

$$Lv_i = b * \lambda_i v_i$$

$$v_i{}^H L v_i = \lambda_i$$

$$v_i{}^H b = 1$$

The eigenvalues computed by NetworkX's laplacian_spectrum function must satisfy the above equivalencies where L is the Laplacian matrix of the graph, v_i is the eigenvector of the i^{th} eigenvalue, and b is positive semidefinite.

Table 2 *TCGA Projects from which Data was Processed*

Project ID	Project Name	Primary Site Analyzed	Sample Size	Reference
TCGA-KIRC	Kidney Renal Clear Cell Carcinoma	• Kidneys	20	https://portal.gdc.cancer.gov/projects/TCGA-KIRC
TCGA-LUAD	Colon Adenocarcinoma	• Colon • Rectosigmoid Junction	20	https://portal.gdc.cancer.gov/projects/TCGA-COAD
TCGA-COAD	Lung Adenocarcinoma	• Bronchus • Lung	20	https://portal.gdc.cancer.gov/projects/TCGA-LUAD
TCGA-LUSC	Lung Squamous	• Bronchus • Lung	20	https://portal.gdc.cancer.gov/projects/TCGA-LUSC

	Cell Carcinoma			
TCGA-THCA	Thyroid Carcinoma	• Thyroid Gland	20	https://portal.gdc.cancer.gov/projects/TCGA-THCA

From each project, a set of twenty patients' RNA-Seq data represented by fragments per kilobase million (FPKM) files were collected.

Analysis of Data

Table 3 *Results of Calculations*

Dataset	λ_1	Degree of Zero Eigenvalues	Number of Node Pairs with a Pearson Correlation Coefficient > 0.75
TCGA-KIRC	0.144396908844	9	22,775,833
TCGA-LUAD	0.0388956653568	7	784,859
TCGA-COAD	0.236749027287	4	93,020,861
TCGA-LUSC	0.118303780644	7	16,692,745
TCGA-THCA	0.167892899534	14	33,186,621

In the above table, λ_1 represents the second smallest eigenvalue, with

$$\lambda_0 < \lambda_1 < \lambda_2 < \cdots < \lambda_n$$

The TCGA-LUAD dataset had the smallest output in terms of nodes with a Pearson correlation value greater than 0.75. Conversely, the TCGA-COAD had the largest collection of nodes with a Pearson correlation value greater than 0.75 by far, with the file representing the correlation values nearing five gigabytes, and the next greatest runner up reaching ~1.7 gigabytes.

It is interesting to note that the size of λ_1 directly correlates with the number of node pairs with significant Pearson correlation coefficients (greater than 0.75). This follows logical

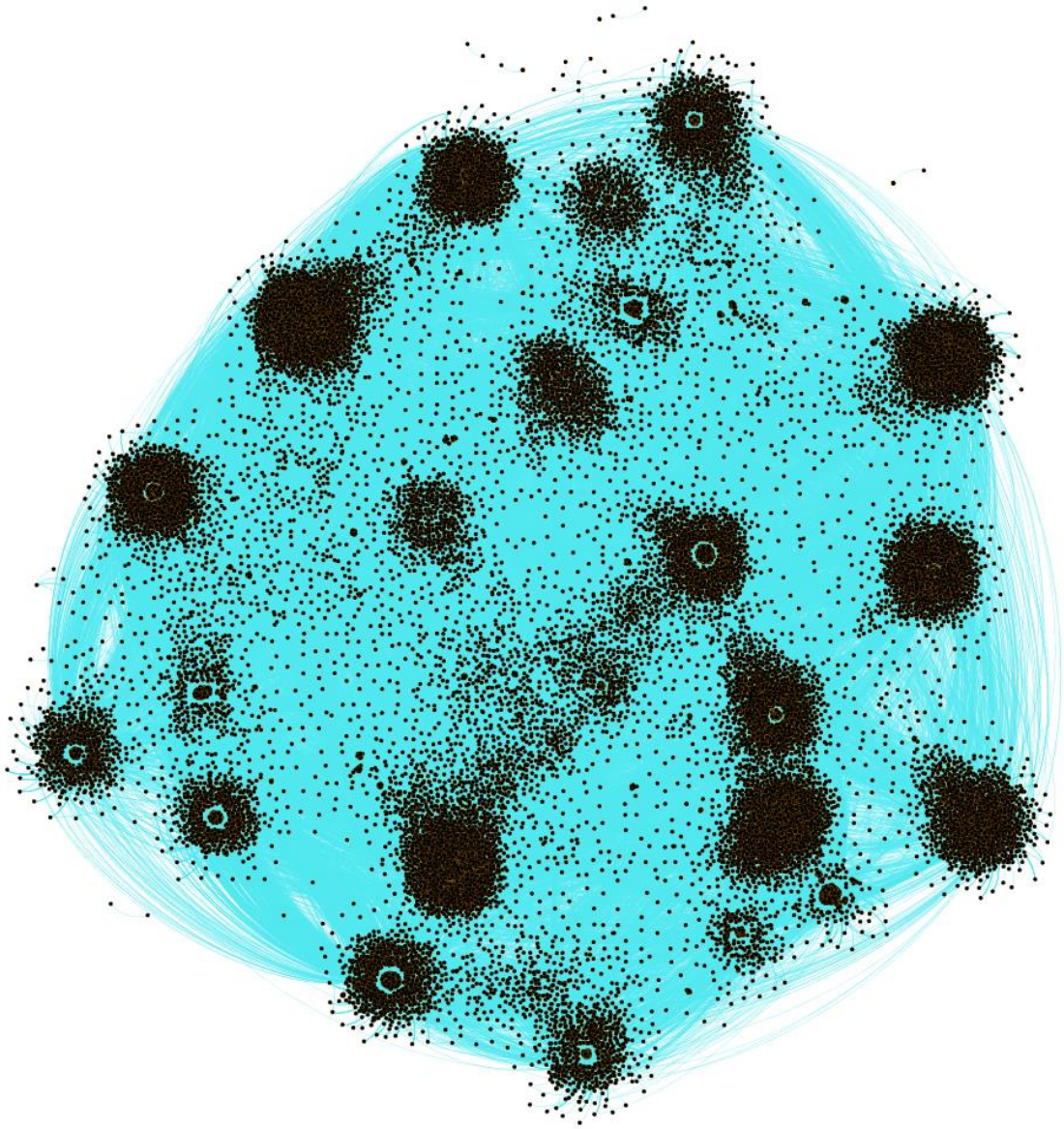
assumptions as the greater the number of node pairs implies the greater connectivity of the graph, meaning a higher algebraic connectivity (represented by λ_1).

Another interesting fact is that of the datasets, the two with the lowest algebraic connectivity are those that originate from the same area of interest. TCGA-LUAD and TCGA-LUSC are projects aimed at the analysis of the gene data of patients with bronchus and lung related cancer growth.

The degree of zero eigenvalues directly equates to the number of connected-components that a graph contains. For instance, TCGA-KIRC's GCN representation contains nine connected components within its graph. In comparing the degree of zero eigenvectors to other data gathered in this research, there seems to be no distinguishable correlation between the number of connected components and the number of node pairs with significant Pearson correlation coefficients.

Visual Representation of Graph

Figure 1. *Visualization of TCGA-LUAD Gene Co-Expression Network*



The above graph was generated using the OpenOrd algorithm provided by the Gephi program. As can be seen, there are several large clusters where nodes (genes) are gathered and many empty areas where the nodes of the graph are more sparsely distributed. These clusters are

strongly connected components of the graph, and it can be assumed that the algebraic connectivity of each of these components would be greater than the average across the graph.

Conclusions

Based on the data found above, there is some merit to the analysis of gene-co-expression networks through the lens of Spectral Graph Theory. While the application above of understanding a GCN network's connectedness through the analysis of its Laplacian matrix's second-smallest eigenvalue only reveals one aspect of the graph and serves to compare the connectedness of different grpahs, there are several other aspects of a graph that can be determined from its Laplacian spectra as was shown in Table 1 from the earlier "Spectral Graph Theory" section. The usage of studies pertaining to graph analysis in Spectral Graph Theory to determine and approximate potentially difficult to find and complex graph properties are bountiful, and these same analyses are very likely applicable in the same way that the algebraic connectivity demonstrated by λ_1 was. The potential for the usage of SGT in the analysis of GCNs may lead to a greater understanding of how genes interact in order to induce the growth of cancer cells.

Future Work

There are several broad-spanning conclusions that were made from this analysis that leads to questions regarding the effectiveness of spectral graph analysis in the examination of gene co-expression networks. A potential future endeavor would be to take a graph and split it down to its individual strongly-connected components, then finding the algebraic connectivity of

each in order to potentially understand the contribution that strongly-connected genes of a cluster may have to cancer growth. Another potential study to conduct is the relevancy of different Laplacian spectrum values to the analysis of GCNs, as the Laplacian spectrum values can give approximations of values that are difficult to accurately compute, and therefore have not been as thoroughly explored in GCN analysis.

There is room to explore the direct relation of algebraic connectivity to cancer growth. A study can be conducted concerning the relationship of the second smallest eigenvalue and other spectral analysis values derived from gene data to other well-understood scientific studies of that data. From that analysis, a more solid understanding of the link between certain properties that can be ascertained from spectral graph analysis and real world effects of genes of cancer patients can be reached.

Stemming from the derivation of the algebraic connectivity of graphs conducted in this analysis, one can question how effective the comparison of GCNs would be through the usage of SGT. Algebraic connectivity is not a concrete value and is more an approximation of other values relating to the connectivity of a graph. The potential usage of approximated values seem strongest in the comparison of graphs, where prior research of the genetic data used to generate said graphs can be compared and analyzed with those values to understand how SGT can play a part in more deeply understanding different GCNs and their origins.

There are other aspects of graphs that may be useful in the analysis of GCNs and can relate to spectral graph analysis, such as the aforementioned diameter and girths of graphs. Computation of these values was not accomplishable due to time constraints within this research, but may give way to interesting results in terms of the relationships of those values to the genetic data that they derive from.

REFERENCES

Chu, Y., & Corey, D. R. (2012). RNA sequencing: Platform selection, experimental design, and data interpretation. *Nucleic Acid Therapeutics*, *22*(4). https://doi.org/10.1089/nat.2012.0367

Chung, F. (1985). Spectral Graph Theory - Chapter 1. In *Linear and Multilinear Algebra* (Vol. 18, Issue 2).

Chung, F., Lu, L., & Vu, V. (2003). Spectra of random graphs with given expected degrees. *Proceedings of the National Academy of Sciences of the United States of America*, *100*(11). https://doi.org/10.1073/pnas.0937490100

Chung, F. R. K. (1995). *Eigenvalues of Graphs BT - Proceedings of the International Congress of Mathematicians* (S. D. Chatterji (Ed.); pp. 1333–1342). Birkhäuser Basel.

Correlation Test Between Two Variables in R. (n.d.). Retrieved April 12, 2021, from http://www.sthda.com/english/wiki/correlation-test-between-two-variables-in-r

Das, K. C. (2004). The Laplacian spectrum of a graph. *Computers and Mathematics with Applications*, *48*(5–6). https://doi.org/10.1016/j.camwa.2004.05.005

Fiedler, M. (1989). Laplacian of graphs and algebraic connectivity. *Banach Center Publications*, *25*(1). https://doi.org/10.4064/-25-1-57-70

Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. *7th Python in Science Conference (SciPy 2008)*.

Marguerat, S., & Bähler, J. (2010). RNA-seq: From technology to biology. In *Cellular and Molecular Life Sciences* (Vol. 67, Issue 4). https://doi.org/10.1007/s00018-009-0180-6

Nica, B. (2018). A Brief Introduction to Spectral Graph Theory. In *A Brief Introduction to Spectral Graph Theory*. https://doi.org/10.4171/188

Pavlopoulos, G. A., Secrier, M., Moschopoulos, C. N., Soldatos, T. G., Kossida, S., Aerts, J., Schneider, R., & Bagos, P. G. (2011). Using graph theory to analyze biological networks. In *BioData Mining* (Vol. 4, Issue 1). https://doi.org/10.1186/1756-0381-4-10

Tomczak, K., Czerwińska, P., & Wiznerowicz, M. (2015). The Cancer Genome Atlas (TCGA): An immeasurable source of knowledge. In *Wspolczesna Onkologia* (Vol. 1A). https://doi.org/10.5114/wo.2014.47136

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … Vázquez-Baeza, Y. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3). https://doi.org/10.1038/s41592-019-0686-2

APPENDIX A

SCRIPTS UTILIZED IN THE PROCESSING AND CREATION OF DATA

consolidateGeneData.py

The following Python code was used to take separated FPKM files and combine them into a table with columns being patients and rows being genes.

```
# The purpose of this code is to take the separated genetic files of the
patients and consolidate them into a one-dimensional table.
# Input: The folder containing all of the raw genetic data of the patients.
# Output: A tab-delimited text file. X-axis will be patients, Y-axis will be
genes, and data will be the quantity of the gene that each patient has.

import os
import argparse

parser = argparse.ArgumentParser()
parser.add_argument('-i', '--input', required=True, help="Input directory")
parser.add_argument('-o', '--output', required=True, help="Output directory")
args = parser.parse_args()

## READ INPUT SECTION

# Getting files through input of directory.
readFilePath = args.input

pathExists = os.path.isdir(readFilePath)
while pathExists == False:
    filePath = input("ERROR:File path entered does not exist. Please enter a
valid file path.\n")
    pathExists = os.path.isdir(readFilePath)

geneFiles = os.listdir(readFilePath)
os.chdir(readFilePath)
# print(geneFiles) - debug print statement

# This section takes the files and creates dictionaries of them, key being
gene, value being gene value.
listOfDicts = []
for gfile in geneFiles:
    dict = {}
    with open(gfile) as f:
        for line in f:
            lineSplit =  line.split("\t")
            lineSplit[1] = lineSplit[1].replace("\n", "")
            dict[lineSplit[0]] = lineSplit[1]
    listOfDicts.append(dict)

# Getting a list of all genes that are present between files.
listOfGenes = [x.keys() for x in listOfDicts]
allGenes = list(set().union(*listOfGenes))
allGenes = list(dict.fromkeys(allGenes))

## WRITE OUTPUT SECTION
```

```
# Get the directory to output the gene table to.
writeFilePath = args.output

# Open the file to be written to.
os.chdir(writeFilePath)
writeFile = open("patientTable.txt", "w+")

# Constructing the first line.
# The first line will contain patient by number, so the first patient is p1,
second is p2, and so forth.
writeFile.write("name\t")
lineString = ""
for i in range(1,len(listOfDicts)):
    lineString += "p" + str(i) + "\t"
lineString += "p" + str(i + 1) + "\t" #need this tab to match with later gene
tabs
lineString += "\n"
writeFile.write(lineString)

# This code block writes each subsequent line past the first line.
# Each line begins with the gene, then tab delimiters between the data for
each patient in order.
for gene in allGenes:
    lineString = str(gene)
    lineString += "\t"
    for x in listOfDicts:
        try:
            lineString += x[gene]
        except:
            lineString += "NULL"
        lineString += "\t"
    lineString += "\n"
    writeFile.write(lineString)
writeFile.close()
```

computeCors.R

The following code is an R script which takes the table generated by the previous section's code and generates the edgelist with the Pearson correlation values acting as weights. This edgelist has columns of a gene, another gene, and their Pearson correlation value.

```
rawData <-
read.table("C:\\Users\\Persons\\Documents\\Research\\OrganizeResearch\\tarfil
es\\TCGA-LUAD\\patientTableLUAD.txt", header = TRUE, sep = "\t")
formattedData <- rawData[,-1]
rownames(formattedData) <- rawData[,1]
transpose <- t(formattedData)

totalGenes <- nrow(formattedData)

filePath <-
"C:\\Users\\Persons\\Documents\\Research\\OrganizeResearch\\testOutput\\RTabl
eLUAD.txt"
file.create(filePath)
outFile <- file(filePath, open = "wt")

for (i in 1:totalGenes)
{
  if(i+1 <= totalGenes){
    for (j in (i+1):totalGenes)
    {
      geneOne <- colnames(transpose)[i]
      geneTwo <- colnames(transpose)[j]
      corVal <- cor(transpose[,i], transpose[,j], use = "complete.obs")
      if(!is.na(corVal) & corVal >= 0.75) {
        lineString <- paste(geneOne, geneTwo, corVal, sep = "\t")
        writeLines(c(lineString), outFile)
      }
    }
  }
}
close(outFile)
```

buildLaplacian.py

The Python script below takes the edgelist created in the computeCors.R R script and outputs the eigenvalues, second smallest eigenvalue's eigenvector, and diameter of the graph.

```
import argparse
import os
import sys

import networkx as nx
import numpy as np
from scipy.sparse.linalg import eigs
from math import isclose
import time

# Sets any NumPy or SciPy print operations to print without truncation.
np.set_printoptions(threshold=sys.maxsize)


def main():
    start_time = time.time()
    try:
        parser = argparse.ArgumentParser()
    except:
        print("ERROR: Argument parsing error.")
        sys.exit(1)

    parser.add_argument("-i", "--input", required=True, help="REQUIRED: The
input file. Must be a weighted edge list.")
    parser.add_argument("-e", "--eig_output", required=True,
                        help="REQUIRED: The output location and name of the
eigenvalue and eigenvector output file.")
    parser.add_argument("-l", "--lap_output", required=False,
                        help="OPTIONAL: The output location and name of the
Laplacian Matrix output file.",
                        default=None)
    parser.add_argument("-a", "--adj_output", required=False,
                        help="OPTIONAL: The output location and name of the
adjacency matrix output file.",
                        default=None)

    try:
        args = parser.parse_args()
        user_input = args.input
        eig_output = args.eig_output
        laplacian_output = args.lap_output
        adjacency_output = args.adj_output
    except ValueError as e:
        print("ERROR:" + str(e))
        sys.exit(1)

    # Open consolidated gene file
    fh = open(user_input, "rb")
```

```
    # Reads gene file as edgelist and creates graph from it
    cor_graph = nx.read_edgelist(fh, nodetype=str, data=(("weight", float),))
    fh.close()

    # Debug statement
    # print(cor_graph)

    # Optional output for the adjacency matrix
    if adjacency_output is not None:
        with open(adjacency_output, "w+") as adj:
            adj_matrix = nx.linalg.graphmatrix.adjacency_matrix(cor_graph,
nodelist=cor_graph.nodes(), weight=None)
            adj_matrix_print = adj_matrix.todense().tolist()
            adj.write(str(adj_matrix_print))

    # Creates scipy sparse matrix object that is the laplacian matrix of the
graph
    lap_matrix = nx.linalg.laplacianmatrix.laplacian_matrix(cor_graph,
nodelist=cor_graph.nodes(),

weight=None).asfptype()  # , weight="weight")

    # Optional output for the Laplacian matrix
    if laplacian_output is not None:
        with open(laplacian_output, "w+") as lap:
            lap_matrix_print = lap_matrix.todense()
            lap.write(str(lap_matrix_print))

    # Use NetworkX laplacian_spectrum function to get all Laplacian
eigenvalues of the graph sorted in ascending order
    eigenvalues = nx.laplacian_spectrum(cor_graph)

    # Loop that finds the index of the second smallest eigenvalue
    smallest eig ind = 0
    for i in range(0, len(eigenvalues)-1):
        if not isclose(eigenvalues[i], 0.0, abs_tol=1e-09):
            smallest_eig_ind = i
            break

    # Use SciPy's sparse library to get the first two eigenvectors of the
graph's Laplacian matrix
    # smallest_eig_ind is one less than the number of eigenvectors to compute
    # TODO: find a way to compute an eigenvector for a known eigenvalue so
that this operation can take much less time
    eigenvectors = [np.real(x) for x in eigs(lap_matrix, which='SM',
k=(smallest_eig_ind+1))[1]]

    # Use NetworkX to find the diameter of the graph
    connected_components = [list(cc) for cc in
nx.connected_components(cor_graph)]
    diameter = 0
    for node_list in connected_components:
        temp_diameter = nx.diameter(cor_graph.subgraph(node_list))
        if temp_diameter > diameter:
```

```
            diameter = temp_diameter

    # Write eigenvalues and vectors to new file
    with open(eig_output, "w+") as eig:
        eig.write("eigenvalues: ")
        for i in range(len(eigenvalues) - 1):
            write_string = str(eigenvalues[i]) + ", "
            eig.write(write_string)
        eig.write(str(eigenvalues[i + 1]))

        eig.write(os.linesep)
        eig.write("diameter of graph: {:.2f}".format(diameter))

        eig.write(os.linesep)
        eig.write("eigenvector of second smallest
eigenvalue({0:0.6f}):\n".format(eigenvalues[smallest_eig_ind]))
        for i in range(len(eigenvectors) - 1):
            write_string = str(eigenvectors[i][smallest_eig_ind]) + ", "
            eig.write(write_string)
        eig.write(str(eigenvectors[i + 1][smallest_eig_ind]))

    print('It took {0:0.1f} seconds to run the program computing eigenvalues
for .'.format(time.time() - start_time) +
          user_input)


if __name__ == "__main__":
    main()
```