UiO : Department of Informatics
University of Oslo

Ranklust

A bioinformatics solution to identify network biomarkers in cancer

Henning Lund-Hanssen

Master's Thesis Spring 2016

Ranklust

Henning Lund-Hanssen

28th April 2016

Abstract

This is where the abstract goes...

Contents

List of Figures

List of Tables

Part I

Introduction

To this day we still struggle with cancer. Even with all our modern equipment and knowledge we have still not been able to tame this horrible disease.

And even if we have all of this new equipment, most molecular biologists does not have the experience nor the competence to use the newfound equipment, or interpret the results. This is where bioinformaticians come in.

Incidence is the increase in the rate of new occurrences. So if we were to analyze the amount of new HIV incidents over 10 years and we start with 20 incidents, then after 5 years we get 20 new incidents and then after 10 years, we get 30 new incidents. We now have a total of 75 new incidents since the start (25 + 20 + 30). Because we do not know when the 20 first incidents in the first 5 years happened, we assume they happened after 2,5 years (halfway to 5 years). So we will get 20 * 2,5 and end up with 50 possible new cases. Do the same with the next 5 years and the next 30 new occurrences! The only exception to the previous calculation is that we know they happened somewhere between 5 and 10 years, so we go halfway, to 7,5 ((5+10)/2 = 7,5) and do the same calculation: 7,5 * 30 = 225. Then combine them, 225 + 50 = 275. We ignore the first change, since they are what we started with. Let us also define the sample size of the population to be 225. We then know that 150 people did not get HIV during the 10-year period. So if we were to add these 150 people over the 10 years that did not receive HIV, we get 150 * 10 = 1500. We add the increase of HIV, which is 275 and get 275 + 1500 = 1775.

My thesis is about making a tool, named Ranklust, which gives cancer researchers an easier way of identifying network biomarkers in cancer.

Chapter 1

Pathway and network analysis of cancer genomes

In the cases of pancreatic, lung, breast, brain and overian cancer, the somatic distribution of SNP's has a few altered genes that occur with a frequency higher than 10% and many other genes that are mutated occur with a frequency of 5% or lower[25]. On the other hand, prostate cancer have relatively few SNPs and CNAs, so that kind of cancer is more likely to be driven by something else, namely DNA methylation. DNA methylation is when a methyl group (CH3) is added[1] to the DNA strand. CONTINUE HERE!!!

Chapter 2

Motivation

Biomarkers are at the centre of this thesis. They are what Ranklust should be able to detect and rank in order to identify network biomarkers, and not just single molecules of them. A biomarker is a "biological measure of a biological state" [26]. Among other things, it can be represented by the levels of a specific protein in our blood, a specific gene, or a combination the two.

Biomarkers can be used for different purposes. They can be used to measure the effect of cancer drug treatment. That the drug does what it is supposed to do. It can be used to predict disease development or the current stage of the disease. Here is a list of what biomarkers currently are being used for:

Usages for biomarkers: [33]

- Disease disposition
  - What is a patient's risk of developing cancer in the future?
- Screening
  - Does earlier detection of patients with cancer decrease mortality?
- Diagnostic
  - Who has cancer? What is the grade of the cancer?
- Prognostic
  - What clinical outcome is most likely if therapy is not administered?

- Predictive
  - Which therapy is most appropriate?
- Monitoring
  - Was therapy effective? Did the patient's disease recur?
- Pharmacogenomic
  - What is the risk for adverse reaction to the prescribed therapeutic dose?

Characteristics for a good biomarker:

- Safe and easy to measure
- Cost efficient to follow up
- Modifiable with treatment
- Consistent across gender and ethnic groups

The National Institutes of Health Biomarkers Definitions Working Group has defined biomarkers as "a characteristic that is objectively measured and evaluated as an indicator of normal biological processes, pathogenic process, or pharmacologic responses to a therapeutic intervention.". The World Health Organization (WHO) has created a guideline for defining biomarkers: "almost any measurement reflecting an interaction between a biological system and a potential hazard, which may be chemical, physical or biological. The measured response may be functional and physiological, biochemical at the cellular level, or a molecular interaction". So a biomarker can be anything from the pulse of the heart or blood pressure. In this thesis, I will see if we can get better and more information from ranking gene and protein clusters based on biomarker information. The result could in the best case be a new biomarker in itself, and can be a step in the direction of hierarchical biomarkers (biomarkers creating new biomarkers).

Biomarkers versus Clinical Endpoints Some of the important characteristics of biomarkers is that they are objective and quantifiable as biological processes. They are used as a state indicator of the biological object that is under scrutiny. When the biological object is a human being screned for cancer, biomarkers may help.

Biomarkers can be an indicator of how far cancer has progressed in the human body. But the human subject may feel no difference at all. It can also be the opposite, that the human subject feels huge differences during several weeks that the cancer might have developed at a grand scale, but the biomarkers show no objective change. This proves that the human

subject's experience and sense of state that it is in does not necessarily have to correlate.

Clinical endpoints is the opposite. They describe how the subjects feel or describe how they function. It is not as objective as biomarkers and it demands more resources to gather the information, as the subjects has to be interviewed in some form, rather than interpreting pure data. But one thing has to be noted; patients does not seek treatment for cancer due to their biomarkers being "off the charts". They seek treatment because they *feel* that they do not feel ok or do not function sufficiently. So biomarkers are not by any means far superior to clinical endpoints in all aspects.

Biomarkers can also be ruled out as a reliable predictor when population differences are too big. As with clinical endpoints, people describe their feelings differently and might hold information back. As pointed out before, people's feelings are subjective, and different ways of interpreting their own body's state might skew statistical results with erronous feedback. Erronous feedback in this case can be examplified as two persons who are at the exact same state of cancer, with the same prerequisites, but they report how they feel differently. One person might be ignoring certain pains or lose hair after radiation treatment, but not the other. This can be mitigated to some degree with careful and accurate screening of patients admitted to treatment, but a totally unified group in terms of how they describe pain and other attributes that are interesting might be seen as borderline impossible. We are after all humans and very much fallible.

In 1990, cancer was the third leading cause of death. By 2013, it has taken second place and the amount of deaths related to cancer is considered to be doubled by 2030. For men in particular, the highest incidents of cancer came from prostate cancer, with a count of 1.4 million incidents. For women, breast cancer tops the statistics at 1.8 million incidents. For both sexes, tracheal, bronchus and lung cancer tops the list at 1.6 *deaths*, not incidents.

The increasing number of cancer-related incidents is a big threat in countries that are not adequate equipped to deal with complex and expensive cancer treatments.

With a stable population growth and age structure, we would have seen a 13% decrease in incident cases between 1990 and 2013. Instead, incident cases has increased by 67%

Incidence cases for women has risen slowly whereas for men, it has decreased.

Breast cancer is the most lethal cancer for women. Only 1% of breast cancer cases occur in men. Between 1990 and 2013, incidence cases of breast cancer has increased by 99%. Taking into account a stable population and

age structure, it falls to about 26%, which still is a significant increase in incidents.

Prostate cancer incident cases increased by 217% in period 1990 to 2013. Increased life expectancy is expected to contribute somewhere between 20% and 43% to the increase in this occasion.

217% on a global scale, 167% in developed countries and 367% in developing countries. Age alone cannot explain the HUGE increase in developing countries. But it may show a bigger need for prostate screening in low-/middle-income countries.

Even though developed countries has 70% higher ASIR, the ASDR is only 20% higher in developed countries, showing that the cancer is detected, patients in developed countries has a higher chance of survival (not very surprising...).

An example of a single molecule biomarker is the prostate specific antigen (PSA). This is a protein produced by the prostate gland in male humans. The identification of cancer with PSA is simple, the higher the level of PSA, measured in ng/mL (nanograms per milliliter), the higher is the chance of the patient having prostate cancer [2].

Today PSA is used for both identifying and evaluating the current stage of prostate cancer. This biomarker can be found by analyzing blood examples from patients, thus fulfilling some of the demands for a good biomarker, but not all of them. It is easy to measure and easy to acquire, but not reliable enough to be used as the only marker to identify and determine the stage or remission of prostate cancer. The low grade of reliability comes from the fact that even though higher levels of PSA shows higher chance of having prostate cancer, prostate cancer is not the only reason to have elevated levels of PSA [2]. Namely inflammation and enlargement of the prostate. Though, a man with both of these cases may or may not develop prostate cancer.

So the conclusion for the PSA biomarker is that it is not reliable enough and are causing faulty treatment of prostate cancer that may not even exist. Because even if a patient has prostate cancer, it may not be harmful, promoting the case of not taking any action against it at all. So there is need for a new biomarker, or at least a better way to diagnose and predict the right treatment.

Chapter 3

Next-generation sequencing

Today, rapid analyzing of genes and proteins are made available through Next-Generation Sequencing (NGS)[3]. Networks offers us an informatics, algorithmic, visual and mathematical tool to study this bigger picture. This project will integrate the opportunity networks offer to discover new biomarkers in cancer. Through acquiring more data, faster than before, there now exists databases with much information that is easy to access. This makes room for building huge networks of proteins and genes, allowing for more extensively and thorough assays to be done. For example, what if something that is classified as a prostate cancer biomarker only is viable when proteins that has not been classified as a biomarker, also is present? Together they could represent a more appropriate *network biomarker*. The amount of data that can be analyzed also opens up for another more personalized approach to each cancer patient. Finding patient-specific biomarkers could make a huge impact on the quality of treatment [32].

Chapter 4

Next generation of biomarkers

The PSA biomarker is over 20 years old [27]. Through those years, it has been discovered other and better biomarkers for prostate cancer than PSA. Among those, PCA3, which is detectable through urine samples from patients. It also has the benefit of not being affected by the size of the prostate gland [29]. But still the results could be better. Therefore, it has been tried to combine these two biomarkers in order to see if it is beneficial to see the results from each biomarker in light of each other [33]. The results from these tests is that they complement each other to a level of significance that makes it compelling to analyze them both to diagnose prostate cancer. It is important to point out that even if these biomarkers are not the best at indicating if a patient has cancer or not, these biomarkers are good at indicating progression and recurrence of prostate cancer.

But all of this is based on single genes or proteins. What if we looked at whole networks as biomarkers?

Chapter 5

Networks as a tool in cancer research

Viewing the cell as a network of proteins and genes presents us with several assumptions to make. Firstly, there is no physical connection between the proteins and genes, which represents the nodes in the network. So a way of defining edges between nodes has to be established. It exists several ways of doing this, but there has yet to be discovered a silver bullet of defining the edges.

Bayesian probability networks are one way of defining edges [30]. It is based on the probability that if a node A exists, then node B exists. That way it is possible to create networks based on the assumption that if node A exists, then node B is 80% likely to exist. Maybe node C and D have a 100% chance of existing if node B exists. These percentages represents how strong the edges between the nodes are, and makes it able to construct a directed acyclic graph, DAG for short. It is important to note that a true bayesian network can not be achieved through random observations. Rather should some constant value(s) be introduced to be able to really measure the effect of the other variables.

The way of defining edges in a network also determines what kind of information that is possible to get from it. The different bio-databases has different ways of calculating edges. Should the database alone define the edges? If the database supply weights in the edges and/or nodes, should it be used, changed or ignored? The final decision of how and in what way the edges should be defined has yet to come.

They have studied 50 cancer types that have been cataloged by the International Cancer Genome Consortium (IGCC). They found that only a few well-studied driver genes are frequently mutated, in contrast to many infrequently mutated genes that may also contribute to tumor biology.

There is only a few altered genes that have a mutation frequency higher than 10%, and a shitload of genes that have a mutation frequency of 5% and lower. Prostate cancer is a type of cancer that has relatively few Single Nucleotide Polymorphisms (SNPs) and Copy-Number Alterations (CNAs). This indicates that prostate cancer is probably driven by some other type of somatic variation. The way driver genes are found is by identifying positive-selection signals.

With this approach the genes that are less frequently mutated, but are functionally important genes, will not be detected by statistical analysis. Another way detecting driver genes is by grouping the genetic alterations we can find through knowledge about cellular mechanisms. This way will provide information about the affected oncogenic pathways.

Pathways are small networks of well studied processes where many areas are well documented. Networks on the other hand are bigger and less explored, but when properly analyzed, it might provide new information unknown to pathway systems.

Analyzing pathways and networks have an advantage over individual genes. They may reveal information that comes from molecular events that covers multiple genes or pathways. Aggregating this data increases the chances of detecting driver genes through statistical analysis (source 15). Also, information gathered through pathways and networks may be enriched through genomic, transcriptomic and proteomic data and create a more unified view of the tumor biology.

Chapter 6

Network clustering: Toward network based biomarker discovery

Networks is the pre-processing of single gene prioritization that is necessary in order to come to the next step, clustering of the network. Clustering is about making a hierarchical view of a network to be able to look at the bigger picture of the cell. The reason to group a network into clusters and rank them is that the edges we create represents different connections. They can represent function, probability of existence, interactions or contents of the cell [24]. The function of the cell is what Ranklust is after. There is several algorithms to create clusters, and all need to be heavily researched in order to find the best suited one. The major differences on how these algorithms work is centered around how they handle vertices and edges. For example how the cluster is expanded, what density level it is aiming for, how robust the algorithm is towards incomplete networks. It is feasible if the cluster-making algorithm is easily, or even already implemented as an app in Cytoscape. ClusterMaker2 is such an app and will be considered [4]. But there is faster cluster-creating algorithms than those implemented in ClusterMaker2, namely the SPICi [31] algorithm. So a possible solution for Ranklust could be to implement the SPICi algorithm into the ClusterMaker2 app, in order to reuse most of the code that represents the GUI, single gene prioritization and network creation. In addition to the cluster-creating algorithms, there is a need for cluster-ranking algorithms. They should be ranked in the order of being a biomarker for cancer.

The secretome of certain genes can be interpreted as biomarkers. CONTINUE HERE!

Chapter 7

Cytoscape: An implementation to rank clusters

Cytoscape is the open-source software platform the Ranklust app will be developed on. Its main purpose is to visualize molecular interaction networks and biological pathways. It is easy to integrate ***Apps*** and even combine multiple apps to solve new problems, given that the source code of the apps is available or that it exists an easy way of piping results from one app to another.

The goal of Ranklust is to cluster the networks we get from single gene prioritization and rank them in order to identify network biomarkers. Apps taking care of making the networks already exists, but there still has to be made a decision about whether or not they could be modified in order to better support the clustering.

Cytoscape is based on the Java programming language, which is a little bit untraditional for a software platform used to develop bioinformatics tools. The reason to choose Java above Python, Perl or other popular programming languages is simply because I am more versed in Java programming than any other language. Java is known for being this big bloated enterprise programming language, and Python as the fast and easy mockup tool to develop good programs fast. Python also has big biological computation libraries like *Biopython* [5], that makes it easy to build your own standalone apps. Though when used in a bigger environment as Cytoscape, Java shines, having sturdy packaging and modelling standards. The Cytoscape community is big, alive and has standards for how the architecture of apps should look like. The community promotes this through the use of the *OSGi* standard [6].

Developing OSGi software should promote modularization [7] unreliable source? of the code and increase the probability of the app being launched as an official Cytoscape app; in addition to provide other devel-

opers with the possibility of reusing my modules in their own apps. Also, it seems like Java 9 is aimed at making it easier to modularize apps along the lines of the OSGi standard, so it might be easier to refactor an application in the future from Java 8 to Java 9 when the architecture already is in place. There exists several design patterns that could prove to be useful in the development of Ranklust. Another strategy to follow may not be a direct design pattern [8], but more of a collection of them, is the clean code principles by Robert C. Martin [9]. More thorough examination of these strategies will follow.

Prototyping of different parts of the app will be done in Python and the Galaxy environment [10]. Python is an easy language to prototype with, and Galaxy is an easy environment to test small scripts in. Galaxy also has great logging of previous experiments combined with settings, so recreating simulations and comparing results is easy to do and reliable. However, in my experience, Java comes up to par with Python in development speed once all of the boilerplate code is written and a good deployment tool is used. Therefore the Cytoscape platform is used for the final deployment of Ranklust.

Chapter 8

Methods

Chapter 9

Databases

Using databases helps with several things. CONTINUE!!!

Which databases to use has to be considered. The reason to use databases is because they have information about how protein and genes form a network based on how they interact with each other. The initial database candidates in Ranklust are iRefIndex [11], GeneMania [12] and STRING [13]. These databases all have in common that it exists Cytoscape apps made to use these databases. STRING however, does not have any repository available through the Cytoscape app store, so interacting with the database through a new app in Cytoscape without making new plugins may be difficult. On the other hand, both iRefIndex and GeneMania have their repositories easily available to the public together with decent documentation. However, the difference between them is what they contain information about. iRefIndex contains data about protein-protein interaction (PPI), while GeneMania contains data about genes. Since proteins come from genes, GeneMania can also give us some information about proteins. Differences between the two databases will be discussed in greater depth at a later stage. The open-source plugins in Cytoscape to communicate with the databases are iRefScape [14] for iRefIndex and GeneMANIA [15] for GeneMania.

A problem with the Neo4J database is the dump it creates. The dump creates a single Cypher (source: cypher) commit for the whole database. This way, if anything goes wrong while importing the data, it will get rolled back. It hinders faulty data and relationship between several parts of the data to be imported into the database. The problem with a single commit to import the data is the required memory. My personal computer and the computers at University of Oslo that I have access to does not have enough memory to import the database as a single commit in Cypher. Splitting up the dump to several commits does not work without quirks. It conquers the memory problem, but the way the relationships are created from the Neo4J dump requires the import process to be done in a single commit. A

way to fix this is either to not use Cypher queries as a way of importing the data, but instead use GraphML. Another way is to refactor the cypher dump so that the creation of relationships does not need to be done in a single commit. I will use both ways, because not being able to use the Cypher queries because of too little memory will most likely be a problem for several people and not just me. Using GraphML is good for exporting data from CytoScape and from the database, but interaction between the two of them while performing the algorithms is not feasible at the time. This last drawback is where Cypher queries are good, because it exists plugins that aid in using data from the database directly in CytoScape. So initializing a database for testing could easily be done with GraphML, and then use Cypher queries to perform the computations on the data.

Creating a script to refactor the Cypher queries does not only help in regards of using the data with CytoScape, but also for everyone using a Neo4J Cypher dump to create a database and does not have much memory. A problem that might come up is performance. Because the current way of creating relationships use node labels in Neo4J and it takes only a single line to create a relationship between two nodes. If I refactor the relationship-part of the Neo4J dump, it will increase to a two instruction query. Firstly, two nodes has to be found based on their indexed node name, which will relate to the protein/gene name. Secondly, I do the same thing that was done before the refactor, I create the relationship between the two nodes. The difference between the old and the new way of creating the relationship is how the nodes are found. The Neo4J dump creates temporary node variables prepended with an underscore and an integer ID. These ID's only exist within the commit the nodes are created. So splitting creation and relationship-making leaves us with worthless IDs that does not refer to any known node. That is when Neo4J tries to be smart and then creates new nodes based on the underscore ID and then creates a relationship between them. The end result, two new nodes that only contains the undescore IDs. So the refactored version aims at matching the underscore IDs in the relationship creation to the node creation, and to replace them with the name of the nodes instead of the temporary underscore ID. But as mentioned, matching the nodes with the name takes up more time and performance might be an issue. For comparison, with a Cypher dump of 4500 lines, it takes about 20 seconds before the query engine gets a stack overflow (explain stack overflow?). With GraphML, an XML file with 250000 lines takes under 1 second to import and it gets relationships right without any form of refactoring.

Chapter 10

Clustering algorithms

Part II

Images testing...

Figure 10.1: Cancers Ranked by Number of Incident Cases in Both Sexes, Globally, by Development Status, and in the 50 Most Populous Countries, 2013

Region	Country	Breast cancer	Tracheal, bronchus, and lung cancer	Colon and rectum cancer	Prostate cancer	Stomach cancer	Liver cancer	Cervical cancer	Non-Hodgkin lymphoma	Esophageal cancer	Leukemia	Lip and oral cavity cancer	Bladder cancer	Uterine cancer	Pancreatic cancer	Brain and nervous system cancer	Kidney cancer	Malignant skin melanoma	Ovarian cancer	Thyroid cancer	Gallbladder and biliary tract cancer	Larynx cancer	Other pharynx cancer	Multiple myeloma	Hodgkin lymphoma	Nasopharynx cancer	Testicular cancer	Mesothelioma
Global		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
Developed		3	4	2	1	5	11	17	7	20	12	14	6	13	8	16	10	9	15	18	19	22	23	21	24	27	25	26
Developing		1	2	4	6	3	5	7	11	8	10	9	16	13	14	12	20	23	18	15	19	17	21	25	24	22	26	27
High-income Asia Pacific	Japan	4	2	1	6	3	5	17	10	11	13	14	9	18	7	22	12	23	15	16	8	21	20	19	27	26	25	24
	South Korea	5	2	1	8	3	4	13	10	15	14	18	11	19	9	16	12	23	17	6	7	21	22	20	25	24	27	26
High-income North America	Canada	4	3	2	1	12	18	20	5	19	10	14	6	8	11	13	9	7	15	17	21	22	23	16	24	27	25	26
	United States	2	3	4	1	13	18	20	7	19	11	12	6	9	10	17	8	5	15	14	23	21	22	16	24	27	25	26
Southern Latin America	Argentina[a]	1	4	3	2	5	18	6	8	14	12	16	10	7	9	19	11	15	17	21	13	20	25	23	24	26	22	27
Western Europe	France	3	4	2	1	10	14	19	6	18	13	12	5	7	11	16	9	8	17	22	23	21	15	20	24	27	25	26
	Germany	3	4	2	1	6	16	21	9	18	11	15	5	12	8	14	7	10	13	22	17	23	19	20	26	27	24	25
	Italy	2	4	1	3	6	8	21	7	22	11	19	5	12	9	14	10	13	15	18	16	20	24	17	25	27	26	23
	Spain	4	3	1	2	6	11	20	7	23	12	13	5	8	9	15	10	14	16	22	19	17	21	18	24	27	25	26
	United Kingdom	4	3	2	1	7	18	19	6	12	11	17	5	14	9	15	10	8	13	24	23	20	22	16	26	27	25	21
Central Asia	Uzbekistan[a]	1	3	8	11	2	9	4	12	5	6	10	20	13	16	7	17	18	19	24	21	14	23	25	15	26	22	27
Central Europe	Poland	3	1	2	4	6	19	11	14	20	12	16	5	9	8	13	7	17	10	21	18	15	23	22	25	26	24	27
Eastern Europe	Russia	3	2	1	5	4	17	9	14	19	12	11	8	10	7	18	6	16	13	15	21	20	22	24	23	26	25	27
	Ukraine	2	3	1	4	5	21	9	15	20	11	8	10	17	7	14	6	13	12	16	19	18	22	24	23	27	26	25
Andean Latin America	Peru[a]	3	6	4	1	2	9	5	7	21	8	18	20	10	12	15	14	17	16	11	13	24	25	19	23	27	22	26
Central Latin America	Colombia[a]	2	6	4	1	3	10	5	9	17	7	16	23	13	12	11	20	18	14	8	15	19	25	22	21	26	24	27
	Mexico[a]	2	7	3	1	5	10	4	9	22	8	18	24	16	12	14	11	20	13	6	17	19	25	23	21	27	15	26
	Venezuela[a]	2	4	5	1	6	13	3	7	19	8	16	22	10	11	17	12	20	15	9	18	14	25	21	23	26	24	27
Tropical Latin America	Brazil[a]	2	4	3	1	5	17	6	11	14	8	10	21	16	12	7	18	13	20	9	22	15	19	23	24	26	25	27
East Asia	China[a]	5	1	4	9	2	3	12	11	6	8	19	14	7	13	10	20	26	21	17	18	16	25	24	22	15	27	23
	North Korea[a]	4	1	5	11	3	2	7	12	6	9	14	17	8	13	10	22	23	21	20	19	16	25	24	18	15	27	26
Southeast Asia	Indonesia[a]	1	2	4	7	5	10	3	8	21	11	6	18	13	15	12	19	23	14	9	16	20	22	25	24	17	26	27
	Malaysia[a]	1	2	3	4	7	6	10	5	19	8	14	15	12	17	18	16	22	11	9	23	20	21	25	24	13	26	27
	Myanmar[a]	1	2	5	11	12	4	3	9	17	7	6	22	8	15	14	23	24	10	13	16	19	20	25	21	18	26	27
	Philippines[a]	1	2	4	3	12	7	5	11	23	6	10	20	9	15	14	18	21	13	8	19	17	22	26	25	16	24	27
	Thailand[a]	3	1	4	5	7	2	6	10	19	11	8	13	17	16	14	18	23	15	12	9	20	21	26	25	22	24	27
	Vietnam[a]	4	2	5	12	3	1	8	6	10	11	7	18	13	17	9	21	25	20	14	19	15	16	26	24	22	23	27
South Asia	Afghanistan[a]	3	2	8	10	1	9	4	12	17	5	14	11	13	18	6	20	23	22	16	21	15	25	24	7	19	26	27
	Bangladesh[a]	3	4	7	11	6	2	8	5	10	9	1	19	17	22	14	23	24	12	26	18	15	13	25	20	16	21	27
	India[a]	1	6	4	15	5	8	3	11	7	10	2	19	24	22	12	21	16	14	17	20	13	9	25	18	23	26	27
	Nepal[a]	1	4	6	13	7	9	3	12	5	8	2	21	22	17	14	23	25	11	20	16	15	10	24	18	19	26	27
	Pakistan[a]	1	3	6	8	13	12	15	5	4	7	2	9	17	24	14	22	25	11	18	20	10	16	23	19	21	26	27
North Africa and Middle East	Algeria[a]	1	3	2	10	4	15	7	6	23	5	21	14	20	16	8	22	24	17	12	9	18	25	19	13	11	26	27
	Egypt[a]	1	6	7	3	9	2	11	13	20	4	14	8	15	10	5	16	23	19	12	18	17	24	25	22	26	21	27
	Iran[a]	2	5	6	3	1	9	15	10	4	7	14	12	24	17	8	18	21	19	13	16	11	26	23	20	25	22	27
	Iraq[a]	1	2	5	8	6	7	10	11	19	3	15	14	9	12	4	16	26	17	13	20	18	24	23	21	25	22	27
	Morocco[a]	1	2	6	3	5	9	4	10	20	8	13	15	11	12	7	21	23	17	16	14	19	25	26	18	22	24	27
	Saudi Arabia[a]	1	4	2	6	9	3	16	5	17	8	13	14	22	11	7	15	26	18	10	12	19	21	23	24	25	20	27
	Sudan[a]	1	2	4	5	3	8	11	9	15	6	14	10	18	17	7	19	22	20	12	21	16	26	24	13	23	25	27
	Turkey[a]	2	1	3	4	5	13	18	7	24	6	20	10	11	9	8	16	22	14	12	19	15	27	21	23	25	17	26
	Yemen[a]	1	2	5	8	3	9	6	10	19	4	15	11	12	18	7	22	23	20	13	16	17	26	24	14	21	25	27
Central sub-Saharan Africa	DRC[a,b]	1	7	5	3	4	6	2	9	8	11	10	16	17	15	12	19	14	20	24	18	21	23	22	13	25	26	27
Eastern sub-Saharan Africa	Ethiopia[a]	1	8	4	3	7	6	2	9	5	18	11	21	15	16	10	12	13	19	22	17	25	24	20	14	26	23	27
	Kenya[a]	1	10	5	4	6	9	3	8	2	14	7	15	23	13	12	20	18	11	17	24	16	25	19	21	22	26	27
	Mozambique[a]	2	7	4	1	5	6	3	8	10	14	11	19	18	16	9	13	15	20	25	17	24	23	21	12	26	22	27
	Tanzania[a]	1	10	4	2	8	5	3	7	6	14	11	21	18	17	9	13	15	19	23	16	25	24	20	12	26	22	27
	Uganda[a]	3	9	6	1	8	7	2	4	5	12	11	16	15	19	21	18	20	10	13	25	24	22	23	14	17	26	27
Southern sub-Saharan Africa	South Africa[a]	2	4	3	1	10	11	6	7	5	13	8	15	14	9	21	17	12	16	20	23	18	22	19	24	27	25	26
Western sub-Saharan Africa	Ghana[a]	3	10	5	1	6	4	2	8	13	12	17	15	7	9	11	20	19	14	21	18	25	26	22	16	24	23	27
	Nigeria[a]	3	9	5	4	6	1	2	7	16	11	17	13	8	12	14	18	20	10	21	19	23	25	22	15	26	24	27

Figure 10.2: Cancers Ranked by Number of Deaths in Both Sexes, Globally, by Development Status, and in the 50 Most Populous Countries, 2013

Region	Country	Tracheal, bronchus, and lung cancer	Stomach cancer	Liver cancer	Colon and rectum cancer	Breast cancer	Esophageal cancer	Pancreatic cancer	Prostate cancer	Leukemia	Cervical cancer	Non-Hodgkin lymphoma	Brain and nervous system cancer	Bladder cancer	Ovarian cancer	Gallbladder and biliary tract cancer	Lip and oral cavity cancer	Kidney cancer	Larynx cancer	Multiple myeloma	Other pharynx cancer	Uterine cancer	Nasopharynx cancer	Malignant skin melanoma	Mesothelioma	Thyroid cancer	Hodgkin lymphoma
Global		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
Developed		1	3	7	2	4	11	5	6	8	17	9	14	10	13	15	18	12	21	16	22	20	26	19	23	24	25
Developing		1	3	2	4	6	5	9	12	8	7	11	10	14	15	16	13	18	17	22	19	21	20	24	26	23	25
High-income Asia Pacific	Japan	1	2	4	3	9	7	5	8	11	16	10	18	12	14	6	17	13	23	15	19	20	24	25	22	21	26
	South Korea	1	3	2	4	8	7	5	9	11	13	10	14	12	16	6	20	15	18	17	21	22	24	23	25	19	26
High-income North America	Canada	1	7	13	2	3	12	5	4	8	20	6	9	10	11	17	18	14	22	15	23	19	25	16	21	24	26
	United States	1	9	8	2	3	13	4	5	7	18	6	12	14	10	20	19	11	21	15	22	17	26	16	23	24	25
Southern Latin America	Argentina[a]	1	4	8	2	3	7	6	5	11	10	13	16	14	15	12	19	9	17	20	23	18	26	21	24	22	25
Western Europe	France	1	7	6	2	4	11	5	3	8	20	10	14	9	13	17	16	12	21	15	18	22	25	19	23	24	26
	Germany	1	6	10	2	3	14	4	5	7	19	11	13	9	12	15	18	8	23	16	20	21	26	17	22	24	25
	Italy	1	3	6	2	4	16	5	7	9	22	10	12	8	13	14	18	11	20	15	23	21	26	17	19	24	25
	Spain	1	3	7	2	5	14	6	4	9	21	10	11	8	12	16	18	13	17	15	22	19	26	20	23	24	25
	United Kingdom	1	7	14	2	3	6	5	4	10	21	8	13	11	9	20	19	12	22	15	23	18	26	16	17	24	25
Central Asia	Uzbekistan[a]	2	1	5	7	4	3	10	18	6	9	11	8	16	15	17	12	14	13	24	20	19	23	21	27	26	22
Central Europe	Poland	1	3	12	2	4	16	5	6	11	14	15	8	7	9	13	18	10	17	21	22	19	25	20	26	23	24
Eastern Europe	Russia	1	3	7	2	4	13	5	6	10	12	16	11	14	9	18	15	8	17	22	21	19	25	20	26	23	24
	Ukraine	1	3	14	2	4	16	5	8	11	10	15	12	9	7	18	13	6	17	23	20	21	25	19	24	22	26
Andean Latin America	Peru[a]	2	1	6	3	7	17	8	4	9	5	10	12	16	14	11	20	13	22	18	23	15	27	19	24	21	25
Central Latin America	Colombia[a]	2	1	7	3	5	12	9	4	8	6	10	11	15	14	13	18	17	16	19	24	22	25	21	27	20	23
	Mexico[a]	1	2	3	5	6	15	8	4	9	7	10	13	17	12	14	19	11	16	18	26	22	27	23	25	20	21
	Venezuela[a]	1	2	7	4	5	13	8	3	9	6	10	14	17	11	16	19	12	15	18	23	20	25	21	27	22	24
Tropical Latin America	Brazil[a]	1	3	8	2	5	7	6	4	11	9	12	10	13	14	18	15	17	16	20	19	21	25	22	26	23	24
East Asia	China[a]	1	3	2	5	8	4	6	15	7	12	11	9	13	18	14	20	17	19	22	23	16	10	26	21	24	25
	North Korea[a]	1	3	2	4	7	5	10	20	6	9	12	8	11	16	14	19	18	17	22	24	15	13	26	25	23	21
Southeast Asia	Indonesia[a]	1	3	6	2	5	14	8	19	7	4	10	9	16	12	13	11	17	20	24	18	21	15	23	25	22	26
	Malaysia[a]	1	5	3	2	4	13	7	12	6	10	8	17	15	11	18	16	14	21	20	19	23	9	24	27	22	25
	Myanmar[a]	1	7	2	4	5	13	11	20	6	3	9	14	15	8	12	10	17	21	25	19	18	16	24	26	22	23
	Philippines[a]	1	6	2	4	3	16	8	11	5	7	10	12	19	9	15	13	17	22	23	21	20	14	24	27	18	25
	Thailand[a]	2	4	1	3	6	12	9	16	8	5	13	11	15	14	7	10	17	20	24	19	22	18	23	26	21	25
	Vietnam[a]	2	3	1	4	11	5	12	14	9	8	6	7	16	22	19	10	21	15	24	13	18	20	26	27	17	25
South Asia	Afghanistan[a]	2	1	5	7	3	12	13	14	4	6	10	9	8	18	17	20	16	11	24	23	21	19	26	25	22	15
	Bangladesh[a]	2	3	1	8	9	7	19	18	5	10	4	13	14	11	16	6	20	15	24	12	21	17	25	26	27	22
	India[a]	4	1	2	7	5	3	15	20	10	6	11	13	16	12	17	8	19	14	21	9	25	18	22	27	23	24
	Nepal[a]	2	3	6	5	4	1	16	20	8	7	11	15	17	10	14	9	19	13	21	12	23	18	25	27	22	24
	Pakistan[a]	1	10	5	9	2	3	18	19	8	16	4	13	7	11	15	6	20	12	21	14	22	17	26	27	23	24
North Africa and Middle East	Algeria[a]	1	3	9	4	2	18	13	15	5	12	6	8	10	14	7	22	19	16	17	23	24	11	25	26	20	21
	Egypt[a]	2	7	1	8	3	11	9	10	5	14	12	4	6	17	16	18	13	15	23	21	20	27	24	25	19	22
	Iran[a]	3	1	9	5	8	2	12	7	4	17	14	6	10	16	13	18	15	11	20	24	26	19	22	27	21	23
	Iraq[a]	2	5	4	7	1	15	8	11	3	12	10	6	9	13	17	19	14	16	21	23	18	22	25	27	20	24
	Morocco[a]	1	3	5	4	2	15	9	6	10	7	11	8	12	14	13	19	17	16	24	23	18	20	25	26	21	22
	Saudi Arabia[a]	2	6	1	3	4	12	8	10	7	17	9	5	13	15	11	16	14	18	22	21	25	20	24	27	19	23
	Sudan[a]	1	2	6	4	3	12	11	10	5	13	9	7	8	16	15	18	17	14	22	24	23	19	25	27	20	21
	Turkey[a]	1	2	10	3	4	16	5	6	7	18	11	8	9	12	15	22	13	14	17	27	19	23	24	21	20	25
	Yemen[a]	1	2	6	5	3	13	12	11	4	10	9	7	8	16	14	18	17	15	23	24	20	19	25	26	21	22
Central sub-Saharan Africa	DRC[a,b]	6	2	5	4	3	7	11	9	10	1	8	14	12	17	13	15	16	18	21	22	20	24	23	27	25	19
Eastern sub-Saharan Africa	Ethiopia[a]	7	6	4	3	5	2	10	9	14	1	8	12	16	17	13	15	11	22	18	21	20	26	23	27	25	19
	Kenya[a]	8	4	5	6	3	1	9	11	14	2	7	15	13	10	19	12	18	16	17	22	25	20	24	27	23	21
	Mozambique[a]	7	3	4	1	5	9	12	6	13	2	8	11	15	18	14	16	10	21	19	20	22	26	23	25	27	17
	Tanzania[a]	8	6	2	3	4	5	13	9	12	1	7	10	16	17	14	15	11	22	19	20	21	25	23	26	27	18
	Uganda[a]	9	8	4	6	7	2	13	5	11	1	3	20	12	10	24	16	15	21	18	17	23	14	25	27	19	22
Southern sub-Saharan Africa	South Africa[a]	1	9	8	3	5	2	7	4	11	6	10	17	14	12	21	13	16	20	15	22	19	25	18	23	24	26
Western sub-Saharan Africa	Ghana[a]	9	3	1	6	5	10	7	4	14	2	8	15	12	11	16	19	17	25	18	24	13	23	21	26	22	20
	Nigeria[a]	7	3	1	5	4	10	9	8	13	2	6	15	11	12	14	18	17	21	20	23	16	25	22	26	24	19

Part III

The project

blablabla...

Part IV

CytoScape

The structure of the code in CytoScape follows the already established structure in the ClusterMaker2 plugin. This is to make it easy to maintain for the other contributors of ClusterMaker2. At the same time, the code is modularized enough to make it easy to extract the code, in order to make a separate plugin. The motivation for having a separate plugin to rank clusters in CytoScape gets bigger if ranking clusters provides useful information, in regards of discovering cancer.

Adding new ranking algorithms should be easy both to implement and unit test. Where the algorithms should be placed and how to connect it to the rest of the GUI in order to choose it, should be a no-brainer. Unit testing of the algorithms should be possible without using the GUI. So all of the logic in the algorithms should be excluded from the GUI. If it is not excluded from the GUI, testing the algorithms would require menu interaction and automatization of testing goes out the window. Every test should be able to just run in the terminal without launching CytoScape.

The "Factory" design pattern is used in a original way in ClusterMaker2, and for the ranking cluster algorithms it is done a little bit different. The GUI will not be responsible for knowing every available algorithm, it will be the algorithm factory's. The GUI's way of knowing which algorithms the user should be able to choose from, will be to ask the factory for a list of all of the available algorithms. This is an example that shows by separating the amount of components that have knowledge about a specific piece of information, it becomes easier adding new features. If the GUI also would have to construct a list by itself about the available algorithms, in addition to the factory, we would end up making changes in two places instead of just one. Twice the work for just one change - adding a new algorithm.

Chapter 11

Implementation

11.1 Intro

The *clusterMaker2* documentation for implementing new parts that is not directly connected with clustering algorithms is non-existing (cite documentation!). So I will make a "walk through" of how to do it and how I did it.

11.2 Preparations

The POM-file has to be updated because libraries connected to the pdf exporting functionality is outdated. Updating the libraries, imports and the usage of these libraries in the source code is enough to make the whole project compile at the mentioned date.

11.3 Registering a service

In order to make it easy to implement new cluster ranking algorithms, a new task factory for the ranking of clusters is required. It has to be registered as a service to make it visible to the GUI menu, aswell as maintain the *OSGi* standard of components added to the *clusterMaker2* project. Also, creating a standalone plugin at a later stage will be easier if Ranklust is properly compartmentalized.

Registering a new service listener is a way of keeping the Ranklust part out of clusterMaker2. Though, it is still packaged within clusterMaker2's API, so that it does not become a mess, if extra functionality is to be

implemented while the plugin resides within clusterMaker2.

11.4 GetNetworkClusterTask

ClusterMaker2 already has some REST (insert: reference to REST services + explanation). It is used in Ranklust as a way of getting the clustering results without changin the previous code. Though, there is a flaw in the REST service. If the clustering algorithm you want the results from is not the last clustering algorithm that was run on the network, it will not be able to get the results. For now, this will stay as it is, but it should be brought up for discussion to rather look up the table of the nodes instead of the table of the network, in order to get the clustering results. This is because the table of the network can only hold a single value for clustering algorithms, which will either be nothing, or the last algorithm run on the network. The drawbacks of looking up the table of nodes instead is the time it consumes. But the time consumed to get all of the nodes with a specific clustering result attribute will take a shorter time than to rerun the algorithm you want to have the results from. A third solution can be to just ignore the algorithm used to cluster the network, as it should not have anything to say for the ranking of the clusters. This may end up being the final solution, as it easily can be viewed as a more "sane" (formulate this in a different way). This option requires a new REST function to be added to clusterMaker2. As a whole, it is highly possible that this new REST functionality should replace the existing one. The argument to replace the already existing algorithm is that REST functionality should hide state and implementation from the services using the REST service to get data. The way it is now, this is not true and it should be changed anyway.

11.5 Tables vs pure OO

One thing in particular that deserves to be mentioned is the way networks are handled. The *CyNode* objects themselves does not contain much. This is because all of the information is saved in the form of plain cells in a spreadsheet. This may at first seem like a way to make it easy to show information to the user, but it is also a way of working more efficient with network data (insert: reference to working with tables on networks is efficient). Creating objects with attributes for each node in a huge network will increase the amount of overhead by a pretty noticeable amount. Working with all of the information in the way of a spreadsheet with rows and columns includes a decrease in overhead. A new node is a new row, so in relation to build the network structure, it is not a difference to speak of. The difference comes in when the nodes have several attributes.

In a table or spreadsheet, attributes can be represented as a single column and be created once for the whole network, instead of once for each node object created. This assumes that getting the objects out of the table is possible by either indexing on a number for arrays, or a unique string for map structures. The result is both a lookup, insertion and deletion time of O(1) (insert: reference to Big-O notation). These times is as law as the Big-O notation goes in terms of speed related to the size of a collection inside a data structure. So both the creation of objects goes down, and the retrieval of attribute information is as low as it is possible to get, when we choose to represent the time by Big-O notation. The only drawback with this implementation is that there is no current type of wrapper around the row and column system. So the retrieval of information is not done the most intuitive way. But this is the way CytoScape works as a whole, so changing the way this works is either done through changing the CytoScape source code, or implementing a wrapper as a standalone function inside clusterMaker2 or as a standalone plugin.

11.6 Single value ranklust

A simple, but effective way of ranking clusters, is just to add together the biomarkers values and sum them up in each respective cluster. This will tell us which cluster will have the most cancerous genes and proteins. It is not very advanced, but an analysis show how effective it is. An alternative is also to add scaling to the values. Should clusters with a large amount of biomarkers receive a bonus because of the sheer amount? Should values alone be scaled exponential in order to magnify the differences between more and lesser cancerous biomarkers? This will be tested in the final stages of single value ranking in Ranklust. But as a beginning, the results will come from a pure additive algorithm. The more advanced ones will be implemented only if the additive algorithm show promising results.

11.7 Multivalue Ranklust

Multivalue ranking of clusters creates a larger context. More data can help making more fine tuned results and get the validation of ranking clusters to find biomakers, hit the significance ceiling of 95% certainity. The question here is how should several values be combined in order to create a single value score to rank the clusters by? The best way should be to list every attribute to the user, and present them with options as to how the values should be scaled. An extra option could be how to scale the values in relation each other. The first way is the simplest way to implement so it will be tested first. And as with the single value ranking, if the first scaling of each individual value shows promising results for multivalue ranking,

the values in relation to each other will be considered as a way of producing more accurate results.

11.8 How to get the clusters

As of now, the SimpleRanking algorithm for ranking the clusters has to have access to the cluster results inside the clusterMaker2 source code. The alternative is to work directly on the CytoScape tables containing the same information. The drawback working on the CytoScape tables is the need for going through every single node in the table in order to add it to a cluster, which is something the source code in clusterMaker2 can serve with a single *getter* method. The drawback using the clusterMaker2 code and not the CytoScape tables is that the getter method to clusterMaker2 demands knowledge in the SimpleRanking algorithm of the cluster algorithm that was used. Coupling these two object classes together is not necessary, because of the two mentioned ways of completing this, and the getter method used is actually implemented with the intention of creating a REST API [16] for applications outside CytoScape. A third way could be to implement a way for all of the clustering algorithms to save their results as a *List<NodeCluster>* list. Once again this just moves the problem a bit around and we will end up with a good amount of duplication code to the REST API method of getting the clusters. Then again, having the REST API getter rely on the *List<NodeCluster>* getter could make for a cleaner solution. This is definetively more work and requires every new clustering algorithm to implement a way of creating these node clusters.

Creating the list of node clusters every time a ranking algorithm should run is maybe not such a bad idea. Atleast when we consider that having every clustering algorithm saving its results could result in massive memory constraints. Every node in the node cluster list should only be a java object reference to a node in the CytoScape table. A Java object reference consists of 8 bytes [17]. So we will end up having 8 bytes for every node in the list to a single cluster. Also, to organize it properly, a list of the clusters also has to wrap round all of the clusters, adding another 8 bytes per cluster entry. One simplification can be made, which is using the indexes in the list as a cluster score attribute from the cluster algorithms. That could exclude clustering algorithms that produce more than a single score as a result of the clustering. A sample clustering with the affinity propagation algorithm produces 839 clusters. That is 6,7 megabytes for only the cluster entries in the outer list. From the same results, the biggest cluster consists of over 700 nodes. That is 5,6 megabytes of memory on a single cluster. Total amount of nodes in the network is ends up using extra memory of 89,7 megabytes. And that is just per clustering algorithm.

11.9 Edge attribute information

In order to work with attributes in the edges, you would think that an additional datastructure like a simple java class like **Map** or a class which inherits attributes from the Java **Set**-class[18] would be preferrable. However, this is not necessary, as CytoScape provides both the source and target node in an edge, together with methods to get the primary key from both of them. This allows us to index the node table and get the cluster information in O(1) time.

Recipe for working the nodes and edges. Only the edges require step one, nodes can start at step 2.

1. Find common or best ranked cluster between two nodes (one edge).
2. Add score to the cluster
3. Repeat step 1-2 for every edge
4. Sort clusters based on rank and create a column to represent the score

Step 1 is the thing that will take extra time for the ranking algorithms including the edges.

There is also a question on how we should add the edge scores to the cluster. Should we add the score of the edge when we visit both the source and the target of the edge? Only from when we visit the source, or only the target? Should the user be prompted about this? Should the algorithm take into account if the edge are directional. Or maybe the user should always provide edges in a specified form. A form that says always specify the source node first, then the target node. Maybe it should be specific to the attribute we are working with. The ways to go here are so many. So many that I will just specify one way, and stick to it.

11.9.1 The way I chose

When iterating over the list of edges in the network, check first the source node for a match, then, if it does not have a match in the node table, look after the target node. Then add the score to the cluster. I assume that the direction, if any, is from the source node to the target node. That way, the score only gets added once to the cluster per edge and it does not have anything to say if it is the source or target node. The user should have responsibility for having the edges directed as from the source node, to the target node, if a direction exists at all.

Part V

Methods

Chapter 12

Algorithms

12.1 Single Attribute Additive Method

12.2 Multiple Attribute Additive Method (MNEA)

12.3 Multiple Attribute Multiplication Method (MNEM)

12.4 Seed-weighted Random Walks Ranking Method (SW-RWR)

Chapter 13

Creation of Neo4J database

13.1 Data preparation

I created a Neo4J database and used a slightly modified CytoScape app to communicate with it. Input to the database is a modified version of data. The data started as a PPI file from the StringDB. The file had a size of 3,9GB and consisted of interactions between two proteins. Each line had two Ensembl Protein IDs (ENSP), the type of interaction, what direction the interaction went in and a score for the interaction.

```
item_id_a item_id_b mode action a_is_acting score
9606.ENSP00000000233 9606.ENSP00000000233 binding 0 800
```

I strip this file in order to get a file with just the ENSP IDs. One line per ID.

```
ENSP00000000233
ENSP00000005340
```

Next up is the conversion from protein IDs to gene IDs coupled with Entrez ID. This information I got from UniProt through the stripped ENSP IDs. The file with the data is formatted like this:

```
Entry EnsembleID     Gene names
P84085 ENSP00000000233 ARF5
```

I match the *EnsembleID* from UniProt, which is a ENSP ID, to the *item_id_a* and *item_id_b* data from StringDB (without the 9606. prefix). Then combine the Entrez id, which is the *Entry* from UniProt with *Gene names* with a square (#) between them as a separator to make it easier to split them when creating Cypher and GraphML import queries for Neo4J. This combination makes up a single gene, and each line in the file I write this

information to consist of two genes and a type of binding between them. The type of binding is provided by the StringDB file mentioned first in this section.

13.2 Neo4J setup and data import

13.2.1 Setup

Not much of a setup is needed to get the database up and running. Every setting that was changed in testing and using the database was set in the *conf/neo4j-server.properties* file, located in the directory relative to the Neo4J database installation.

The database location does not need to be set up, but knowing the location of the data that is saved in the Neo4J database is somewhat crucial when working with several database instances.

```
org.neo4j.server.database.location=data/string_mini.db
org.neo4j.server.webserver.port=7474
```

NB! That path is relative to the location the neo4j database is run from.

Turning authentication on/off is also smart for testing purposes. Communicating with a Neo4J database instance that demands authentication also requires the usage of the modified CytoScape app *CyNeo4J*. This is because the app does support everything we need except from authenticating with a database. For the database, this is just having a boolean value *true/false*.

```
dbms.security.auth_enabled=true
```

13.2.2 Import

Importing data into Neo4J is not very mature at this point. You have the possibility to just use Neo4J's query language, *Cypher*. However, importing the whole gene data from StringDB and UniProt combined takes a long time in Cypher, even if the language is the native language to Neo4J. The alternative we used was *neo4j-shell-tools* [19]. It is easy to install and supports import with *CSV*, *Geoff* and *GraphML* filetypes. A GraphML import took under 1 minute on my old and underperforming laptop with 4GB memory, 2GHz CPU and a 128GB SSD [20]. In comparison, the Cypher import took several hours, with the same amount of data. I did not test importing with CSV, because CytoScape does not support exporting data to a CSV file. On the other hand, CytoScape has great support exporting

data to a GraphML file.

The Python scripts used to initialize the Neo4J database with data create either GraphML or Cypher. Importing any of them result in the same type of data in the database, but because of the heavily increased speed using GraphML over Cypher, we used GraphML exclusively to initialize the database. The nodes are created with the forementioned *Entrez ID* as primary key and *Gene name* as the displayed name in the Neo4J GUI.

Chapter 14

CytoScape implementation

The secretome values of genes can be expressed by a single integer value, though, for compatibility reasons Ranklust will require double values. The clustered network can be constructed with an AP, short for Affinity Propagation, clustering algorithm [28]. Affinity propagation clustering concentrates on the nodes in the network that are binding the rest of them together. Using affinity propagation for clustering will produce results that represent a grouping of nodes that are coupled by seemingly unimportant nodes to most clustering algorithms. But the AP algorithm is good at expressing nodes that are not highly connected to many nodes, but rather the nodes that are binding other highly connected nodes together. This results in bigger clusters that can be a target for methods that cure cancer by severing the interaction between biomarker genes.

It is also discussed how AP performs versus Markov clustering (source, both markov algorithm and the "vs" paper). And since Markov algorithm performs better on protein interaction, it will also be used to cluster the networks. An analysis between the rankings that come from the Markov and AP clusterings will be performed, which hopefully will give concrete results as to the pro's and con's of each algorithm. Some questions should be raised as the analysis is done. For example, is both of the algorithms good, but have different uses, even if they are not directly involved with the ranking done after the clustering? Are either AP or Markov useless for a particular type of ranking afterwards?

The information provided through the whole process from what type of nodes (protein or gene), interaction between the nodes and what kind of biomarker we want to identify are all factors that will have a great impact on how all of this should be combined. The order of operations on the network will also effect the result. For example, AP clustering may create a few big clusters and many small ones. At this stage, the results will consist of the biggest clusters constructed from AP clustering that has focus on pure connection between nodes and not their attributes. At this point, the

cluster ranking algorithm of Ranklust will be run in order to produce a picture of potential biomarkers. This picture is the first and simplest step that will be used as a result for an analysis. The analysis in this thesis will focus on validating the cluster ranking scores as ways of indicating potential biomarkers.

In order to validate the scores, I will need to know the state of the patient that the data to create the network came from. As there is no use to just generate results without knowing what they show. They might show connections between them, but without some sort of context the results are useless. The context needed is not very high, but the results from the ranking should be tested for several purposes. For example, is biomarkers from ranking of clusters best for disease disposition, screening, diagnostic, prognostic, prediction or monitoring cancer. I will aim for screening, diagnostic and most of all prognostic usages. As mentioned earlier, the prognostics of prostate cancer often results in 50% of patients receiving treatment that was not needed.

Part VI

Results

Chapter 15

Neo4J

Neo4J was chosen as the database I used to interact with the data from *CytoScape*.

I ended up with implementing a python script to refactor the Neo4J database dump. When run with the command:

```
time python relations_refactoring.py movember.cql refactored_movember.cql
```

The inputs here are time, which benchmarks the time and CPU power used by the program, python which runs the script, the name of the script, inputfile that we want to refactor and the name of the file, which the refactored output resides in. The runtime resulted in:

```
python relations_refactoring.py movember.cql refactored_movember.cql
0,04s user 0,00s system 42% cpu 0,092 total
```

The size of the refactored file was 4500 lines, where about half of them were queries that created relationships between two nodes. The size of the database that was dumped was 2,11 megabytes according to the Neo4J browser interface. Also, the program used 500 megabytes of memory when run on my personal computer with Python 3.4. The queries that created the nodes was bundled into several commits, which ended up being approximately 500 queries per commit. The relationship queries was singled out and can be seen as a single commit for each relationship created between two nodes. This can be improved further, but at the time it was not necessary, as a runtime below 1 second does not introduce any difficulties or problems at the time.

15.1 Data communication

Data communication between the Neo4J database server and the CytoScape instance is done through the CyNeo4J app to CytoScape [21]. CyNeo4J is a CytoScape app that was developed during the Google Summer Code 2014 arrangement. It supports connecting to a Neo4J instance, as well as syncing data up and down, from and to the database server. One thing it did not support was authentication on Neo4J servers. Not having authentication is a serious problem, so I implemented a simple way of getting access to the database server by providing a possibility to insert username and password at the same time the user has to provide a URL to the Neo4J database server instance. Implementation-wise, this only required an extra header to be included in each http request going to the password protected Neo4J database server instance. Every request used the static **Request** class to Get/Post/Put HTTP requests to the Neo4J database server instance. The refactor looked something along these lines in all of the files.

Before:

```
Request req = Request.Post(url)
        .bodyString(call.getPayload(), ContentType.APPLICATION_JSON);
```

After:

```
Request req = Request.Post(url)
        .addHeader("Authorization", auth64EncodedInfo)
        .bodyString(call.getPayload(), ContentType.APPLICATION_JSON);
```

Synchronization time between Neo4J and CytoScape through the CyNeo4J app is a huge timesink. As of now, the time it takes to populate an empty CytoScape network with the gene information from STRING is about 2 hours. This could be shortened by exporting the Neo4J data with GraphML and into CytoScape. Because after the initial data is inside CytoScape, updates to the Neo4J instance goes much faster.

The CyNeo4J also uses a legacy HTTP library to get information from the Neo4J database [22]. It is possible that the performance increases with the new library [23]. The new library supports creating transactions, which implicit gives support for rollbacks in case something goes wrong with the query.

A future improvement to the CyNeo4J app could be to change the communication between Neo4J and CytoScape to be done in GraphML and not Cypher. This is because through this whole process of importing and exporting data from and to Neo4J, GraphML has shown itself to be a superior format over Cypher. Though, to this day, direct queries to a running Neo4J instance has to be done in Cypher, and is not possible in GraphML. So until the Cypher query language gets faster, a convertion

method using GraphML as way of exchange data between CytoScape and Neo4J, could be a candidate for a much better solution than the current one, which is 100% Cypher query based.

Part VII

Analysis

Analysis...

Part VIII

Conclusion

Conclusion...

```
#!/usr/bin/python3


version_str = '<?xml version="1.0" encoding="UTF-8"?>\n'
graphml_xml = '<graphml xmlns="http://graphml.graphdrawing.org/xmlns" xmlns:xsi="http://www.w3.org/2001/XMLSchema-insta
graph_id = '<graph id="G" edgedefault="directed">\n'
graph_id_end = '</graph>\n'
graphml_xml_end = '</graphml>\n'

written_genes = set()

def write_edge(split_line, gml_file, eid):
    """
    Writes an edge in GraphML format to the gml_file
    """

    from_gene_ID = split_line[0].split('#')[0]
    relation = split_line[1]
    to_gene_ID = split_line[2].split('#')[0]

    gml_file.write('<edge id="{0}" source="{1}" target="{2}" label="{3}"></edge>\n'.format(eid, from_gene_ID, to_gene_I


def write_node(split_line, gml_file):
    """
    Writes a node in GraphML format to the gml_file
    """

    from_gene = split_line[0].split('#')
    to_gene = split_line[2].split('#')

    from_entrez = from_gene[0]
    to_entrez = to_gene[0]
    from_gene_symbol = from_gene[1]
    to_gene_symbol = to_gene[1]

    if from_entrez not in written_genes:
        gml_file.write('<node id="{0}" labels=":Gene"><data key="entrezID">{0}</data><data key="name">{1}</data></node>
        written_genes.add(from_entrez)

    if to_entrez not in written_genes:
        gml_file.write('<node id="{0}" labels=":Gene"><data key="entrezID">{0}</data><data key="name">{1}</data></node>
        written_genes.add(to_entrez)


with open('string_db.gml', 'w') as gml_file, open('gene_links_entrez.txt', 'r') as gene_links:
    # Write start of GraphML format
    gml_file.write(version_str)
    gml_file.write(graphml_xml)
    gml_file.write(graph_id)

    # Create nodes
    for line in gene_links.readlines():
        write_node(line.split(), gml_file)

with open('string_db.gml', 'a') as gml_file, open('gene_links_entrez.txt', 'r') as gene_links:
    # Create relation between the nodes
    eid = 0
    for line in gene_links.readlines():
        write_edge(line.split(), gml_file, eid)
        eid += 1

    #Write end of GraphML format
    gml_file.write(graph_id_end)
    gml_file.write(graphml_xml_end)
```

```
#!/usr/bin/python3
open_commit = False
id1 = 1
id2 = 2


def make_query(node1, node2, string_prop1, string_prop2, relationship):
    global id1
    global id2

    name1 = '{name: "' + string_prop1 + '"}'
    name2 = '{name: "' + string_prop2 + '"}'
    new_query = 'MATCH (_{0} {1}), (_{2} {3}) CREATE _{0}-[:{4}]->_{2};\n'.format(id1, name1, id2, name2, r
    id1 += 2
    id2 += 2

    return new_query

def create_nodes():
    global open_commit
    commit_limit = 1000
    lines_before_commit = commit_limit
    created_nodes = 0

    with open('gene_links.txt', 'r') as gene_links, open('string_db.cql', 'w') \
            as cypher_file:
        cypher_file.write('CREATE INDEX ON :Gene(name);\n')
        cypher_file.write('BEGIN\n')
        open_commit = True

        written_genes = set()

        for line in gene_links.readlines():
            splitted = line.split()
            from_gene = splitted[0]
            to_gene = splitted[2]

            if lines_before_commit == commit_limit and created_nodes != 0:
                cypher_file.write('BEGIN\n')
                open_commit = True

            created_nodes = 0

            if from_gene not in written_genes:
                cypher_file.write('CREATE (:Gene {{name: "{}"}})\n' \
                        .format(from_gene))
                written_genes.add(from_gene)
                created_nodes += 1
            if to_gene not in written_genes:
                cypher_file.write('CREATE (:Gene {{name: "{}"}})\n' \
                        .format(to_gene))
                written_genes.add(to_gene)
                created_nodes += 1

            lines_before_commit -= created_nodes

            if lines_before_commit <= 0:
                cypher_file.write(';\n')
                cypher_file.write('COMMIT\n')
                lines_before_commit = commit_limit
                open_commit = False


def create_relations():
    with open('gene_links.txt', 'r') as gene_links, open('string_db.cql', 'a') \
            as cypher_file:
        if open_commit:
            cypher_file.write(';\n')
```

```
            cypher_file.write('COMMIT\n')

        for line in gene_links.readlines():
            splitted = line.split()
            from_gene = splitted[0]
            relation = splitted[1]
            to_gene = splitted[2]

            cypher_file.write(make_query('Gene', 'Gene', \
                    from_gene, to_gene, relation))

create_nodes()
create_relations()
```

```
#!/usr/bin/python3
with open('9606.protein.actions.v10.txt', 'r') as actions_file, open('UniProt_entrez.tab', 'r') as gene_file, open('ger
    actions_file.readline()
    gene_file.readline()

    protein_to_gene = dict()

    for line in gene_file.readlines():
        link_info = line.split()
        if len(link_info) > 2:
            protein_to_gene[link_info[1]] = (link_info[0], link_info[2])

    for link in actions_file.readlines():
        link_info = link.split()
        from_protein = link_info[0][5:]
        to_protein = link_info[1][5:]
        mode = link_info[2].upper()

        if from_protein in protein_to_gene and to_protein in protein_to_gene:
            transformed_link = protein_to_gene[from_protein][0] + '#' + \
                               protein_to_gene[from_protein][1] + \
                               ' ' + mode + ' ' + \
                               protein_to_gene[to_protein][0] + '#' + \
                               protein_to_gene[to_protein][1] + '\n'
            links_file.write(transformed_link)
```

```
#!/usr/bin/python3
with open('9606.protein.actions.v10.txt', 'r') as actions_file, open('UniProt_protein_gene_mapping.tab', 'r') as gene_f
    actions_file.readline()
    gene_file.readline()
    protein_to_gene = dict()
    for line in gene_file.readlines():
        link_info = line.split()
        if len(link_info) > 3:
            protein_to_gene[link_info[2]] = link_info[3]
    for line in actions_file.readlines():
        link_info = line.split()
        from_protein = link_info[0][5:]
        to_protein = link_info[1][5:]
        mode = link_info[2]
        if from_protein in protein_to_gene and to_protein in protein_to_gene:
            transformed_link = protein_to_gene[from_protein] + ' ' + mode.upper() + ' ' + protein_to_gene[to_protein] +
            links_file.write(transformed_link)
```

```
#!/usr/bin/python3

with open('9606.protein.actions.v10.txt', 'r') as infile, open('actions_refactored.txt', 'a') as outfile:
    dictionary = {}
    mappings =
    infile.readline()
    for line in infile.readlines():
        splitted = line.split()
        filtered1 = splitted[0][5:]
```

```
        filtered2 = splitted[1][5:]
        if filtered1 not in dictionary:
            dictionary[filtered1] = 1
            outfile.write(filtered1 + '\n')
        elif filtered2 not in dictionary:
            dictionary[filtered2] = 1
            outfile.write(filtered2 + '\n')
```

```
package edu.ucsf.rbvi.clusterMaker2.internal.algorithms.ranking.advanced;

import edu.ucsf.rbvi.clusterMaker2.internal.api.ClusterManager;
import edu.ucsf.rbvi.clusterMaker2.internal.utils.ModelUtils;
import org.cytoscape.model.CyNetwork;
import org.cytoscape.work.Tunable;
import org.cytoscape.work.util.ListMultipleSelection;

import java.util.List;

public class MNEAContext {
    private CyNetwork network;
    public ClusterManager manager;

    @Tunable(description = "Node attributes", groups = "Biomarker information", gravity = 1.0)
    public ListMultipleSelection<String> nodeAttributes;

    @Tunable(description = "Edge attributes", groups = "Biomarker information", gravity = 10.0)
    public ListMultipleSelection<String> edgeAttributes;

    public MNEAContext(ClusterManager manager) {
        this.manager = manager;
        network = this.manager.getNetwork();
        updateContext();
    }

    public void updateContext() {
        nodeAttributes = new ListMultipleSelection<>(updateNodeAttributes());
        edgeAttributes = new ListMultipleSelection<>(updateEdgeAttributes());
    }

    private List<String> updateEdgeAttributes() {
        if (this.network != null) {
            return ModelUtils.updateEdgeMultiAttributeList(network, null).getPossibleValues();
        }

        return new ListMultipleSelection<>(ModelUtils.NONEATTRIBUTE).getPossibleValues();
    }

    private List<String> updateNodeAttributes() {
        if (this.network != null) {
            return ModelUtils.updateNodeAttributeList(network, null).getPossibleValues();
        }

        return new ListMultipleSelection<>(ModelUtils.NONEATTRIBUTE).getPossibleValues();
    }

    public List<String> getSelectedNodeAttributes() {
        return nodeAttributes.getSelectedValues();
    }

    public List<String> getSelectedEdgeAttributes() {
        return edgeAttributes.getSelectedValues();
    }

    public void setNetwork(CyNetwork network) {
        this.network = network;
    }

    public CyNetwork getNetwork() {
```

```
61            return network;
62        }
63    }
```

Bibliography

[1] URL: http://www.news-medical.net/life-sciences/What-is-DNA-Methylation.aspx (visited on 22/02/2016).

[2] URL: http://www.cancer.gov/cancertopics/types/prostate/psa-fact-sheet (visited on 11/05/2015).

[3] URL: http://www.illumina.com/technology/next-generation-sequencing.html (visited on 11/05/2015).

[4] URL: http://apps.cytoscape.org/apps/clustermaker2 (visited on 22/05/2015).

[5] URL: http://biopython.org/wiki/Main_Page (visited on 20/05/2015).

[6] URL: http://wiki.cytoscape.org/Cytoscape_3/AppDeveloper (visited on 08/05/2015).

[7] URL: http://www.javaworld.com/article/2878952/java-platform/modularity-in-java-9.html (visited on 20/05/2015).

[8] URL: http://www.techopedia.com/definition/18822/design-pattern (visited on 20/05/2015).

[9] URL: http://www.amazon.com/Clean-Code-Handbook-Software-Craftsmanship/dp/0132350882 (visited on 20/05/2015).

[10] URL: https://usegalaxy.org/ (visited on 08/05/2015).

[11] URL: http://irefindex.org/wiki/index.php?title=iRefIndex (visited on 07/05/2015).

[12] URL: http://www.genemania.org/ (visited on 07/05/2015).

[13] URL: http://string-db.org/ (visited on 07/05/2015).

[14] URL: http://apps.cytoscape.org/apps/irefscape (visited on 07/05/2015).

[15] URL: http://apps.cytoscape.org/apps/genemania (visited on 07/05/2015).

[16] URL: http://www.restapitutorial.com/lessons/whatisrest.html (visited on 18/03/2016).

[17] URL: http://www.oracle.com/technetwork/java/javase/tech/vmoptions-jsp-140102.html (visited on 18/03/2016).

[18] URL: https://docs.oracle.com/javase/7/docs/api/java/util/Set.html (visited on 13/04/2016).

[19] URL: https://github.com/jexp/neo4j-shell-tools (visited on 28/04/2016).

[20] URL: http://www.samsung.com/us/computer/pcs/NP900X3E-A02US-specs (visited on 28/04/2016).

[21] URL: https://github.com/gsummer/cyNeo4j (visited on 17/03/2016).

[22] URL: http://neo4j.com/docs/stable/rest-api-cypher.html (visited on 17/03/2016).

[23] URL: http://neo4j.com/docs/stable/rest-api-transactional.html (visited on 17/03/2016).

[24] Anne-Ruxandra Carvunis and Trey Ideker. 'Siri of the Cell: What biology Could Learn from the iPhone'. In: *CellPress* 157 (Apr. 2014).

[25] Pau Creixell, Jüri Reimand, Syed Haider, Guanming Wu, Tatsuhiro Shibata, Miguel Vazquez, Ville Mustonen, Abel Gonzalez-Perez, John Pearson, Chris Sander, Benjamin J Raphael, Debora S Marks, B F Francis Oulette, Alfonso Valencia, Gary D Bader, Paul C Boutros, Joshua M Stuart, Rune Linding, Nuria Lopez-Bigas and Lincoln D Stein. 'Pathway and network analysis of cancer genomes'. In: *Nature Methods* 12.7 (July 2015).

[26] MD Dr Ananya Mandal. URL: http://www.news-medical.net/health/What-is-a-Biomarker.aspx (visited on 12/05/2015).

[27] Michael F.Berger et al. 'The genomic complexity of primary human prostate cancer'. In: *Nature* 1 (2011).

[28] Brendan J. Frey and Delbert Dueck. *Clustering by Passing Messages Between Data Points*. 16th Feb. 2007. URL: http://www.psi.toronto.edu/affinitypropagation/FreyDueckScience07.pdf.

[29] Alexander Haesea, Alexandre de la Tailleb, Hendrik van Poppelc, Michael Marbergerd, Arnulf Stenzle, Peter F.A. Muldersf, Hartwig Hulandg, Clément-Claude Abboub, Mesut Remzid, Martina Tinzld, Susan Feyerabende, Alexander B. Stillebroerf, Martijn P.M.Q. van Gilsf and Jack A. Schalkenf. 'Clinical Utility of the PCA3 Urine Assay in European Men Scheduled for Repeat Biopsy'. In: *European Urology* 54 (2008).

[30] David Heckerman. *A Tutorial on Learning With Bayesian Networks*. MSR-TR-95-06. Microsoft Research, Mar. 1995. URL: http://research.microsoft.com/apps/pubs/default.aspx?id=69588.

[31] P. Jiang and M. Singh. 'SPICi: a fast clustering algorithm for large biological networks'. In: *Bioinformatics* 26.8 (15th Apr. 2010), pp. 1105–1111. ISSN: 1367-4803, 1460-2059. DOI: 10.1093/bioinformatics/btq078. URL: http://bioinformatics.oxfordjournals.org/cgi/doi/10.1093/bioinformatics/btq078 (visited on 22/05/2015).

[32] Rebecca J. Leary, Isaac Kinde, Frank Diehl, Kerstin Schmidt, Chris Clouser, Cisilya Duncan, Alena Antipova, Clarence Lee, Kevin McKernan, Francisco M. De La Vega, Kenneth W. Kinzler, Bert Vogelstein, Luis A. Diaz Jr. and Victor E. Velculesc. 'Development of Personalized Tumor Biomarkers Using Massively Parallel Sequencing'. In: *Science Translational Medicine* 2 (2010).

[33] John R. Prensner, Mark A. Ruin, John T. Wei and Arul M. Chinnayiyan. 'Beyond PSA: The next generation of prostate cancer biomarkers'. In: *Sci Transl Med* 1 (2012).