REPUBLIC OF TURKEY
ADANA ALPARSLAN TÜRKEŞ SCIENCE AND TECHNOLOGY UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF BIOENGINEERING

DCNA: A TOOL FOR DIFFERENTIAL CO-EXPRESSION NETWORK ANALYSIS OF GENE EXPRESSION DATA

TALİP YASİR DEMİRTAŞ

MASTER OF SCIENCE

ADANA 2021

REPUBLIC OF TURKEY
ADANA ALPARSLAN TÜRKEŞ SCIENCE AND TECHNOLOGY UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF BIOENGINEERING

DCNA: A TOOL FOR DIFFERENTIAL CO-EXPRESSION NETWORK ANALYSIS OF GENE EXPRESSION DATA

TALİP YASİR DEMİRTAŞ
MASTER OF SCIENCE

SUPERVISOR
ASSOC. PROF. DR. ESRA GÖV

ADANA 2021

I hereby declare that all information in this thesis has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all information that is not original to this work.

[Signature]

Talip Yasir DEMİRTAŞ

ABSTRACT

DCNA: A TOOL FOR DIFFERENTIAL CO-EXPRESSION NETWORK ANALYSIS OF GENE EXPRESSION DATA

Talip Yasir DEMİRTAŞ

Department of Bioengineering

Supervisor: Assoc. Prof. Dr. Esra GÖV

July 2021, 68 pages

Recently, network-based approaches have been utilized to provide context-specific network models for complex human diseases. Differential co-expression networks which are built a biological network and analysis of differential co-expression networks via topological assessment offer valuable knowledge regarding biological alterations based on gene expression by comparing two context-specific conditions, such as diseased tissues and healthy tissues. This study aims to design and develop a web-based tool that all researchers can use to make important inferences about pathologies by finding the differential co-expression network between the differentially expressed genes (DEGs) regulated in the context-specific state.

A method was developed based on comparing gene expressions of tissues in two context-specific conditions (i.e., disease and normal samples or chemical applied and not applied samples), and differential-co-expression analysis was performed based on comparing the correlation relationships between the expression levels of genes in both states. Gene clusters (modules) with high interaction with each other in the created network structures were obtained. In this thesis, the web-based tool named DCNA was developed by using the new algorithm belonging to our research group. Codes and supplementary data are available in http://www.github.com/tyasird/differential-coexpression-network-analysis. Besides, the most important point of the study, a user-friendly web-based tool was designed, where researchers can easily analyze gene expression data (microarray & RNA-seq) without wasting time with coding. DCNA is accessible at http://dcna.computationalbiology.org/.

Herewith method, gastric cancer microarray datasets were used as a case study and a cancer-specific correlation network was constructed. Significant gene modules were determined with the help of the DCNA web-based tool. We assert that disease-specific candidate gene modules found by the DCNA web-based tool will help to develop strategies for the prognosis and treatment of the disease.

Keywords: gene correlation networks, bioinformatics tool, gene expression, visualization

ÖZET

DCNA: GEN İFADE VERİLERİYLE KO-EKSPRESYON ANALİZİ YAPAN AĞ TABANLI UYGULAMA

Talip Yasir DEMİRTAŞ

Biyomühendislik Anabilim Dalı

Danışman: Doç. Dr. Esra GÖV

Temmuz 2021, 68 sayfa

Son zamanlarda, karmaşık insan hastalıkları için bağlama özgü ağ modelleri sağlamak için ağ tabanlı yaklaşımlar kullanılmıştır. Biyolojik bir ağ oluşturan ve farklı gen ifade ağlarının topolojik değerlendirme yoluyla analizleri olan karşılaştırmaya dayalı gen ifade ağları, hastalıklı dokular ve sağlıklı dokular gibi bağlama özgü iki durumu karşılaştırarak gen ifadesine dayalı biyolojik değişiklikler hakkında değerli bilgiler sunar. Bu çalışma, araştırmacıların bağlama özgü durumda farklılık gösteren gen ifadelerini (DEG) kullanarak oluşturacağı farklılaşan ortak gen ifadeleri ağları ile patolojiler hakkında önemli çıkarımlar yapmak için kullanabilecekleri bir web aracı tasarlamayı ve geliştirmeyi amaçlamaktadır.

Bağlama özgü durumlardaki (hastalık ve sağlık durumunda veya kimyasal uygulanan ve uygulanmayan durumda) dokuların gen ifadesi değerlerinin karşılaştırılmasına dayanan ve iki farklı durumdaki gen anlatım seviyelerinin korelasyon ilişkilerine dayanarak, farklılaşan ortak gen ifadesi analizlerinin yapıldığı bir metot geliştirildi. Oluşturulan ağ yapılarında birbirleriyle yüksek etkileşime sahip gen kümeleri (modüller) elde edildi. Bu tezde, araştırma grubumuza ait olan yeni bir algoritma kullanılarak, DCNA olarak adlandırılan bir web-tabanlı araç geliştirildi. Kodlar ve ek veriler http://www.github.com/tyasird/differial-coexpression-network-analysis adresinde mevcuttur. Çalışmanın en önemli noktası, araştırmacıların kodlama ile zaman kaybı yaşamadan, gen ekspresyon verilerini kolayca analiz edebilecekleri kullanıcı dostu bir web tabanlı aracın tasarlanmasıdır. Araştırmacılar, DCNA aracına http://dcna.computationalbiology.org/ adresinden erişebilir.

Bu metot ile mide kanseri mikrodizi veri setleri vaka çalışması olarak kullanıldı ve mide kanserine özgü korelasyon ağları oluşturuldu. İleri istatistiksel yöntemler ve algoritmalar ile geliştirilen DCNA web-tabanlı araç sayesinde anlamlı gen modülleri belirlendi.

Geliştirilen web-tabanlı araç ile bulunan hastalığa özgü aday gen modüllerinin, hastalığının prognozu ve tedavisine yönelik stratejiler geliştirilmesine yardımcı olacağını öne sürüyoruz.

Anahtar Kelimeler: gen korelasyon ağları, biyoinformatik araç, gene ifadesi, görselleştirme

nosce te ipsum

*

ACKNOWLEDGEMENTS

This amazing journey started in 2019 with a rainy day in winter. I must be accepted; it was a difficult decision to change of study field. However, some people helped me with handling this hassle.

Assoc. Prof. Dr. Esra Göv, I would like to express my deep and sincere gratitude to my research supervisor for allowing me to do research and providing invaluable guidance throughout this thesis. When I look back to that day, I understand better how lucky I am to have the opportunity to work with her. I am very grateful to her for helping me come up with these ideas well above the level of simplicity and into something concrete. She is the one who made this difficult path easy for me. I deeply thank her for such great suggestions; without her, I would have been lost in the void of ideas.

Dr. Danesh Moradigaravand, an excellent lecturer who accepted me into the working group despite my limited knowledge. I am deeply indebted to him for offering me the internship opportunity in his research group and leading me to work on diverse exciting projects in a limited time frame. His vision, sincerity, and motivation have deeply inspired me. It was a great privilege and honor to work and study under his guidance. I cannot express enough thanks to him for his continued support and encouragement mean so much to me.

Dr. Özgur Cem Erkin, great lecturer that makes biology meaningful for me with its magnificent narrative skills, which makes me combine puzzles and patiently answers all my nonsense questions.

I would like to thank my thesis committee members, Prof. Dr. Muhammet Şakiroğlu and Prof. Dr. Kazım Yalçın Arğa, for all their guidance through this process; your ideas and feedback will intellectually contribute to my development.

There are also several people I want to thank for their spiritual support: Emin Oğuz, Mert Alğan, Fared Alkurdi, Aydın Girici, Gani Yücel, Uğur Taşar.

E. Dila Kurtul, I met her before the MSc interview, and I still remember the ideas she gave me for my interview. We overcame all the difficulties and shared many things and feelings from the first day until today. To my dear friend, who is trying to produce good things within difficult conditions, believe me, your spiritual effort is so much in this thesis. My completion of this thesis would not have been accomplished without your support. I am truly happy to know you, sincerely thank you.

I guess we have come to an end. At the time being, I am feeling the clearance inside me while filling a small academic gap. This adventure that I started in 2019 is about to end in the spring of 2021, and I will start new challenges from here on. With love and respect to the people, I met at this time of my life. See you in another life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

DCNA	: Differential Co-Expression Network Analysis
GCN	: Gene Co-expression Network
IDE	: Integrated development environment
REST-API	: Representational state transfer - Application Programming Interface
ORM	: Object–relational mapping
MVT	: Model View Template
NIGMS	: U.S. National Institute of General Medical Sciences
WWW	: World Wide Web
RDBMS	: Relational Database Management System
SQL	: Structured Query language
HTTP	: Hypertext Transfer Protocol
FTP	: File Transfer Protocol
SSH	: Secure Shell Protocol
IIS	: Internet Information Services
SCC	: Spearman correlation coefficients
PCC	: Pearson correlation coefficients
GUI	: Graphical user interface
UML	: Unified Modeling Language
WSGI	: Web Server Gateway Interface
DNA	: Deoxyribonucleic acid
GC	: Gastric Cancer
GEO	: Gene Expression Omnibus
RMA	: Robust Multi-Array Average
LIMMA	: Linear models for microarray data
DEGs	: Differentially Expressed Genes
JSON	: JavaScript Object Notation
CSS	: Cascading Style Sheets
HTML	: HyperText Markup Language
MCODE	: Molecular Complex Detection

1. INTRODUCTION

1.1. Motivation

There are many comprehensive methods used to understand the depth of the biological system. To understand and analyze this fully mobile structure, it is necessary to examine the structures with a holistic approach. Various approaches can be used to analyze and interpret transcriptome data. Network-based analysis has great importance in understanding complex biological systems with a holistic perspective.

Recently, network-based approaches have been utilized to provide context-specific network models for human complex diseases, and there are some effective approaches about this subject in the scientific literature (Gov & Arga, 2017). Differential co-expression networks which are built a biological network and analyses of differential co-expression networks via topological assessment offer valuable knowledge regarding biological alterations based on gene expression by comparing two context-specific conditions, such as diseased tissues and healthy tissues (Gov, 2020; Hsu et al., 2015; Ideker & Krogan, 2012).

1.2. Goal

Our goal is to develop an algorithm that works in all diseases and create a network to find significant gene modules through this algorithm. The most important point is to design all these steps user-friendly and create an application where researchers can easily analyse gene expression data without wasting time with coding.

We aim to find the algorithm to give the best results as soon as possible without dealing with complex mathematical operations. For this reason, it is necessary to try with many datasets to optimize the algorithm and get meaningful results.

1.3. Contribution

With this thesis, the differential co-expression algorithm in the literature has been developed with python to give more meaningful results. Moreover, it has been brought to the most optimal level by performing tests on it.

Contribution of the study to develop the differential co-expression network analysis model of gene expression data (microarray & RNA-seq). We want to create a web-based tool for determining key gene groups affected by diseases at the transcriptional level with this model.

2. LITERATURE REVIEW

2.1. Gene Expression

The conversion of genetic data in the DNA molecule into a functional product is called a gene. Genes encode proteins; proteins determine cell function. Thus, the operation of the cell is determined by thousands of genes expressed in it (Cook, 2019; Reece et al., 2014). With Microarray Technology, thousands of gene expressions in the cells can be calculated, and simultaneously to see which one is expressed or repressed by comparing control groups. Such high throughput expression data can diagnose clinical conditions and identify prognostic biomarkers (Tarca et al., 2006).

The challenge for researchers is to analyze the main biological mechanisms by making sense of large-scale data. The first step is to provide a vision about the behavior of the differently expressed gene list in the cell mechanism (Khatri & Drăghici, 2005).

2.2. Gene Co-expression Network

Gene expression data requires advanced statistical analysis and complex data mining methods for interpretation. One of these popular methods is the network-based data structure. The advantage of the network-based methods is extracting meaningful information from complex data and presenting the ease of visual interpretation by grouping big data. In addition, networks provide a natural way to model interactions between gene groups. Through network biology methodologies and concepts, gene co-expression network (GCN) has become popular to understand the molecular mechanism behind the diseases or studied research subject. Gene expression correlation analysis in different biological conditions has provided a new perspective about disease formation and progression. In GCN, the nodes represent the genes, and the edges between the nodes indicate that the correlation coefficient score is statistically significant (Gov & Arga, 2017).

According to Dehmer et al., all gene expression values analyzed to generate GCN are compared in binary mode. After the comparison, a similarity score is calculated using statistical models. Gene pairs that exceed the similarity score threshold are used to construct the co-expression network. (Dehmer et al., 2011).

2.3. Differential Co-expression Network

Recently, network-based approaches have been utilized to provide context-specific network models for complex human diseases. Several studies have been reported from our research group, particularly in ovarian cancer (Gov & Arga, 2017). Differential co-expression networks which are built a biological network and analyses of differential co-expression networks via topological assessment offer valuable knowledge regarding biological alterations based on gene expression by comparing two context-specific conditions, such as diseased tissues and healthy tissues (Gov, 2020; Hsu et al., 2015; Ideker & Krogan, 2012). Significant inferences can be made by finding the differential co-expression network between the genes regulated in the disease state (Chowdhury et al., 2020).

Finding gene correlations in human diseases is important for different group studies. Some of these studies; finding diagnostic or prognostic biomarkers, classification of disease and disease stages or monitoring response to treatment, monitoring the occurrence of disease processes (Tarca et al., 2006).

2.4. Differential Co-expression Network Tools

The production and spread of biological data have accelerated from past to present. With this acceleration, different fields of science have created new disciplines by conducting joint studies. Bioinformatics, one of these sciences, is a discipline that uses computer science and statistics to manipulate biological data. Bioinformatics can quickly analyze biological data types that are growing with new technology. It can also help researchers to analyze biological data using computer-based web tools. In order to process and evaluate the gene expression data, statistical-based computational tools need to be developed. (Aamer Mehmood, 2014). Differential co-expression analysis tools in the literature can be seen in Table 2.1. Commonly most of the differential co-expression analysis tools are R-based. For this reason, researchers must be able to use the R programming language to use these tools.

Table 2.1. Differential co-expression network tools in literature

Tool	Application Type	Reference
CrosstalkNet	Web based	(Manem et al., 2018)
dcanr	Not web-based, R package	(Bhuva et al., 2019)
MODA	Not web-based, R package	(Li et al., 2016)
R/EBcoexpress	Not web-based, R package	(Dawson et al., 2012)
diffcoexp	Not web-based, R package	(Wei et al., 2018)
WGCNA	Not web-based, R package	(Langfelder & Horvath, 2008)
CEMiTool	Not web-based, R package	(Russo et al., 2018)
DINGO	Not web-based, R package	(Ha et al., 2015)
BFDCA	Not web-based, R package	(Wang et al., 2017)
DCGL v2. 0	Not web-based, R package	(Yang et al., 2013)
DGCA	Not web-based, R package	(McKenzie et al., 2016)
DICER	Not web-based, R package	(Amar et al., 2013)
DiffCoEx	Not web-based, R package	(Tesson et al., 2010)
CoDiNA	Not web-based, R package	(Morselli Gysi et al., 2020)
BioNetStat	Not web-based, R package	(Jardim et al., 2019)
DiffCorr	Not web-based, R package	(Fukushima, 2013)
MAGIC	Not web-based, Matlab	(Hsiao et al., 2016)

2.5. Literature Review of Used Technologies

The programming languages, packages, and technologies we used for the research are listed below.

2.5.1. Server-side and Clint-side

Server-side means that the program runs on another machine or server, afterward send a response to the user or client. Clients are not able to see a server-side process. The server

compiles a request sent by the client, subsequently shows program output (Figure 2.1). Therefore, important information is stored and processed by server-side languages. Because the client cannot access or change the information. To give an example, assume that registration to the database is done on the client-side rather than the server-side; the client would be able to manipulate the database as they wished (Pollock, 2010). The user who sends a request to the server with software or protocol is called the client. In a web application, the client-side indicates all operations performed and shown on the client computer, such as images and text relevant to the user interface (Pollock, 2010).

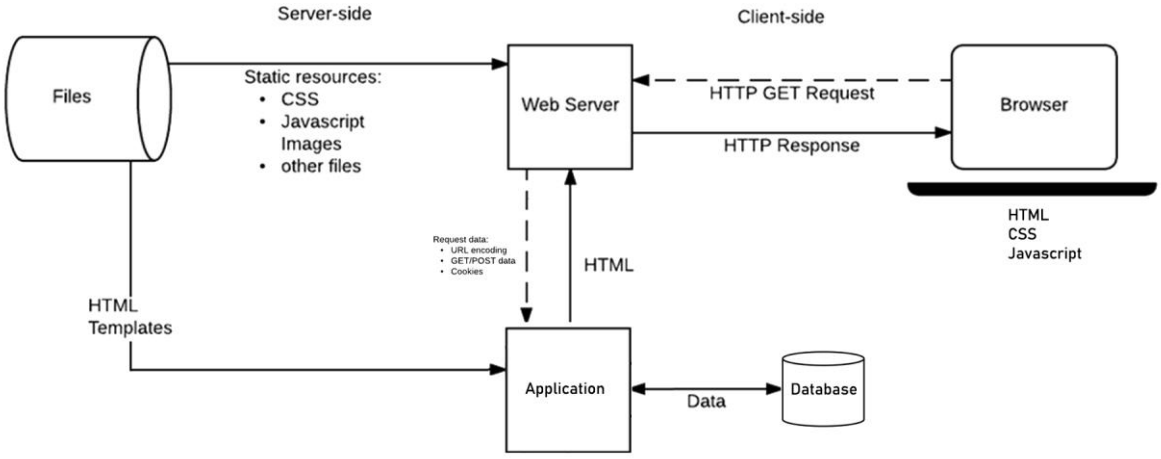


Figure 2.1 Demonstrate how server-side and client-side concepts works (*Introduction to the Server Side*, 2021).

2.5.2. Python Language

Python is an open-source server-side language, and it was released in 1991 by Guido Van Rossum. Python 2 was released in 2000, and Python 3 was released in 2008. There is no significant difference between Python2 and Python3, although they do not continue each other. Python2 and Python3 cannot be run together. It is not ideally backward compatible (The Python Development Team, 2008; Van Rossum & Drake Jr, 1995; van Rossum & The Python Development Team, 2020).

Python is an object-oriented, platform-independent programming language. It can work with various operating systems, such as Windows, Linux, OSX, etc. Python language is easy to learn as well as to write because of its pure structure. Mainly the programming languages use

special characters or braces to specify code blocks, whereas Python uses indentation structure. Code blocks (like function, class, loop, etc.) start with indentation. The code indentations of the same code block should be equal. The indentation gives Python an organized structure and makes it an easy-to-read language (Van Rossum & Drake Jr, 1995; van Rossum & The Python Development Team, 2018, 2020). Indentations are started with four spaces, and they must be a multiple of four. However, generally, coders use tabs due to equality to four-space. The enforcement of indentation in Python makes the code tidier and cleaner.

Besides many packages, it has a sheer variety of loadable packages. In order to import an internal package, It is needed to write "import package_name". However, when we need to install external packages, Python has its package installer called "pip" (Van Rossum & Drake, 2012). Its default installed in Python 2 >=2.7.9 or Python 3 >=3.4 versions. With using pip, it is very easy to install external packages (PyPA, 2020).

Python internal package management has 279,815 libraries in pip (PythonSoftwareFoundation, 2020). Due to the large number and variety of packages, it enables Python to be used in every field. With Python, applications can be developed in various fields such as console applications, desktop applications with a graphical interface, network modeling, web applications, mobile applications, and 3d modeling. Its opulent libraries are the language used in academic studies, scientific calculations, and computational biology (Srinath, 2017).

2.5.3. Python Packages

The list of libraries used with python in this study is given below.

2.5.3.1. py4cytoscape

The py4cytoscape library communicates with the Cytoscape program using the python language and REST API (Figure 2.2). It acts as a bridge between Cytoscape in the python language (Demchak, 2021; Otasek et al., 2019; TheCytoscapeConsortium, 2020).

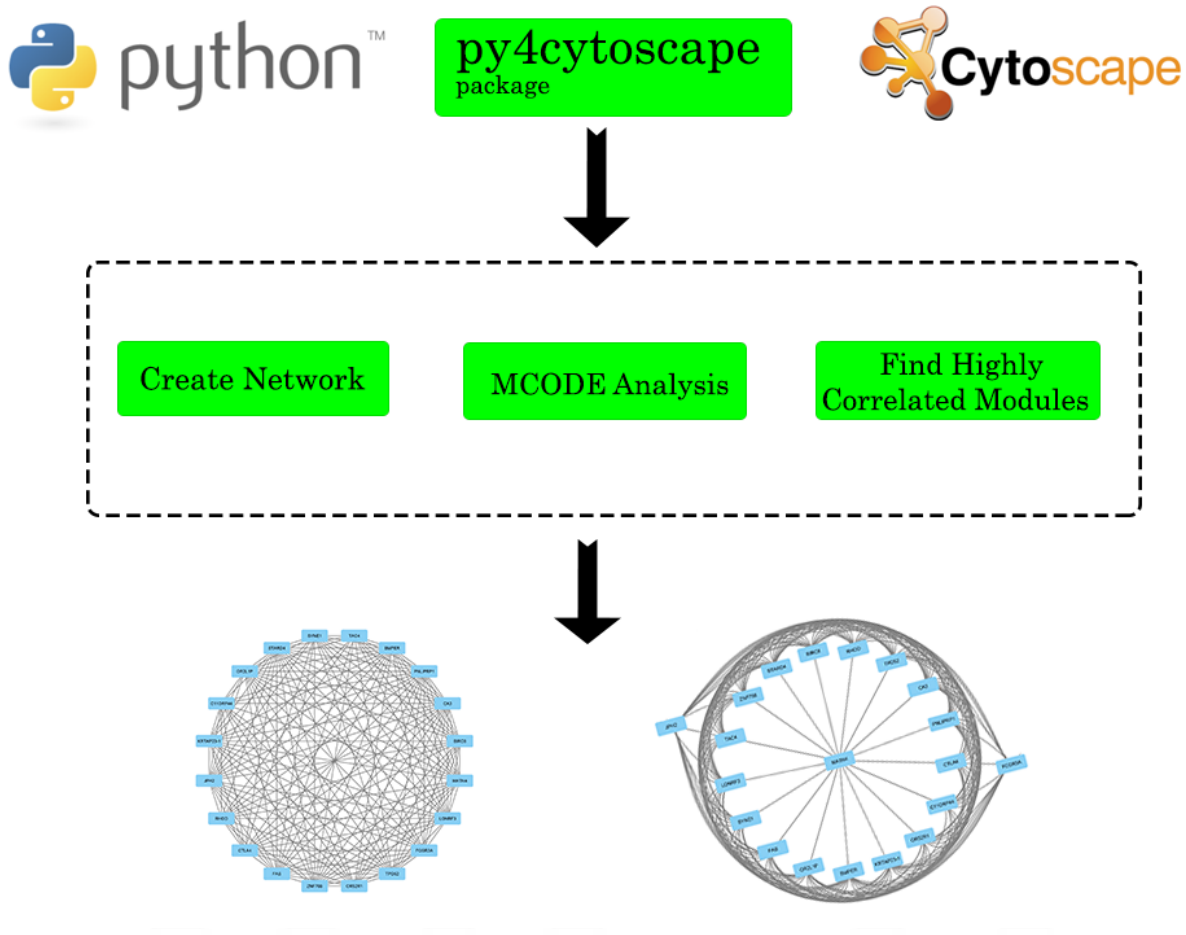


Figure 2.2 The mechanism of use of the py4cytoscape package

There are different libraries to communicate between programming languages and Cytoscape. Although Py2Cytoscape for Python and RCy3 libraries for R are popular, these two libraries have different functions and responses (TheCytoscapeConsortium, 2020).

The goal of the py4cytoscape library is to work on the same principle as RCy3 (R library for Cytoscape). In this way, it will be much easier to adapt a code written in RCy3 to python. There are approximately 270 functions in the library. With the help of these libraries, Cytoscape can do the following operations:

- Load high-throughput data to create biological networks.
- Powerful visualization options.
- Enrichment analysis with integrating other omics data to the networks.
- Different analysis algorithms and applications such as MCODE, clusterMaker2, etc. (Demchak, 2021; TheCytoscapeConsortium, 2020).

2.5.3.2. Pandas

PANDAS is an open-source package created in 2008. It is a decent library for data analysis and manipulation purposes. It enables working with multi-dimensional data to fast, flexible, and easy way. It is used for manipulating and analysing high-throughput data with merging, cleaning, reshaping, slicing, handling missing data, alignment, inserting, and deleting by Data Frame and Series structures functions. Another good thing about the PANDAS package is easy to import and export data. It can import into most tabular formats such as CSV, HTML, Excel, txt, SQL, etc. PANDAS package is often used with other data analysis packages such as scikit-learn, matplotlib, and NumPy due to its flexibility. Data scientists need to clean, adjust, reshape, group, and merge multiple data forms. Pandas is an ideal package for the above purposes (McKinney, 2012).

2.5.3.3. Concurrent Log Handler

It is used to record transactions performed with Python codes. Since applications consist of many layers, they are used to record and detect errors.

2.5.3.4. NumPy

NumPy is an open-source library used for linear algebra, Fourier transform, and matrices in Python. It works with arrays. Library name means Numerical Python, which looks like python lists; however, it is faster than lists. It is used for manipulating multidimensional arrays and matrices for sorting, selecting, reshaping, creating, and fundamental linear algebra operations. NumPy object is called N-Dimensional array (ndarray) (McKinney, 2012; NumPyCommunity, 2020).

2.5.3.5. Django Framework

A wide variety of web libraries are available for running the Python language on the web. According to the JetBrains company's research on developers, the most popular of the libraries are Django and Flask (JetBrains, 2020; Taneja & Gupta, 2014; Van Rossum & FL Drake Jr, 2010).

There are also libraries such as CherryPy, Tornado, web2py, Bottle, aiohttp, which are not popular. Although the main purpose of these libraries is to run python code on the web, they differ from each other in concepts and goals (Ghimere, 2020; Petlovana, 2020; Saabith et al.,

2019). Libraries such as Flask, CherryPy, Hug are designed to publish python code on the web in a simple and fast way. Libraries such as Falcon and Bottle are provided extra performance and reliability (Table 2.2) (Falconframework.org, 2020; Taneja & Gupta, 2014).

Table 2.2. Python web frameworks speed comparisons (Falconframework.org, 2020)

Framework	**speedup**	**req/sec**	**µs/req**
Falcon (2.0.0)	75x	330.676	3.02
Bottle (0.12.16)	37x	162.274	6.16
Werkzeug (0.15.4)	11x	47.395	21.1
Flask (1.0.2)	8x	36.746	27.21
Django (2.2.1)	1x	4.436	225.41
Falcon (2.0.0)	75x	330.676	3.02
Bottle (0.12.16)	37x	162.274	6.16

Django is a library designed for larger and more complex projects, positioned within the full-stack framework category. As a full-stack framework, It includes many useful functions such as authorization, database ORM, and intuitive admin interface. Thanks to these features, developers can quickly do what they want without installing packages and writing much more codes (Demchenko, 2020; Ghimere, 2020).

It was listed the features of Django as follows:

- Big Community: Developers have community support where they can ask questions and help where they have trouble. Django has the largest community support among frameworks in this regard.
- Simple Database Connection: Django is installed with database ORMs. It does not need to deal with third-party ORM packages for database operations (Hourieh, 2008).
- MVT architecture: MVT (model-view-template) architecture is similar to the MVC (model-view-controller) structure. The template is the section where the interface design seen by the users is created. The view is the structure that connects the model,

and template layers, where the algorithms are executed, the operations performed by the users are evaluated and finalized. The model is where the data used in the application is created and stored (Hourieh, 2008).

- Administrator Panel: Django automatically creates an administrator panel interface for CRUD (create-read-update-delete) operations. The ready-made administrator panel is very practical for developers because it saves the trouble of creating an administrator panel. (Ghimere, 2020).
- Wide Usage Area: Django is suitable for developing all kinds of applications with its full-stack and flexible structure. Sites with a large user base such as Instagram, Udemy, Mozilla, and Pinterest also use Django (Dauzon, 2014).
- Security: Security vulnerabilities should be checked in any application contacted with the user. Django performs security checks on their forms by default to prevent popular web vulnerabilities such as SQL Injection and (XSS) Cross-Site Scripting (Hourieh, 2008). Django allows users to eliminate basic security weaknesses with its internal security measures (Ghimere, 2020).

2.5.4. Cytoscape

Cytoscape is a java-based, cross-platform, open-source, and powerful network creation and visualization application funded by the U.S. National Institute of General Medical Sciences (NIGMS) (Cytoscape, 2020b; Smoot et al., 2011). Cytoscape application aims to shows relations between high throughput omic data via creating networks. It provides the opportunity to import large data using various formats such as text, JSON, Excel, etc. (Cytoscape, 2020c; Su et al., 2014). A graph can be created and visualized by uploading genes, proteins, or other molecular-level data (Cytoscape, 2020c; Su et al., 2014). It can be downloaded from http://www.cytoscape.org, and it requires Java 11 to work, and it is installed automatically. It is recommended to use on a 64bit computer (Cytoscape, 2020b).

In addition to Cytoscape's core features also support users in developing plug-ins with the API and documents they prepare. These add-ons created by users can be downloaded from the Cytoscape App Store (https://apps.cytoscape.org) (Cytoscape, 2020b; Su et al., 2014). Each plugin page includes the number of downloads, screenshots, descriptions, version information, and citation information of that plugin. The most downloaded plugins are ClueGO, BiNGO, CluePedia, stringApp, MCODE (Cytoscape, 2020a). In addition, these

applications can be installed automatically from within the Cytoscape software. Cytoscape estimates memory usage for normal users, but when users want to load large data, the maximum and minimum memory limit to be used can be determined (Cytoscape, 2020b; Su et al., 2014).

It shows the quantitative relationship between the data by constructing a graph with node and edge attributes. For instance, if the loaded data is the gene set, a node represents the gene, and an edge represents the link between genes (Cytoscape, 2020c; Su et al., 2014). Cytoscape allows the integration of global data sets into the data uploaded, using the "Public Databases" menu. In the Public Databases section, there are databases such as PSICQUIC, Pathway Commons, STITCH, NDEx (Pablo & Millán, 2013). In addition to all these possibilities, thanks to the add-ons that can be downloaded from the app store. Enrichment analysis can be made for the established networks (Pablo & Millán, 2013). Created Networks can be exported in many extensions, such as JPG, PNG, SVG, and PDF. The recommended export format is PDF. (Cytoscape, 2020b; Su et al., 2014).

Visualization works can be done in two different areas as Layout and Style. Using layout menu, the presentation style can be changed with different network algorithms. The hierarchy will change as the layout makes changes based on different algorithms. In the style menu, settings are made on the appearance, such as edge and node sizes and colors (Cytoscape, 2020b; Su et al., 2014).

2.5.5. Molecular Complex Detection - MCODE

MCODE (Molecular Complex Detection) is a clustering algorithm based on the graph theory, developed to find the connection between protein-protein interaction sections (Bader & Hogue, 2003). Cytoscape plug-in of MCODE algorithm can be downloaded from https://apps.cytoscape.org/apps/mcode. The number of downloads to date is 87033, and the latest version is Version 2.0.0, released on 1 November 2020.

Apart from MCODE, various clustering algorithms can be used for complex molecular identification such as CFinder, CMC, ClusterONE, COPRA, DPClus, MCL, MINE, RNSC, and SPICi (Ou-Yang et al., 2013).

We can define the algorithm as three stages: weighting, prediction, post-processing. MCODE assigns a weight to all nodes by calculating the local density. The high-weight scored node is selected as the center, the nodes connected to it expand outward. The center node is then pulled out (Ou-Yang et al., 2013). By default, MCODE has been set parameters to give the best results for normal users. However, it still allows fine-tuning to improve the network (Pavlovic, 2009). These parameters can be divided into two groups as Scoring Parameters and Finding Parameters.

Scoring Parameters

- Include Loops: MCODE will include self-edges in the calculation of neighborhood density.
- Degree Cutoff: Degree means the number of connection and this parameter adjust minimum degree for scoring of node (Pavlovic, 2009).

Finding Parameters

- Node Score Cutoff: It is an effective parameter during cluster construction. While creating the cluster, new members will only be added if their node score is lower than the set cutoff from the seed node of the cluster. This numerical value corresponds to the percentage value.
- Haircut: If the haircut is allowed, MCODE eliminates all the independently attached nodes.
- Fluff: After activation, MCODE extends cluster centers by a nearby shell, depending on the fluff parameter Node Density Cutoff. MCODE shows better performance when fluffing is turned off (Ou-Yang et al., 2013).
- K-Core: It is a connection limit per node. By reducing its value, users obtain small clusters.
- Max. Depth: Limits the distance of the central node to other members in the clusters. If it is wanted to reduce the number of elements in the fabric, this parameter should be decreased (Pavlovic, 2009).

2.5.6. Hypertext Markup Language

Due to document sharing problems at CERN in 1989, CERN scientist Tim Berners-Lee established the World Wide Web (WWW) system. Afterward, he developed the HTML language to share documents (T. Berners-Lee, 1989). The Hypertext Markup Language

(HTML) is a fundamental markup language used to produce hypertext contents. It can be prepared many HTML contents such as e-mail, news, documentation, media, graphics, etc. (Eltahawi, 2016). Content written in HTML language is interpreted by the browser and displayed to users. HTML language can be written in any text editor. The text data, consisting of tags determined by HTML standards, should be saved as .html or .htm to be read by the browser. The recorded document must be uploaded to a server to be accessible from the web by users. (Sheward & Napier, 2005). In summary, when the user enters an address on the web (example: example.com), the browser interprets the HTML codes and shows the user the rendered page.

HTML documents consist of tags specified by standards. With these tags, they perform the document's formatting, but they are not visible when the page is opened. The browser recognizes the tag between the <> (greater and less than) signs and interprets the values between this tag accordingly. Users see the formatted version of the page without seeing any tag. To give an example, <h1> is known as a title tag, and the strings written between these tags are shown to users in bigger format (<h1> Title </h1>) (T. J. Berners-Lee, 1992; Eltahawi, 2016; Purewal, 2014).

2.5.7. Cascading Style Sheets

There are a limited number of formatting options available in the HTML language used to format basic content. Although HTML shapes the content according to a certain format, it remains very primitive for presentation. Therefore, The Cascading Style Sheets (CSS) language is produced in 1994 to format the style of HTML documents (Kalló, 2015). CSS is a language used to format the interface and make it more readable (York, 2005).

It is used to determine how the HTML tags will appear by assigning individual styles to the generated HTML tags. The style file is stored in the file with the .css extension, and all created styles can be accessed here. CSS codes allow the creation of visually impressive designs by changing the properties of HTML elements such as color, background, length and width, font style, position, and visual effect (Figure 2.3). An effective CSS coding provides high usability on the page (Stark & Jepson, 2012; Takei et al., 2015).

CSS is also a very important point in WEB and mobile programming because 90% of websites today use CSS. CSS is used not only for web applications but also for mobile and desktop applications developed as a hybrid.

The CSS file should be written according to a set of rules and the generated CSS file should be included in the HTML page. Codes are given a class name and written in braces (Mazinanian et al., 2014).

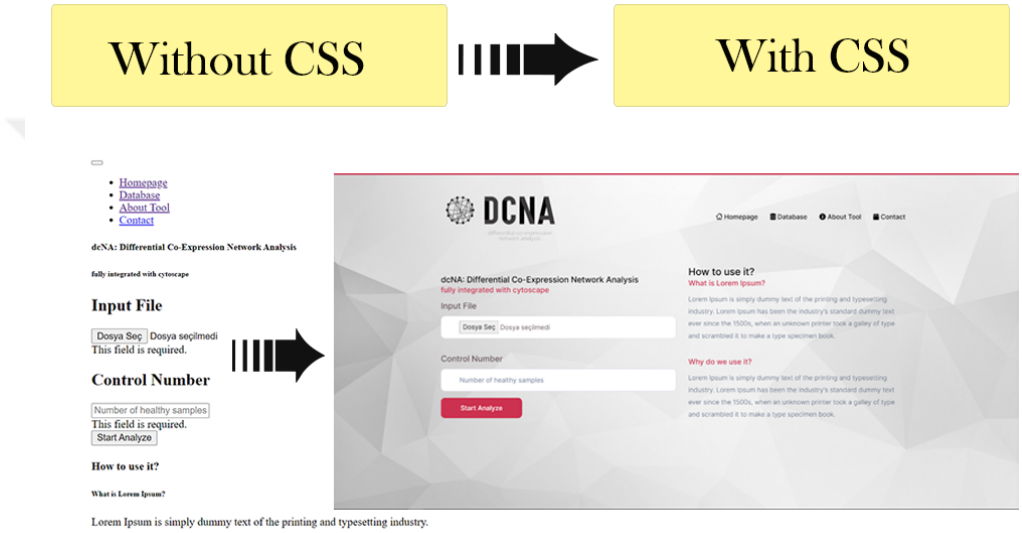


Figure 2.3 Web page comparison with CSS and without CSS

The CSS file is consisting of style declarations in selectors. There can be multiple declarations in selectors. For instance, class1 selector refers to create red colored content. Also, to call the selector in HTML file, we must specify in HTML tags in class attribute (Figure 2.4).

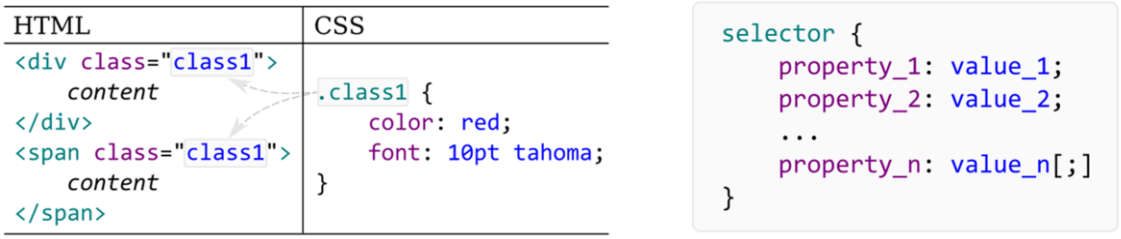


Figure 2.4 CSS rules (Mazinanian et al., 2014)

2.5.8. Bootstrap Framework

Bootstrap is a robust open-source CSS framework that helps to create and design HTML websites. It is made by Twitter and very popular in the CSS framework list. The advantage of the Twitter Bootstrap framework is that it provides prepared classes and a grid system. Namely, we do not have to create all declarations and selectors one by one. It is needed to be imported into HTML code to start use (Matsinopoulos, 2020).

Through the widespread use of the Internet and technology, it has become possible to access websites from many different devices. However, different devices mean different resolutions and different designs. Twitter Bootstrap framework is used to develop a suitable design for every resolution. It has own grid and resolution system (Shenoy et al., 2014).

Bootstrap has a responsive grid structure. As a result of this, the contents are automatically adjusted according to the resolution. Using Bootstrap, a website designed on a computer can be accessed from a smartphone or tablet without design problems. Twitter bootstrap has many free tools and layouts such as buttons, dropdowns, forms, tables, and other useful components. For instance, when we need to create a contact form in front-end design, we can access and use on the bootstrap website different designed form components (Shah, 2015).

2.5.9. JavaScript Language

JavaScript is a fluent object-oriented programming language interpreted by the browser. It is created in 1995 as a client-side language to interact with users and pages (Pollock, 2013).
However, it has started to be used in many areas, showing a great change in the past. At this time, JavaScript can be used by server-side language (Ferguson, 2019). Although used for 2d enhancements once it was first released, nowadays it has great developments. It is used for mobile, game, 3d, application development, and websites (Dirksen, 2013).

The JavaScript language does not need a compiler. The scripts we write are automatically interpreted by the browser when the page is loaded and the result is displayed to the user.
It is used to add dynamism to web pages and interact between the page and the user. JavaScript can access and modify all HTML and CSS elements on the page. It is needed to write codes among <script> tags in order to use JavaScript. For instance, <script>

alert(‘error!’);</script> code shows users to error alert (Maria, 2018; Niederst et al., 2012; Osmani, 2012; Pollock, 2013).

As seen in the study by StackOverflow in 2020, it is the most popular programming language due to its ability (Figure 2.5).

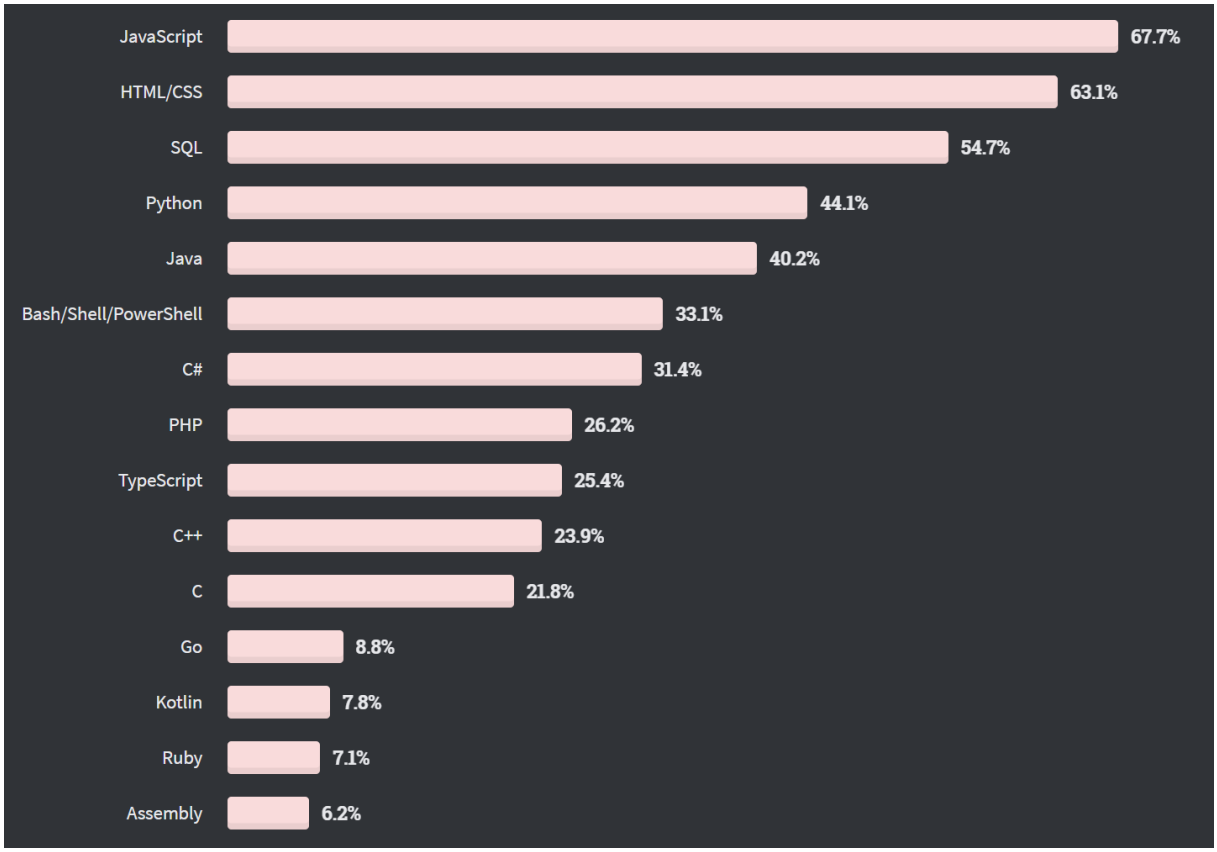


Figure 2.5 StackOverflow most popular technologies research (Stackoverflow.com, 2020)

The main reason is that JavaScript can affect many areas (Ciman et al., 2014). Also, it is possible to create desktop applications (Maria, 2018). A language used in such a wide range of areas is inevitably popular.

2.5.10. jQuery Library

jQuery is a decent, fleet, and easy to read JavaScript framework to handle HTML documents. The main point about jQuery, it is simplifying to JavaScript actions. It is needed to import jQuery library in HTML file for use jQuery functions. Since jQuery is a JavaScript library, it

must be written between script tags. jQuery syntax starts with $ character. For instance, $("button").hide() code is helps to hide button element (Ahmed, 2014).

2.5.11. MySQL

We can define the database as the place where data collection is stored on an orderly basis. However, the data must be controllable by a software called a database management system (DBMS). The basic mission of the DBMS is to store data in an organized way, manipulate and query in data (Widenius et al., 2002).

There are many Database Management systems available such as MySQL, PostgreSQL, Microsoft Access, Oracle, Microsoft SQL Server (Foster & Godbole, 2016). Most RDBMS use SQL (Structured Query language) to communicate with data. Syntax in SQL commands may differ slightly between RDMSs, but generally, the way it works is the same (Schuurman, 2011).

MySQL is the world most popular, open-source, relational database management system and written in C and C++. It uses SQL (Structured query language) for handling data. SQL has general commands to data definition, data manipulation, data transaction, and control commands (Foster & Godbole, 2016).

Relational Database Management System consists of tables and different data types in tables such as string, integer, boolean, and DateTime. The RDMS table consist of rows, and every row refers to a record with unique identity. The rows are divided into columns to group different types of data. In addition, RDBMS provides communication between different tables (Letkowski, 2014).

Important features of MySQL are listed below:

- MySQL has a very small size and easy to install withal it is a powerful capability. It is not needed to make long time settings to install. In Linux operation systems, MySQL installation can be done automatically with a single command (Tahaghoghi & Williams, 2006).
- Using SQL as a communication language with data. SQL is standard in most RDBMS (Tahaghoghi & Williams, 2006).

- The most important thing is that MySQL is free and open source. We do not need to pay anything as users.
- MySQL can support and handle big data.
- MySQL can be used with most programming languages. For instance, if it is wanted to use MySQL with python, it should be just needed to import the python-MySQL library (Tahaghoghi & Williams, 2006).
- It is growing and improving over time.

2.5.12. World Wide Web

The web can be defined as a constantly growing environment that contains files, documents, media, and resources that users can access. There are two structures within its architecture. The section where the files or documents are stored is called the server, and the person accessing them is called the client (Taneja & Gupta, 2014).

2.5.13. Server

A server is hardware with an operating system installed on it, connected to a network, and accessible by other computers. The server can be connected to the internet or can only communicate with computers connected to its network as a closed network. The prepared application must be deployed to the server for accessions by users. Without deployment to the server, users will not be able to access the application. Therefore, designed and coded application files need to upload server to serve other users (Gupta et al., 2017; McMahon, 2003; Shen et al., 2010).

2.5.14. Hypertext Transfer Protocol

The client can connect or communicate with the server using different ways. HTTP is the most popular form of communication. Different protocols are used to communicate with the server other than HTTP (Comer & Stevens, 1993). Standard protocols are used for different operations such as FTP for file transfer, SMTP for mail transmission, SSH for remote connection, etc. (Comer & Stevens, 1993). These application services must be set up to detect and process the different operations performed by the users on the server-side. These services have standard port numbers, for example, mail service needs to be installed on the server for mail communication, and mail service communicates using port 25 by default. On the other

hand, an HTTP service can be installed for web communication, and thus, this server can be accessed over the web. HTTP service uses port 80 by default (Comer & Stevens, 1993).

When a client enters a domain address through a browser, it is directed to an IP address. The IP address is directed to a machine that is currently open and connected to the internet network. If an HTTP server is installed on this machine, the server receives the request and responds (Taneja & Gupta, 2014).

Various applications can be used as an HTTP server. IIS, Apache, Nginx are popular web services (Jader et al., 2019).

2.5.15. Nginx

Nginx is a free, fast, and easy to use web server made by Igor Sysoev in 2004. It shares the top rank with apache in the web server market. Studies in the literature show that the response time is much faster than the Apache web server, and Nginx is more successful in memory usage (Kunda et al., 2010; Nguyen, 2017).

Nginx, besides being easy to install, it can operate with high performance even without configuration. If we are talking about web servers, we expect them to be constantly open and accessible. Nginx can remain open and serve even during update processes (Soni, 2016).

2.5.16. Gunicorn

Gunicorn is a bridge to connect python codes with an HTTP server (Gunicorn.org, 2020).

2.6. A case study: Gastric Cancer

2.6.1. Gastric Cancer

Gastric cancer (GC) is one of the prevalent malignant cancers in the world (Shi et al., 2014). The development and progression of GC initiate in epithelial cells of stomach mucosa (Shi et al., 2014). In 2018, it was reported that about 1,000,000 new GC cases were where about 783,000 deaths, which leading GC as the fifth-highest prevalent cancer type. However, considering mortality rate, it ranked third highest mortality in cancer (Bray et al., 2018). The prognostic outcome is very low since most GC cases are diagnosed in the advanced stage of the tumor at the time of diagnosis, making the treatment opportunities limited (Van Cutsem et

al., 2016). Therefore, detecting a prognostic gene signature for patients with GC may improve the patients' survival outcomes.

Despite important findings from these systems biology studies, the mechanism of GC is still unclear, and new biomarker candidates are required. Our approach is significantly different from the previous studies where we have considered differential co-expression networks in GC and healthy tissues to provide genes that may behave collectively in modules and assess those modules for prognostic performance.

The present study investigated gastric tumors and normal gastric tissues, and three datasets obtained from different population groups were integrated. Microarray datasets were statistically analyzed, and differentially expressed genes (DEGs) were identified by compared tumor vs. control tissues. Mutual DEGs were identified, and the co-expression network of mutual DEGs was constructed and altered modules were determined using tumor and control tissues. The prognostic and diagnostic performance of these modules were evaluated. The presented altered correlation sub-networks here will be a crucial resource for the understanding of gastric cancer pathogenesis.

3. MATERIALS AND METHODS

3.1. Algorithm

An integrative differential co-expression network analysis methodology (Gov & Arga, 2017) was applied to transcriptome datasets to identify differentially co-expressed networks. Differentially expressed genes (DEGs) were identified via statistical analysis, and mutual DEGs were used in the co-expression network analysis. The mean level of expression of each mutual DEGs was calculated, and Z score normalization was applied.

$$Z = \frac{x - \mu}{\sigma}$$

Z = standard score

x = observed value

μ = mean of the gene expression values

σ = standard deviation of the gene expression values

Normal distribution of gene expression values was checked for diseased and healthy tissues separately with the D'Agostino and Pearson's normality test (D'agostino et al., 1990). If the normalized values are normally distributed, Pearson correlation coefficients (PCC) were calculated. Conversely, if values are not normally distributed, Spearman correlation coefficients (SCC) were calculated.

$$p = 1 - \frac{6\Sigma\, d_i^2}{n(n^2 - 1)} \qquad r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \Sigma(y_i - \bar{y})^2}}$$

p = Spearman's rank correlation coefficient

d_i^2 = difference between the two ranks of each observation

n = number of observations

r = Pearson correlation coefficient

x_i = values of the x-variable in a sample

$\bar{x}$ = mean of the values of the x-variable

y_i = values of the y-variable in a sample

$\bar{y}$ = mean of the values of the y-variable

Subsequently, we calculated the mean and standard deviation of coefficient scores of the entire square matrix to reach the p-critic value (confidence interval). The PCC/SCC cut-off was determined (p-value < 0.05) to identify statistically significant pairwise correlations of mutual DEGs.

Confidence Interval:

$$pcritic = \mu + 1.96 * \sigma$$

μ = mean of the correlation coefficient scores

σ = standard deviation of the correlation coefficient scores

We reached significant gene pairs by filtering coefficient scores outside of the confidence interval. To identify key gene modules of the co-expression network, the MCODE plugin (Saito et al., 2012) of the Cytoscape (Cline et al., 2007) was used with the help of the py4cytoscape package (Demchak, 2021). Modules with significantly altered topological metrics (i.e., network density) were considered altered modules compared to the PCC or SCC values of tumor and control samples. Two threshold conditions (using parameter y') were defined to classify genes and express the significance of the altered expression profiles between disease and control states.

i) Genes show significantly high co-expression in the diseased samples but no significant co-expression in the control samples.
ii) Genes show significantly high co-expression in both diseased and control samples but change the co-expression direction in the disease and control samples:

$$y' = |\ PCC_d - PCC_h\ | \geq 0.5$$

3.2. Implementation of Algorithm

The algorithm was written with python and improvements have been made to make it more stable. Gene expression data was read with the help of PANDAS library (McKinney, 2012). Then the gene names and string data were deleted to perform mathematical operations. Each column represents the gene expression value of the same sample of different genes, and each line shows the gene expression value of different samples of the same gene. It was found the

average and standard deviation values on all columns. Subsequently, It was subtracted the average from each expression value and divide this value into standard deviation. In this way, it was reached the normalized gene expression value. Then, disease and healthy gene expression values were separated to perform the Pearson correlation coefficient test for each group separately. Thus, it was found a square matrix with correlation coefficients. The mean and standard deviation were found of the entire square matrix and reached the p-critic value (confidence interval). All these mathematical and statistical operations were done using the NUMPY (McKinney, 2012) and PANDAS library. The obtained data was written into an excel file for the researchers to review.

3.3. Back-end and Front-end Development

Python was chosen as the programming language for back-end development and coding of the algorithm. The reason for this is Python's flexible structure, comprehensive library support, and widespread use of scientific, statistical, and computational engineering fields.

Django Web Framework was chosen to transfer the coded algorithm to the web environment and reach all users. The reason for selecting Django is its MVT (Model View Template) architecture and the convenience it provides with its internal library variety.

Cytoscape program and MCODE library were used for networking and module finding with the algorithm output. This process has been done with the py4cytoscape library, which communicates the Python language and the Cytoscape program.

Front-end is necessary for codes and algorithms written in any language to be used by the end-user. All operations related to the design, such as the fonts, colors, design of the pages, text size used in the interface, were carried out. HTML5, CSS, JavaScript languages were used for front-end development. Bootstrap was chosen as the CSS library. The most important reason is that it provides the opportunity to create a responsive and modern design. jQuery was chosen as the JavaScript library.

3.4. Workflow

- Upload raw gene-expression data and web-based tool separates into disease and control.
- Find significant genes after normalization, correlation, and statistical analysis methods.
- In the background, The MCODE plugin of the Cytoscape plugin was used to create a correlation network to find highly correlated gene groups (Figure 3.1).

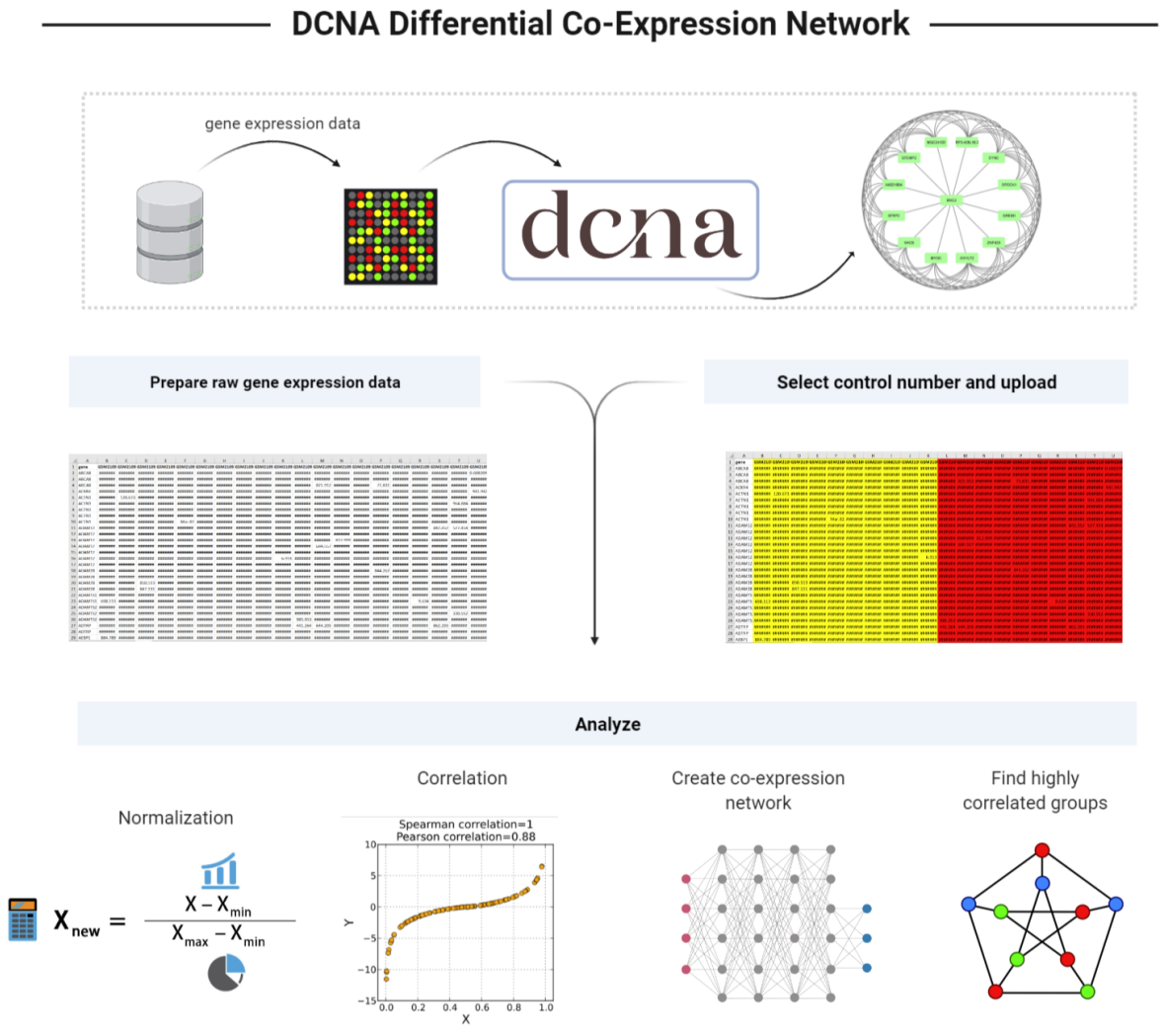


Figure 3.1 Workflow of DCNA

3.5. Usage

- DCNA needs an excel file with gene expression data to start the analysis. The excel file prepared by the researchers which contain gene expression values of two context will be selected from the DCNA homepage, and the analysis will be started. Users must enter two form inputs to use the DCNA tool. The first input is an excel file with gene co-expression data inside two different conditions. The second input must be the number that the user specifies the number of control samples (Figure 3.2).
- After uploading and specifying the control number, the tool makes correlation analysis and redirects the user to the result page. On this page, users can download correlated genes with a correlation score.
- Afterward, users can create a gene co-expression network with these scores. When the user clicks the analyze network button, the tool will run the Cytoscape application and return with clusters which means highly correlated gene groups for the selected condition.
- On the network result page, users can reach scores and network density for each cluster.

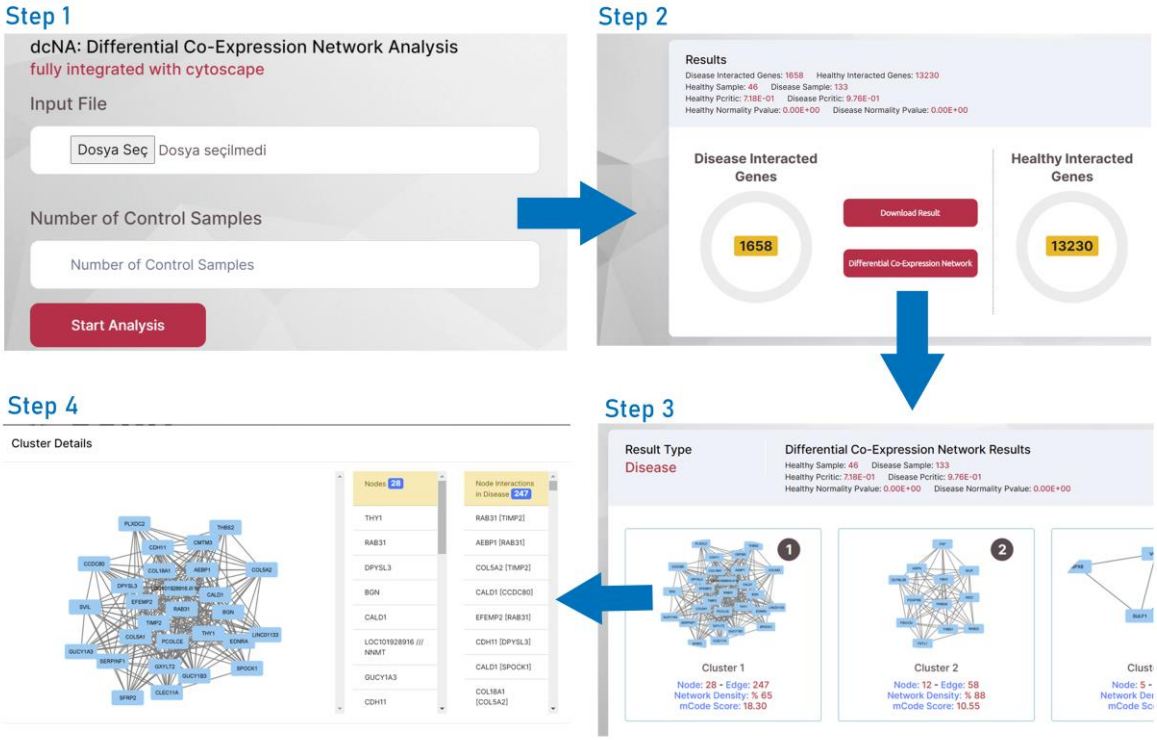


Figure 3.2 Usage of DCNA

3.6. Deployment of Web-Based Tool

For the deployment process, there are some applications and software's need to be used.

3.6.1. Server Installation

Ubuntu operating system selected for server. The server configuration created is as follows:

- Memory/Ram: 4 GB
- Hard Disk: 25 GB
- CPU: 2 Cores Shared Cpu
- Bandwidth: 4 TB

3.6.2. Secure Shell Protocol

The necessary applications were installed by connecting to the server via Secure Shell Protocol (SSH). The Putty program is used for SSH access to the server via windows. The default access port for SSH is 22.

3.6.3. Nginx

It is necessary to install Webserver such as Apache, Nginx, Microsoft IIS for remote access to server / IP address via browser. After logging into the server with SSH, Nginx is installed as the HTTP server. The HTTP server allows users to access the server using the HTTP protocol over the browser. The HTTP default port is 8080. After Nginx is installed, it is necessary to configure Nginx to determine which folder on the server users will be directed to when they enter the server IP Address using a browser. By opening a folder in the server, the web-based tool files are loaded into this folder. By configuring Nginx, the IP Address is redirected to this folder. The Nginx commands used, and the configuration file is given below.

~ ssh@root

```
sudo apt install nginx
systemctl start nginx
sudo mkdir -p /var/www/dcna/html
sudo chown -R $USER:$USER /var/www/dcna/html
sudo chmod -R 755 /var/www/dcna/html
nano /var/www/dcna/html/index.html
sudo nano /etc/nginx/sites-available/dcna
ln -s /etc/nginx/sites-available/dcna.ini /etc/nginx/sites-enabled/
sudo systemctl restart nginx
```

~/etc/nginx/sites-available/dcna.ini

```
server {
        listen 80;
        listen [::]: 80;

        root /var/www/dcna/html;
        index index.html index.htm index.nginx-debian.html;
        server_name dcna.computationalbiology.org;
        location / {
                try_files $uri $uri/ = 404;
        }
```

3.6.4. File Transfer Protocol

File Transfer Protocol (FTP) provides the Transfer of files between a client and server on the network. FTP sends data unencrypted. All FTP transmissions are sent as regular text, including usernames, passwords, commands, and the files themselves. However, SFTP is safe. By default, SFTP is already available on all Linux. The files were uploaded to the server using the SFTP protocol with the FileZilla program. By default, SFTP is already available on all Linux SFTP protocol default port is 22.

3.6.5. Django Framework

Django installation, database migration, admin user creation processes have been done by applying the following commands.

~ ssh@root

```
pip install -r requirements.txt Django ~=3.1.1
django-admin.exe startproject config .
python manage.py startapp app
python manage.py migrate
python manage.py makemigrations app
python manage.py migrate app
python manage.py createsuperuser
python manage.py runserver
```

After installing Django, the following configuration has been made for static files to work with the Django framework.

config/settings.py

```
STATIC_URL = '/static/'
STATICFILES_DIRS = [
    os.path.join (BASE_DIR, "static"),
    '/var/www/dcna/static/',
MEDIA_ROOT = os.path.join (BASE_DIR, 'uploads')
MEDIA_URL = '/uploads/'
```

3.6.6. Database

The web-based tool uses MySQL (relational database management system) as its database. Installation commands and configuration information are given below.

~ ssh@root

```
sudo apt install mysql-server
sudo mysql_secure_installation
sudo service mysqld start
pip install mysqlclient
```

~ nano config/settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'OPTIONS': {
            'database': 'dcna',
            'user': 'root',
            'password': '',
        }
    }
}
```

There are two tables named app_diffcoexpression and app_network in the database. These two tables are related to each other. Table schema information is shared below (Figure 3.3).

- app_diffcoexpression: In this table, input data, output data, p-critic values, normality p-values, health and disease sample numbers are stored.
- app_network: In this table, cluster information of differential co-expression results is stored. These clusters are created with the MCODE plugin.

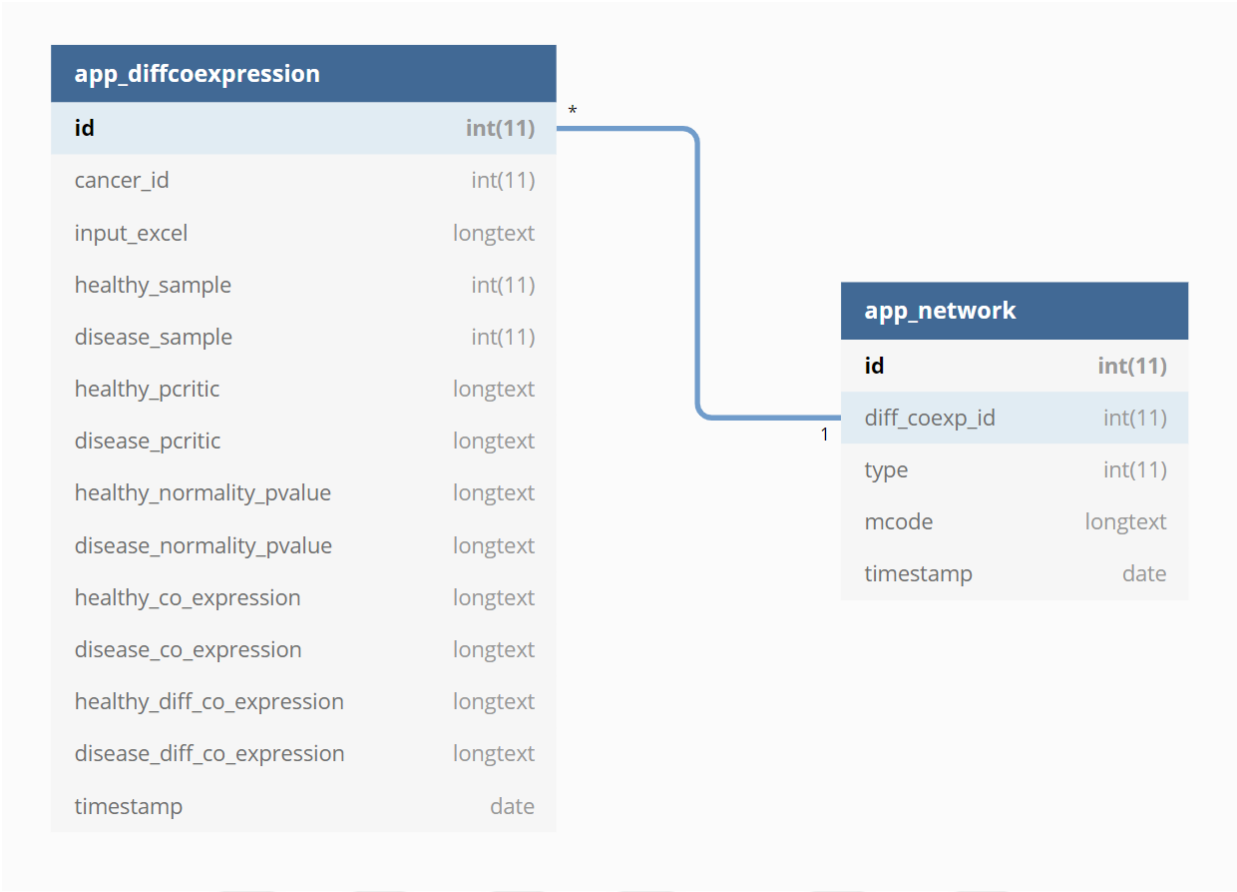


Figure 3.3 UML diagram of DCNA database

3.6.7. Gunicorn as a Web Server Gateway Interface

Python code can run on Ubuntu by default. However, WSGI (Web Server Gateway Interface) is used for python codes to communicate with the web interface. There are libraries like uWSGI and Gunicorn for this. Gunicorn is preferred for the web-based tool. Gunicorn has been installed using the Python library management system, and the related setup commands and configuration files are listed below.

~ ssh@root

```
pip install gunicorn
nano conf/gunicorn_config.py
gunicorn -c conf / gunicorn_config.py config.wsgi
```

gunicorn.conf

```
command = '/var /www/dcna/env/bin/gunicorn'
pythonpath = '/var/www/dcna/env/bin'
bind = ('159.65.204.133:8000']
#bind = 'unix: /var/www/dcna/tool.sock'

workers = 3
```

```
accesslog = '/var/www/dcna/logs/access.log'
errorlog = '/var/www/dcna/logs/error.log'
timeout = 900
worker_class = 'gevent'
log_level = 'debug'
```

3.6.8. Cytoscape

Cytoscape installation and configuration on the server was done with the following commands.

~ ssh@root

```
wget https://github.com/cytoscape/cytoscape/releases/download/3.8.2/Cytoscape_3_8_2_unix.sh
chmod u + x Cytoscape_3_8_2_unix.sh
sudo sh Cytoscape_3_8_2_unix.sh
apt install default-jre
apt install openjdk-11-jre-headless
sudo apt install default-jre
sudo sh Cytoscape_3_8_2_unix.sh -q
Cytoscape &
```

3.7. Project Files

Project file information is shared below (Figure 3.4).

- Migrations: This folder is used for database migrations. Database tables is created for each model file.
- Config: This folder includes two important files.
- settings.py: In this file, we can specify global settings about the project such as; database configuration, template folder, application installation, debug management, time settings, language settings, upload folder destination, etc.
- Urls.py: In this file, we can specify our URLs.
- Env: This folder includes third-party python packages which are necessary for the project.
- Helpers: This folder includes custom helper files for preparation for include to project. It is not compulsory, however.
- Logs: This folder includes log files.
- Static: This folder includes asset files such as images, CSS, fonts, etc.
- Templates: This folder includes HTML template files.
- Tools: This folder includes two python scripts that are used for tool algorithms.

- dcna_tool.py: This file is used for gene expression analysis with input files given by users. Output file becomes binary excel file which shows genes relations.
- network_tool.py: This file is used for network analysis to create a differential co-expression network.
- Uploads: This folder includes uploaded input and output files.

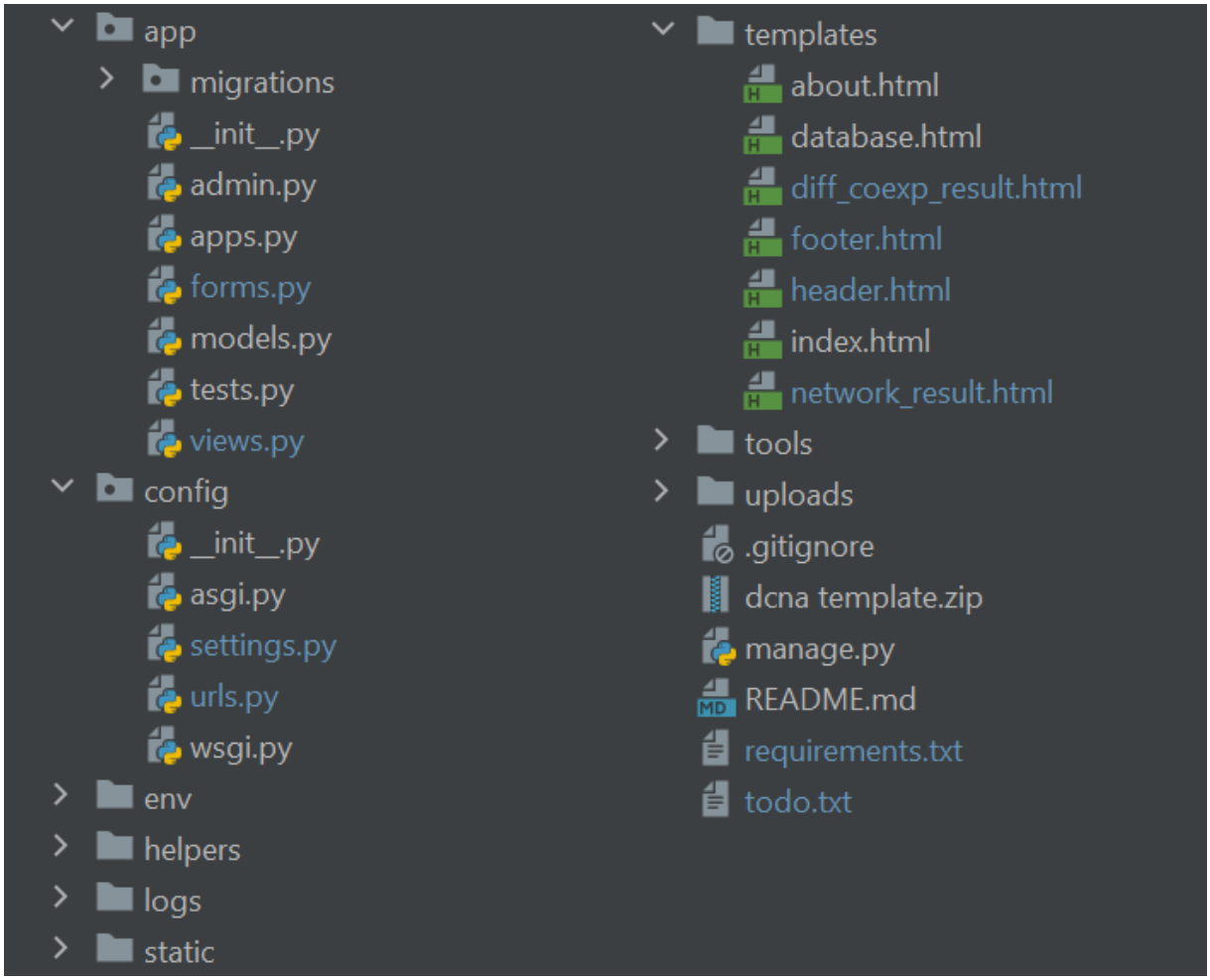


Figure 3.4 Project files of DCNA

3.8. As a Case Study: Gastric Cancer

3.8.1. Gene Expression Datasets

The raw data of gene expression datasets of gastric cancer (GC) from a total of three studies (with accession no: GSE19826, GSE54129, and GSE GSE79973) have been obtained from Gene Expression Omnibus (GEO) (Barrett et al., 2012) and analyzed to characterize gene expression profiles of GC. The samples in these datasets were taken from the Affymetrix gene

chip microarray platform (Figure 4.5A). A total of 179 samples were explored, including 133 GC and 46 normal gastric tissue samples.

3.8.2. Identification of Differentially Expressed Genes

To identify differentially expressed genes (DEGs), firstly, each dataset was normalized through the Robust Multi-Array Average (RMA) expression measure (Bolstad et al., 2003) as implemented in the "affy" package (Gautier et al., 2004) in R language (Gentleman et al., 2006). DEGs in each dataset were identified from the normalized log-expression values using the multiple testing options of LIMMA (linear models for microarray data) (Smyth, 2005). Benjamini-Hochberg's method was used to control the false discovery rate. A p-value threshold of 0.05 with a fold-change cut-off of 2 was used to determine the statistical significance of DEGs.

3.8.3. DCNA Web-Based Tool Usage

DCNA web-based tool was used due to the creation of a differential co-expression network and identify significant modules. An input file consisting of expression values has been prepared to upload to the DCNA to start the analysis. For the preparation of the input file, Expression profiles of mutual DEGs of three datasets were separated as tumor and control samples. Two new data lists (consisting of 133 tumors and 46 control samples) were formed using these common DEGs expressions.

3.8.4. Prognostic and Diagnostic Performance Analysis of Gene Modules

The prognostic performances of module genes were evaluated using independent gastric cancer datasets from TCGA. Cox proportional hazards regression analysis was performed via the SurvExpress validation tool (Aguirre-Gamboa et al., 2013). In SurvExpress, the gastric cancer adenomocarcinoma (STAD) and stomach and esophagus adenocarcinoma (STES) were grouped into low- and high-risk groups according to their prognostic index. The prognostic capabilities of the resultant genes were identified via Kaplan-Meier (KM) plots and the log-rank test. An independent dataset, GSE66229, was used to identify their potential to differentiate between tumor and control tissues. Raw gene expression values of putative gene groups were extracted, and heatmaps and dendograms were drawn via CIMminer online tool (https://discover.nci.nih.gov/cimminer/home.do) according to the Euclidean distance of the biomarker candidates' gene expression.

4. RESULTS AND DISCUSSIONS

4.1. DCNA Structure

The web-based tool consists of 4 different pages.

- Home Page: This section allows users to upload input files to the system (Figure 4.1).
- Co-expression Result Page: The data file uploaded by the users is processed by the algorithm in the background, and the result is shown. Here the co-expression output file can be downloaded in excel format. By clicking the network analysis button, the network creation process is started with Cytoscape (Figure 4.2).
- Network Analysis Page: This page shows the networks created from the Cytoscape program (Figure 4.3).
- Module page: The page where the modules created with the MCODE plugin is displayed (Figure 4.4).

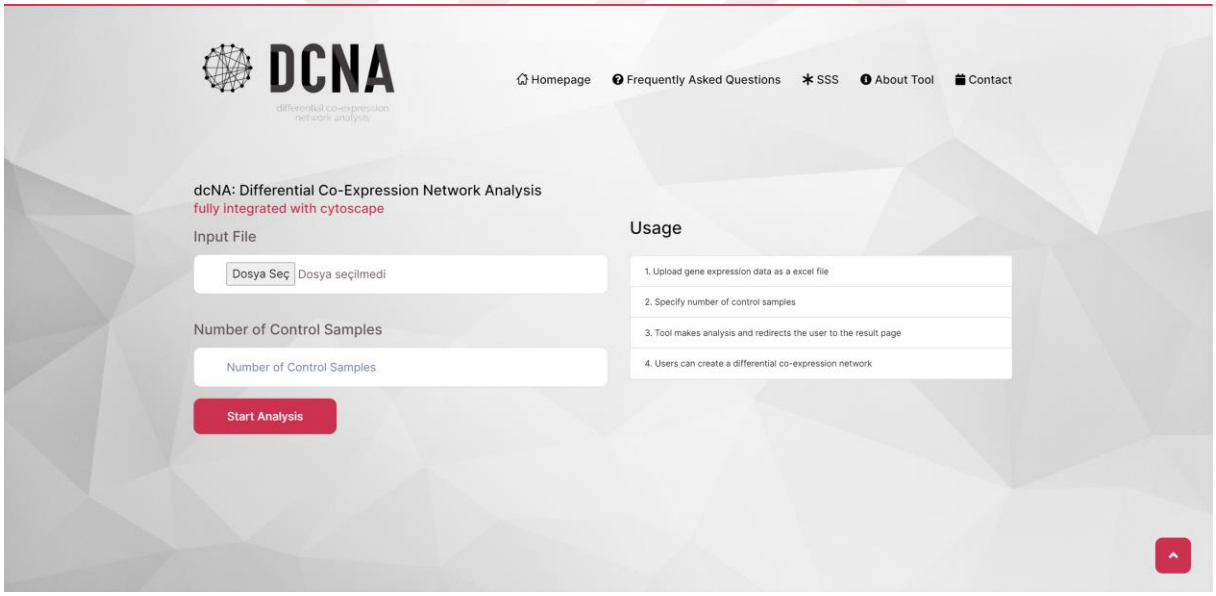


Figure 4.1 DCNA web-based tool home page

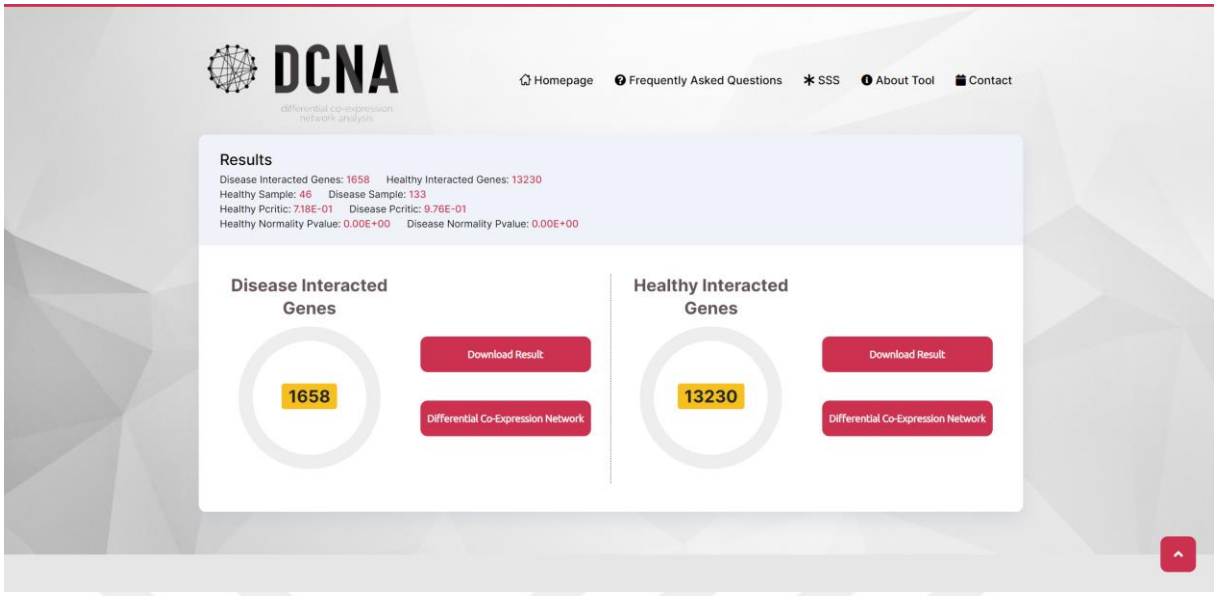


Figure 4.2. DCNA web-based tool co-expression result page

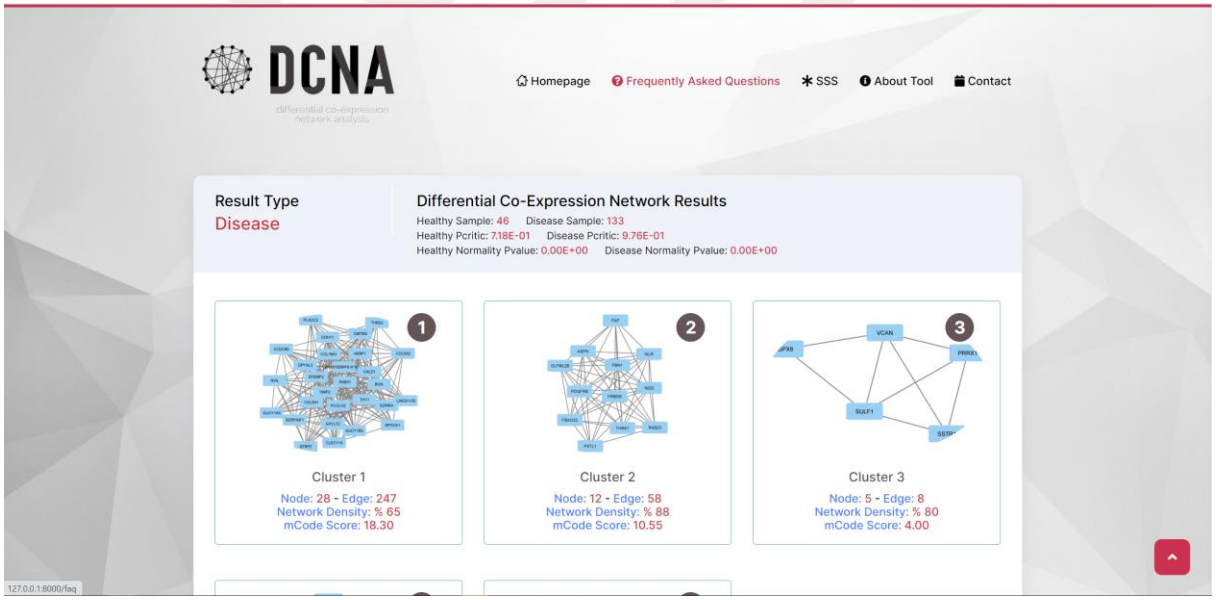


Figure 4.3 DCNA web-based tool network analysis page

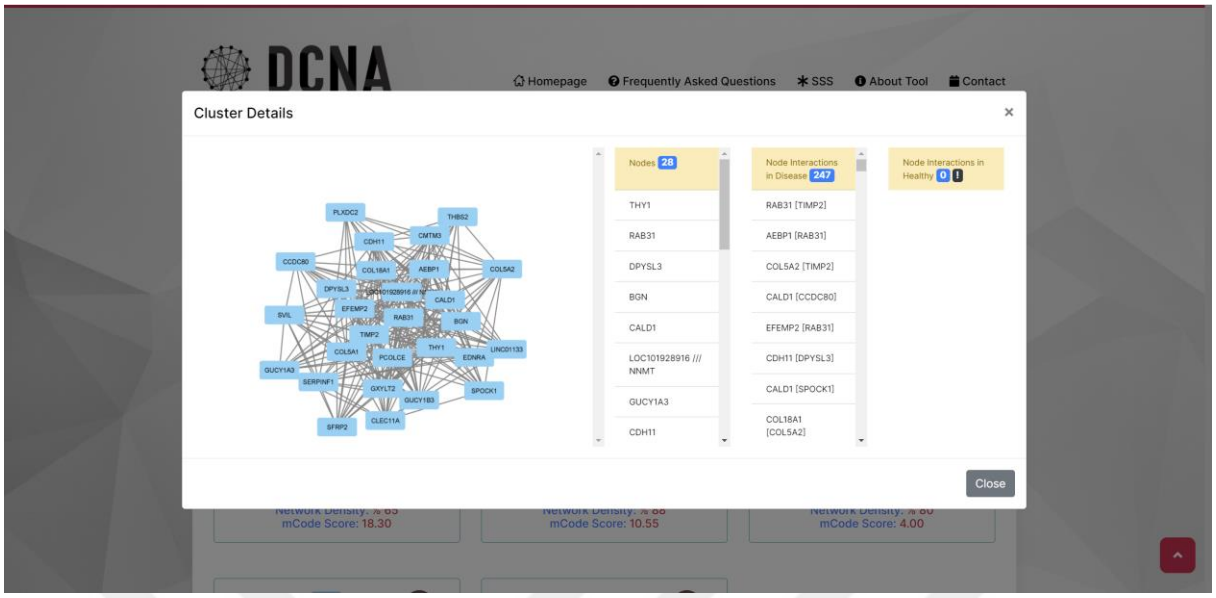


Figure 4.4 DCNA web-based tool module page

4.2. Data Privacy and Protection

Uploaded gene expression data is not stored on the server. Analysis results are stored for 1 week for the user to access again, after which the data is automatically deleted from the database.

4.3. Connection Issues

If the connection between the client and the server is lost during the analysis, the process will stop. In this case, the user will have to start the analysis again. If the server does not receive any response from the user for 30 seconds, the server will automatically finish the process.

4.4. Input File Format

Before analysis process, users should prepare files according to a certain format. Accepted format is simple excel file (.xls, or .xlsx). Sample names in the first line are not necessary for analysis, they can be written for informational purposes. First column contains “Gene Identifiers”, followed by the control and uncontrol samples in the next columns. The remaining rows contain the gene identifiers, in the following columns, their expression values.

4.5. Parallel Use

DCNA allows parallel use in the computation part. However, since the Cytoscape does not allow multiple uses on the networking part, there is a parallel usage restriction. If the Cytoscape program is used by another user, the warning "Try again in 10 seconds" is shown.

4.6. Sample Size Limit

There is no sample size limit on the differential co-expression analysis part. However, on the network creation part, since the number of edges will increase when the input file contains a large number of genes, the processing speed may slow down depending on the Cytoscape capability and server configurations. The tests performed depending on the gene and sample size are shown in the table. The recommended number of DEGs is 300.

Table 4.1. Sample and gene size speed comparisons

Sample Size	Gene Size	Analysis Time (second)	Interacted Genes	Network Creation Time (second)
100	100	4	~500	17
200	100	12	~500	17
300	100	19	~500	17
500	100	34	~500	17
1000	100	91	~500	17
100	200	5	~1900	36
100	300	5	~4000	48
100	500	9	~18000	Java heap space error
100	1000	88	~74000	Java heap space error
100	2000	104	~290000	Java heap space error
100	3000	213	~680000	Java heap space error

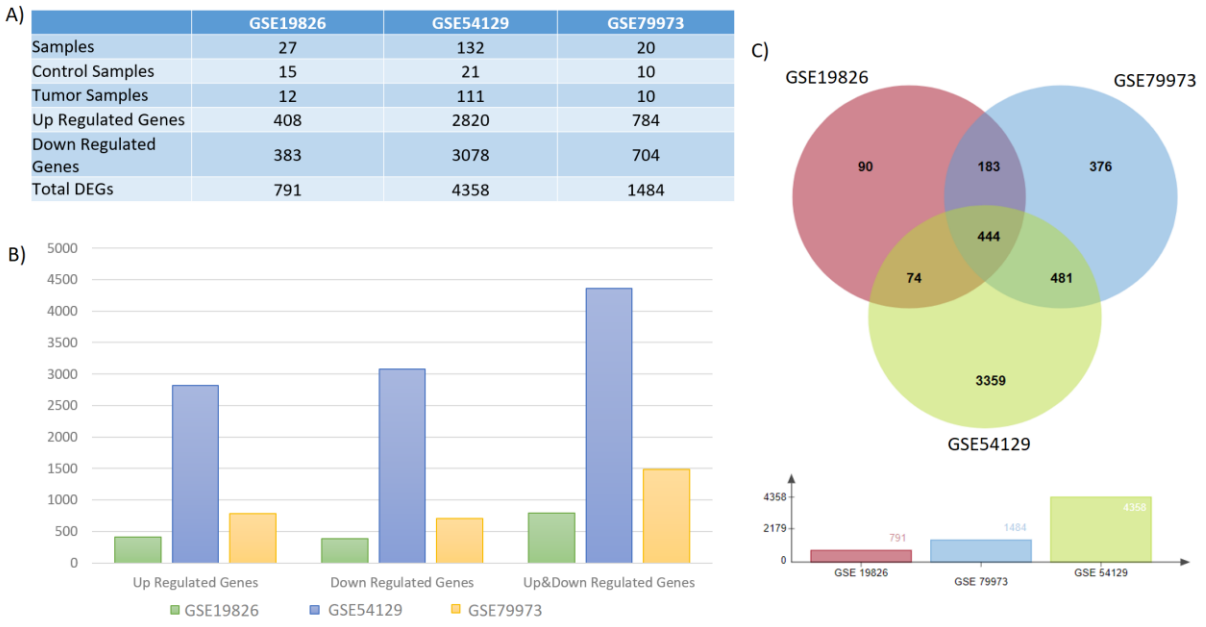
4.7. Availability and Supplementary Information

DCNA is accessible at http://dcna.computationalbiology.org/. Codes and supplementary data are available in http://www.github.com/tyasird/differential-coexpression-network-analysis.

4.8. Case Study

Gastric Cancer datasets were analyzed with the DCNA after the development process finished due to verification.

We have analyzed three microarray gene expression datasets in this study to identify DEGs in GC compared to controls. We obtained 791 DEGs (FDR<0.05 and logFC=2) where 408 upregulated and 383 DEGs downregulated genes from gse19826 datasets. Analysis of gse54129 showed 4358 DEGs (FDR<0.05 and logFC=2), 2820 upregulated, and 3078 downregulated DEGs. Our analysis of gse79973 showed 1484 DEGs (FDR<0.05 and logFC=2) were 784 upregulated and 704 downregulated genes (Figure 4.5A, B). We then sought to detect DEGs common among the three datasets to identify the most consistent DEGs that are likely to be observed in GC samples. Our comparative analysis showed 444 DEGs common among the three above-mentioned GC datasets, which were considered for further downstream analysis (Figure 4.5C).



	GSE19826	GSE54129	GSE79973
Samples	27	132	20
Control Samples	15	21	10
Tumor Samples	12	111	10
Up Regulated Genes	408	2820	784
Down Regulated Genes	383	3078	704
Total DEGs	791	4358	1484

Figure 4.5 A) Table of gastric cancer datasets, control, and tumor sample counts B) DEG counts of each datasets C) Venn diagram of DEGs and common DEG count

After analysis, the DCNA web-based tool determined two crucial gene modules (Figure 4.6), which were significant prognostically in predicting the survival outcome in GC in independent datasets.

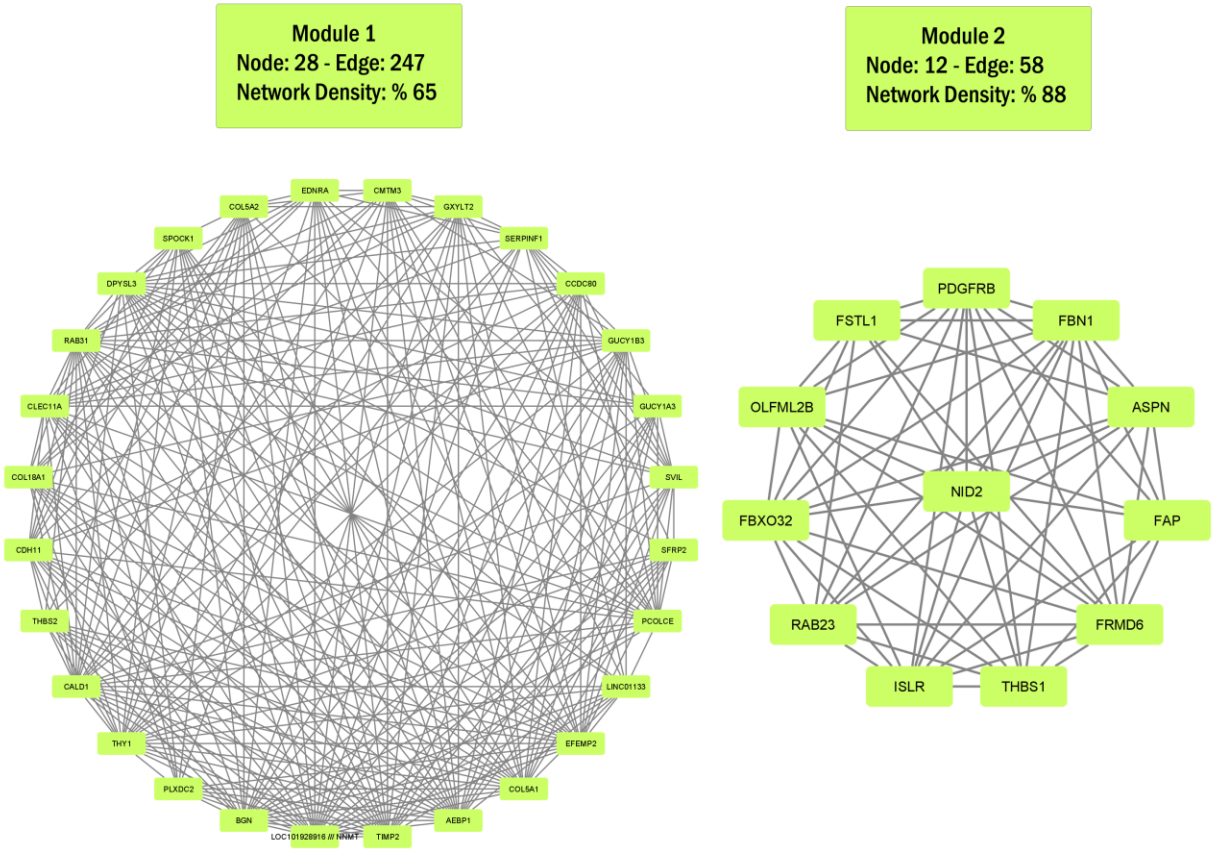


Figure 4.6 Two crucial gene modules determined by DCNA web-based tool

Using an independent data set, the expressions of the module genes were plotted as dendrogram by using Euclidean distance (Figure 4.7). GC tumor and control samples were divided into almost two main groups, and it was determined that modules could differentiate between tumor and normal gastric tissues.

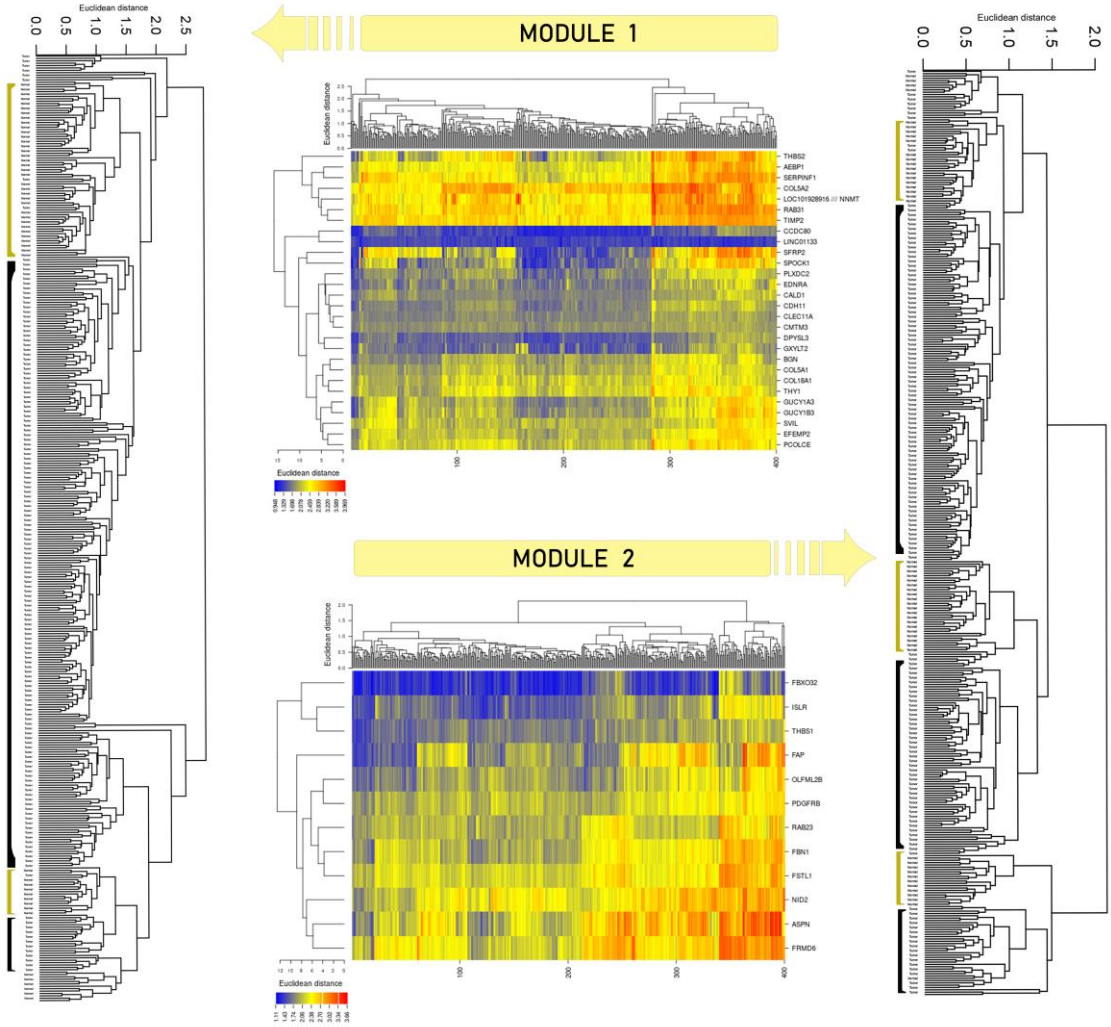


Figure 4.7 Euclidean distance dendrogram of significant gene modules

To predict the survival of GC using the co-expressed modules, we performed KM plot and long-rank hazards text using independent RNA-Seq data from the TCGA database. We have shown that the dysregulated expression of three gene co-expressed modules was associated with the worst outcome in both STAD and STES (Figure 4.8). Our analysis suggested that the identified modules may be used as prognostic and diagnostic biomarkers in GC.

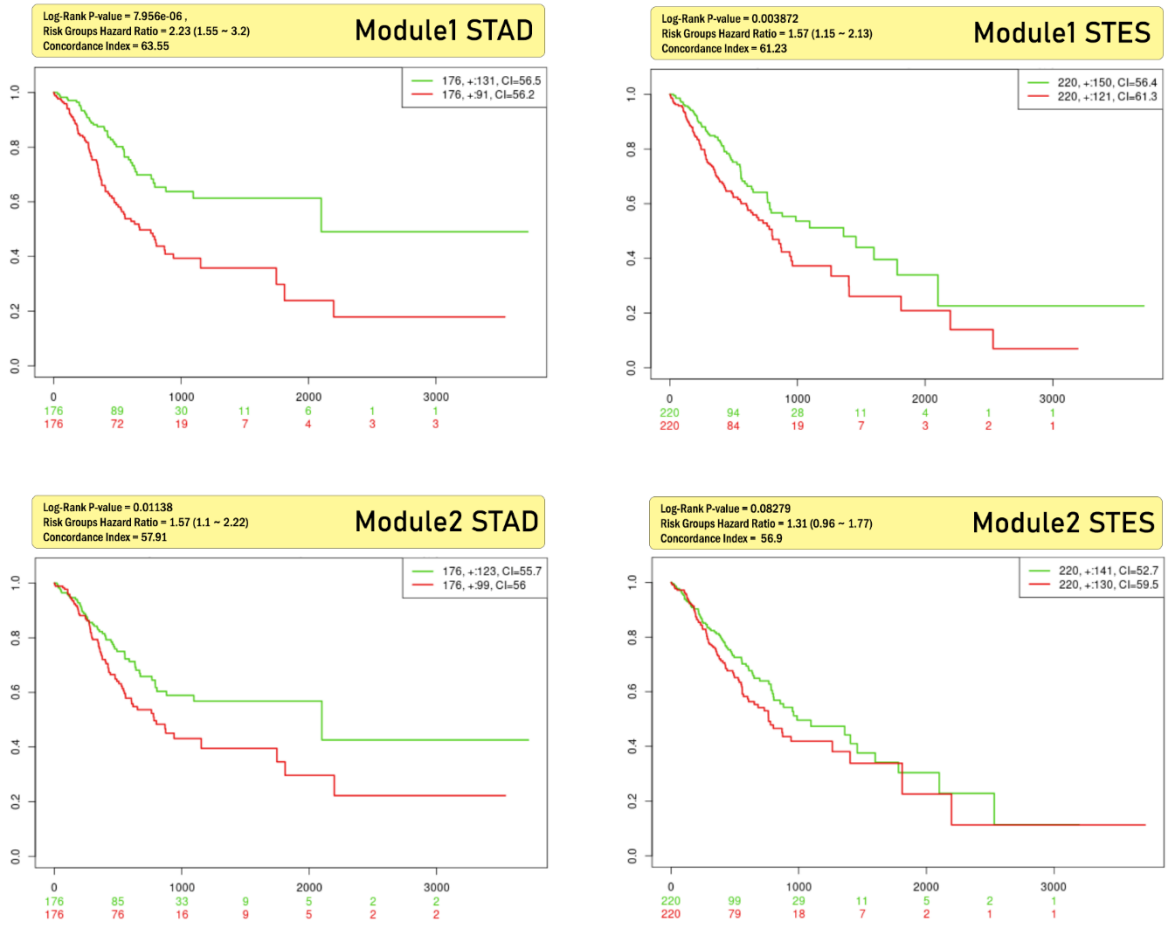


Figure 4.8 The prognostic performances of module genes on independent datasets

As a result, we investigated the microarray gene expression datasets of GC comprehensively using a combination of bioinformatics approaches with the DCNA web-based tool. As seen on diagnostic and prognostic performance analysis, DCNA has targeted significant gene groups correctly. We detected two co-expressed gene modules in these common genes, which were significant prognostically in predicting the survival outcome in GC in independent datasets. In addition, besides stomach cancer, other cancer types were also tested in DCNA, and as a result, it has been seen that the algorithm found significant gene groups for each cancer type.

5. CONCLUSIONS

Identification of key gene modules provides a depth understanding of the genes which behave similarly in expression pattern. The importance of this study is improving the unique differential co-expression network method applicable in all diseases or various states and publishing it as a web-based tool for all researchers to use.

Thus, we aimed to determine co-expressed key gene modules, calculating gene expression with a bioinformatics approach. In this aim, the DCNA web-based tool was developed. The DCNA is designed to be user-friendly and aimed at researchers who study network medicine or systems biology fields. The DCNA is open to the use of all researchers, and it can easily use without coding experience.

Choosing Python as the programming language facilitated the transfer of the algorithm to the web environment. We made a modular system in order to improve with new updates and packages in the future.

As we have presented in the Gastric Cancer case study, the application has found important gene modules that could be biomarker candidates.

6. RECOMMENDATIONS

6.1. Limitations

The web-based tool and the algorithm performed well in several of the aims as mentioned above; however, network creation time might be longer for large input files due to Cytoscape capability.

6.2. Future Works

In the future perspective, some studies can be done to develop the web-based tool.

- Prognostic and diagnostic analyses of significant gene modules can be performed automatically.
- The design and style of the modules can be changed as the user wants.
- Gene modules can be used in literature search with NLP and text-mining methods.
- Results can be sent by e-mail in PDF format instead of showing on the page.

7. APPENDICES

tools/dcna_tool.py

```
import pandas as pd
import numpy as np
import os
from scipy.stats import normaltest
import logging


class DcnaTool:

    def __init__(self):
        logging.basicConfig(filename='tool.log', level=logging.DEBUG, format='%(asctim
e)s - %(message)s', datefmt='%d-%b-%y %H:%M:%S')
        console = logging.StreamHandler()
        console.setLevel(logging.INFO)
        formatter = logging.Formatter('[%(asctime)s][%(levelname)s]: %(message)s')
        console.setFormatter(formatter)
        logging.getLogger('').addHandler(console)
        logger = logging.getLogger(__name__)


    def read_input(self, input_folder, file):

        logging.info('read_input')
        filename, file_ext = os.path.splitext(os.path.basename(file))
        df = pd.read_excel(input_folder + file, header=None)
        if type(df.iloc[0, 1]) == str:
            df = df.iloc[1:, 0:]
        for i in df.columns:
            if i != 0:
                df[i] = df[i].astype('float64')
        # Ortalama aliyoruz
        df = df.groupby(df.columns[0]).mean().reset_index()
        # Gen isimlerimin bulundugu kolonu siliyoruz ki normalizasyon yapabilelim.
        df_values = df.iloc[:, 1:]
        # Gen isimlerini alıyoruz
        df_genes = df.iloc[:, 0]

        return filename, df_genes, df_values

    def normalization(self, df_values):

        logging.info('normalization')
        # Sutun bazinda ortalama aliyoruz. axis=1 olursa satir bazinda ortalama alir.
        mean = df_values.mean(axis=0)
        # Sutun bazinda standart sapma aliyoruz.
        std = df_values.std(axis=0)
        # Normalizasyon
        df_norm = ((df_values - mean) / std)

        return df_norm

    def explode_samples(self, normalizated_data, healthy_sample):

        logging.info('explode_samples')
```

```
        # sağlıklı ve hastalıklı örnekleri bölüyoruz.
        healthy = normalizated_data.iloc[:, 0:int(healthy_sample)]
        disease = normalizated_data.iloc[:, int(healthy_sample):]
        # örnek sayıları
        disease_sample = len(disease.columns)
        healthy_sample = len(healthy.columns)

        return healthy, disease, disease_sample, healthy_sample

    def normality_test(self, healthy, disease):

        logging.info('normality_test')
        z_h, pval_healthy = normaltest(healthy.stack().to_numpy())
        z_d, pval_disease = normaltest(disease.stack().to_numpy())

        return pval_healthy, pval_disease

    def correlation(self, healthy, pval_healthy, disease, pval_disease):

        logging.info('correlation')
        # Pearson ve Spearman uyguluyoruz.
        if (pval_healthy < 0.05):
            healthy_corr = healthy.T.corr(method='spearman')
        else:
            healthy_corr = healthy.T.corr()


        if (pval_disease < 0.05):
            disease_corr = disease.T.corr(method='spearman')
        else:
            disease_corr = disease.T.corr()

        return healthy_corr, disease_corr

    def find_pcritic(self, healthy_corr, disease_corr):

        logging.info('find_pcritic')
        # Sağlıklı bölümler
        df_healty_mean = healthy_corr.values[np.triu_indices_from(healthy_corr.values,
 1)].mean()
        df_healty_std = healthy_corr.values[np.triu_indices_from(healthy_corr.values,
1)].std()
        pcritic_healty = df_healty_mean + 1.96 * df_healty_std
        # Hastalıklı bölümler
        df_disease_mean = disease_corr.values[np.triu_indices_from(disease_corr.values
, 1)].mean()
        df_disease_std = disease_corr.values[np.triu_indices_from(disease_corr.values,
 1)].std()
        pcritic_disease = df_disease_mean + 1.96 * df_disease_std
        # İki pcritic değerinden ufak olanı seçiyoruz.
        pcritic = min(pcritic_healty, pcritic_disease)

        if (pcritic > 0.7):
            pcritic = 0.7

        return pcritic, pcritic_healty, pcritic_disease

    def make_binary(self, healthy_corr, disease_corr, df_genes):

        logging.info('make_binary')
```

```
        # Pearson Uygulanmış temiz verinin satır ve sütunlarına gen isimlerini ekliyor
uz.
        healthy_with_genes = pd.DataFrame(healthy_corr.values, index=df_genes.values,
columns=df_genes.values)
        disease_with_genes = pd.DataFrame(disease_corr.values, index=df_genes.values,
columns=df_genes.values)

        # Veriyi binary dizi haline getiriyoruz.
        # melted = pd.melt(df_healty2.reset_index(), id_vars='index', value_vars=df_ge
nes.values)
        dfHealty = healthy_with_genes.stack().reset_index()
        dfHealty.columns = ['gene1', 'gene2', 'score']
        dfDisease = disease_with_genes.stack().reset_index()
        dfDisease.columns = ['gene1', 'gene2', 'score']

        dfDisease['opposite'] = dfHealty.score
        dfHealty['opposite'] = dfDisease.score

        # Genlerin kendileri ile olan eşleşmelerini siliyoruz
        df_healthy_filter = dfHealty[dfHealty.gene1 != dfHealty.gene2]
        df_disease_filter = dfDisease[dfDisease.gene1 != dfDisease.gene2]

        # Tekrarlayan capraz gen satırlarını siliyoruz
        h_filter = pd.DataFrame(df_healthy_filter, columns=['gene1', 'gene2', 'score',
 'opposite'])
        h_filter = h_filter[h_filter.gene1.astype(str) < h_filter.gene2.astype(str)]

        d_filter = pd.DataFrame(df_disease_filter, columns=['gene1', 'gene2', 'score',
 'opposite'])
        d_filter = d_filter[d_filter.gene1.astype(str) < d_filter.gene2.astype(str)]

        return h_filter, d_filter

    def filter_by_pcritic(self, healthy_binary, disease_binary, pcritic):

        logging.info('filter_by_pcritic')
        # Pcritic değerlerinden büyük olan Anlamlı verileri filtreliyoruz
        filtred_healthy = healthy_binary[
            (healthy_binary.score > pcritic) | (-
healthy_binary.score > pcritic)].reset_index(drop=True)
        filtred_disease = disease_binary[
            (disease_binary.score > pcritic) | (-
disease_binary.score > pcritic)].reset_index(drop=True)

        return filtred_healthy, filtred_disease

    def differential_co_expression(self, healthy, disease, pcritic, threshold):

        logging.info('differential_co_expression')
        # hastalık durumu için;
        # pcc değeri pcritic değeriden büyük olan verileri alıyoruz
        # bu veriler içerisinde ki her etkileşimin sağlıklı pcc değeri pcriticden küçü
k olanları filtreleyip seçiyoruz.
        # hastalık ve sağlıklı durumdaki pcc değerlerinin farkı 0.5den büyükse bu grub
a dahil ediyoruz.

        healthy['differential'] = healthy.score - healthy.opposite
        healthy = healthy[(healthy.opposite < pcritic) | (healthy.differential > thres
hold)]
```

```
        healthy = healthy[(healthy.opposite > -
pcritic) | (healthy.differential > threshold)]
        healthy = healthy.sort_values(by="opposite", ascending=False).reset_index(drop
=True)

        disease['differential'] = disease.score - disease.opposite
        disease = disease[(disease.opposite < pcritic) | (disease.differential > thres
hold)]
        disease = disease[(disease.opposite > -
pcritic) | (disease.differential > threshold)]
        disease = disease.sort_values(by="opposite", ascending=False).reset_index(drop
=True)

        return healthy, disease

    def export_excel(self, filename, folder, healthy, disease):

        logging.info('export_excel')
        healthy.to_excel(folder + 'healthy_' + filename + '.xlsx', engine='xlsxwriter'
, index=None, header=True)
        disease.to_excel(folder + 'disease_' + filename + '.xlsx', engine='xlsxwriter'
, index=None, header=True)

    def export_json(self, healthy, disease):

        # if (len(healthy.values) == 0):
        #     raise ValueError('Result is Empty.')
        return healthy.to_dict(), disease.to_dict()

    def interaction_count(self, healthy, disease):

        return len(healthy.index), len(disease.index)

    def find_module_interactions(data, opposite, cluster, pcritic):
        # verilan tablo icinde bu clustera ait genlerin birbirleriyle olan etkilesimle
rini cekiyoruz.
        interactions = data.loc[((data['gene1'].isin(cluster)) & (data['gene2'].isin(c
luster)))].reset_index(drop=True)
        interactions = interactions[interactions['gene1'] < interactions['gene2']]
        # buldugumuz bu etkilesimler zit tabloda var mi ona bakiyoruz. Varsa pcritic u
zerinde olanlari aliyoruz.
        search_genes = opposite[opposite[['gene1', 'gene2']].isin(interactions[['gene1
', 'gene2']]).all(axis=1)]
        # once tum etkilesimleri gosteriyoruz, sonra etkilesimlerin zit tabloda arama
sonuclarini
        return interactions, search_genes[search_genes['score'] > float(pcritic)]
```

tools/network_tool.py

```
import py4cytoscape as py4
import pandas as pd
import os
import uuid
from time import sleep
import json


class NetworkTool:

    def __init__(self):
```

```
        self.session_folder = 'uploads/cytoscape/session/'
        self.network_folder = 'uploads/cytoscape/network/'
        self.style_folder = 'static/cytoscape/'
        self.session_name = None
        py4.set_summary_logger(True)
        py4.py4cytoscape_logger.set_summary_logger(True)

    def create_session(self):
        self.session_name = uuid.uuid4().hex
        py4.save_session(self.session_folder + self.session_name)
        return self.session_name

    def get_session(self):
        session_name = py4.commands.cyrest_get('session/name')
        return os.path.basename(session_name['name'])

    def delete_session(self):
        py4.session.close_session(False)

    def read_excel(self, input_excel):
        results = pd.read_excel(input_excel, converters={'gene1': str, 'gene2': str})
        results = results.rename(columns={'gene1': 'source', 'gene2': 'target'})
        return results

    def create_network(self, input, network_name):
        network_suid = py4.networks.create_network_from_data_frames(
            edges=input,
            title=network_name,
            collection=network_name + '_collection'
        )
        return network_suid

    def analyze_network(self):
        analyze = py4.commands.commands_post('analyzer/analyze')
        return analyze

    def run_mcode(self):
        mcode = py4.commands.commands_post(
            'mcode cluster degreeCutoff=2 fluff=false fluffNodeDensityCutoff=0.1 hairc
ut=true includeLoops=false kCore=2 maxDepthFromStart=100 network=current nodeScoreCuto
ff=0.2 scope=NETWORK')
        sleep(.5)
        return mcode

    def get_clusters(self, mcode, min_nodes, layout=None):

        if mcode is not None:

            limited_clusters = [i for i in range(len(mcode['clusters'])) if
                                len(mcode['clusters'][i]['nodes']) >= min_nodes]
            clusters = []
            for i in range(len(limited_clusters)):
                hash = uuid.uuid4().hex
                view_id = py4.commands.commands_post('mcode view id=1 rank=' + str(i +
 1))
                if layout is not None:
                    py4.layout_network(layout)
                data = {
                    "id": i,
                    "hash": hash,
```

```
                    "node": py4.networks.get_node_count(),
                    "edge": py4.networks.get_edge_count(),
                    "score": mcode['clusters'][i]['score'],
                    "table": py4.tables.get_table_columns()
.drop(columns=['id', 'name', 'selected', 'MCODE::Clusters (1)', 'SUID'])
.rename(columns={'shared name': 'gene', 'MCODE::Node Status (1)': 'node_status','MCODE
::Score (1)': 'mcode_score'}).sort_values(by=['mcode_score'], ascending=False).reset_i
ndex(drop=True).to_dict()}

                clusters.append(data)
                py4.export_image(self.network_folder + str(hash))

            #return json.dumps(clusters), None  # None means no error
            print(type(clusters))
            return clusters, None  # None means no error

        else:
            return "", "no clusters found."

    def set_style(self, number=None, default=None, name=None):
        styles = ['Marquee', 'Universe', 'Directed', 'DisGeNETstyleV2', 'default', 'Mi
nimal', 'Sample3', 'Ripple','Sample1', 'Big Labels', 'default black', 'Sample2', 'Grad
ient1', 'Nested Network Style', 'Solid', 'Curved']
        if default == True:
            return py4.set_visual_style('default')
        if name is not None:
            return py4.set_visual_style(name)

        return py4.set_visual_style(styles[number])

    def set_layout(self, number, default=None):
        layouts = ['attribute-circle', 'stacked-node-layout', 'degree-
circle', 'circular', 'attributes-layout', 'kamada-kawai', 'force-
directed', 'cose', 'grid', 'hierarchical', 'fruchterman-rheingold', 'isom', 'force-
directed-cl']
        return py4.layout_network(layouts[number])

    def load_style(self, file):
        load = py4.styles.import_visual_styles(self.style_folder + file)
        return load[0]
```

app/views.py

```
from django.shortcuts import render, redirect, reverse, get_object_or_404
from django.http import HttpResponse, HttpResponseNotFound, HttpResponseRedirect, Stre
amingHttpResponse
from django.core.files.storage import FileSystemStorage
from django.core.serializers import serialize
from django.template.defaulttags import register
from django.db.models.query import QuerySet
from django.utils.safestring import mark_safe
from django.forms.models import model_to_dict
from app.models import DiffCoExpression, Network
from django.template import Library
from django.utils import safestring
from django.utils import timezone
import helpers.helper as helper
from .forms import analyzeForm
from tools.dcna_tool import DcnaTool
```

```
from tools.network_tool import NetworkTool
import pandas as pd
import uuid
import json
import math
import os
import glob

def index(request):
    status_hash = uuid.uuid4().hex[0:10]
    return render(request, 'index.html', {'status_hash': status_hash})


def about(request):
    return render(request, 'about.html')


def database(request):
    # file_list = glob.glob("uploads/output/disease_*.xlsx")
    # file_list = [f.replace('\\','/') for f in file_list]
    networks = Network.objects.all()
    for i in networks:
        print(i.diff_coexp.input_excel)
    return render(request, 'database.html', {'networks': networks} )

def status(request, status_hash):
    with open('uploads/status/' + status_hash + '.txt', 'r') as f:
        lines = f.readlines()
    return HttpResponse('\n'.join(lines))


def logdetail(detail,status_hash):
    f = open('uploads/status/'+status_hash+'.txt', "a")
    f.write("{0}\n".format(detail))
    f.close()


def analyze(request):
    input_folder = 'uploads/input/'
    output_folder = 'uploads/output/'
    error = None
    analyze_result = None

    # METHOD POST
    if request.method == 'POST':

        form = analyzeForm(request.POST, request.FILES)
        if form.is_valid():

            status_hash = request.POST['status_hash']
            with open('uploads/status/' + status_hash + '.txt', 'w') as f:
                f.write('')

            file = request.FILES['input_excel']
            if not allowed_file(file.name):
                form.add_error('input_excel', 'file ext error')
                return render(request, 'index.html', {'form': form})

            fs = FileSystemStorage(location=input_folder)
            _, ext = os.path.splitext(file.name)
```

```
            new_filename = uuid.uuid4().hex + ext
            fs.save(new_filename, file)

            try:
                dcna = DcnaTool()
                logdetail('uploading..', status_hash)
                filename, genes, values = dcna.read_input(input_folder, new_filename)
                logdetail('normalization..', status_hash)
                normalizated_data = dcna.normalization(values)
                logdetail('explode samples..', status_hash)
                healthy, disease, disease_sample, healthy_sample = dcna.explode_sample
s(normalizated_data,request.POST['healthy_sample'])
                logdetail('normality test..', status_hash)
                pval_healthy, pval_disease = dcna.normality_test(healthy, disease)
                logdetail('correlation..', status_hash)
                healthy_corr, disease_corr = dcna.correlation(healthy, pval_healthy, d
isease, pval_disease)
                logdetail('find pcritic..', status_hash)
                pcritic, pcritic_healty, pcritic_disease = dcna.find_pcritic(healthy_c
orr, disease_corr)
                logdetail('make binary..', status_hash)
                healthy_genes, disease_genes = dcna.make_binary(healthy_corr, disease_
corr, genes)
                logdetail('filter by pcritic..', status_hash)
                filtred_healthy, filtred_disease = dcna.filter_by_pcritic(healthy_gene
s, disease_genes, pcritic)
                logdetail('differential coexpression analysis..', status_hash)
                diff_healthy, diff_disease = dcna.differential_co_expression(filtred_h
ealthy, filtred_disease, pcritic,
                                                                             0.5)
                logdetail('export to excel..', status_hash)
                dcna.export_excel(filename, output_folder, diff_healthy, diff_disease)
                logdetail('export to json..', status_hash)
                diff_healthy_json, diff_disease_json = dcna.export_json(diff_healthy,
diff_disease)
                logdetail('create db record..', status_hash)
                create = DiffCoExpression.objects.create(
                    cancer_id=0,
                    input_excel=new_filename,
                    healthy_sample=healthy_sample,
                    disease_sample=disease_sample,
                    disease_pcritic=pcritic_disease,
                    healthy_pcritic=pcritic_healty,
                    healthy_normality_pvalue=pval_healthy,
                    disease_normality_pvalue=pval_disease,
                    healthy_co_expression=0,
                    disease_co_expression=0,
                    healthy_diff_co_expression=diff_healthy_json,
                    disease_diff_co_expression=diff_disease_json,
                    timestamp=timezone.now()
                )

                return redirect('diffcoexp', id=create.id)

            # manually add error
            except ValueError as e:

                return render(request, 'index.html', {'error': e.args[0]})

        # form error
```

```
        else:
            return render(request, 'index.html', {'form': form})

    # METHOD GET
    else:
        return redirect('/')


def diffcoexp(request, id):
    response = get_object_or_404(DiffCoExpression, id=int(id))
    data = {}
    data['healthy_diff_co_expression_len'] = len(response.healthy_diff_co_expression['
gene1'])
    data['disease_diff_co_expression_len'] = len(response.disease_diff_co_expression['
gene1'])
    return render(request, 'diff_coexp_result.html', {'response': response, 'data': da
ta})


def network(request, id, type_id):
    # 0 = disease, 1 = healthy
    prefix = type_id == 0 and 'disease_' or 'healthy_'
    output_folder = 'uploads/output/'

    diff_coexp = DiffCoExpression.objects.get(id=id)
    network = Network.objects.filter(diff_coexp_id=id, type=type_id).first()

    # no records, open cytoscape and do analysis
    if network is None:

        try:
            network_tool = NetworkTool()
            if network_tool.get_session():
                return render(request, 'diff_coexp_result.html',
                              {'error': 'Cytoscape is busy. Please try again later or
Refresh the page.',
                               'response': diff_coexp})
        except Exception as e:
            return render(request, 'diff_coexp_result.html', {'error': str(e), 'respon
se': diff_coexp})

        else:
            try:
                session = network_tool.create_session()
                read = network_tool.read_excel(output_folder + prefix + diff_coexp.inp
ut_excel)
                create_network = network_tool.create_network(read, session)
                mcode = network_tool.run_mcode()
                clusters, error = network_tool.get_clusters(mcode, 5)
                delete = network_tool.delete_session()
                create = Network.objects.create(diff_coexp_id=id, type=type_id, mcode=
clusters)

                clusters_with_interactions = []
                for c in clusters:
                    interactions, opposite_interactions = find_module_interactions(
                        diff_coexp.disease_diff_co_expression, diff_coexp.healthy_diff
_co_expression,
                        c['table']['gene'].values(), diff_coexp.disease_pcritic)
                    c['interactions'] = interactions
```

```
                    c['opposite_interactions'] = opposite_interactions
                    clusters_with_interactions.append(c)

                return render(request, 'network_result.html',
                              {'clusters': clusters_with_interactions, 'network': crea
te, 'diff_coexp': diff_coexp})

            except Exception as e:

                delete = network_tool.delete_session()
                return render(request, 'diff_coexp_result.html', {
                    'response': diff_coexp,
                    'error': str(e),
                })

    # record found in db, just show it
    else:
        clusters = network.mcode
        clusters_with_interactions = []
        for c in clusters:
            interactions, opposite_interactions = find_module_interactions(diff_coexp.
disease_diff_co_expression,
                                                                           diff_coexp.
healthy_diff_co_expression,
                                                                           c['table'][
'gene'].values(),
                                                                           diff_coexp.
disease_pcritic)
            c['interactions'] = interactions
            c['opposite_interactions'] = opposite_interactions
            clusters_with_interactions.append(c)

        return render(request, 'network_result.html',
                      {'clusters': clusters_with_interactions, 'network': network, 'di
ff_coexp': diff_coexp})


def allowed_file(filename):
    allowed_extensions = {'.csv', '.xls', '.xlsx'}
    _, ext = os.path.splitext(filename)
    return ext in allowed_extensions


@register.filter
def get_item(dictionary, key):
    return dictionary.get(key)


@register.filter
def format_pvalue(value):
    return format(float(value), "10.2E")


@register.filter
def jsonify(object):
    if isinstance(object, QuerySet):
        return mark_safe(serialize('json', object))
    return mark_safe(json.dumps(object))
```

```
jsonify.is_safe = True


@register.filter
def find_network_density(value, arg):
    return int(round(arg / ((value * (value - 1)) / 2), 2) * 100)


def find_module_interactions(data, opposite, cluster, pcritic):
    data = pd.read_json(json.dumps(data))
    opposite = pd.read_json(json.dumps(opposite))
    # verilan tablo icinde bu clustera ait genlerin birbirleriyle olan etkilesimlerini
 cekiyoruz.
    interactions = data.loc[((data['gene1'].isin(cluster)) & (data['gene2'].isin(clust
er)))].reset_index(drop=True)
    interactions = interactions[interactions['gene1'] < interactions['gene2']]
    # buldugumuz bu etkilesimler zit tabloda var mi ona bakiyoruz. Varsa pcritic uzeri
nde olanlari aliyoruz.
    search_genes = opposite[opposite[['gene1', 'gene2']].isin(interactions[['gene1', '
gene2']]).all(axis=1)]
    search_genes = search_genes[search_genes['score'] > float(pcritic)]
    # once tum etkilesimleri gosteriyoruz, sonra etkilesimlerin zit tabloda arama sonu
clarini
    # returns data_interactions, opposite_interactions
    return interactions[['gene1', 'gene2']].values.tolist(), search_genes[['gene1', 'g
ene2']].values.tolist()
```

app/models.py

```
from django.db import models
from django.utils import timezone

# Create your models here.
class DiffCoExpression(models.Model):

    cancer_id = models.IntegerField(blank=True, null=True)
    input_excel = models.TextField(blank=True, null=True)
    healthy_sample = models.IntegerField(blank=True, null=True)
    disease_sample = models.IntegerField(blank=True, null=True)
    healthy_pcritic = models.TextField(blank=True, null=True)
    disease_pcritic = models.TextField(blank=True, null=True)
    healthy_normality_pvalue = models.TextField(blank=True, null=True)
    disease_normality_pvalue = models.TextField(blank=True, null=True)
    healthy_co_expression = models.JSONField(blank=True, null=True)
    disease_co_expression = models.JSONField(blank=True, null=True)
    healthy_diff_co_expression = models.JSONField(blank=True, null=True)
    disease_diff_co_expression = models.JSONField(blank=True, null=True)
    timestamp = models.DateField(default=timezone.now,blank=True, null=True)

    def publish(self):
        self.timestamp = timezone.now()
        self.save()

    def __str__(self):
        return str(self.id) + "_" + str(self.timestamp)


class Network(models.Model):
```

```
    diff_coexp = models.ForeignKey(DiffCoExpression,null=True,on_delete=models.CASCADE
)

    #diff_coexp_id = models.IntegerField(blank=True)
    type = models.IntegerField(blank=True,null=True) #0 Disease
    mcode = models.JSONField(blank=True, null=True)
    timestamp = models.DateField(default=timezone.now,blank=True, null=True)

    def publish(self):
        self.timestamp = timezone.now()
        self.save()

    def __str__(self):
        return str(self.id) + "_" + str(self.timestamp)
```

app/forms.py

```
from django import forms
from .models import DiffCoExpression


class analyzeForm(forms.Form):
    status_hash = forms.CharField(label='Hash', required=True)
    healthy_sample = forms.IntegerField(label='Healthy Sample', required=True)
    input_excel = forms.FileField(label='Input', required=True)

    class Meta:
        model = DiffCoExpression
        exclude = ['disease_sample', 'pcritic_min', 'pcritic_max', 'healthy_normality_
pvalue',
                   'disease_normality_pvalue', 'healthy_co_expression', 'disease_co_ex
pression',
                   'healthy_diff_co_expression', 'disease_diff_co_expression', 'timest
amp']
```

app/apps.py

```
from django.apps import AppConfig
class AppConfig(AppConfig):
    name = 'app'
```

app/admin.py

```
from django.contrib import admin
from .models import DiffCoExpression, Network
admin.site.register(DiffCoExpression)
admin.site.register(Network)
```

config/settings.py

```
from pathlib import Path
import os

BASE_DIR = Path(__file__).resolve().parent.parent
SECRET_KEY = '$u#n)kdux58@1ji)p*uo4oyb*ltbur0(=yqyn28*o&tck5vlpa'
```

```
DEBUG = True
ALLOWED_HOSTS = []

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.humanize',
    'app'
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'config.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': ['templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ]
        },
    },
]

WSGI_APPLICATION = 'config.wsgi.application'


DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': 'db',
    }
}


AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValida
tor',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
```

```
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'Europe/Istanbul'

USE_I18N = True

USE_L10N = True

USE_TZ = True

STATIC_DIR = os.path.join(BASE_DIR, 'static')
STATIC_URL = '/static/'
STATICFILES_DIRS = [STATIC_DIR, ]

MEDIA_ROOT = os.path.join(BASE_DIR, 'uploads')
MEDIA_URL = '/uploads/'
```

config/urls.py

```
from django.conf import settings
from django.conf.urls.static import static
from django.contrib import admin
from django.urls import path, re_path
from app import views


urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.index, name='index'),
    path('about', views.about, name='about'),
    path('database', views.database, name='database'),
    path('analyze', views.analyze, name='analyze'),
    path('status/<slug:status_hash>', views.status, name='status'),
    path('diffcoexp/<int:id>/', views.diffcoexp, name='diffcoexp'),
    path('network/<int:id>/<int:type_id>', views.network, name='network'),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT) + static(settings.ST
ATIC_URL, document_root=settings.STATIC_ROOT)
```

REFERENCES

Aamer Mehmood, M. (2014). Use of Bioinformatics Tools in Different Spheres of Life Sciences. *Journal of Data Mining in Genomics & Proteomics*, *05*(02). https://doi.org/10.4172/2153-0602.1000158

Ahmed, M. Z. (2014). Which one is better - JavaScript or jQuery. *International Journal of Computer Science and Mobile Computing*, *3*(6), 193–207.

Amar, D., Safer, H., & Shamir, R. (2013). Dissection of regulatory networks that are altered in disease via differential co-expression. *PLoS Comput Biol*, *9*(3), e1002955.

Bader, G. D., & Hogue, C. W. V. (2003). An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, *4*, 1–27. https://doi.org/10.1186/1471-2105-4-2

Barrett, T., Wilhite, S. E., Ledoux, P., Evangelista, C., Kim, I. F., Tomashevsky, M., Marshall, K. A., Phillippy, K. H., Sherman, P. M., & Holko, M. (2012). NCBI GEO: archive for functional genomics data sets—update. *Nucleic Acids Research*, *41*(D1), D991–D995.

Berners-Lee, T. (1989). Information Management: A Proposal. *Cern*, *March 1989*, 1–15. http://www.w3.org/History/1989/proposal.html

Berners-Lee, T. J. (1992). The world-wide web. *Computer Networks and ISDN Systems*, *25*(4–5), 454–459. https://doi.org/10.1016/0169-7552(92)90039-S

Bhuva, D. D., Cursons, J., Smyth, G. K., & Davis, M. J. (2019). Differential co-expression-based detection of conditional relationships in transcriptional data: comparative analysis and application to breast cancer. *Genome Biology*, *20*(1), 1–21.

Bray, F., Ferlay, J., Soerjomataram, I., Siegel, R. L., Torre, L. A., & Jemal, A. (2018). Global cancer statistics 2018: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries. *CA: A Cancer Journal for Clinicians*, *68*(6), 394–424.

Chowdhury, H. A., Bhattacharyya, D. K., & Kalita, J. K. (2020). (Differential) Co-Expression Analysis of Gene Expression: A Survey of Best Practices. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *17*(4), 1154–1173. https://doi.org/10.1109/TCBB.2019.2893170

Ciman, M., Gaggi, O., & Gonzo, N. (2014). *Cross-platform mobile development*. *March*, 757–759. https://doi.org/10.1145/2554850.2555104

Cline, M. S., Smoot, M., Cerami, E., Kuchinsky, A., Landys, N., Workman, C., Christmas, R., Avila-Campilo, I., Creech, M., Gross, B., Hanspers, K., Isserlin, R., Kelley, R.,

Killcoyne, S., Lotia, S., Maere, S., Morris, J., Ono, K., Pavlovic, V., … Bader, G. D. (2007). Integration of biological networks and gene expression data using cytoscape. *Nature Protocols*, *2*(10), 2366–2382. https://doi.org/10.1038/nprot.2007.324

Comer, D. E., & Stevens, D. L. (1993). *Internetworking with TCP/IP Vol. III: Client-server programming and applications*. Prentice-Hall, Inc.

Cook, L. (2019). Genetics and Biotechnology. In *edtechpress*.

Cytoscape. (2020a). *Cytoscape App Manager*. https://apps.cytoscape.org

Cytoscape. (2020b). *Cytoscape User Manual*.

Cytoscape. (2020c). *Cytoscape welcome letter*. https://cytoscape.org/cy3_welcome_letter_v8.pdf

D'agostino, R. B., Belanger, A., & D'Agostino Jr, R. B. (1990). A suggestion for using powerful and informative tests of normality. *The American Statistician*, *44*(4), 316–321.

Dauzon, S. (2014). *Django Essentials*. Packt Publishing Ltd.

Dawson, J. A., Ye, S., & Kendziorski, C. (2012). R/EBcoexpress: an empirical Bayesian framework for discovering differential co-expression. *Bioinformatics*, *28*(14), 1939–1940.

Dehmer, M., Emmert-streib, F., Graber, A., & Salvador, A. (2011). *Applied Statistics for Network Biology*. Wiley-Blackwell.

Demchak, B. (2021). *Overview of py4cytoscape*. https://py4cytoscape.readthedocs.io/en/latest/

Demchenko, M. (2020). *Top Python Web Frameworks 2020*. https://ncube.com/blog/top-python-web-frameworks-2020

Dirksen, J. (2013). *Learning Three. js: The JavaScript 3D Library for WebGL*. http://books.google.com/books?hl=en&lr=&id=6TVeAQAAQBAJ&oi=fnd&pg=PT10&dq=Learning+Three+.+js+:+The+JavaScript+3D+Library+for+WebGL&ots=oZje_vkIfH&sig=8E-JDs88BoaFo2qi1rZEFajzk9M%5Cnhttp://books.google.co.uk/books?id=6TVeAQAAQBAJ

Eltahawi, A. O. (2016). *Hyper Text Markup Language HTML : A Tutorial*. *November*, 1–63. https://www.researchgate.net/publication/321213969_Hyper_Text_Markup_Language_HTML_A_Tutorial

Falconframework.org. (2020). *Falcon*. https://falconframework.org/#sectionAbout

Ferguson, R. (2019). Beginning JavaScript. *Beginning JavaScript*, 1–10.

https://doi.org/10.1007/978-1-4842-4395-4

Foster, E. C., & Godbole, S. (2016). *Database systems: a pragmatic approach*. Apress.

Fukushima, A. (2013). DiffCorr: an R package to analyze and visualize differential correlations in biological networks. *Gene*, *518*(1), 209–214.

Gautier, L., Cope, L., Bolstad, B. M., & Irizarry, R. A. (2004). affy—analysis of Affymetrix GeneChip data at the probe level. *Bioinformatics*, *20*(3), 307–315.

Gentleman, R., Carey, V., Huber, W., Irizarry, R., & Dudoit, S. (2006). *Bioinformatics and computational biology solutions using R and Bioconductor*. Springer Science & Business Media.

Ghimere, D. (2020). *Comparative study on Python web frameworks: Flask and Django*. *May*. https://www.theseus.fi/handle/10024/339796

Gov, E. (2020). Co-expressed functional module-related genes in ovarian cancer stem cells represent novel prognostic biomarkers in ovarian cancer. *Systems Biology in Reproductive Medicine*, *66*(4), 255–266.

Gov, E., & Arga, K. Y. (2017). Differential co-expression analysis reveals a novel prognostic gene module in ovarian cancer. *Scientific Reports*, *7*(1), 1–10. https://doi.org/10.1038/s41598-017-05298-w

Gunicorn.org. (2020). *Gunicorn*. https://docs.gunicorn.org/en/stable/

Gupta, S., Narang, S., Arora, S., & Jindal, S. (2017). A Brief Discussion About Client-Server System. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, *6*(3), 2278–1323. http://ijarcet.org/wp-content/uploads/IJARCET-VOL-6-ISSUE-3-338-341.pdf

Ha, M. J., Baladandayuthapani, V., & Do, K.-A. (2015). DINGO: differential network analysis in genomics. *Bioinformatics*, *31*(21), 3413–3420.

Hourieh, A. (2008). Learning Website Development with Django. In *Birmingham: Packt Publishing Ltd*. ftp://forum.olvi.net.ua/Library/share/Django/Learning Website Development with Django [Ayman Hourieh] (2008)/Learning.Website.Development.with.Django.Ayman.Hourieh.2008.pdf

Hsiao, T.-H., Chiu, Y.-C., Hsu, P.-Y., Lu, T.-P., Lai, L.-C., Tsai, M.-H., Huang, T. H.-M., Chuang, E. Y., & Chen, Y. (2016). Differential network analysis reveals the genome-wide landscape of estrogen receptor modulation in hormonal cancers. *Scientific Reports*, *6*(1), 1–16.

Hsu, C.-L., Juan, H.-F., & Huang, H.-C. (2015). Functional analysis and characterization of

differential coexpression networks. *Scientific Reports*, *5*(1), 1–14.

Ideker, T., & Krogan, N. J. (2012). Differential network biology. *Molecular Systems Biology*, *8*(1), 565.

Introduction to the server side. (2021). https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Introduction

Jader, O. H., Zeebaree, S. R. M., & Zebari, R. R. (2019). A state of art survey for web server performance measurement and load balancing mechanisms. *International Journal of Scientific and Technology Research*, *8*(12), 535–543.

Jardim, V. C., Santos, S. de S., Fujita, A., & Buckeridge, M. S. (2019). BioNetStat: A tool for biological networks differential analysis. *Frontiers in Genetics*, *10*, 594.

JetBrains. (2020). *Devecosystem 2020*. https://www.jetbrains.com/lp/devecosystem-2020/python/

Kalló, B. (2015). *Extending the Cascading Style Sheets (CSS) Language with Programming Constructs*.

Khatri, P., & Drăghici, S. (2005). Ontological analysis of gene expression data: current tools, limitations, and open problems. *Bioinformatics*, *21*(18), 3587–3595.

Kunda, D., Chihana, S., & Sinyinda, M. (2010). Web Server Performance of Apache and Nginx: A Systematic Literature Review. *Computer Engineering and Intelligent Systems*, *8*(2), 43–52. https://iiste.org/Journals/index.php/CEIS/article/view/35842

Langfelder, P., & Horvath, S. (2008). WGCNA: an R package for weighted correlation network analysis. *BMC Bioinformatics*, *9*(1), 1–13.

Letkowski, J. (2014). Doing database design with MySQL. *Journal of Technology Research*, *6*(I), 1–15. https://pdfs.semanticscholar.org/6dab/89a3e32c994cff67d0c0f40c69ef9446db2c.pdf%0A

Li, D., Brown, J. B., Orsini, L., Pan, Z., Hu, G., & He, S. (2016). MODA: MOdule Differential Analysis for weighted gene co-expression network. *ArXiv Preprint ArXiv:1605.04739*.

Manem, V., Adam, G. A., Gruosso, T., Gigoux, M., Bertos, N., Park, M., & Haibe-Kains, B. (2018). CrosstalkNet: A visualization tool for differential co-expression networks and communities. *Cancer Research*, *78*(8), 2140–2143.

Maria, P. (2018). JavaScript Beyond the Browser. *Open Repository Theseus, University of Turku*, *1*, 25. https://www.theseus.fi/handle/10024/151106

Matsinopoulos, P. (2020). Practical Bootstrap. In *Practical Bootstrap*. https://doi.org/10.1007/978-1-4842-6071-5

Mazinanian, D., Tsantalis, N., & Mesbah, A. (2014). Discovering refactoring opportunities in cascading style sheets. *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, *16-21-Nove*, 496–506. https://doi.org/10.1145/2635868.2635879

McKenzie, A. T., Katsyv, I., Song, W.-M., Wang, M., & Zhang, B. (2016). DGCA: a comprehensive R package for differential gene correlation analysis. *BMC Systems Biology*, *10*(1), 1–25.

McKinney, W. (2012). *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*.

McMahon, R. (2003). *Introduction to networking*. McGraw-Hill Osborne Media.

Morselli Gysi, D., de Miranda Fragoso, T., Zebardast, F., Bertoli, W., Busskamp, V., Almaas, E., & Nowick, K. (2020). Whole transcriptomic network analysis using Co-expression Differential Network Analysis (CoDiNA). *Plos One*, *15*(10), e0240523.

Nguyen, V. N. (2017). *Comparative Performance Evaluation of Web Servers*. *31*(3), 28–34.

Niederst, J., Beijing, R., Cambridge, Farnham, Köln, Sebastopol, & Tokyo. (2012). *Learning Web Design, Fourth Edition*. http://www.winnystudio.com/kmcweb/lectures/LearningWebDesig4thEdition_01.pdf

NumPyCommunity. (2020). *NumPy User Guide 1.19*. 214.

Osmani, A. (2012). *Learning JavaScript Design Patterns: A JavaScript and jQuery Developer's Guide*.

Otasek, D., Morris, J. H., Bouças, J., Pico, A. R., & Demchak, B. (2019). Cytoscape Automation: Empowering workflow-based network analysis. *Genome Biology*, *20*(1), 1–16. https://doi.org/10.1186/s13059-019-1758-4

Ou-Yang, L., Dai, D.-Q., & Zhang, X.-F. (2013). Protein complex detection via weighted ensemble clustering based on Bayesian nonnegative matrix factorization. *PloS One*, *8*(5), e62158.

Pablo, A., & Millán, P. (2013). Network generation and analysis through Cytoscape and PSICQUIC. *European Bioinformatics Institute*, 26.

Pavlovic, V. (2009). *MCODE Documentation*. http://www.baderlab.org/Software/MCODE/UsersManual

Petlovana, Y. (2020). *Top 13 Python Web Frameworks to Learn in 2020*.

https://steelkiwi.com/blog/top-10-python-web-frameworks-to-learn/

Pollock, J. (2010). JavaScript A Beginner's Guide Third Edition. In *McGraw-Hill*.

Pollock, J. (2013). *JavaScript Fourth Edition*. McGraw-Hill.

Purewal, S. (2014). *Learning Web App Development*.

PyPA. (2020). *pip documentation*. https://pip.pypa.io/en/stable/installing/

PythonSoftwareFoundation. (2020). *pypi.org*. https://pip.pypa.io/en/stable/installing/

Reece, J. B., Urry, L. A., Cain, M. L., Wasserman, S. A., Minorsky, P. V, & Jackson, R. B. (2014). *Campbell biology* (Issue s 1309). Pearson Boston.

Russo, P. S. T., Ferreira, G. R., Cardozo, L. E., Bürger, M. C., Arias-Carrasco, R., Maruyama, S. R., Hirata, T. D. C., Lima, D. S., Passos, F. M., & Fukutani, K. F. (2018). CEMiTool: a Bioconductor package for performing comprehensive modular co-expression analyses. *Bmc Bioinformatics*, *19*(1), 1–13.

Saabith, A. L. S., Fareez, M. M. M., & Vinothraj, T. (2019). Python current trend applications-an overview. *International Journal of Advance Engineering and Research Development*, *6*(10).

Saito, R., Smoot, M. E., Ono, K., Ruscheinski, J., Wang, P. L., Lotia, S., Pico, A. R., Bader, G. D., & Ideker, T. (2012). A travel guide to Cytoscape plugins. *Nature Methods*, *9*(11), 1069–1076. https://doi.org/10.1038/nmeth.2212

Schuurman, D. (2011). A brief introduction to MySQL. *Perspectives in Public Health*, *131*(6), 248–249. https://doi.org/10.1177/1757913911425736

Shah, M. Q. (2015). Responsive Web Development Using the Twitter Bootstrap. *Bachelor's Thesis Degree Programme: Information Technology Specialisation: Internet Technology 2015*, 1–42.

Shen, X., Buford, J., Yu, H., & Akon, M. (2010). Handbook of peer-to-peer networking. *Handbook of Peer-to-Peer Networking*, 1–1405. https://doi.org/10.1007/978-0-387-09751-0

Shenoy, A., Sossou, U., Reviewers, U. S., Gupta, R. K., Raval, H., & Sarmento, F. (2014). *Learning Bootstrap, Unearth the potential of Bootstrap to create responsive web pages using modern techniques*.

Sheward, D., & Napier, C. (2005). A guide to HTML – the HyperText Markup Language. *STAFFORDSHIRE UNIVERISTY*.

Shi, J., Qu, Y.-P., & Hou, P. (2014). Pathogenetic mechanisms in gastric cancer. *World Journal of Gastroenterology: WJG*, *20*(38), 13804.

Smoot, M. E., Ono, K., Ruscheinski, J., Wang, P.-L., & Ideker, T. (2011). Cytoscape 2.8: new features for data integration and network visualization. *Bioinformatics*, *27*(3), 431–432.

Smyth, G. K. (2005). Limma: linear models for microarray data. In *Bioinformatics and computational biology solutions using R and Bioconductor* (pp. 397–420). Springer.

Soni, R. (2016). *Nginx*. Springer.

Srinath, K. R. (2017). Python – The Fastest Growing Programming Language. *International Research Journal of Engineering and Technology*, 354–357. https://d1wqtxts1xzle7.cloudfront.net/55458585/IRJET-V4I1266.pdf?1515226715=&response-content-disposition=inline%3B+filename%3DPython_The_Fastest_Growing_Programming_L.pdf&Expires=1593202307&Signature=HBD7oa85wDxqRzTWX01uVRBlMacGX5mkGk1b~SVVTTkENJ6cf5diKz

Stackoverflow.com. (2020). *Stack Overflow Developer Survey*. https://insights.stackoverflow.com/survey/2020

Stark, J., & Jepson, B. (2012). *Building Android Apps with HTML, CSS, and JavaScript, Second Edition*.

Su, G., Morris, J. H., Demchak, B., & Bader, G. D. (2014). Biological network exploration with Cytoscape 3. *Current Protocols in Bioinformatics*, *47*(1), 8–13.

Tahaghoghi, S., & Williams, H. (2006). Learning MySQL: Get a Handle on Your Data. In *O'Reilly Media*.

Takei, N., Saito, T., Takasu, K., & Yamada, T. (2015). Web Browser Fingerprinting Using only Cascading Style Sheets. *Proceedings - 2015 10th International Conference on Broadband and Wireless Computing, Communication and Applications, BWCCA 2015*, 57–63. https://doi.org/10.1109/BWCCA.2015.105

Taneja, S., & Gupta, P. R. (2014). Python as a Tool for Web Server Application Development. *International Journal of Information, Communication and Computing Technology*, *2*(1), 77–83. https://www.jimsindia.org/8i_Journal/VolumeII/Python-as-a-tool-for-web-server-application-development.pdf

Tarca, A. L., Romero, R., & Draghici, S. (2006). Analysis of microarray experiments of gene expression profiling. *American Journal of Obstetrics and Gynecology*, *195*(2), 373–388. https://doi.org/10.1016/j.ajog.2006.07.001

Tesson, B. M., Breitling, R., & Jansen, R. C. (2010). DiffCoEx: a simple and sensitive method to find differentially coexpressed gene modules. *BMC Bioinformatics*, *11*(1), 1–

9.

The Python Development Team. (2008). *Python 3.0 Release*. https://www.python.org/download/releases/3.0/

TheCytoscapeConsortium. (2020). *py4cytoscape Documentation*.

Van Cutsem, E., Sagaert, X., Topal, B., Haustermans, K., & Prenen, H. (2016). Gastric cancer. *The Lancet*, *388*(10060), 2654–2664.

Van Rossum, G., & Drake, F. L. (2012). Python Tutorial. *Python Software Foundation*, 1–136.

Van Rossum, G., & Drake Jr, F. L. (1995). *Python tutorial* (Vol. 620). Centrum voor Wiskunde en Informatica Amsterdam.

Van Rossum, G., & FL Drake Jr. (2010). *HOWTO Use Python in the web*.

van Rossum, G., & The Python Development Team. (2018). *The Python Language Reference - Release 3.8.1*. 1–160. https://docs.python.org/3/download.html

van Rossum, G., & The Python Development Team. (2020). *Python Frequently Asked Questions - Release 3.8.1*. 1–109.

Wang, D., Wang, J., Jiang, Y., Liang, Y., & Xu, D. (2017). BFDCA: A comprehensive tool of using Bayes factor for differential co-expression analysis. *Journal of Molecular Biology*, *429*(3), 446–453.

Wei, W., Amberkar, S., & Hide, W. (2018). *About diffcoexp*.

Widenius, M., Axmark, D., & Arno, K. (2002). *MySQL reference manual: documentation from the source*. " O'Reilly Media, Inc."

Wiese, R., Eiglsperger, M., & Kaufmann, M. (2004). yfiles—visualization and automatic layout of graphs. In *Graph Drawing Software* (pp. 173–191). Springer.

Yang, J., Yu, H., Liu, B.-H., Zhao, Z., Liu, L., Ma, L.-X., Li, Y.-X., & Li, Y.-Y. (2013). DCGL v2. 0: an R package for unveiling differential regulation from differential co-expression. *PloS One*, *8*(11), e79729.

York, R. (2005). Beginning CSS: Cascading Style Sheets for Web Design. In *Css* (Issue February).