UNIVERSITY OF TROMSØ UIT

FACULTY OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE

NOWAC Data Exploration

Bjørn Fjukstad

INF-3993 Individual Special Curriculum

June 2013

Abstract

Cancer is the leading cause of death in economically developed countries, and the second leading cause of death in developing countries. There have been significant advances in the different instruments collecting biological data, reducing both cost and data collection time. With the access to big data sources containing peta-scale genomic datasets, the are great possibilities for novel discoveries by analysing and visualizing this data. Advances in cancer research are reliant upon inter-disciplinary collaboration between different sciences, ranging from biology to computer science.

This report will give a thorough description of the entire data analysis and knowledge discovery pipeline for epidemiological studies. It will describe the different epidemiological studies, among case-control and prospective cohort studies, and how these builds a foundation for biology research. It gives an introduction to molecular biology for computer scientists, including the high-throughput instruments used, and describes how state of the art big data systems can be used to manage the immensely large quantities of biological data generated. The report will outline the challenges of visualizing multi-variate data such as biological data, and the modern techniques used in the data exploration community. Finally the first prototype of the NOWAC Explorer, a biological data exploration tool for fast and interactive visualizations in the modern web browser, is described. It will investigate if it can be visualized with sufficient scalability and performance in a modern web browser.

The main lesson learned in this project is that recent advances in epidemiology, biology instruments, and big data systems are making it possible to conduct new studies using massive data sets. However, we found that current biology visualization systems have not fully reached their potential to support novel biological discoveries.

Acknowledgements

I would like to thank my adviser Associate Professor Lars Ailo Bongo and my co-adviser Professor Eiliv Lund for their continuous feedback and support during the course of this project.

I would like to thank Hege Marie Bøvelstad, Dr Vanessa Dumeaux and Knut Hansen for their invaluable help with access to the NOWAC dataset.

Finally, I would like to thank Karina Standahl Olsen for her phenomenally helpful input throughout the project, especially on the first four chapters of this report.

Contents

Abstract **iii**

Acknowledgements **v**

1 Introduction **1**

2 Epidemiology **5**

2.1 Terms and background 5

2.2 Epidemiological studies 7

2.2.1 Ecological studies 7

2.2.2 Cross-sectional studies 8

2.2.3 Case-control studies 8

2.2.4 Prospective cohort studies 8

2.2.5 Randomized controlled trial 10

3 Molecular Biology **11**

3.1 Cells 12

3.2 The working units of the cell 12

3.3 Genetic code 13

3.3.1 Genes 15
3.4 Protein synthesis 16
3.4.1 Transcription 16
3.4.2 Translation 17
3.5 Replication and Expression of Genetic Information 17
3.5.1 Cancer 19

4 Genomics Data 21

4.1 History 22
4.2 Motivation 22
4.3 Genomic Data 23
4.4 Next-generation Sequencing 24
4.4.1 Preparation 24
4.4.2 Sequencing and imaging 25
4.5 Microarray 26
4.5.1 Illumina Human WG-6 Chip 27

5 Big Data Systems 29

5.1 Storage 30
5.1.1 Relational databases 30
5.1.2 GFS 31
5.1.3 Non-relational databases 33
5.2 Analysis 37
5.2.1 MapReduce 37
5.2.2 Apache Hadoop 38

6 Visualization Systems 41

6.1 Backends . . . 42

6.1.1 Tableau . . . 42

6.1.2 Spotfire . . . 46

6.1.3 Wrangler . . . 47

6.2 Biological Data Visualization . . . 49

6.2.1 Challenges . . . 50

6.2.2 Current techniques . . . 52

6.3 Multi-Dimensional Data Visualization . . . 56

6.3.1 Visualization and visual analysis of scientific data . . . 56

6.4 Display wall data visualization and interaction . . . 60

6.4.1 Interaction Spaces . . . 60

6.4.2 WallScope . . . 62

7 NOWAC Explorer 65

7.1 Architecture . . . 66

7.2 Design . . . 67

7.3 Background . . . 69

7.3.1 D3 . . . 69

7.3.2 three.js . . . 69

7.3.3 KEGG . . . 70

7.3.4 NOWAC dataset . . . 70

7.4 Implementation . . . 71

7.4.1 Average gene expression histogram . . . 71

7.4.2 Average year to diagnosis histogram 72
7.4.3 NOWAC Heatmap 73
7.4.4 KEGG Pathway 74
7.5 Discussion and Future Work 74

8 Summary 77

References 79

Appendices

Appendix A Source Code 87

List of Figures

1.1 Outline of the report 3

2.1 Epidemiology is the first stage in the pipeline of designing a bio data exploration tool 5

2.2 A visual comparison of case-control and prospective epidemiological studies. 9

3.1 The fundementals of molecular biology is the second stage of the pipeline towards the NOWAC Explorer 11

3.2 Visualization of the 3D structure of protein Myoglobin. Figure retrieved from [8]. 14

3.3 An illustration of a series of codons of a mRNA molecule. Figure from [61] 15

3.4 Dogma of molecular biology 16

4.1 The production of genomics data is an important step in the pipeline, generating the datasets to be visualized. 21

5.1 Systems for managing the peta-scale genomic datasets has become increasingly important as the datasets are produced faster containing increased ammounts of data. 29

5.2 Different types of data management systems 30

5.3 GFS architecture. Figure from [50]. 33

5.4 The different types of non-relational database systems 34

5.5 An example table in BigTable 36

6.1 Visualization systems 41

6.2 Screen shot of the Tableau Desktop Application 44

6.3 Screen shot of the Spotfire Application 47

6.4 TIBCO ActiveSpaces is a distributed peer-to-peer in-memory data grid that enables heterogeneous applications to share, exchange, and process data in real time. Figure from [60]. . . 48

6.5 A screenshot of the Wrangler interface. The left panel contains history of previously performed operations and suggestions for new transformations. The right panel contains an interactive table with the dataset the user is currently working on. Figure from [36] 49

6.6 Molecular graphics. A Porin molecule shown side by side to put emphasis on how rendering effects can improve the 3D shape. Images from [4, 5]. 50

6.7 Visualization of a genomic microarray dataset using a high-resolution display wall. Image retrieved from [53]. 51

6.8 Graph of interacting protein pairs with proteins (nodes) colored according to cell cycle expression profile cluster membership [10]. 53

6.9 Heatmap generated from DNA microarray data reflecting gene expression values [6]. 54

6.10 Global KEGG map. Figure from [37]. 55

6.11 An illustration of the Interaction Spaces concept. Figure from [52] 61

6.12 The WallScope Architecture. Figure from [32]. 63

7.1 The NOWAC Explorer is the last stage in the pipeline, combining knowledge from the previous steps, integrating them into a single system for interactive biological data exploration 65

7.2 General architecture of the NOWAC Explorer prototype. . . 66

7.3 General design of the system 67

7.4 Visualizations using D3. Figure from [13] 70

7.5 Visualizations using three.js. Figure from [58]. 70

7.6 Histogram showing the average difference in gene expression between the case-control pairs. 72

7.7 Simple histogram showing the average years to diagnosis in the NOWAC dataset. 72

7.8 Heatmap visualization of the NOWAC dataset. 73

7.9 Heatmap show in Figure 7.8 zoomed in to reveal more information 74

7.10 KEGG Pathway visualization 75

7.11 Illustration of the different zoom-levels of the graph visualization. The graph is randomly generated and consists of 50 000 nodes and 25 000 edges. 76

List of Abbreviations

Amazon S3	Amazon Simple Storage Service.
API	application programming interface.
cDNA	complimentary DNA.
CSS	Cascading Style Sheets.
CSV	Comma Separated Values.
CT	Computer Tomography.
D3	Data-Driven Documents.
DBMS	Database management system.
DNA	Deoxyribonucleic acid.
dNTP	deoxyribonucleoside triphosphate.
DOM	Document Object Model.
emPCR	Emulsion PCR.
GFS	Google File System.
GPU	Graphics Processing Unit.
GUI	graphical user interface.
HDFS	Hadoop Distributed File System.
HTML	HyperText Markup Language.

KEGG	Kyoto Encyclopedia of Genes and Genomes.
KGML	KEGG Markup Language.
MDS	Multi-Dimensional Scaling.
MR	MapReduce.
MRI	Magnetic Resonance Imaging.
mRNA	messenger RNA.
NAC	Network Accessible Compute.
NAD	Network Accessible Display.
NGS	Next-generation sequencing.
NOWAC	Norwegian Women and Cancer.
ODBC	Open Database Connectivity.
PCA	Principal Component Analysis.
PCR	Polymerase chain reaction.
PDBMS	Parallel database management systems.
RDBM	Relational Database Management System.
RNA	Ribonucleic acid.
rRNA	ribosomal RNA.
SQL	Structured Query Language.
SVG	Scalable Vector Graphics.
TDE	Tableau Data Engine.
TQL	Tableau Query Language.
tRNA	transfer RNA.
URL	Uniform Resource Locator.
XML	eXtensible Markup Language.

Chapter 1

Introduction

Cancer is the leading cause of death in economically developed countries and the second leading cause of death in developing countries. It is continuing to increase as the world's population is aging and growing. Also because of the cancer-causing behaviour, like smoking, being adopted in developed countries. It is estimated that in 2008 there were about 12.7 million cancer cases and 7.6 million cancer deaths [34]. Many of these cases could have been prevented by introducing tobacco control programs, promoting physical activity and healthier dietary intake, earlier detection and treatment. Of the many types of cancer, breast cancer is the most frequently diagnosed and the leading cause of cancer death among women. The advances are reliant upon collaboration between sciences such as biology, statistics, computer science and epidemiology.

Scientists in different disciplines are collaborating develop new techniques for diagnosis and treatment of cancer patients. In an inter-disciplinary collaboration it is neccessary for researchers from the different fields to have some insight into the different fields they are collaborating with, in order to understand the possibilities and limitations within these. Computer scientists would have to have some understanding in the human body in order to provide solutions usabe for biology end-users

In this research field the analysis of large quantities of data is necessary. The data is immensely heterogeneous, and can for example consist of Deoxyribonucleic acid (DNA) samples from humans, or lifestyle choices, e.g. if a person is a smoker or non-smoker. Data must be collected and stored in a computer system in order for scientist to investigate them. This investigation is usually done using some software allowing researchers to perform advanced statistical analysis of their data, and fast visualization techniques for discovering patterns leading to scientific discoveries. Currently, epidemiology visualization tools are primitive both in their support for large amounts of data as well as the visualization techniques used. Usually these tools are designed as native applications to take advantage of the underlying architecture of the platforms they are running on thus allowing access to hardware resources such as storage and graphics devices. With the improvements in web browsers and HTML 5 [65], web applications are seeing an increased popularity. With new technologies such as WebGL [39], developers are able to utilize Graphics Processing Units (GPUs) through the web browser, allowing more advanced visualizations. With such web applications, researchers can perform data exploration without having to install any third-party software.

One example of a big dataset is the Norwegian Women and Cancer (NOWAC) study, a prospective study which aims at identifying the relationship between lifestyle and the risk of cancer. The NOWAC study started its data collection in 1991. In 2006 the study contained questionnaire information from 170 000 women with repeated collection of information after 4-6 years (2 or 3 times) [42]. In 2006 the collection of blood samples to The NOWAC Postgenome Study [22] started, and after 50 000 blood samples the data collection was finished. In addition there have been collected over 800 biopsies (small samples of breast tissue), half of which are from healthy women and the other half from women with breast cancer.

The goal of the project is to investigate the techniques possible for creating interactive visualizations for this dataset, and implementing a prototype for such a visualization. The project will make use of the advances in web technologies, to aid visualizations of the NOWAC dataset. The data used in this project is a subset of the NOWAC data, where the individuals have a follow-up time of maximum three years. The data contains gene expression measurements for over 400 individuals, collected at different times. This allows for the investigation on how gene expression changes over periods of time. All individuals have been made anonymous, hence do not require (by law) a secure storage and processing facility. Scientifically and comercially valuable, meaning that nor the data or the visualizations will not be publicly available.

In order to get an understanding of the dataset at hand, and the technologies available to design an interactive visualization, the project is split into three parts, (i) a literature study of the concepts in epidemiology, molecular biology and genomic data generation; (ii) a literature study to understand systems for visualizing and handling big data; (iii) development of a prototype of the NOWAC Explorer.

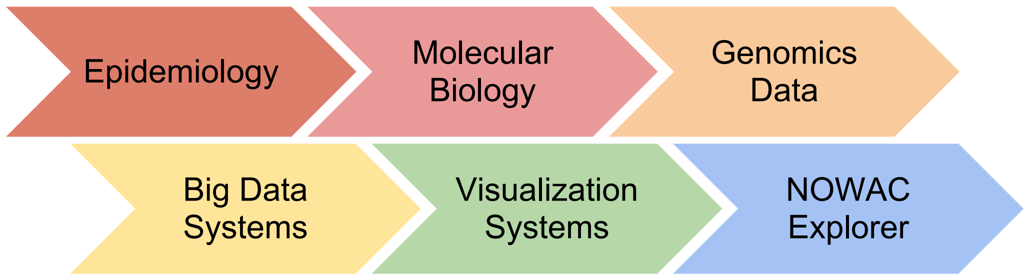


Figure 1.1: Outline of the report

The rest of the report will follow the structure shown in figure 1.1. It starts with chapter two, giving an introduction to epidemiology, the systematic search of causes of diseases. It will look into the different types of epidemiological studies as well as the many terms used in epidemiology. The chapter will give an understanding of the different approaches taken by researchers to design studies, as well as a description of what type of study the NOWAC study is. The third chapter will give an introduction to molecular biology, which contain descriptions of basic concepts like cells, genes, DNA and proteins. This chapter will give an understanding of the building blocks of the human body and introduction to cancer. The Genomics Data chapter, chapter 4, gives an description of the technologies and methods used for collecting biological data. The chapter gives a description of how these methods generate the vast ammounts of data needed to be stored and processed. The fifth chapter will give an overview to the different big data management systems used to store and mange large quantities of data. It describes the possible solutions that can be used to store large quantities of biological data generated by the genomics data tools in chapter 4. Chapter six will investigate different visualization techniques and systems. It discusses how visual analytics tools such as Tableau and Spotfire manage their big data. In addition it presents challenges and methods to visualizing biological and other multi-dimensional data. Finally the chapter discusses future technologies for visualization systems and possible interaction methods in these. The seventh chapter presents a prototype of the NOWAC Explorer, an interactive data exploration tool for investigating genomic data from the NOWAC study.

Chapter 2

Epidemiology

When designing visualization tools for researchers, it is fundamental to understand the type of study being used. If researchers are using a specific method of conducting a study, the visualization may have to adapt to this, e.g. if the study is designed to discover differences between individuals some disease, compared to (similar) healthy individuals.

The NOWAC study is a prospective study designed to identify the possible relationships between lifestyle and the risk of cancer. This chapter gives an introduction to epidemiology and the different types of epidemiological studies.

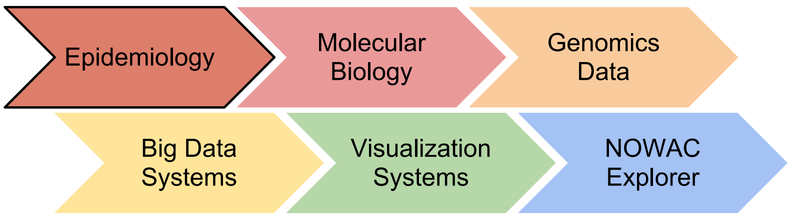


Figure 2.1: Epidemiology is the first stage in the pipeline of designing a bio data exploration tool

2.1 Terms and background

Epidemiology is the systematic search for causes of diseases. In epidemiology the researchers are interested in predicting who will get ill and why, in order to prevent diseases. To do this, epidemiologists study the number of sick individuals in a population, the *risk factors* involved, and *causal relationships*. Risk factors are variables that describe an increase in risk of a disease or being infected. Examples of risk factors are age, ethnicity and gender. Causal relationships are relationships that are causally related, that is one event occurring as a direct result of another. An example of such a relationship can be the relationship between smoking and the risk of lung cancer.

Epidemiological studies often collect large datasets, often from different sources, to detect patterns and gain knowledge. The data can be collected for example through questionnaires or blood samples of individuals om the population. The detection of patterns is often done using advanced statistical methods and is often a collaboration between different scientific disciplines.

There are two main branches of epidemiology, *descriptive* and *analytical* epidemiology. In descriptive epidemiology, the task is to describe how frequently a condition (i.e. a disease) occurs. In analytic epidemiology the main goal is to describe the underlying reasons behind a disease to find the risk factors, and thereby diagnosis and treatment tools.

In order to describe the frequency of a condition, there are two measures used; *prevalence* and *incidence*. Prevalence describes the condition at the current time, for example the number of smokers in a population. Incidence gives a number of how many new occurrences of a given condition within a time frame, e.g. the number of new smokers during a year.

Often in epidemiology the researchers are interested in finding a quantitative measure of the relationship between a risk factor and a disease. This measure can be expressed as either a *relative risk* or an *attributable risk*. A relative risk describes how many times the risk increases when an individual is exposed, compared to the risk when not exposed. This risk is calculated as the ratio between the incidences of the exposed divided by the incidences of the non-exposed individuals. If we can not determine the incidence, like in a *cohort study*, section 2.2, this risk can be estimated by the ratio between the odds of individuals with a disease being exposed (*case*) divided by the odds of individuals without the disease (*control*) was exposed. Attributable risk gives a measure of how many more cases of a disease occurs because of an exposure. Relative risk describes the causal relationship of the disease while attributable risk gives an indication of preventive measures of removing the exposure.

In epidemiological studies the main focus is to find *causal* relationships between risk factors and diseases. Other relationships like *random*, *bias* or *confounding* are challenges epidemiologists face during this process.

A random relationship describes a relationship occurring in a epidemiological study but does not reflect the real world. To determine if a relationship is random or if there is actually a pattern, the p-value is calculated. If this value is below a given significance level, the null hypothesis (there is no relationship between two events, e.g. smoking and lung cancer) is rejected.

Bias is an error in the design, execution or analysis of a epidemiological study, which gives an erroneous indication of the relationship between exposure and the risk of being infected. There are two types of bias; *information* and *selection bias*. Information bias is information researchers are not aware of during the study, e.g. errors in height measurements of individuals. Selection bias covers the samples within the study and how representative they are with regards to the population.

Confounding relationships are relationships where the variables studied leads to over- or underestimation of a relationship. If we suppose a statistical relationship between ice cream consumption and drowning deaths over a given time period. These two correlate positively, resulting that one might try to explain that eating ice cream causes drowning or the other way around. A likely explanation is that there is another variable, the confounding variable, which influences them both, namely the season.

If the observed relationship is not because of chance, bias or confounding, but is actually real, we have a causal relationship. An example of such a relationship between smoking and lung cancer, where the risk of getting lung cancer increases dramatically for smokers.

2.2 Epidemiological studies

There are four main types of epidemiological studies; *ecological studies*, *cross-sectional studies*, *case-control studies* and *prospective studies*.

2.2.1 Ecological studies

Ecological studies concentrates on using groups as observational units, meaning that these studies are concerned with measuring frequency, risk factors and causal relationships between different groups. A group can be any subset of a population where the members have something in common, e.g. ethnicity, age or whether they smoke or not. These types of studies are easy to perform as well as cheap. However there are many causes of errors in these studies, the primary being that the observational unit are groups not individuals.

2.2.2 Cross-sectional studies

In a cross-sectional study the observational unit are individuals in a population. Information retrieval from the individuals are gathered at one single time, meaning that any surveys and/or biological data are sampled at the same time. When collecting multiple pieces of information from the individuals, one is able to study any statistical correlation between the observed information. This study will be able to retrieve results fast, making it possible for conclusions to be drawn relatively fast. However in these studies it is difficult to make any good predictions about the causal relationships between being exposed to an illness and being infected by it.

2.2.3 Case-control studies

Case-control, or retrospective, studies focuses on studying individuals in two groups, one group consisting of individuals infected by a disease (case) and a group consisting of individuals that are not infected (control). The persons within the control group should be representative for the population, but this can prove to be a difficult task. Another difficulty with case-control studies is that they cannot be used to study diseases where individuals cannot be interviewed. Further patients are aware of their illness which may bias their answers in a possible survey. Case-control studies are however very useful, especially when there is a need for an answer fast, for instance locating the source during E.coli outbreaks.

2.2.4 Prospective cohort studies

The previous studies have been retrospective, in the sense that the individuals being studied have already been infected by the disease. In a prospective, or cohort, study the individuals are selected from a healthy sample of the population. This sample can be representative of the population, but is not necessarily required. The individuals are then studied with regards to exposure, for example by monitoring the use of birth control pills or the individual's age. Then one awaits for the number of infected individuals to reach a certain number, in order to draw statistical conclusions from the observations. With such a study the possible risk of biased answers will be largely avoided since the individuals do not know if they will be infected. Since the monitoring may be over a longer time frame, depending on the prevalence of the disease, this study may take a longer time to produce answers.

The NOWAC study is classified as a prospective cohort study, since it collects blood samples of women both before they get breast cancer, as well as at the time of diagnosis.

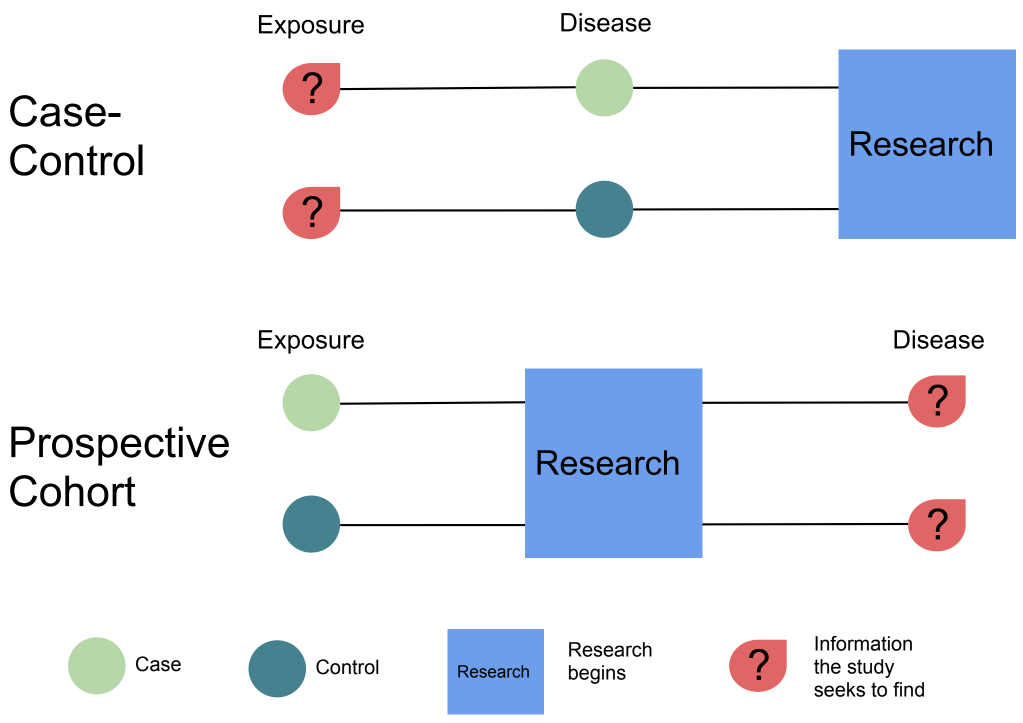


Figure 2.2: A visual comparison of case-control and prospective epidemiological studies.

2.2.5 Randomized controlled trial

A randomized controlled trial is described as the gold standard within epidemiological studies or trials. Like cohort studies this type of study also start with a sample of healthy individuals, but this study will have control over which of the individuals are exposed or not. The sample of individuals are split into two groups randomly, one group being exposed (i.e. given a hopefully effective treatment) and the other not (i.e. given a placebo drug). The study is then performed as long as required to determine if the treatment is effective or not. The study is said to be double blind if neither the recipients of the treatment or the researchers are aware of the individual's membership to the experimental group or the control group.

The main advantage with such a study is the groups created are completely comparable with regards to the factors introduced to influence the risk of being infected. This is because of the randomization done when allocating individuals to the treatment groups. Downsides to this type of study is that they are quite expensive and may take long time to complete. Another comment is that the results found may not be generalizable to the population, since the sample chosen for the study often is a homogeneous group with a common set of features on the contrary to the general population. Another downside to such a study is that it may not even be possible to generate the two groups. In the NOWAC study, having full control over the subjects and dictating their lifestyle choices would not be ethically responsible or even possible.

Chapter 3

Molecular Biology

The goal of the NOWAC study is to understand the impact of lifestyle choices on the risk of having cancer. To understand what cancer is and how it develops in the human body, we must have some base knowledge of how the human body works and the building blocks of our body. With this knowledge we can create helpful visualizations and tools for the researchers working in the field, since we have some understanding of what the visualizations must contain.

This chapter gives an introduction to the biological aspects of this project. First a discussion on cells will be presented, continued by a brief introduction to proteins, followed by a section on DNA and protein synthesis. The chapter ends with a section on genetic engineering which completes the foundation for the rest of the report.

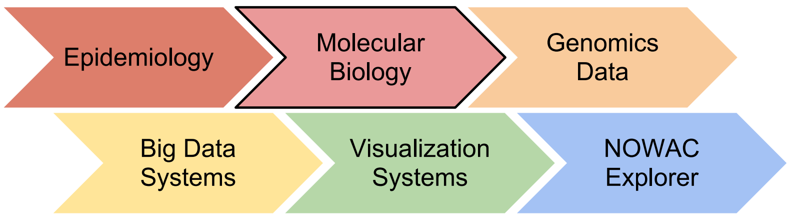


Figure 3.1: The fundementals of molecular biology is the second stage of the pipeline towards the NOWAC Explorer

3.1 Cells

Cells are the smallest units an organism can be divided into, that still possesses the functions performed by living organisms. These cells perform many of the same tasks, involving exchanging of materials with their environment, self duplication, transmitting and receiving signals with their environment, and the synthesizing of *molecules*. The human body contains about 100 trillion cells [41], while the simple organisms like bacteria are single celled. Within the human body all cells are specialized. This specialization task is known as *cell differentiation*. When cells have been specialized they migrate to form tissues. These tissues form *organs*, which in turn form *organ systems*.

Cells are categorized very broad into four categories: *muscle cells*; *nerve cells*; *epithelial cells*; and *connective-tissue cells*. Muscle cells generate moment and mechanical forces within the human body, for example to move limbs or to pump blood within the heart. Nerve cells are the cells responsible for controlling the activities of other cells. These cells achieve this by initiate and conducting electrical signals. The epithelial cells are mainly barriers within the human body, located on the surfaces that either cover the body, organs or other structures within the body. The last type of cells, the connective-tissue cells, are as the name hints, responsible for connecting and maintaining the inter-structure of the body. Importantly, many other cell types exist, for example those of the immune system.

As mentioned above specialized cells form tissues, and as with cells there are four main types of tissue; *muscle tissue*, *nerve tissue*, *epithelial tissue* and *connective tissue*. Tissue is defined both as aggregate of a single type of specialized cell and the general cellular fabric of any organ or structure.

Organs are compositions of any these four types of tissues, which can be structured in various ways. Often organs are organized into small subunits referred to as *functional units*. Organ systems are a collection of organs who together perform some function, for example the digestive system.

3.2 The working units of the cell

Poteins are large molecules found in any living organism. They vary in size and function. Some function as messengers within an organism, transmitting messages between cells. In the human body, proteins aid in many processes, for example by turning food into energy, the process of cell development, and contribute to the distribution of oxygen in the body. Proteins are fundamental

A protein is composed of carbon, hydrogen, oxygen, and nitrogen molecules, as well as other small amounts of other material. It is a macromolecule in the sense that its a chain of more than 50 subunits, known as *amino acids*. Chains with less than 50 amino acids are known as *peptides*. Amino acids are molecules with special properties. There are 20 different amino acids, where all except proline has a common formula. Since the chains can have up to 1000 or more amino acids, and there are 20 amino acids in total, there number of different proteins is extremely large. Proteins can also be composed by a number of polypeptide chains, these are known as *multimeric proteins*.

The structure of the proteins are determined by two factors; (i) the number of amino acids; and (ii) the order of amino acids in the chain. This structure is essential for how the protein functions, and the 3D structure is known as the conformation of a protein. Figure 3.2 shows the 3D structure of the protein Myoglobin.

3.3 Genetic code

All cells within an organism contain the same genetic information, this genetic information is stored within nucleic acids which are responsible for storage, transmission and expression. There are two types of nucleic acids, Deoxyribonucleic acid (DNA) and Ribonucleic acid (RNA), each with its properties. DNA are responsible for the storage of genetic information, while RNA is in charge of decoding the information stored within DNA. Both of these acids are composed of a sequences of subunits known as *nucleotides*.

DNA are composed of two classes of nucleotides, the purine bases adenine (A) and guanine (G), and the pyrimidine bases cytosine (C) and thymine (T). DNA is structured as two chains of nucleotides, held together by hydrogen bonds between the purine and pyrimidine bases. Because of the molecular structure of the nucleotides, G is always connected to C, and A is always connected to T. This is the principle of complimentary base pairing.

From these four bases, DNA codes information using three-letter keywords (codons) from one strand of DNA. The codons specifies the sequence of amino acids, but can also be used as stop signals to denote the end of a sequence.

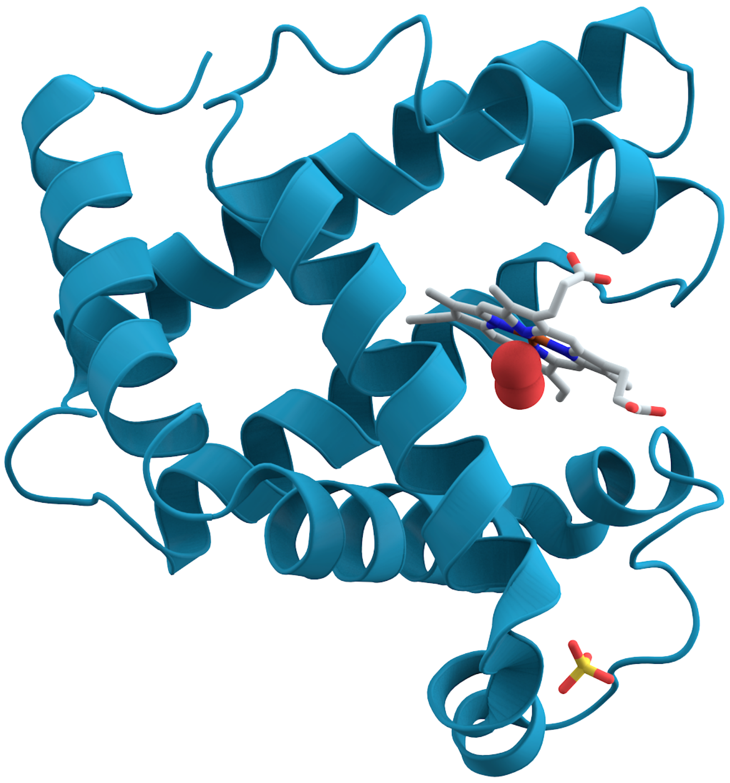


Figure 3.2: Visualization of the 3D structure of protein Myoglobin. Figure retrieved from [8].

98% of the human genome do not contain any protein coding information [23], but serve mainly regulatory function.

There can be more than one sequence of bases specifying a single amino acid. As an example, the sequences C-C-A, C-C-G, C-C-T and C-C-C all specify the amino acid glycine. From this we can deduce that there are multiple sequences of amino acids that can describe a protein.

RNA on the other hand consist only of a single chain of nucleotides, but where DNA has the pyrimidine base thymine, it is replaced by a pyrimidine base uracil (U). This results in a A-U pairing in the nucleotide chain. Figure 3.3 illustrates a messenger RNA (mRNA) molecule containing series of codons.

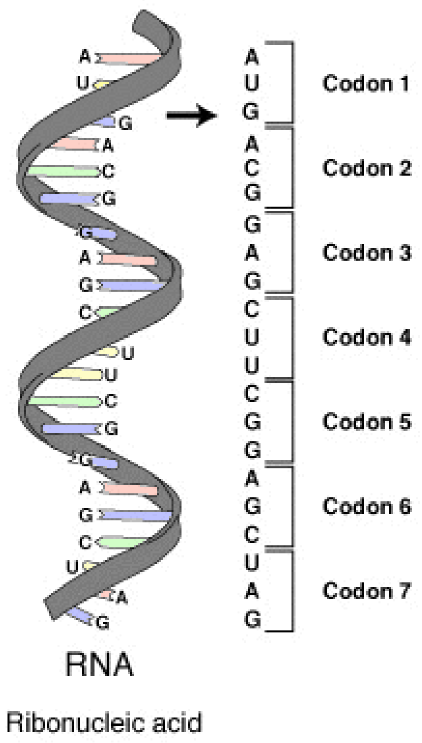


Figure 3.3: An illustration of a series of codons of a mRNA molecule. Figure from [61]

3.3.1 Genes

A *gene* is a sequence of DNA nucleotides which specifies the amino acid sequence of a single polypeptide chain. Genes are used during the protein synthesis to specify how particular proteins should be constructed. genes of an organism have been passed on to it from its ancestors. A *genome* is the total genetic information stored in DNA within a typical cell of an organism. The human genome contains approximately 20 500 genes, but the number of proteins is considerably higher due to redundancy of the genetic code, and multiple steps of regulation and modification. In each cell there are 46 *chromosomes*, which is a piece of DNA containing many genes. Chromosomes also contain a class of proteins called histones. These proteins package DNA and controls its functions.

3.4 Protein synthesis

It is important to understand how proteins are put together, or *synthesized* in order to understand the importance of genes. These genes specify exactly how a protein should be put together, so that they can contribute in the different processes within the body.

In order to synthesize the proteins within an organisms, both DNA and RNA are used. The protein synthesis happens within a specific part of the cell, separated from where the DNA is stored. The sequence information of DNA is transferred to RNA in a process called *transcription*, in order for the genetic information to be used in the protein synthesis. RNA can then be used to synthesize a protein in a process called translation. See figure 3.4.

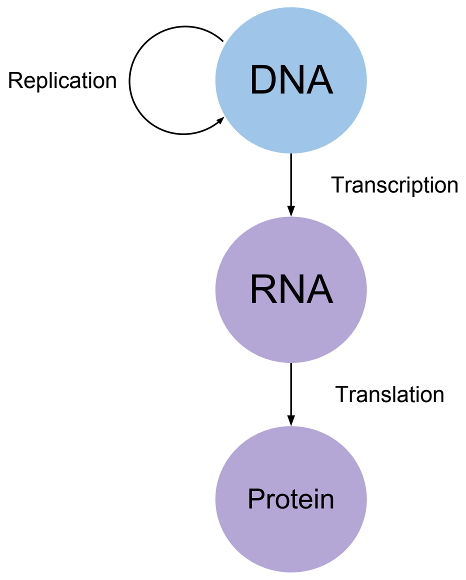


Figure 3.4: Dogma of molecular biology

3.4.1 Transcription

As mentioned, information needs to be transfered to RNA which transports the message from DNA and to the location in the cell where protein synthesis occurs. Information is transcribed from DNA to a special type of RNA called mRNA. The DNA strands are separated from each other, in order for the DNA strand to act as a template for the mRNA, according to the rules of complementary base pairing. A special *enzyme*, which is simply a protein which acts as a catalyst, initiates this process and is responsible for the creation of the mRNA. Before the RNA is transfered, some modifications to it is performed. These modifications remove the areas of the RNA which do not contain any protein coding information. This process is termed splicing.

3.4.2 Translation

When the mRNA has been spliced it is transfered to a part of a cell called a ribosome. The ribosome is constructed from about 80 different proteins as well as ribosomal RNA (rRNA). This structure is responsible for translating the RNA and constructing proteins from these translations. When the ribosome receives a mRNA it generates a chain of amino acids based on the sequence in the mRNA. Another type of RNA, transfer RNA (tRNA) is responsible for transferring the correct amino acid to the correct mRNA sequence. tRNA are the smallest of the different RNA molecules, but is very important in that it binds the correct amino acid to the correct RNA codon. Within the ribosome the incoming amino acid is connected to the protein chain generated.

When the protein has been assembled it must be folded to complete its creation. It is this folding that characterizes each protein. Smaller proteins are folded simultaneously as they emerge from the ribosome, while larger proteins must be folded when the protein chains are complete. When proteins have been folded they can undergo some post-translational processing if necessary. This process could split a protein into several smaller peptide chains.

3.5 Replication and Expression of Genetic Information

In humans the egg and sperm cell each contain 23 molecules of DNA, each containing a different set of genes. From these 23 chromosomes, 22 contain genes which produce proteins which govern most cell structures and functions. These are known as *autosomes*. The last chromosome is known as the sex chromosome determining the gender. When the egg is fertilized it contains in total 46 chromosomes, 44 of these are autosomes and two sex chromosomes.

DNA replication is the process of taking a single DNA molecule and producing two identical copies of the molecule. As with the synthezising of RNA, an enzyme is involved with splitting a DNA molecule and creating a new DNA. This enzyme is known as DNA polymerase. When the DNA's double helix structure is divided into two individual strands, the DNA polymerase reads from the two strands and pairs each base according to complementary base pairing. The process completes with two identical DNA molecules. Of course, since there are 40 trillion cells in the adult human body, coming from this one fertilized egg cell, errors are bound to occur. After the creation of the new DNA molecule, they go through a process of proof reading which should correct any errors. However, some errors goes through unnoticed after this process.

The rate of cell division varies from cell type, but the fastest growing cells can divide about once every 24 hours. The capacity of undergoing cell division also depends on the type of cell. Some cells can undergo cell division continuously trough is lifetime (like skin cells), while others (like nerve cells) rarely or never divide.

When DNA replicates, there is a large chance for it to be corrupted. Any alteration in the nucleotide sequence in the DNA is known as a *mutation*. Mutations can occur if a base is replaced by another one or if parts of the DNA strands are deleted or added. The latter may cause the loss of an entire gene or gene group.

There are three ways a mutation might affect a cell; either it may cause no noticeable changes to the cell, it could modify a cell but still be compatible with cell growth and replication, or it could cause cell death. There are mechanisms for repairing DNA, but they depend on the error only occurring in one of the two DNA strands. Repairing DNA is performed by special enzymes and is crucial for long-lived cells which rarely divide.

The effect of a mutation can effect either the organism or its offspring. If the mutation occurs in an egg or sperm cell the mutation will be passed on to its offspring, however it may not affect the organism itself.

Genetic diseases are diseases which are a result of inheritance of mutant genes, rather than infections or viruses. There are over 4000 of these [62] which categorize into three categories; *single gene diseases*, *chromosomal* and *polygenic diseases*. The first only require a single gene to produce the disease, while the last two require multiple genes. Chromosomal diseases are the result of addition or deletion of chromosomes or partions of these. A classic example of a chromosomal disease is Down's Syndrome. Plygenic diseases require interactions between multiple genes, which individually can cause no harm, but together produces disease. Examples of polygenic diseases are diabetes or cancer.

3.5.1 Cancer

As mentioned cancer is a polygenic disease, but it is not however an inherited genetic disease depending on mutations within the reproductive cells. Cancer however are results of gene mutations which can occur in any cell anytime. A cancer cell is a cell which has a capacity of uncontrolled growth. Of the different types of cancer, about 90% develop in epithelial cells and are known as carcinomas. Of the remaining 10%, cancer developing in muscle and connective tissues are known as sarcomas and cancer derived from white blood cells are called leukemias and lymphomas [62]. Cancer is primarily a disease of old age, due to accumulation of mutations over many years. Some inherited mutations may increase the risk of cancer considerably, e.g. the BRCA gene mutation that increases the risk of breast cancer.

Tumors are results of the uncontrolled growth of cancer cells. These are a growing mass of tissue, which could be either benign or malignant. Benign tumors have not invaded surrounding tissues, while malignant tumors have spread to their surrounding tissues interrupting their functions.

Chapter 4

Genomics Data

In order to collect genetic information about cancer patients, one must be able to measure or collect information about a patient's DNA. This is done in a process called *DNA sequencing* which determines the the ordering of nucleotides within a strand of DNA. This process determines the sequence of the four bases adenine, guanine, cytosine and thymine. To sequence a DNA molecule, the main strategy is to first *reverse transcribe* the RNA strand into a complimentary DNA (cDNA) molecule and then use the same sequencing technique as for DNA molecules.

In the NOWAC study, this DNA sequencing is crucial in order to collect information about the subjects in the trial. With new technology this is achievable in just a work day, and has helped researchers.

This chapter will give an introduction to the technologies used in the collection of genomic data. First a discussion on the topic of genomic data and the problems concerning it is given, before the main techniques and technologies used in DNA sequencing is presented.

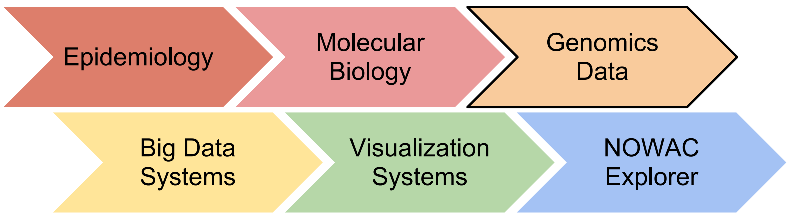


Figure 4.1: The production of genomics data is an important step in the pipeline, generating the datasets to be visualized.

4.1 History

The sequencing of the human genome stands as one of the major scientific achievements of the twentieth century [49]. It is a recent idea to sequence the human genome, but it could not be possible if not for the decades of advances within biology and compute power. Already in 1953 researchers Watson and Crick discovered the DNA structure. In 1973 the first genes were replicated by Berg and Cohen, before Sanger in 1975 sequenced the first small molecule. In the early 1990's the idea behind sequencing the entire human genome was conceived, but not until twenty years later the work was near completion. In 2003 about 20 500 genes was described. The next-generation sequencing-era started in 2004 with the introduction of massively parallel sequencing platforms [18].

4.2 Motivation

The expression of genes in different cells, tissues or organisms, are constantly changing during health, adaption, toxicity, disease and aging. However an individual's genetic blueprint remains relatively static during its lifespan. At any point in time only a portion of a genome is expressed in specific cells and tissues, this portion is named the *transcriptome*. Of the 20 500 different genes in the human genome the transcriptome consists of about 10-15 000 different genes expressed in tissue.

There are different levels of gene expression; the *genome level* consisting of the genes, the *transcriptome level* consisting of the genes transcribed at a given point in time, the *proteome level* consisting of proteins making up cells and tissues, and the *metabolome level* consisting of small molecules and metabolites. The motivation behind sequencing DNA is that it allows for the study the genes and their effect on the proteins synthesized and their contribution in the different metabolisms.

As mentioned in the previous chapter there is a possibility for mutations in the DNA during replication. This mutation may cause a number of diseases, and if one wants to detect such a mutation the mutant DNA must be sequenced and compared to the DNA of a healthy individual. However, the problem is that there are many mutations of a DNA that may not produce any observable biological consequence.

Detecting such mutations have been found by sectional resequencing of DNA, using Sanger-methods which could just sequence shorter regions of the human genome. Newer technologies makes it practical sequencing much larger regions of the human genome.

4.3 Genomic Data

The size of a genome in an organism range from just under 4,000 base pairs, found in the bacteria Bacteriophage MS2 [25], up to 670 billion base pairs found in the amoeboid Polychaos dubium [47]. Humans have two copies of their inherited genome of 3.2 billion base pairs each [63].

In the later years, the technology used in the collection of genomic data has seen a tremendous improvement. Today, single week-long sequencing runs can produce as much data as did entire genome centers just a few years ago [35]. With this growth in data generation, the need for effective solutions for both handling such vast amounts of data, as well as analytic tools has become necessary for the advances within the biological and medical sciences.

The growth of the amounts of data produced by the sequencing labs across the world has out outperformed the advances both in compute power and storage capacity [35]. Sequencing output is now doubling every 9 months, leaving disk storage and high-performance computation fields far behind. Analyzing the collected data is also a growing problem, and it is today at a much higher cost than producing the actual data, often as much as a factor of ten higher.

The problems with many of the current sequencing technologies is the raw data produced. Many of the systems use digital imaging for capturing the bases being sequenced, leading to the creation of many terabytes of images every day. These images must then be processed in order to read the specific base encountered. Some systems use real-time processing of the images to reduce the cost of storing the images, and output only the base call. This reduces storage required with several orders of magnitude. Even if the storage capacity required can be reduced, when moving the data downstream of the sequencing machines bottlenecks emerge. Often research institution have access to a high-speed internet connection, often with connection speeds at gigabit level. However, if one should download the entire dataset for the 1000 Genomes Project, which is currently at over 200 terabytes [44], it would take several weeks before the download would be finished. To solve this problem, using service-oriented architectures where the computations are moved closer to the data is in use. However, data security and privacy issues must be taken into account when working with human data and renting services from a third party.

Another solution to the storage problem is to introduce *reference genomes*. With these reference genomes, instead of storing an entire genome, the difference to the reference genome is stored. Since mutations on a genome typically is less than 0.1% of the data, it has potential for reducing storage costs. This method has not been fully adopted, and currently there exists no human reference genome.

4.4 Next-generation Sequencing

Next-generation sequencing (NGS) is a term used to describe the newer technologies that allow massively-parallel sequencing of genes. The different techniques available today use different methods for sequencing, but they all share a common pipeline consisting of three phases: *preparation*, *sequencing and imaging*, and *data analysis*.

Of the many NGS instruments, the Roche 454 Genome Sequencer is considered the pioneer within NGS. It was released in 2004 and was able to simultaneously sequence hundreds of thousands DNA fragments, with read lengths of over 100 base pairs. In 2006 the Illumina Genome Analyzer was released. This sequencer generated tens of millions of shorter 32 base pair reads. Applied Biosystems' SOLiD was the last sequencer to arrive the marked, but with capabilities of sequencing 400 million 50 base pair reads. The first single-molecule sequencer was the Helocos BioScience HeliScope, which was capable of producing 400 million 25-35 base pair reads [18].

4.4.1 Preparation

There are two methods used in the preparation stage, either clonally amplifying templates which originates in single DNA molecules or using single DNA molecules as templates.

Common to the preparation stage in most of the sequencing techniques is to break down DNA into smaller strands, from which templates are created. This template is then attached to a solid surface, usually glass, and with the spatially separated templates the sequencing of thousands to billions of nucleotides can occur simultaneously.

For clonally amplifying DNA two methods are common; either *Emulsion PCR (emPCR)* or *solid-phase amplification.* emPCR builds on the Polymerase chain reaction (PCR) technology, which can from only a few copies of DNA generate millions of copies of the same DNA sequence. emPCR creates a library of fragments or mate-pair targets, on which specific DNA molecules can attach to. After DNA is attached to the targets, it is separated into sngle strands and captured onto specific beads. These beads will only allow one DNA molecule to be attached. When DNA have been attached to the beads they can be immobilized and captured onto a surface or inserted into wells where the sequencing reactions may occur. In solid-phase amplification the DNA is broken into fragments which is attached to 'primers' on each of its ends. These are then attached to a surface, where a reaction that clones each of the DNA fragments occurs, creating clusters of these.

Using the single-molecule temples one can reduce the amount of DNA material needed. These methods do not rely on PCR which could produce mutations in the clonally amplified templates. This method is often used in RNA-seq [43] because of the small amounts of mRNA used. There are three different approaches, each involving the immobilization of DNA onto some surface.

4.4.2 Sequencing and imaging

Depending on the methods used in the preperation stage, sequencing and imaging must be performed accordingly. Upon imaging of the DNA with the clonally amplified templates, the signal recorded is a consensus of the nucleotides or probes added at a given cycle. Because multiple signals must be processed, this demands high efficiency on the imaging devices used. When using single DNA molecules as templates, multiple nucleotide or probes could be added at any given cycle, introducing errors.

Cyclic reversible termination is a imaging strategy used for sequencing DNA. In this strategy the templates are washed with fluorescently modified nucleotides, which attaches to the templates one nucleotide at a time. When a nucleotide is attached to the template a light is emitted, which can be captured with a camera. The different bases are labeled with a different fluorescent dye, emitting light at different wavelengths. When the nucleotide has been added the DNA synthesis process terminates, and the remaining nucleotides not being attached is washed away. Then the process continues with the surface being washed with nucleotides, each hoping to attach to a template. This continues until the DNA has been sequenced. The Illumina/-Solexa Genome Analyzer (GA), which has dominated the Next-generation sequencing marked is using this technique [43].

Sequencing by ligation is different method which uses DNA ligase, a specific enzyme that facilitates the joining of DNA strands together. In this method flourescently labeled probes attaches itself to a complementary sequence, before being joined by the *DNA ligase.* Non-ligated probes are washed away and fluorescent imaging is used to determine the identity of the ligated probe [43]. This is repeated over again until the DNA has been sequenced.

Another method for sequencing is *Pyrosequencing.* Instead of modified nucleotides being added which terminates the DNA synthesis in every step, single deoxyribonucleoside triphosphate (dNTP) are washed across wells that contain template beads and some other enzymes. When the dNTP attaches to the bead, it starts a chemical reaction emitting light which is captured by a camera. The order and intensity of the light peaks are recorded as flowgrams, which reveal the underlying DNA sequence [43].

Real-time sequencing is one of the newest technologies, which involves imaging the continuous addition of the flourescently labeled nucleotides during DNA synthesis. What separates this from reversible terminators, is that the synthesis is not halted when a nucleotide is attached.

Another technology that is gaining momentum, is *semi-conductor sequencing.* One such sequencer is the Ion Proton sequencer which claims to be able to sequence an entire human genome for just $1000 in a few hours. The Ion Proton sequencer uses advances in semi-conductor technology to achieve this low cost, with the sequencing being done on a single chip [54]. The chip itself contain imaging technology for capturing the chemical sequencing reactions.

4.5 Microarray

In principle DNA microarrays are matrices on some solid surface (often glass), with attached DNA oblinucleotide probes attached to. mRNA is sampled from an individual, and then purified, amplified and labeled. Afterwards it is placed on the microarray allowing it to hybridize by complimentary base pairing with the oblinuclotides on the microarray. Any excess material is washed away and image analysis determines the quantities of mRNA in the sample.

There are three main objectives of microarray sequencing, the most important in the NOWAC study being the discovery of differently expressed genes. With

4.5.1 Illumina Human WG-6 Chip

In the NOWAC study, the prospective data collected has been collected using the Illumnia Human WG-6 chip. This is a micoarray chip using 50 mer oblinucleotide probes attached to beads randomly assembled and replicated in each array. The probes used target transcripts/positions in the genome.

Chapter 5

Big Data Systems

There are numerous systems for managing the large quantities of data used in visualization systems. These systems are required to handle the three V's of Big Data: volume, variety and velocity [33]. Volume describes the size of the data stored, often ranging from terabytes to petabytes. Variety describes the differences within the data itself, whether the data is structured, unstructured, or some combination of the two. The last V, velocity, describes the speed data is generated at.

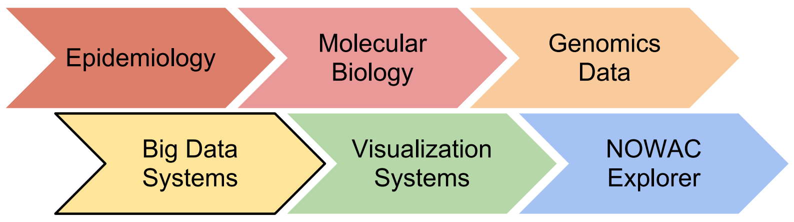


Figure 5.1: Systems for managing the peta-scale genomic datasets has become increasingly important as the datasets are produced faster containing increased ammounts of data.

There are two main approarches to managing and interacting with big data, either using a Database management system (DBMS) or a storage solution built on top of the MapReduce framework on top of Google File System (GFS) or Hadoop Distributed File System (HDFS). DBMS are fit for managing large quantities of structured data, which can fit into a well defined schema. This will require the system to convert the data when loading it into the system, but does not have to worry about any parsing during execution. MapReduce (MR) on the other hand requires the developer to write any logic into the application, which can understand the underlying structure of the data.

This chapter will given an introduction to big data systems. It will discuss the two different branches of management systems, either built on top of Relational Database Management Systems (RDBMs), or GFS or HDFS. The chapter is split in two, discussing both pure storage systems and analytical engines. The storage systems consists of a discussion of relational database systems, followed by an introduction to GFS used in several storage systems. The storage section continues with a discussion of non-relational database systems, such as mongoDB and BigTable, and the chapter ends with an introduction to analytical engines using MapReduce and storage systems built on the popular open source implementation of MapReduce Hadoop. Figure **??** illustrates the different types of data management systems.

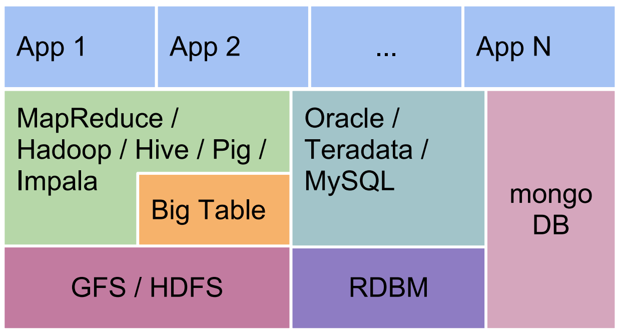

App 1
App 2
...
App N
MapReduce / Hadoop / Hive / Pig / Impala
Big Table
Oracle / Teradata / MySQL
mongo DB
GFS / HDFS
RDBM

Figure 5.2: Different types of data management systems

5.1 Storage

5.1.1 Relational databases

A DBMS is a system for managing quantities of organized data. Traditional database systems were often placed on powerful mainframe computers, often with specialized hardware, to manage database and transaction processing tasks. Traditionally, the relational model have been opted for and used in DBMS, where data is stored in *tables*, and every item is identified by a *key*. The access of such databases is done in a query language, such as Structured Query Language (SQL). What is important in relational databases, is the possibility of linking information stored in different tables together.

From the DBMS systems, Parallel database management systems (PDBMS) have been developed. These systems take advantage of the relational data model, which is suitable for parallelization, and are built on clusters of computers with commodity hardware, communicating and exchanging data over an interconnection network. The database is often partitioned among different computers, allowing different computers to operate on different parts of the dataset, facilitating parallelism. Partitioning is done either by column-wise or row-wise, depending on the different systems. Row-wise partitioning will distribute rows to different computers, while column-wise distributes columns to different computers. Row-wise partitioning is suitable if the database should normally read entire records at a time, e.g. in a phonebook looking up both name, number and address for a single person. If a column-wise partitioning is suitable if the access patterns are for the same attribute of multiple rows, e.g. looking up a range of phone numbers. The partitioning of data can be done using multiple techniques. The simplest being *round-robin* where rows are distributed in round-robin fashion to the available computers. Another approach is to use *hash-partitioning* which computes a hash of each row, and distributes it based on this hash. Given that the hash function distributes rows uniformly, this technique will function as a load balancer, and ensure that data is evenly distributed among the machines. However if the user is likely to access certain ranges at a time, *range partitioning* might be feasible. This will parition the data based on some pre-defined ranges, for example that in a phone book all numbers starting with 555 is stored on a single machine. A problem with range partitioning is that data may be unevenly distributed, with a risk that larger parts of the data is stored on a single node.

A property of DBMS that has made them attractive on the marked, is their use of *indices*. These are data structures that improves the data retrieval time, both for random and sequential access. An index is often just a copy of one or more columns from a database table that requires faster access times.

5.1.2 GFS

GFS is a distributed file system developed at Google. It is designed to provide fault tolerance while running on inexpensive commodity hardware [28]. While it uses an architecture with a single point of failure, it provides high performance to a large number of clients. When designing this file system, the developers identified both current and anticipaded workloads, and created a file system that would target these. They identify four insights, which should be optimized for: (i) failures are the norm; (ii) files are huge, often many gigabytes; (iii) file operations are often appends, not random writes within a file; and (iv) designing the file system application programming interface (API) and application benefits the overall system.

GFS runs on a cluster of commodity machines, and consists of a single master and multiple *chunkservers*. Files are divided into *chunks* of fixed size, 64 MB, which are assigned a *chunk handle* upon creation. The chunks are stored as regular files on the chunkservers, and GFS stores on default three replicas for each chunk, but this can be changed by the user.

The master stores metadata regarding the location of the chunks and the mapping between chunks and files. It also controls which chunkserver should hold specific chunks, migration of chunks between chunkservers and the garbage collection of chunks. To detect any failing chunkserver the master periodically sends heartbeats that records the state of the chunkserver.

When a client wishes to retrieve a file, it contacts the master. This master determines the chunks the file consists of and where these are stored. The client receives this information and continues to contact the chunk servers directly.

In GFS the only atomic operations are file creations, which are handled exclusively by the master. Other operations to chunks and their replicas are handled by a *primary* replica. This primary determines an ordering of operations to the other replicas and ensures serializability. Primaries are given a *lease* and if a primary would fail during execution, a new replica is granted the lease and can carry out further operations.

In GFS the decision to use a single master allows for a simple design and more sophisticated placement policies, because a master can use global knowledge to place chunks. To prevent the master to become a bottleneck, the clients will cache the chunk locations for a limited time. This reduces the interaction required with the master. Also, in the case of a master failure, the master does not store the chunk location persistently, instead it asks all chunkservers at startup and whenever a chunkserver joins the stoarge cluster.

GFS provides data locality in the sense that the master can place replicas of a data item closer to a client, reducing the latency.

Since GFS has chosen a large chunk size, it reduces the interaction with the master, communication can be reduced since clients are likely to perform

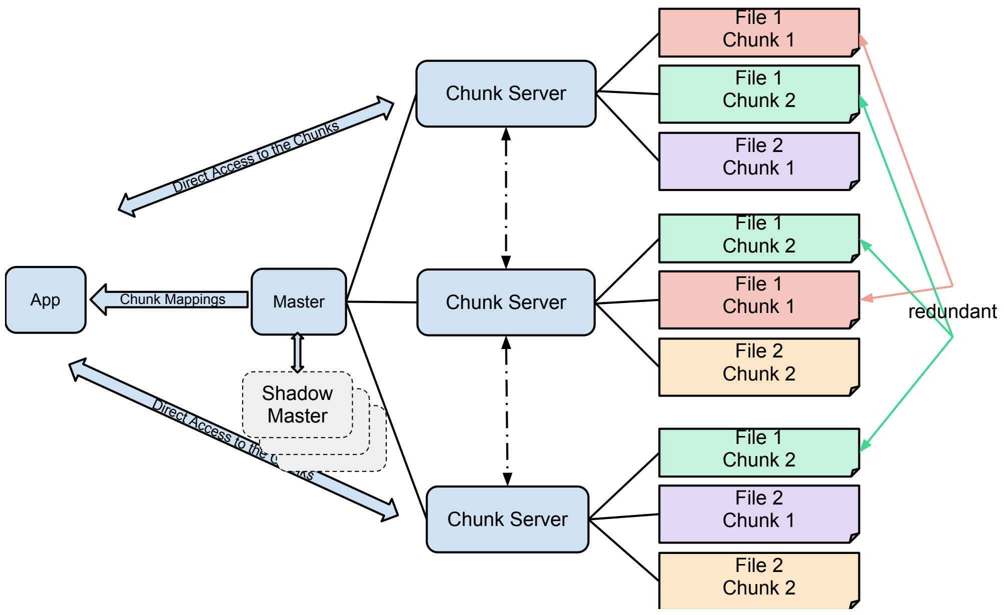


Figure 5.3: GFS architecture. Figure from [50].

multiple operations on a single chunk, and it allows for the chunk to filesystem mapping to fin in main memory.

5.1.3 Non-relational databases

As described above, SQL has been used to access data stored in relational databases. These rely on the underlying relational model, where data must be stored after a specific schema. Newer storage systems has seen the relational model as a limitation, and have opted for the non-relational or NoSQL databases. NoSQL databases provide better scalability and availability at the cost of consistency guarantees. NoSQL databases are often suitable for managing large quantities of data, where the nature of the data does not require a relational data model. Within the NoSQL databases, the mainstream data models are as follows: (i) key-value stores, where data items are specific storage node dictated by a hash function. Examples of these are Amazon Dynamo [21] and Walter [51]; (ii) column family stores, where data is stored by columns. Examples of these is BigTable [15], Cassandra [40] and Hbase [56]; and (iii) document oriented stores which is similar to key-value stores, but the values are stored in a structured document, often in JSON or XML format. Example of document oriented stores are mongoDB [1] and

CouchDB [57].

This report will only discuss column family stores and document oriented stores.

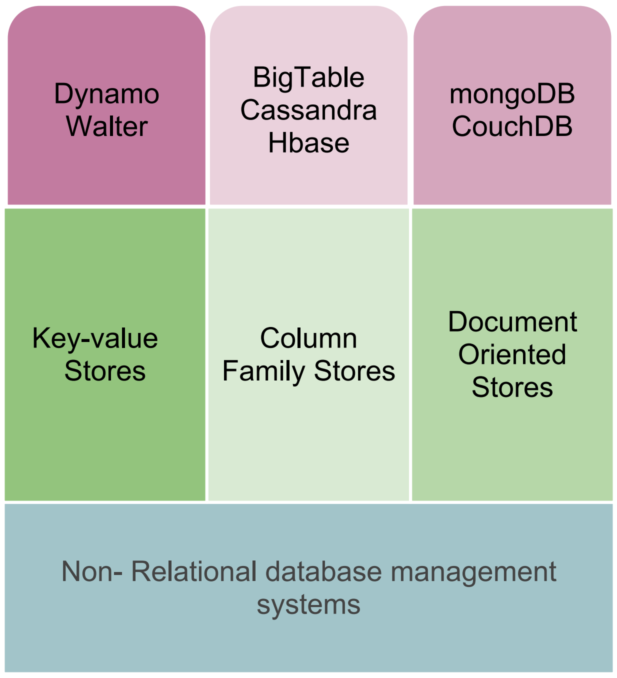


Figure 5.4: The different types of non-relational database systems

mongoDB

mongoDB is an open-source NoSQL-database [56]. The mongoDB database contain multiple collections that contain multiple documents. These documents can be structured differently, and is represented as a JSON structure. Within these documents, key-value pairs are stored. Queries on the mongoDB database are structured as JSON as well, and can be sped up by the use of indicies. To prevent data loss, mongoDB provides the replication of data through *replica sets*. Using this techniques a master is elected to handle all writes on an object. Reading can be performed from any of its replicas, but the master is only allowed to perform writes. To distribute data over a cluster different computers, mongoDB provides *sharding*. To avoid data loss data is stored on logical nodes which consist of multiple physical computers. This is similar to the storage policy used in Haystack [9], to prevent any loss of photos.

With mongoDB there are some clear advantages, first developers can write simple queries in plain query syntax. This alleviates the need to write complex SQL queries. Second, mongoDB allows for sharding to be deployed easily, something that can be troublesome with e.g. MySQL databases. Third, it uses no data schema, which can be helpful if the data schema is not know ahead of time and can change from time to time. However, mongoDB provides no joins or transactions which may be critical in some storage solutions.

Interestingly, the 32-bit version of mongoDB only supports a database of up to 2GB while the 64-bit version does not have any size limitations [2].

BigTable

BigTable is a distributed storage system for managing structured data that is designed to scale to a very large size: petabytes of data across hundreds or thousands of commodity servers [15]. BigTable was developed at Google and is used in the backend solutions for web indexing, Google Earth and Google Finance. Comparing to mongoDB BigTable provides better availability and scalability. It is currently not availible outside Google, but an open source BigTable-like implementation, Hbase, is availible. Also BigTable is column oriented, making it more suitable for certain applications than mongoDB's document oriented storage model.

BigTable shares many implementation strategies with databases, but it provides a different interface than the traditional PDBMS. In BigTable the data is organized in three dimensions: rows, columns and timestamps. A data item with a specific row key, column key and timestamp is called a cell. The data is stored in column-oriented fashion, opposite to traditional relational databases where the data is stored in a row-fashion. BigTable performs a clever optimization by not storing columns without values in specific columns, which is very good for sparse tables.

Rows in BigTable are stored lexicographically by the row key. Reads and writes under a row are serializable, providing an easy interface for users with concurrent updates to the same row. This implies that rows are the unit of transactional consistency. Rows are stored in what is called *tablets*. These tablets hold rows with consecutive row keys, and will grow until a set size before they are split into two. With these tablets, reads within a small range of keys will limit the communication with storage nodes. This property can be exploited by the end users to gain performance.

Column keys are grouped together into *column families*. Data stored within a column family tends to be of similar type and can be compressed. The number of such families tends to be small, at most a couple hundred, but number of columns are unlimited.

Timestamps are introduced to support versioned objects, allowing tables to contain multiple versions of the same data, indexed by timestamp.

Figure 5.1.3 shows an example table storing web pages. It shows two column families, contents and anchor, consisting of web page contents and the text of any achor that reference to this page respectively. NRK's homepage is referenced by both NRK Beta and the home page of the University of Tromsø. Each of the achor cells have only one version, while the contents column contain three versions at different timestamps.

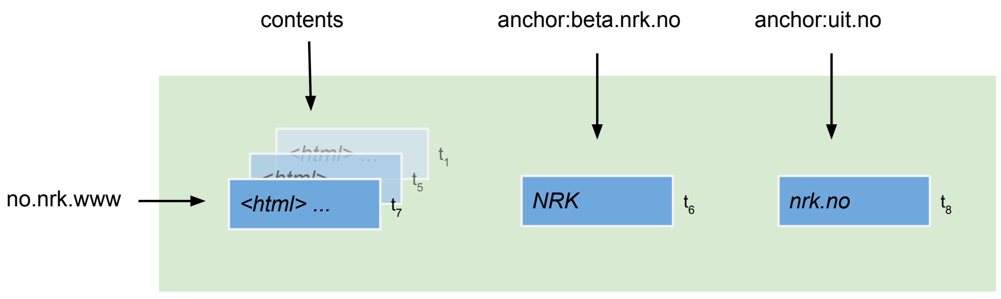


Figure 5.5: An example table in BigTable

To organize servers in a big table cluster, a cluster managemnt system is used for job scheduling, managaing resources, monitoring the nodes and handling machine failures. BigTable uses a single master to serve any request, which is simply forwarded to a BigTable tablet server that responds to the request. These tablet servers can be added dynamically to accomodate changes in workload [15]. BigTable relies on GFS to store files, and uses an in-memory mapping which allows for lookups using a single disk seek. It uses Chubby [14] for ensuring that there is only master to serve requests, as well has holding some metadata for recovery purposes.

5.2 Analysis

5.2.1 MapReduce

MapReduce is a programming model and associated implementation for processing and generating large datasets [20]. In this programming model is up to the user to create *map* and *reduce* functions. The map function should process input (key, value) pairs into a set of intermediate (key, value) pairs. These intermediate pairs are grouped together with regards to the key and is passed to the reducer. The reducer should merge these values together to produce a result.

The implementation of MapReduce should handle partitioning of the input data, scheduling the execution of the program across multiple machines, handling failures and handling any inter-machine communication. This allows the programmer to focus on the problem at hand, and not having to have any prior experience with parallel and distributed systems.

Architecture

The MapReduce framework has a master-slave architecture, with a single master and several workers, often one per cluster node. Users submit MapReduce jobs to the master which assigns the map and reduce tasks to the workers. The workers executes the tasks and handles any data transmission between the map and reduce phases.

Implementation

MapReduce uses re-execution as the primary mechanism for fault tolerance, and will restart any failed job. The initial implementation of MapReduce used GFS for managing the stored data on disk, using replication to provide availability on top of unreliable hardware. During execution the map invocations are distributed multiple machines by the partitioning of the input data into M *splits*. The input splits can then be processed in parallel by different machines. Reduce invocations are distributed by the partitionig of the intermediate keys into R pieces using a partition function specified by the user, usually a simple hash funcion on a key.

The MapReduce implementation handles both *worker failure* and *master failure*. The master pings the workers at reggular intervals to detect node failure. If it detects that any of the nodes are down it schedules the re-execution of the failed job. If the master would fail, the entire MapReduce task must be restarted. It is however very rare that a master would fail, because MapReduce is often run on a cluster with houndreds of nodes and it is very unlikely that one specific node would fail.

The underlying file system used in MapReduce should provide data locality. The MapReduce master will try to schedule jobs that can run on nodes hosting the data, or physically close th the node holding the input data.

Evaluation

There are many benefits of using MapReduce for both storing and interacting with data. It is easier for developers to write any code to access the data, due to the fact that the MapReduce library hides any code dealing with fault tolerance, distribution and parallelization.

Also, the developers using MapReduce does not have to learn a new language in order to write MapReduce jobs. These can be written in any programming language, making it easier for developers to get going and maintaining the applications.

Unfortunately the MapReduce programming model has its limitations. With its one-input two-stage data flow, if the user needs more complex operations like joins or multi stage operations, custom solutions must be written. This leads to code that is hard to maintain and difficult to reuse [46].

5.2.2 Apache Hadoop

Apache Hadoop [26] is a open source software framework which implements the MapReduce programming model. Hadoop provides a distributed filesystem, the Hadoop distributed filesystem (HDFS) [11], for storing large datasets over large collections of computers, and technique for executing work (MapReduce jobs). These MapReduce jobs are run across the same large collection of computers, ensuring that computations are run near the data.

A limitation with Hadoop is that the filesystem used, HDFS is optimized for sequential scans and not for random access [59]. In data exploration the case is often that the data investigated is accessed in a random pattern, which makes HDFS less satisfactory for interactive exploration tools.

Pig

Pig is a system written on top of Hadoop, allowing users to specify SQL-like statements, which are translated into map-reduce jobs. Pig compiles programs written in Pig Latin a language designed to fit in a sweet spot between the declarative style of SQL, and the low level procedural style of map-reduce [46]. It was developed by researchers at Yahoo!, and in their 2008 paper they describe how development and execution of their data analysis tasks was sped up after introducing Pig. Another key aspect of Pig is the debugging environment. Traditionally when writing MapReduce jobs on large datasets, it may take minutes if not hours for the jobs to complete. When writing programs in an iterative approach, developers are dependent on fast feedback to correct any errors that may occur. With MapReduce the run-debug-run cycle may be very slow an inefficient, but the Pig debugging environment speeds up this cycle by generating example data which illustrates the output of every stage. The developer can then use this output to determine if the program must be modified.

Hive

Hive is a data warehousing solution also built on top of the Hadoop software framework. Like Pig [46] it supports queries in a declarative language. This language is termed HiveQL which like Pig Latin is compiled into map-reduce jobs to be executed on Hadoop. Hive supports tables, analogous to tables in relational databases. What separates Hive from Pig is the Hive-Metastore, a system catalog containing metadata about the tables stored in Hive [59]. This metastore is contacted whenever tables are referenced in HiveQL, allowing potential speedup. It is stored within either a relational database, or another filesystem that allows for fast random access.

Impala

Cloudera Impala is a new platform built on top of Hadoop allowing real-time queries using a declarative language. It uses the same metadata, SQL syntax (HiveQL), ODBC driver and user interface as Hive [24]. Unlike Hive and Pig that uses Hadoop's HDFS, Impala accesses data through a distributed query engine like the ones found in RDBMSs, which provides an order-of-magnitude better performance than Hive [24]. Using such an query engine, it allows for systems to use it where interactivity is key. According to Cloudera in [24], Impala is capable of 3-4x speedup on I/O bound queries, 7-45x speedup on queries involving more than one join operation, and 20-90x speedup if the data accessed in the query comes out of its cache.

Chapter 6

Visualization Systems

To gain valuable insights in the peta-scale datasets stored in big data storage systems, visualization systems are often needed. There are a wide range of tools availible, solving many problems across multiple domains. In the NOWAC study over 170 000 women participated in questionnaires over a time period of over 4-6 years. In addition blood samples from over 400 individuals have been collected and stored in a biobank. Creating visualizations that efficiently visualize data from multi-variate data, such as from the NOWAC dataset, is not a straightforward task. Visualizations must both be interactive, provide useful visualizations to the user, and even including advanced statistical tools to generate the visualizations needed.

This chapter will give an introduction to the different visualization systems avalible, both for biological data and data from other sciences, how they handle large datasets, and finally present challenges faced in data visualization tools. In addition a short description on visualization and interaction techniques for high-resolution display walls will be given.

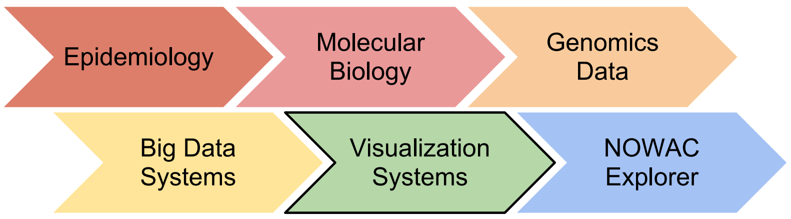


Figure 6.1: Visualization systems

6.1 Backends

In addition to providing a fault tolerant storage solution, the backend systems serving data to data exploration tools must also provide a simple and fast interface for the visualization systems. These systems must present large quantities of data to the user while providing an interactive experience.

There are three main actions performed by data exploration tools: data management, data modeling and visualization [68]. The data management is often done by contacting one or more relational databases, from which data modeling and visualization can be performed. Most of the data exploration tools allow connections to relational database systems such as SQL, PostrgeSQL or Oracle [68]. However there is only a few systems that allows for direct interaction with scalable storage systems such as Hadoop, mongoDB or web-based systems such as Amazon Simple Storage Service (Amazon S3).

The different data exploration tools can be grouped into either stand-alone desktop applications or applications run in a server farm. Of the many commercial applications available Tableau, QlickVeiw and Spotfire support the use of a server farm [19]. Join operations are a common trend within big data systems, and is found very helpful in many scenarios. However, many tools are restricted by the DRAM on the system when performing join operations.

This chapter contains a survey of two tools, Tableau and Spotfire, and how they manage big datasets.

6.1.1 Tableau

Tableau is a visual analytics system for performing ad-hoc exploration an analysis of customer data sets [55]. Tableau Software was founded at Stanfard as a research project before becoming a commercial product [17]. The Tableau application is run as a desktop application, which can either connect to an online data store or operate on its own offline subset of the data.

The online data store may consists of one or more database systems, ensuring consistency across the users. Tableau like many other visual analytics platforms also supports an iPad application, as well as a Dashboard accessible from any web browser. This makes it possible for users to perform analytics tasks on multiple platforms, and on the go. In [55] the authors identified three key arguments for developing an offline data engine: (i) The original data may be unavailible, often the case when travelling; (ii) the data may be stored on some storage device or format which requires is not suited for analytical queries; and (iii) the analysis itself should be presented as a report.

From these Tableau created an *extract* feature which allows for users to extract or retrieve a portion or the data collection and perform any offline analysis. The offline data engine was originally implemented using the Firebird relational database, but Tableau has made some recent development introducing their own data engine. This has allowed Tableau to relax their dependency on third-party software. Also with its own data engine, the unnecessary optimizations for small sets of queries is dropped, so that the data engine will be optimized for the exact usage Tableau is seeing.

Like many other data exploration tools, Tableau needs to handle unstructured exploration of new datasets, as well as advanced aggregation and join operations. Tableau has support for multidimensional data which requires that filtering is also possible to perform on this data. A requirement for the data engine is to support the various other data sources it should connect to. It should understand the semantics of the data, with an easily extensible type system. Further the Tableau system should work on 32-bit Windows laptops with limited resources and available memory.

Queries

In Tableau queries are often generated in a drag-and-drop fasion, where the user performs some interaction with the presented data and the result are then presented to the user in the same interface. Tableau uses VizQL language to describe an analytical query and associated graphical output [55].

Tableau Data Engine

In order for offline exploration, Tableau has developed its own Tableau Data Engine (TDE), a specialized column store [67]. This data engine can be run on desktop machines, or on a shared server. This Data Engine uses special tables for storing metadata which allows for faster access. Using such a columnar store, Tableau is able to compress the data and operate on the data in its compressed form, similar to what is done in BigTable.

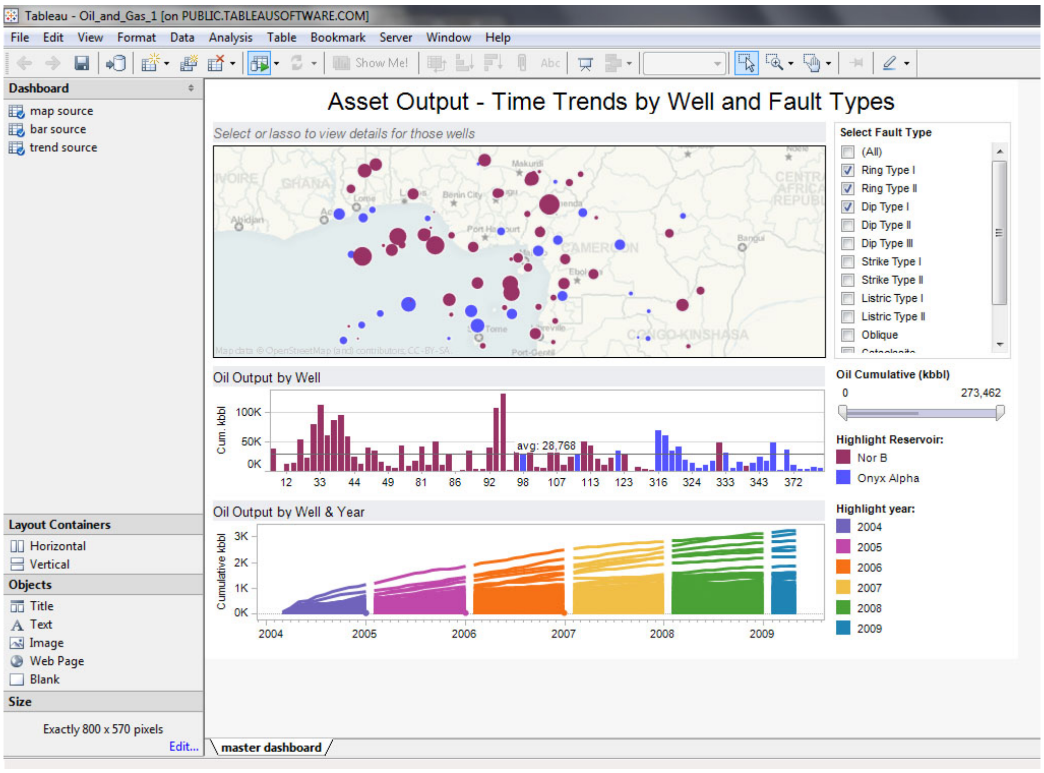


Figure 6.2: Screen shot of the Tableau Desktop Application

The data engine is composed of five layers: (i) a storage model; (ii) a execution engine; (iii) a query parser and optimizer; (iv) a communication interface; and (v) the Tableau VizQL compiler.

Like traditional databases the TDE contains the logical notion of schemas, columns and tables. These are structured as multi-level directories. Since the TDE is column oriented, each table is a simple directory, where the columns are stored as files. A schema is a directory containing the tables, and a database is simply a top-level directory containing the schemas. Columns contain either fixed width arrays of values, or a dictionary file containing tokens. These tokens are just an index into a dictionary, e.g. if we want a specific date within a range, rather than having mulitple entries with long date strings like *10/01/2013* it could just point to a dictionary containing the possible values. These tokens are represented as integers and can be treated as such, if two entries contain the token they point to the same date. In [67] they report that these dictionaries could save up to 50% on the storage requirements.

The execution engine follows from traditional databases, generating a query plan from a set of operators taking rows as inputs and producing output rows. For performance TDE uses block processing, where intermediate records are generated to be used by other operators. In the query tree one can think of a lower-level operators producing such blocks, which can then be operated on by its parent.

The Query parser and Optimizer is used to analyze and perform general optimizations to a query. TDE uses the Tableau Query Language (TQL) for representing a query. The parser accepts commands in TQL and generates an in-memory tree representation which the optimizer further transforms and converts into an executable query plan.

The TDE runs as a separate process and is communicated to through regular sockets. From Tableau queries are sent which the TDE reads and routes them to a *session manager* which executes the queries. The results are then written back to Tableau. The communication interface handles this communication as well as any communication regarding longer running queries that require progress reports to the user.

The VizQL compiler compiles the visualization specifications and compiles them into database queries such as SQL and TQL.

With the column-oriented storage model, Tableau is able to compress data and operate on it in its compressed form [55]. As mentioned in [55], Tableau targets an analytic workload where traditional row-oriented databases suffers to provide sufficient performance. TDE is able to provide efficient utilization of processors available, allocating independent queries to separate processor cores.

Other data sources

Like other vendors Tableau allows its software to connecto to other sources using Open Database Connectivity (ODBC), a standard API for accessing DBMS. It is a middleware that sits between applications that translates general-purpose requests written in SQL to database-specific requests.

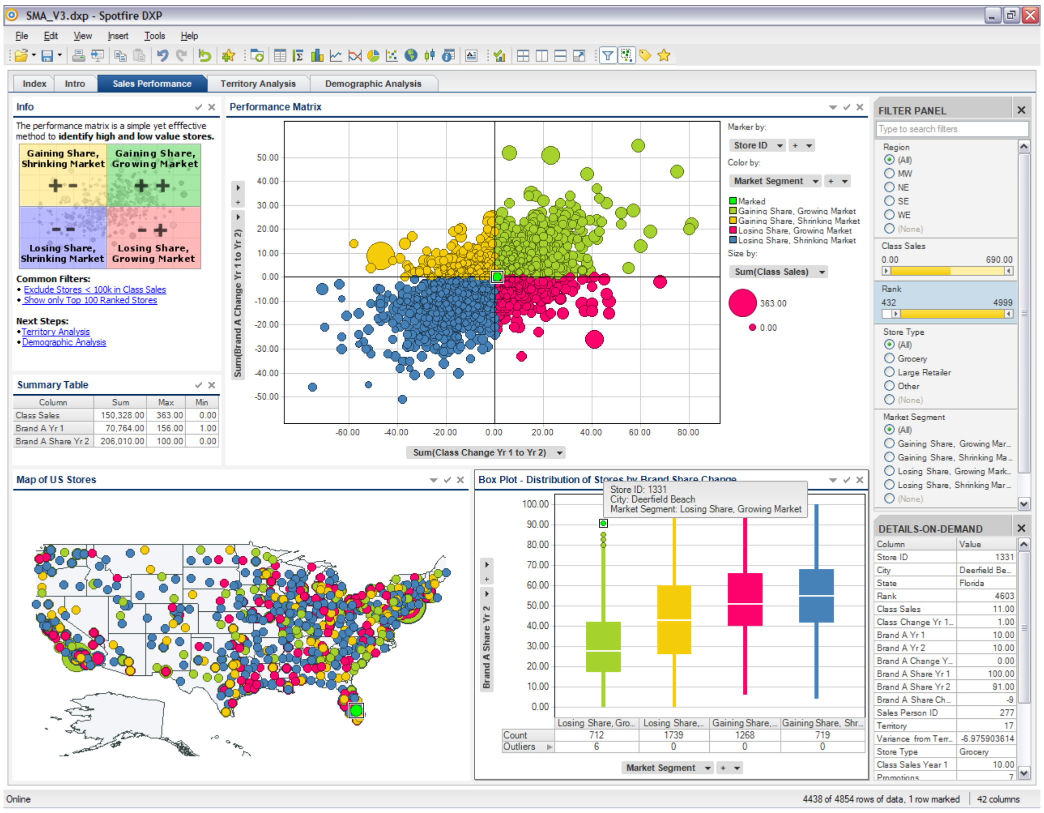
Using ODBC, Tableau has like many other vendors made use of the Apache Hadoop framework. As of the fall of 2012 the company had been supporting Hive as a datastore. However while it is great for bach jobs, it was not able to provide a fast and interactive solution for ad-hoc data exploration [66]. When

Cloudera released Impala in October 2012, they announced that Tableau was one first data exploration tools to use Impala. Integrating Impala in Tableau is done easily and requires only a simple plugin to allow for connection to Impala. The key advantages using another tool on top of the Hadoop stack is that when the data first has been collected, there is no overhead in moving it to another datastore for querying. Since Impala uses a distributed query engine, it is suitable for allowing the ODBC connection. Tableau makes it possible to explore petabytes of data using Impala, making it extremely suitable for big data exploration.

6.1.2 Spotfire

TIBCO Spotfire is another data exploration tool, that has come out of a research group. It is very similar to Tableau in that it allows for exploration of large datasets. These originating in either simple text or excel files, relational database systems or data originating from TIBCO's ActiveSpaces. Figure 6.3 shows a screen shot of a typical user session using the Spotfire Application. Spotfire also provides a client available from web browsers.

TIBCO Spotfire provides support for big data analytics, through its distributed peer-to-peer infrastructure. The ActiveSpaces platform uses a specific membership protocol and does not rely on any central administration [60]. TIBCO reports in [60] that this improves latency as well as efficiency, and it combines database and messaging functionality under a single interface. Peers in the network contribute with compute and storage capability, storing all data in-memory. These peers can be any computer, often servers either running Linux or Windows Server. If a clients wants to join the network in order to perform any query, it can join as a 'leech'. These 'leeches' have full access to the network, or space as TIBCO calls it, without having to contribute with computational or storage resources. Figure 6.4 illustrates the structure of the ActiveSpaces data grid. As the figure illustrates, it allows for any database or application to join the grid and participate with storage and processing power. This architecture allows for elastic scalability, making it easy for developers to add and remove peers.


SMA_V3.dxp - Spotfire DXP
File Edit View Insert Tools Help
Index Intro Sales Performance Territory Analysis Demographic Analysis
Gaining Share, Shrinking Market + -
Gaining Share, Growing Market + +
Losing Share, Shrinking Market - -
Losing Share, Growing Market - +
Performance Matrix
Sum(Brand A Change Yr 1 to Yr 2)
Sum(Class Change Yr 1 to Yr 2)
Marker by: Store ID
Color by: Market Segment
Size by: Sum(Class Sales)
Map of US Stores
Box Plot - Distribution of Stores by Brand Share Change
Brand A Share Yr 2
Market Segment
FILTER PANEL
Region
Class Sales
Rank
Store Type
Market Segment
Online

Info

The performance matrix is a simple yet efffective method to **identify high and low value stores.**

Common Filters:
- Exclude Stores < 100k in Class Sales
- Show only Top 100 Ranked Stores

Next Steps:
- Territory Analysis
- Demographic Analysis

Summary Table

Column	Sum	Max	Min
Class Sales	150,328.00	363.00	0.00
Brand A Yr 1	70,764.00	156.00	1.00
Brand A Share Yr 2	206,010.00	100.00	0.00

Store ID: 1331
City: Deerfield Beach
Market Segment: Losing Share, Growing Market

	Losing Share, Gro...	Losing Share,...	Gaining Share,...	Gaining Share, Shr...
Count	712	1739	1268	719
Outliers	6	0	0	0

DETAILS-ON-DEMAND

Column	Value
Store ID	1331
City	Deerfield Be...
State	Florida
Rank	4603
Class Sales	11.00
Class Change Yr 1...	1.00
Brand A Yr 1	10.00
Brand A Yr 2	10.00
Brand A Change Y...	0.00
Brand A Share Yr 1	100.00
Brand A Share Yr 2	91.00
Brand A Share Ch...	-9
Sales Person ID	277
Territory	17
Variance from Terr...	-6.975903614
Store Type	Grocery
Class Sales Year 1	10.00
Promotions	7

4438 of 4854 rows of data, 1 row marked | 42 columns

Figure 6.3: Screen shot of the Spotfire Application

6.1.3 Wrangler

Wrangler is an interactive system for creating data transformations [36]. Wrangler targets the problem of the tedious task of manipulating data, e.g. reformatting data values or correcting erroneous or missing values, in order to perform analytics tasks on the data. These transformations can be difficult to specify formally and reuse later on different datasets. Like Tableau it translates user interactions and manipulation of the visualizations into queries that modify the underlying data. Wrangler employs clever techniques to leverage the semantics in the data types, e.g. geo locations or dates, to suggest data transformations relevant to the user.

The authors report that data cleaning is responsible for up to 80% of development time and cost for data warehousing projects. And that the *data wrangling* often is done using scripting languages like Python or Perl. However it may be a hard task to specify specific data transformation in a programming

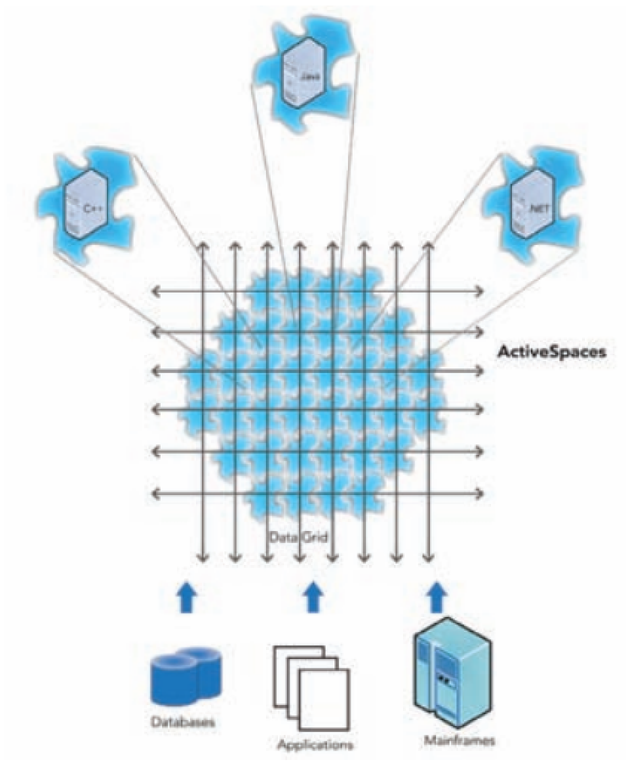
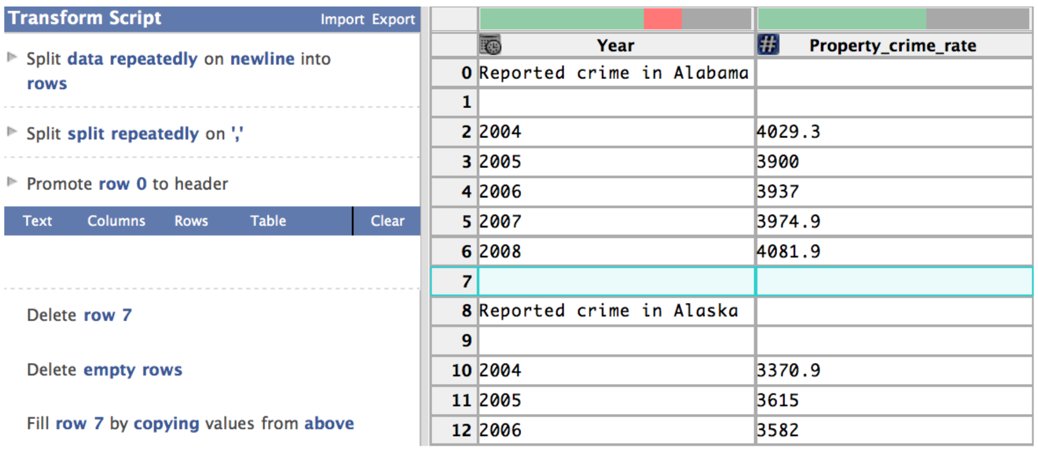


Figure 6.4: TIBCO ActiveSpaces is a distributed peer-to-peer in-memory data grid that enables heterogeneous applications to share, exchange, and process data in real time. Figure from [60].

language, and may often require advanced parsing of the input data. Wrangler allows users to select data through interactions and suggests applicable transformations based on the current visualizations being presented to the user. Users may preview selected data transformations in order to decide whether to apply them or not. Wrangler is able to record the data transformations applied in a script. This script can either be run in a web browser or be converted into Python or MapReduce code. The MapReduce code allows for developers to *wrangle* a smaller subset of the entire dataset on a less powerful device such as a laptop or tablet, and then run the MapReduce jobs on a cluster for the entire dataset.



▸ Split **data repeatedly** on **newline** into **rows**

▸ Split **split repeatedly** on **','**

▸ Promote **row 0** to header

Delete **row 7**

Delete **empty rows**

Fill **row 7** by **copying** values from **above**

	Year	Property_crime_rate
0	Reported crime in Alabama	
1		
2	2004	4029.3
3	2005	3900
4	2006	3937
5	2007	3974.9
6	2008	4081.9
7		
8	Reported crime in Alaska	
9		
10	2004	3370.9
11	2005	3615
12	2006	3582

Figure 6.5: A screenshot of the Wrangler interface. The left panel contains history of previously performed operations and suggestions for new transformations. The right panel contains an interactive table with the dataset the user is currently working on. Figure from [36]

6.2 Biological Data Visualization

In the last decades the methods for visualizing and exploring biological data has improved greatly. With the increased rate data is generated at, the need for tools that can manage and allow for fast exploration has emerged. Especially tools that generate massive amounts of data, like high-throughput DNA sequencers, has driven the need of novel data exploration tools. Today there are a wide spectrum and large number of different tools, but biology research projects often require custom built solutions for their specific problem set.

Molecular graphics are the most mature in the different visualization areas in biology, and is widely used in textbooks and popular media [45]. Figure 6.6 are examples showing a Porin molecule created with the molecular graphics tool QuteMol [64]. Other newer branches within biology that is making use of computer aided visualizations are genome visualizations, which has become popular since the sequencing of the human genome.

This chapter will discuss different methods for visualizing biological data, the challenges with creating exploration tools, as well as the data exploration tools available.

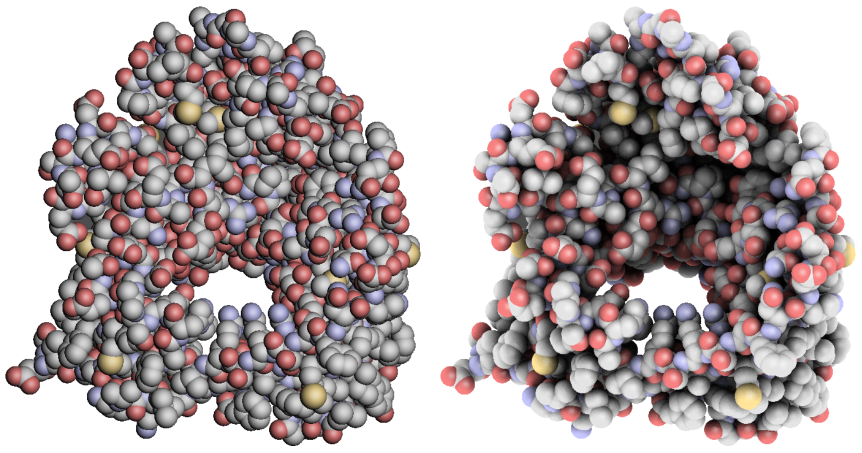


Figure 6.6: Molecular graphics. A Porin molecule shown side by side to put emphasis on how rendering effects can improve the 3D shape. Images from [4, 5].

6.2.1 Challenges

There are multiple challenges that needs to be overcome when designing biological data exploration tools.

- The key challenge is to benefit from the extreme quantities of data without being overwhelmed by it. It may be intersting to both show expression data as well as pathway data in the same presentation, but how this can be done in an intuitive way is still a challenge. This challenge is still largely unfulfilled and will require the development of truly integrated and highly usable tools [45].
- A second challenge is to build tools that are easy to learn and use. This requires an understanding the needs of the researchers and designing graphical user interfaces (GUIs). The tools should be easy to interact with and provide useful visualizations, while not bombarding the user with information.
- A third challenge in data exploration tools used by biologists, is how to represent data from different scales. In genomic exploration, the tools may need to be able to visualize data consisting of whole genomes, down to the nucleotides that form the DNA strands. In order for the visualizations to be useful they must display the correct level of detail

at the different zoom levels. E.g. it would be unnecessary to visualize every atom when looking at cells.

- Another challenge is resolving what can be automated in different data exploration systems. Ideally the tools should provide visualizations for tasks requiring human inspection, while automating others.

- For representing large biological datasets, the size of the display hardware has been a limiting factor for what can be displayed at once. The use of high-resolution displays such as a display wall can introduce new possibilities for researchers. Figure 6.7 shows how high-resolution display walls can motivate collaboration between researchers. Also, in [45] they advocate that large display devices and tiled arrays with improved resolution are likely to be of significant benefit.

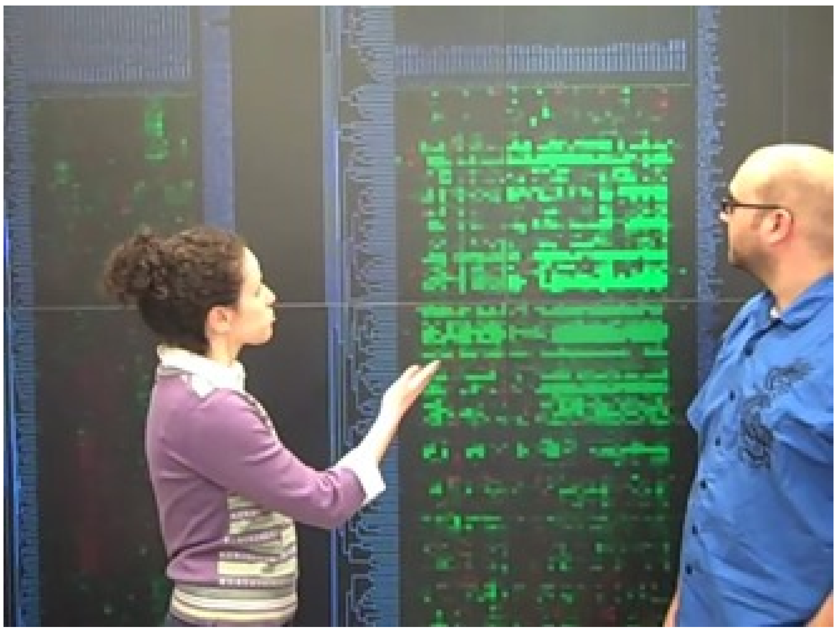


Figure 6.7: Visualization of a genomic microarray dataset using a high-resolution display wall. Image retrieved from [53].

- Another challenge is the computational requirements of the visualization. Often datasets can be terabytes in size, and require advanced statistical methods in the visualizations. For these, desktop computers are not powerful enough to carry out the computations needed.

6.2.2 Current techniques

Visualization techniques has evolved over the last decades. The data exploration tools have evolved from something only domain experts had access to, to something widely available to the general public. The main reason for this is the advances within computer hardware and networks, reducing the cost of computational power. In addition with the faster interconnection networks, tools are being integrated with remote knowledgebases that provide both data and visualizations. Many of todays tools can also be run directly in the web browser and do not require any additional installation. These tools are being used to integrate web applications for data ming and browsing, often using multiple visualization tools [45].

Currently there are two somewhat overlapping categories in the visualization tools available, (i) tools focused on *automated interpretation and exploration of large biological networks*; and (ii) tools focused on *assembly and curation of pathways* [27].

In (i) we find tools for example looking at protein-to-protein interactions, and maybe exposing patterns in such interactions. Also tools for generating heatmaps of expression values fall into this category. Often data may be multi-variate and the tools need to support clustering algorithms and dimensionality reduction, as discussed in **??**. Tools for assembling pathways are typically used for discovering patterns in how cells cooperate and interact. Many of these are web-based and make use of knowledgebases containing these, e.g. Kyoto Encyclopedia of Genes and Genomes (KEGG) [37].

This subsection will cover the techniques used in the following , (i) protein interactions; (ii) gene expression; (iii) metabolic pathways.

Protein interactions

The datasets required to study the interactions between proteins, the datasets are often large in size and require heavy processing. A common trend is to visualize their interactions by plotting each protein as a node in a graph, with the interactions between them as edges in the graph. In order to present these graphs in an *asthetically pleasing* way, they are often manipulated using force-direction layout algorithms. These algorithms minimize the overlapping of edges, producing grahps that are easier to investigate by researchers. Figure 6.8 shows a graph of proteins and their interactions.

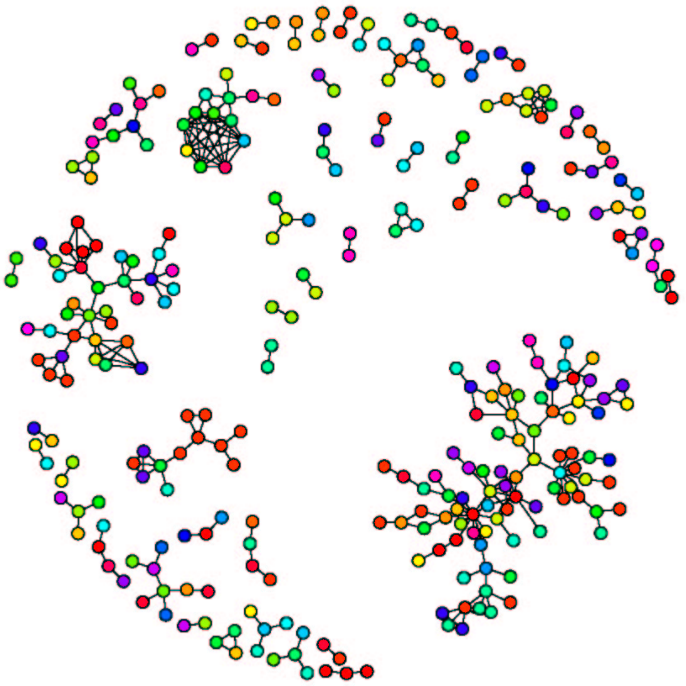


Figure 6.8: Graph of interacting protein pairs with proteins (nodes) colored according to cell cycle expression profile cluster membership [10].

As figure 6.8 shows, there are several clusters in the graph. These may be useful to collapse into a single node, to hide some level of detail to the user. Because the layout itself does not tell anything of where the proteins are found within the organism, it may be necessary to display this using different coloring of nodes.

Gene expression

The goal of gene expression profiling is to usually to find a set of genes or, less typically, proteins that share a related pattern of expression [45]. With these profiles the problem is that they may contain several thousand genes and origin from different samples taken at different time points, making the visualizations possibly complex.

With gene expression, common plots are scatter plots combined with dimensionality reduction, heatmaps and dendrograms are commonly used. Figure 6.9 shows a heatmap visualization. On the figure genes have been rearranged according to the hierarchical clustering shown in the dendrograms.

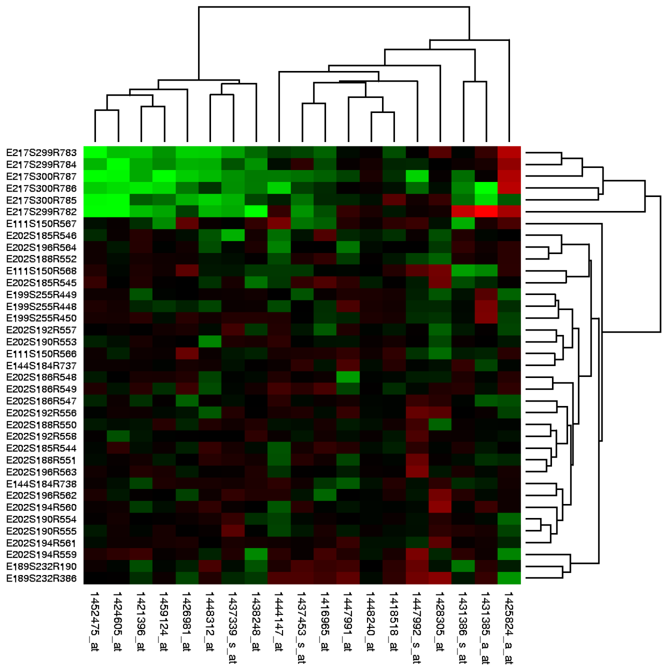


Figure 6.9: Heatmap generated from DNA microarray data reflecting gene expression values [6].

From the heatmaps generated it is often relevant to look at the pathways the different genes are represented in. As we will see in 6.2.2 the pathways are often also plotted as graphs, and a common trend in the visualization society is to map the gene expression values down to the nodes in the graphs. These could either contain the colors from the heatmaps or some bars indicating their expression values.

Metabolic pathways

The study of how molecules interact within a cell to produce reactions and chemical reactions, visualizations is key. These visualizations can be either static providing precomputed images, e.g. KEGG or dynamic that allow users to modify and create their own pathways, like GenMAPP [29]. Figure 6.10 shows a global map from KEGG. As the illustration shows, these maps may be very complicated, and clever visualization techniques can be very helpful for detecting patterns more easily.

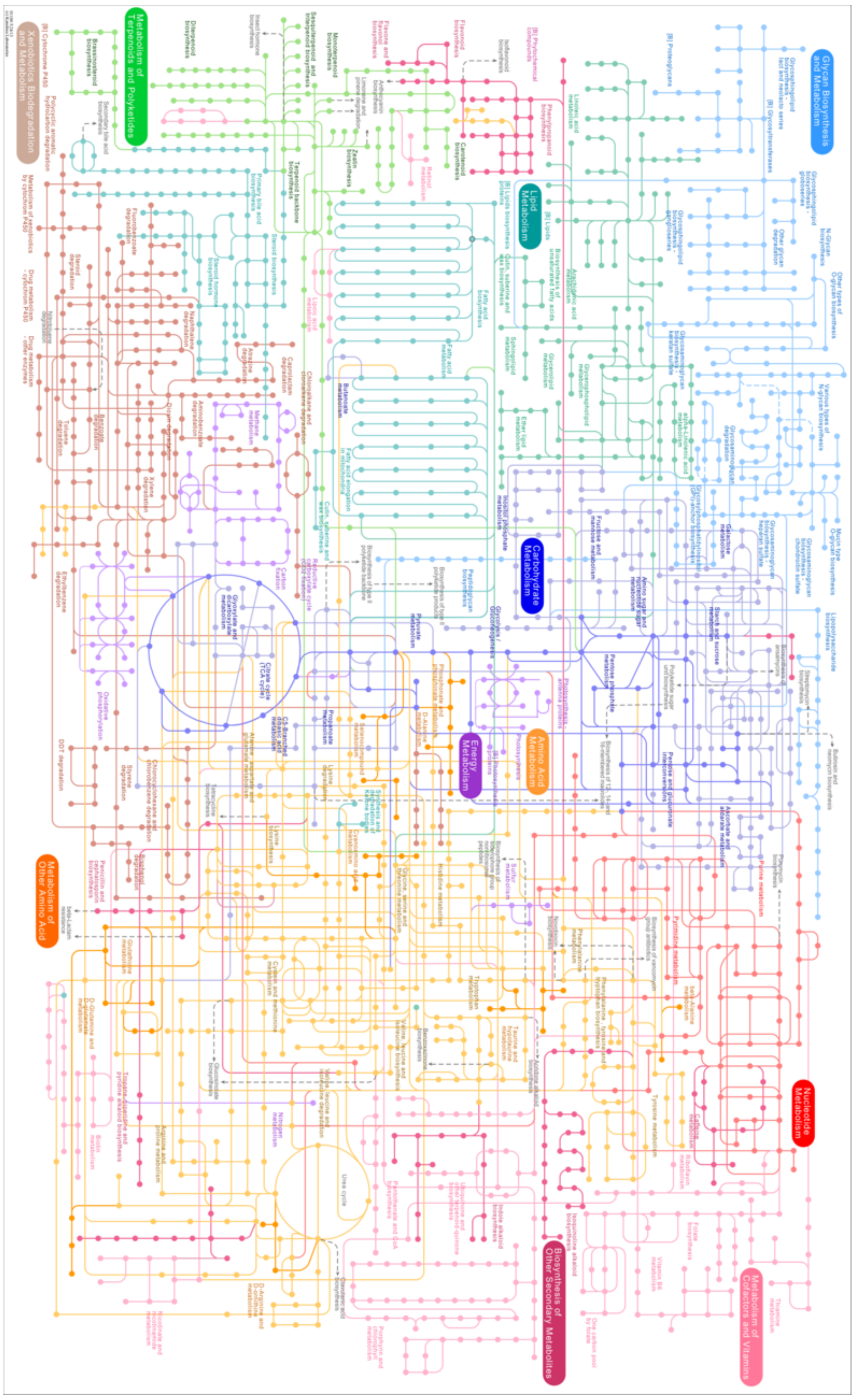


Figure 6.10: Global KEGG map. Figure from [37].

6.3 Multi-Dimensional Data Visualization

Creating understandable and interactive visualizations for scientific data is an often neccessary task in order to gain knowledge into the large datasets collected. One of the many difficulties with creating such visualizations is the trend that data and model is becoming *multi-faceted* [38]. Also creating intuitive ways of interacting with the visualization systems are crucial to their design.

In this chapter, the difficulties regarding creating and interacting with scientific visualizations is discussed. It will emphasis on the survey paper *Visualization and Visual Analysis of Multi-faceted Scientific Data: a Survey*[38] and the fourth chapter of the Ph.D. thesis *Device-Free Interaction and Cross-Platform Pixel Based Output to Display Walls* [52].

6.3.1 Visualization and visual analysis of scientific data

In the literature the notion of multi-faceted data describes data that is (i) *spatio-temporal*, relating to both space and time, e.g. the tracking of an object; (ii) *multi-variate*, consisting of different attributes, e.g. temperature or pressure; (iii) *multi-modal*, with data from different data sources; (iv) *multi-run*, consisting of data from different simulations or runs; and (v) *multi-model*, data from simulaitons using different simulation models.

Generally the technologies perform at good at one of the above types, but not a combination. With newer scientific discoveries, it has become increasingly important to create solutions which integrates different data sources and types into a single representation.

In [38] they identify that the visualization of spatio-temporal and multi-variate data have been broadly studied, while multi-run and multi-model data scenarios are new to the visualization community.

Spatio-temporal data

Managing *spatio-temporal* visualizations is important within several disciplines. Within meteorology it can be interesting to study the movement of hurricanes during the hurricane season, and within medical sciences it may be interesting to study how cancer cells evolve over time. Representing this data in an intuitive manner can be complicated, and techniques involving placing visualization side-by-side for visual inspection. The choice of representing data in 2D or 3D is also something which must be taken into conciderations. This decision often relies on the task at hand [38], but some data inherently suggest 3D representation.

When visualizing spatio-temporal data visualizations use common approaches, like automatically generated animations or interactive sliders for selecting different time ranges. Often the animation generated show the data at different time steps, or along some time axis.

Multi-variate data

Visualizations may also require that multiple attributes and relations must be inspected in order to provide any interesting output. When scientists are faced with data with numerous attributes, the visualization tools must be integrated with statistic analysis tools and dimensionality reduction software. These statistical tools may be computationally intensive, and must provide high performance to better the user experience. Typical tools for dimensionality reduction include Principal Component Analysis (PCA) and Multi-Dimensional Scaling (MDS). PCA transforms multi-dimensional data into a coordinate system that is orthogonal to the variable with the largest variance. PCA can transform a dataset that cannot be visualized using know techniques, into lower-dimensional dataset easily represented in 2D or 3D. MDS maps higher-dimensional data onto a lower dimensional space, still containing the dissimilarities between the data.

Multi-variate data is often visualized using scatterplots or polar coordinates. These expose correlations or outliers often interesting to researchers. With these techniques the human aspect of the computations come in hand, allowing users to spot patterns in the data. *Glyphs* are often powerful when visualizing multi-variate data, where the data is used to determine its shape, color, position and so on. These can again be used to draw attraction by for example using different color schemes to aid human pattern recognition abilities. If the visualization is in 3D further techniques can be employed to enhance the depth perception.

Multi-modal

With multi-modal data coming from different sources, often of the same spatio-temporal location, the challenge is to merge the data together to give more information to the user. An example of this is integration of the two different set of images generated by either Computer Tomography (CT) scans or Magnetic Resonance Imaging (MRI). CT scans are good for imaging bone structures, while MRI are better at imaging soft tissue, but a visualization may want to visualize them together.

Multi-run

Using multi-run simulations, it may be be interesting for researchers to tweak certain parameters in order to inspect the different outcomes. With such simulations it can be interesting for an aggregated summary between each run, but also simultaneous visual inspection. Creating such an aggregated summary can prove to be difficult when the data is multidimensional.

When generating suitable representations for multi-run data, it may prove challenging because of its high-dimensionality, size and because it may be multi-variate.

Multi-Model

When researchers are investigating complicated physics simulations where interactions between different parts, e.g. the interaction betwean ocean and athmosphere in climate research. Creating visualizations that correctly visualize and represent these relationships is a hard task.

As well as interactions, the data may be simulated in different grids, one in 2D the other in 3D. With this in mind the visualization systems must identify the differences and determine how data can be represented using the different models. In [38], the work of creating visualizations of multi-model scenarios have not yet been fully adopted by the community, and there exists only a single system targeting this type of data visualization.

In addition to the different facets described including adding pictures, video or text to the visualiaztions. In a data exploration tool, adding some information or labels through text may prove crucial in order to generate some semantic context to the data.

Data exploration

With data visualizations researchers are able to interact directly with the datasets, not just presenting finds.

There are many standard visualization techniques developed, e.g. *histograms*, *scatter plots*, *parallel coordinates* or *graphs*. These are often in 2D allowing for users to select, or brush, data items for comparison against other plots. Many visualization tools allow for users to interact with several visualizations at once, for easy comparison.

To interact with a visualization users can either navigate manually by some user gestures, automatically or computationally assisted. The gestures or interactions performed by the user can be zooming in, rotating the view or panning. Finding a view that reveals significant ammounts of information to the user can be difficult, and in some cases it may be neccessary to hide some details from the user in order to not overwhelm users with information. Often data may be too large and complex to be represented to the user directly, and must be filtered accordingly.

From the user's perspective the system must provide a fast and snappy interface, meaning the interfaces and data presentations should be instant upon user interaction.

An interesting point in [38] is to integrate machine learning into the analytics process. These systems can learn from previous explorations and reduce exploration tasks by automating many fundamental tasks. If users have a tendency to focus on the time aspect of some data, the visualization system could optimize for this hiding away some low level filtering and algorithms.

Gap in visualization techniques

A key observation in [38] is that they see a gap between the techniques used by domain experts, and the features provided by visualization research. New advances in visualization are rarely used in application domains. They argue that this may be because the methods provided by visualization research are often complex and does not integrate simply in the workflow of the domain. They claim that a major challenge for future development is the bridging of this gap, for example by a collaboration with domain experts when designing visualzation systems. These systems should (i) follow guidelines from perception research and human computer interaction; (ii) provide simple graphical user interfaces and advanced visualizations.

6.4 Display wall data visualization and interaction

Interesting technologies that has not been fully adopted by the community is the use of high-resolution tiled display walls. These often provide screen resolutions orders of magnitude larger than commodity desktop computers or laptops. In addition, these encourage collaboration between researchers in front of the display wall, not hanging over a small monitor. This section gives an introduction to two systems developed at the Tromsø Display Wall, one visualization system and the other an system for interactions with the display wall.

6.4.1 Interaction Spaces

Interactions with computing systems can be done with or without devices. Typical devices for interacting with computers are keyboards and mice. Display walls allow mulitple users to interact with large visualizations, and for these types of interactions, the traditinoal devices have several downsides [52]. Problems with traditional devices is that they cannot easily be carried around, require some tracing surface (like mice), and they do not scale to multiple users. In addition devices are easily lost and since they must be portable they require batteries which must be changed from time to time. Interactions without devices is possible by computer assisted speech recognition or gesture detection. Such systems often require the user to wear special equipment, and speech regognition is not fully developed yet. Gesture detection that do not require that the user wears some equipment, often limits the tracking to 2D.

Wether interacting with a device or without one, it is still restricted to a single computer. In [52] the author presents the concept of *Interaction Spaces*, which is a volume within which user interaction is detected. This volume is not restricted to a single user or computer, but shared among different users and computers. Figure 6.11 illustrates the interaction spaces concept. It shows three interaction spaces, the largest (yellow) created using four microphones, the blue created using floor mounted cameras, and the magneta-colored is created using a camera mounted in the celing behind the user.

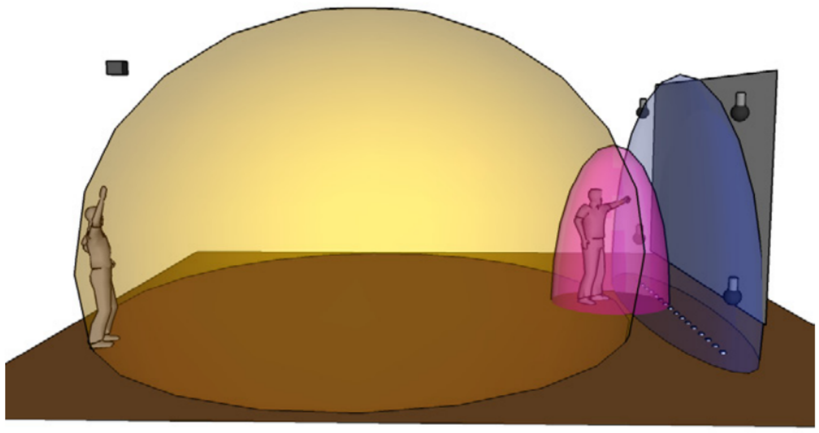


Figure 6.11: An illustration of the Interaction Spaces concept. Figure from [52]

From the concept of Interaction Spaces, three systems are presented in [52]; (i) *Camera-sense*, a system which can track objects in 3D using commodity cameras; (ii) *Snap-detect*, a system capable of locating user by tracking the sound of a user snapping her fingers or clapping her hands; and (iii) *Arm-angle*, a system that can regonize the angle of a user pointing her arm. In 6.11, Snap-detect is shown as the yellow sphere, Camera-sense the blue, and Arm-angle the magneta colored sphere. Of the three different systems, only Camera-sense allows for supports multiple users detecting interactions in 3D.

The Camera-sense system

The Camera-sense system is able to detect user interactions by using many commodity cameras mounted along the floor in from of the display surface. The system can not identify objects, whether its a hand or a pencil, just their location. The Camera-sense system is built using several cameras and a cluster of computers to detect motion from the images gathered by the cameras. The information about detected objects are then passed from the cluster and to a workstation. This workstation does the process of locating objects in 2D and 3D.

The Snap-detect system

The Snap-detect system is able to detect users clapping or snaping their fingers by comparing audio samples with a pre-recorded audio clip with a user snapping her fingers. The system is composed of four microphones which can locate origin of the snap by measuring the arrival time at the different microphones. Even if the system can detect sounds in 3D the actions are mapped down to 2D. In the Snap-detect system four microphones are placed in a rectangular shape in front of the Tromsø display wall, two close to the ceiling and two close to the floor. The signals from the four microphones are sent to a workstation, which detects the sound and localizes its origin.

The Arm-angle system

The Arm-angle can detect users pointing in any direction by looking for straight lines on images captured by a camera. In the system a single steerable camera is mounted in the back of the room, looking over the Troms display wall and any users in from of it. The camera is connected to a workstation that controls its field of view and determines the direction in which an object points.

6.4.2 WallScope

The idea behind WallScope is to separate display resources from computational resources so that computational resources are not constrained by number of display nodes, and the display-side can access customized data for the visualization system. WallScope consists of pull-based Network Accessible Display (NAD) resources requesting visualizations from Network Accessible Compute (NAC) resources [32].

The overall architecture of WallScope is shown on figure 6.12. The display nodes sends requests to the live datasets which translates these into compute messages to the compute nodes, the compute nodes respond with results that are used as a part of the final rendering on the display nodes.

A NAD is defined as a display with functinality that enables usage over a network. With NADs it allows computers to connect to a display over a network connection as if it was physically connected. NAC resources are computational resources providing content any network accessible display. In [32] the author presents NAD and NAC resources for interactive visualization of data on displays ranging for laptops with a limited resolution dislay, to high-resolution tiled display walls, such as the Tromsø Display Wall, a 22-megapixel rear-projected display [7].

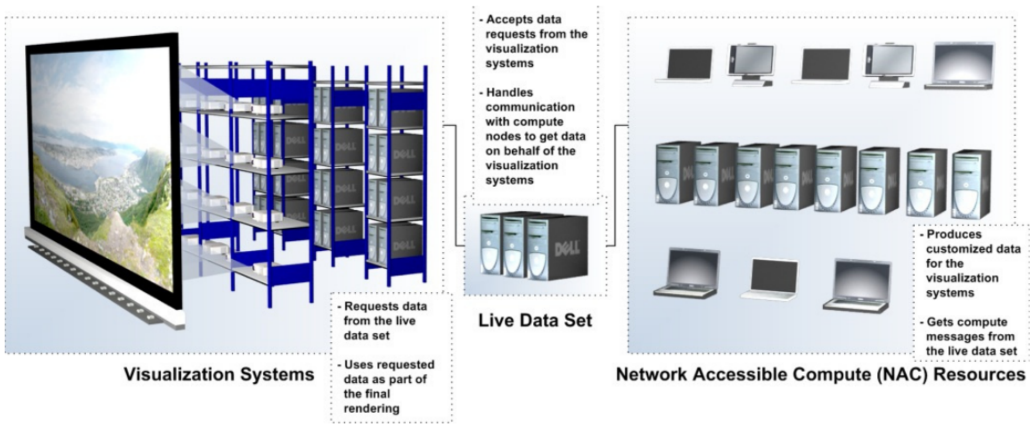

- Accepts data requests from the visualization systems
- Handles communication with compute nodes to get data on behalf of the visualization systems
- Requests data from the live data set
- Uses requested data as part of the final rendering
Visualization Systems
Live Data Set
- Produces customized data for the visualization systems
- Gets compute messages from the live data set
Network Accessible Compute (NAC) Resources

Figure 6.12: The WallScope Architecture. Figure from [32].

In WallScope the NACs can be either static or dymaic. A static NAC can be though of as a cluster or supercomputer permanent to the system. A dynamic NAC is a resource that can register in the system, perform some computations, and later leave the system.

On the display side of WallGlobe, four compnents have been developed: (i) *WallGlobe*, a visualization system for visualizing globes or planets, similar to Google Earth [30] or The WebGL Globe [31]; (ii) *WallView*, a visualization system for high-resulution images; (iii) *LDSView*, a system combining WallGlobe and WallView; and (iv) *The live data set*, a coordinator between display-side and compute-side The visualization systems are composed of tree internal components, a rendering engine, a rendering queue and a request queue.

On the compute-side, three components were designed and implemented: (i) *WallCompute*, a system for computing images and other data for WallGlobe and WallView; (ii) *WallWeather*, a system for computing both images and weather forcasts; and (iii) *Dynamic compute resources*, a system for managing dynamic compute resources

The WallScope system shows a near linear speedup using 1 - 6 compute nodes. Beyond 6 compute nodes the speedup is slightly reduced caused by the round-robin work distribution [32]. Further it identifies computation of customized data to be the main bottleneck of the system. WallScope also demonstrates that personal desktop computers can be made interoperable with high-resolution display walls.

Chapter 7

NOWAC Explorer

We have now described the entire pipeline, starting with epidemiology and ending with biological data visualizations. The last part of the report will contain a description of the prototype, the NOWAC Explorer, a data visualization and exploration tool created for the NOWAC dataset. The idea behind the NOWAC Explorer is to allow ad-hoc exploration of the NOWAC dataset, utilizing modern technologies to provide interactive and intuitive visualizations.

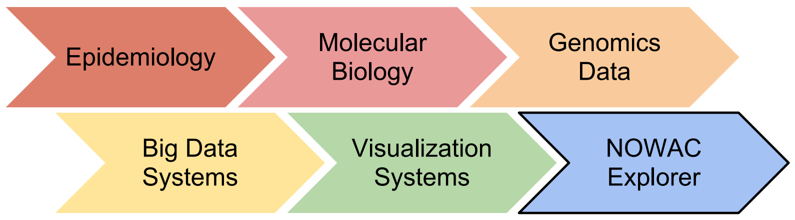


Figure 7.1: The NOWAC Explorer is the last stage in the pipeline, combining knowledge from the previous steps, integrating them into a single system for interactive biological data exploration

It is increasingly common to use web applications for visualization in biology, making use of more powerful web browsers and standards. We have chosen to investigate visualization technologies for web applications, with an emphasis on client performance and scalability.

The prototype consists of the following visualizations: (i) 2D heatmap and histograms for the NOWAC dataset; and (ii) a 3D graph visualization of KEGG pathways. These visualizations were created using the Data-Driven

Documents (D3) and three.js libraries.

This chapter contains the overall architecture and design of the NOWAC Explorer prototype, and its corresponding implementation.

7.1 Architecture

The NOWAC Explorer consists of three separate parts: (i) Visualization Tools used to visualize and explore the NOWAC dataset; (ii) An Analytics and Statistics platform, used to perform analytical or statistical analysis on the dataset; and (iii) A Data Store holding the NOWAC dataset and additional resources. Figure 7.2 outlines the architecture.

The simple NOWAC prototype was focused on the visualization tools, not the underlying storage or analytics engine.

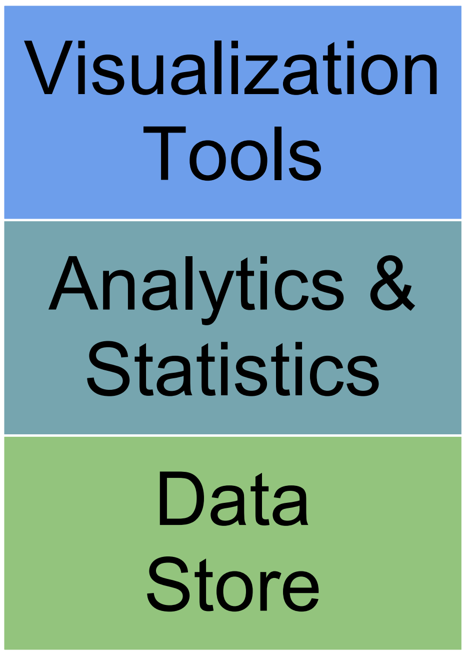


Figure 7.2: General architecture of the NOWAC Explorer prototype.

The visualization tools are exposed to the users, which can run any of these in order to explore the NOWAC dataset. The visualization tools will communicate with the analytics and statistical platform to produce applicable visualizations to the user.

In the NOWAC explorer prototype the analytics and statistics platform as well as the data store is placed on a server or server farm. This server should be capable of handling both storing and analysis of the data needed for the visualizations. The server will perform any computation and return only what is needed to visualize the results at the client.

An advantage using this approach, where all computation is done remotely, is that clients can be lightweight and possibly run on a mobile device. If the data itself should be transferred to the client, it may increase the bandwidth needed for the visualization tool, and the computations may take longer time. However, once the data is downloaded to the client, it can perform any operations to it even if the server should go down, or client get disconnected.

7.2 Design

The general design of the system is outlined on figure 7.3. It follows a three-tiered model consisting of (i) a backend containing providing data collected from different data sources, e.g. sequencing instruments or KEGG; (ii) a webserver providing an interface for access to the data store ; and (iii) a web application containing interactive data visualizations for the user.

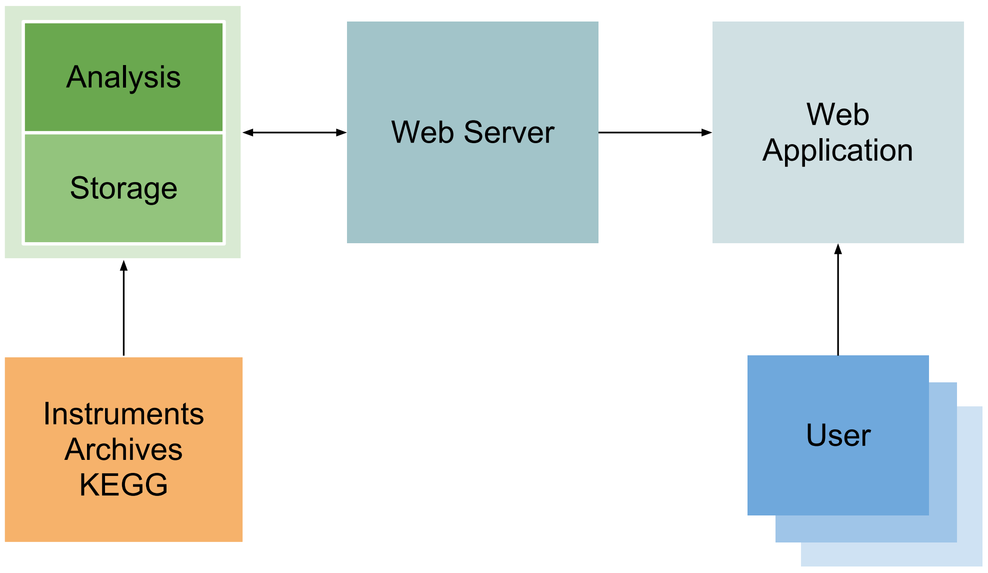


Figure 7.3: General design of the system

The sytem is composed of two major components: (i) a server-side backend component that is responsible for retrieving data from multiple data sources, and exposing the computed data for visualization; (ii) A webserver hosting a web application exposed to users; and (ii) a web application containing a visualization library providing different set of visualizations for clients. This small project will focus on the visualization library.

Currently there are four simple visualizations developed: (i) Histogram of anverage differences in gene expression between cases and controls; (ii) Average years to diagnosis histogram; (iii) Heatmap of the entire NOWAC dataset; and (iv) Graph visualization of a pathway selected from KEGG. These will be discussed in more detail in 7.4.

The visualization tools within the web application are exposed to the clients, which can display them on their local system. The visualization tools query the data store to generate visualizations for the client.

The visualization tools can be entire applications downloaded and installed at the client, or provided through some generic interface, like an applet running in a web browser. Downloadable applications allows for wider access to the local systems, e.g. storage or other hardware, but they have to be written explicitly for different platforms and installed before they can be used. Applet-style visualization tools allow for users to begin data exploration without having to install any additional software. In addition, since the tools are retrieved from the server upon every launch, updates to the software can be done seamlessly without any user interaction. With local applications, upgrades to the software often involve user interaction, which may not be done for years, if ever [3].

The backend system has opted for the latter approach to visualization libraries, where the clients access the tools through a web service tough a browser. Clients request visualizations by contacting the backend system using a specific Uniform Resource Locator (URL).

Data from the remote sources can either be gathered into the local data store of the backend system, the data warehouse approach, or fetched upon query of the specific data items. There are several benefits and drawbacks by using either of the approaches. If data must be collected from remote data sources upon a query, the time taken to fetch this information will effect the time length of the query from the visualization system. In some cases, maybe making the system unresponsive. Another drawback is that if a remote data source goes down for maintenance, the backend system cannot fetch any data from their store. However, a benefit of using such an approach is that there is little or no storage requirements for the backend system itself, everything is stored remotely. With the peta-scale datasets available this would absolutely reduce the storage costs dramatically. If the backend system should choose to pre-fetch data from the remote sources, there are several benefits for this as well. There is no additional latency in accessing data from the remote data sources, since everything is stored locally. The information is available as long as the backend system is available. However, when data is updated at the different sources, the pre-fetched data must also be updated. This may lead to inconsistent data, since the local stores may take some time before they are updated.

In the system there is currently one local data store, containing the NOWAC dataset, and a remote dataset consisting of KEGG pathways.

7.3 Background

7.3.1 D3

D3 is a representation-transparent graphics library for use on the web. It enables direct inspection and manipulation of the standard Document Object Model (DOM) [13]. It provides a JavaScript library for visualizing and manipulating documents. It uses the combined capabilities of HyperText Markup Language (HTML), Scalable Vector Graphics (SVG) and Cascading Style Sheets (CSS) to provide functional visualizations for the web. The project started at the Stanford Visualization Group, and is in continuous development. The library is open source and can be found at [12]. Figure 7.4 shows example visualizations using D3.

7.3.2 three.js

three.js is a lightweight 3D visualization library that supports rendering using WebGL, a JavaScript API that allows for GPU acceleration of graphics in the web browser. WebGL uses the HTML5 canvas element accessed through DOM interfaces similar to D3. three.js is open source and can be found at [58].

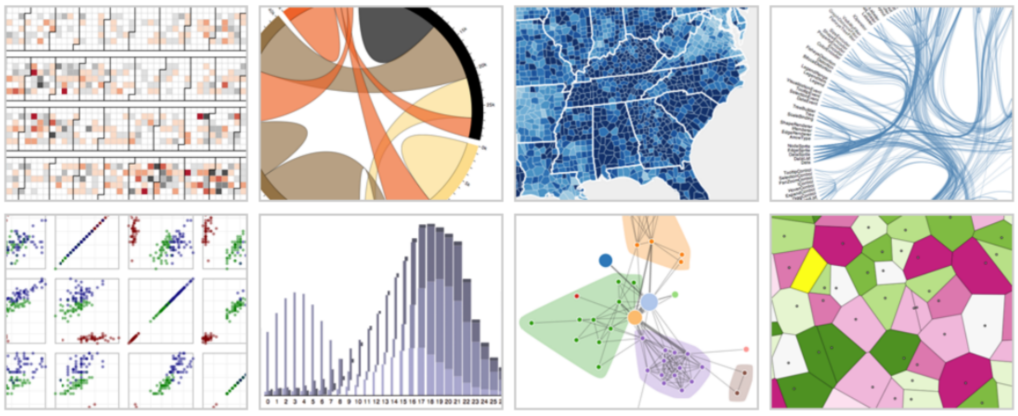


Figure 7.4: Visualizations using D3. Figure from [13]

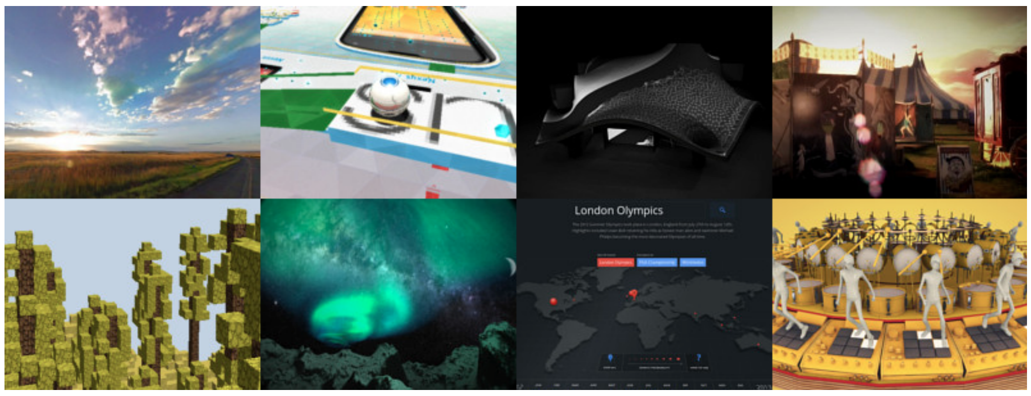


Figure 7.5: Visualizations using three.js. Figure from [58].

7.3.3 KEGG

KEGG is an online data knowledgebase containing information about genomes, metabolic pathways, biological chemicals and proteins. It provides a freely available API and a paid FTP subscription.

7.3.4 NOWAC dataset

The NOWAC dataset contains 263 pairs of case-control and their respective gene expression values for 22782 genes. The case-controls have been anonymized and the genes are given a number from 0 to 22781. Additional labeling of genes are available, but not necessary in this prototype.

7.4 Implementation

The implementation of the backend system follows its design. It consists of a storage system and a web server exposing the different viualizations to different clients.

Currently the backend storage system consists of one large datafile containing the NOWAC dataset, and a smaller file describing a single KEGG pathway. The NOWAC dataset is stored within a single R [48] data file provided by the NOWAC research group. This file is then processed by Python scripts that generate the data used in the visualization. The Python scripts make extensive use of the *numpy* module for matrix operations. The KEGG pathway available for visualization is manually added to the storage system, and there are no tools generated for retrieval of such KEGG pathways. However, KEGG allows for developers to access its open API, and this was used to download the KEGG Markup Language (KGML), an eXtensible Markup Language (XML) like format, file describing the pathway. It is imaginable that there exists libraries with support for downloading the KEGG pathways, e.g. BioPython.

Due to the security requirements for accessing NOWAC data, the computer currently hosting the web service is the same running the different clients. This is because there are not currently implemented any access control regulations in the web service that would permit unauthorized users to access the NOWAC dataset.

The different visualizations are all written in Javascript embedded into different web pages hosted on the web server. The first three make use of the D3 library, while the last uses three.js for visualization. When a web browser accesses a web page, the containing text, images and other content is downloaded to the client. If any Javascript code is found, the built-in interpreter reads the code and runs it. Using Javascript the entire code base is maintained at the server, which can update it anytime allowing seamless updates at the clients.

7.4.1 Average gene expression histogram

The first visualization shows the average difference in gene expression between the case-control pairs per gene. Figure 7.6 shows only the first bars of the visualization. In this visualization a Python script parses the NOWAC dataset and generates a Comma Separated Values (CSV) file that can be parsed by D3 used for plotting. The file contains tuples of gene number and value, which translates to a colored bar.

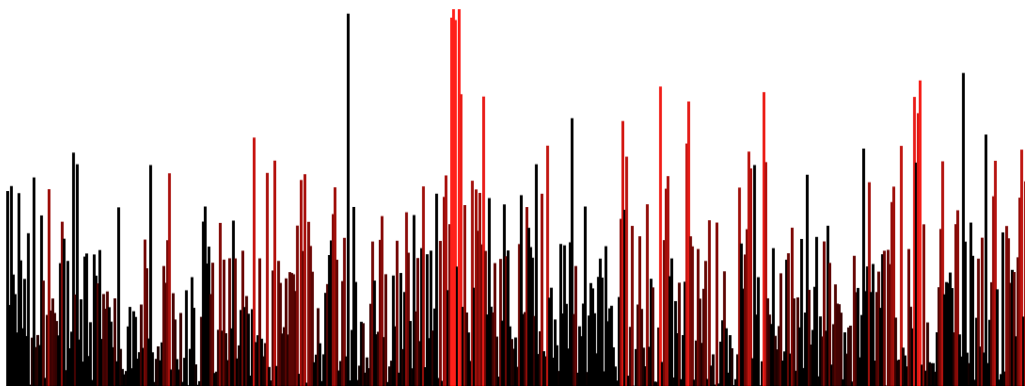


Figure 7.6: Histogram showing the average difference in gene expression between the case-control pairs.

The visualization is interactive and displays more detailed information about the gene numer and corresponding value, whenever a user hovers over the different bars.

7.4.2 Average year to diagnosis histogram

This is a simple interactive three-bar histogram. Like the previous the back-end provides the information required to construct the histogram, and the D3 library is used to visualize it. It was developed to demonstrate an interactive visualization using D3. The visualization is shown on figure 7.7

Figure 7.7: Simple histogram showing the average years to diagnosis in the NOWAC dataset.

7.4.3 NOWAC Heatmap

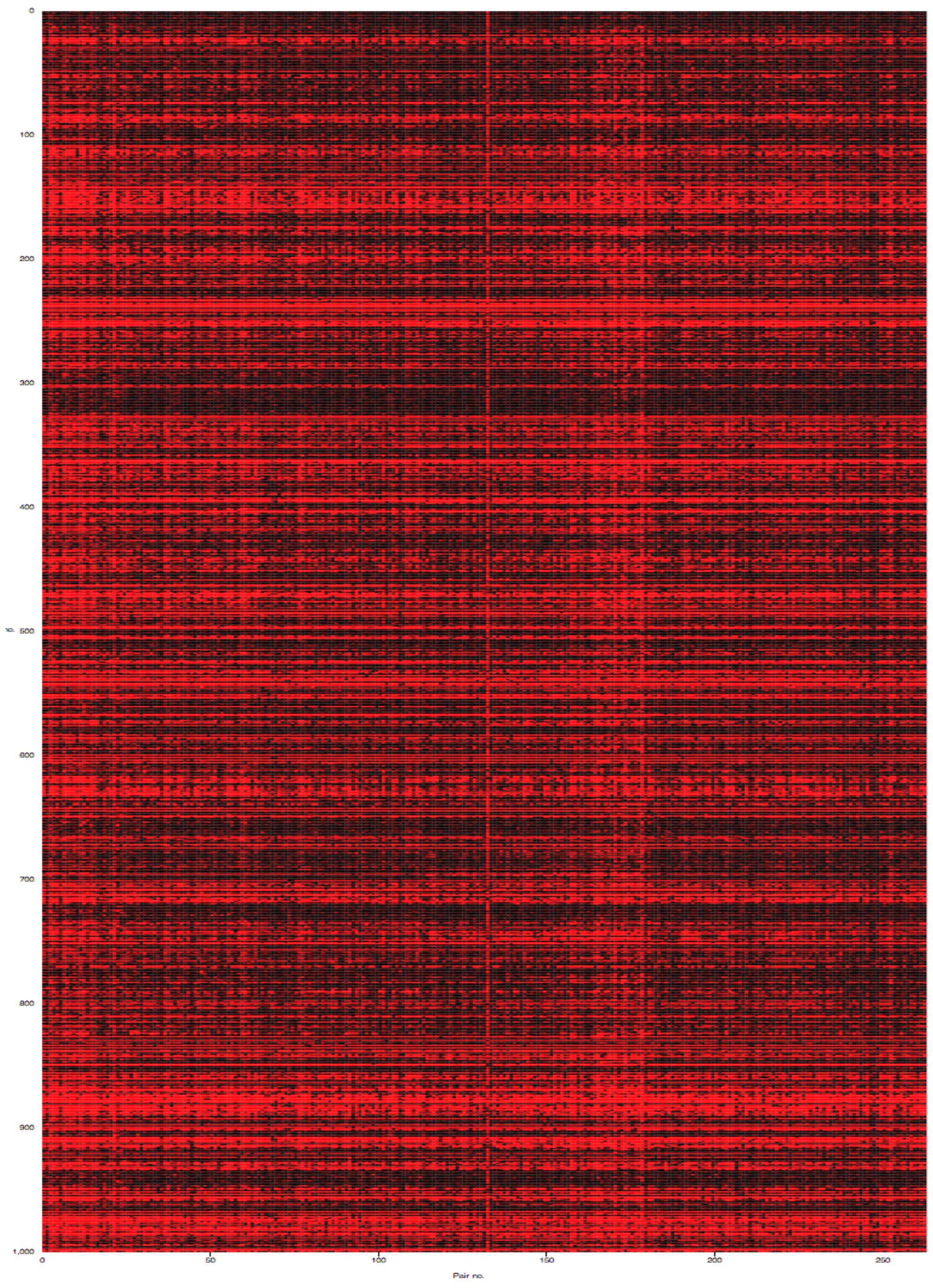


Figure 7.8: Heatmap visualization of the NOWAC dataset.

The Heatmap visualization was developed using two techniques. The first with a simple visualization using D3 that manipulates the DOM tree. This adds a DOM element for every (gene, expression value) pair, creates a square and colors it accordingly. However with the over 20 000 genes and 263 pairs of case-control, it uses large amounts of memory and takes a long time to load. Because of this a second visualization is provided. It uses the HTML5 canvas element, and a Render Queue [16] to render the different squares. This is a much faster and memory efficient visualization reducing load times significantly. Figures 7.8 and 7.9 shows the visualization.

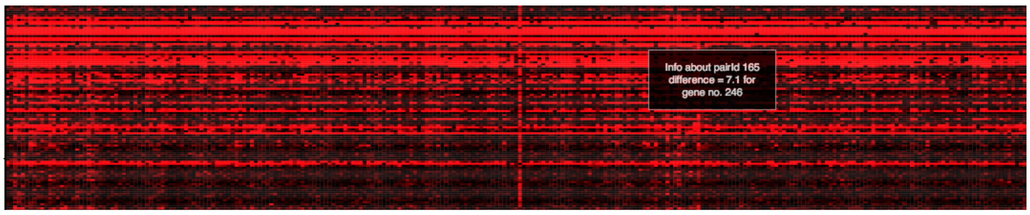


Figure 7.9: Heatmap show in Figure 7.8 zoomed in to reveal more information

7.4.4 KEGG Pathway

The KEGG pathway visualization imports a pathway from KEGG, translates it from KGML to a set of CSV files containing nodes and the paths between them, and visualizes this graph using three.js. All nodes are drawn using a single particle, and the edges between them are lines. To minimize edge and node overlap a force direction algorithm is applied to the graph. It does not make any distinction on what type of node read from the KGML file, so the visualization does not draw cells or genes differently. Figure 7.10 shows the visualization.

7.5 Discussion and Future Work

The prototype visualization tools for the NOWAC explorer show that new technologies using HTML5 and WebGL are suitable for visualization of biological data. The visualizations created are fully interactive and can be used to visualize the NOWAC dataset without any installation of third party software. In addition it has created a springboard for further development withing biological data visualization, a rapidly changing and interesting field.

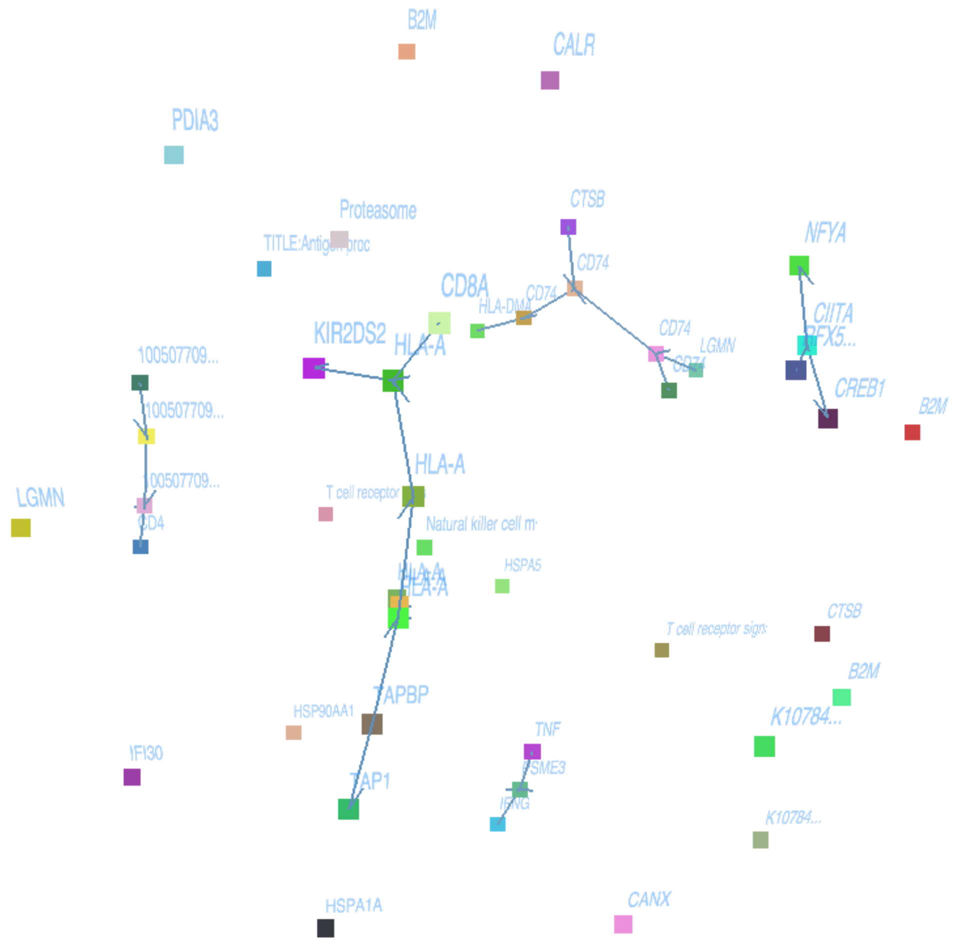


Figure 7.10: KEGG Pathway visualization

The visualizations were all relatively simple to create without any prior knowledge or experience in Javascript or graphics programming. The libraries used, D3 and three.js, provide a simple interface without having to concentrate on the low-level graphics programming, and were a great asset during the prototype development.

The prototype implemented still has plenty of features left to be completed in order to realize the system: (i) It lacks a designated storage system both for storing the NOWAC dataset, but also data from sources like KEGG; (ii) there is no component for downloading and visualizing an arbitrary KEGG pathways; (iii) the files sent over http for the visualizations can grow very large, e.g. the CSV file for the heatmap is 1.2 megabytes large; (iv) the system should be platform independent, but have not yet been fully tested on mobile devices; (v) the system lacks access control mechanisms for authenticating users; and (vi) the system supports only a handful of basic visualizations.

In future work we plan to focus on the visualization parts of the system, creating a generic pathway visualizer that can be integrated with the NOWAC dataset. This integration must be able to capture interesting parts of the dataset, hopefully by statistical analysis, and create an interactive data exploration tool for researchers.

To test the capabilities of three.js a 50.000 node and 25.000 edge graph was generated using three.js without any clever rendering techniques. It can be interactively explored without any problems, and demonstrates the power of WebGL in the web browser. Also it demonstrates that every human gene can be plotted without any problem using three.js. However, using the current force direction implementation causes the visualization to run dead slow.

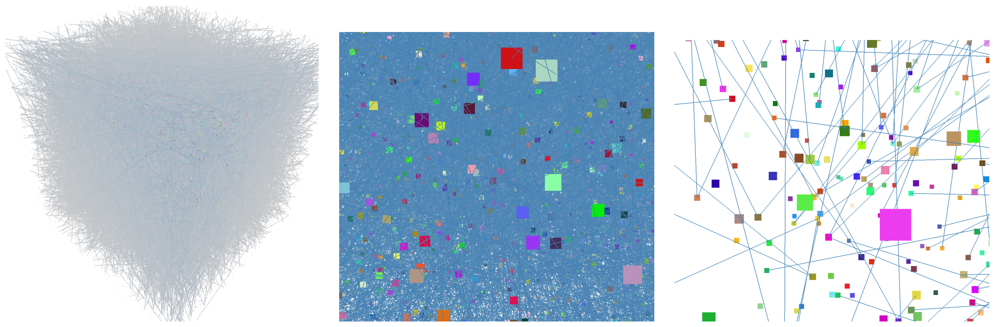


Figure 7.11: Illustration of the different zoom-levels of the graph visualization. The graph is randomly generated and consists of 50 000 nodes and 25 000 edges.

To conclude, our experience show that implementing biological data visualization tools in the modern web browser provide fully functional tools requiring relatively short development time.

Chapter 8

Summary

This report has given a review of the different stages in big data analysis and visualization pipelines.

The report has described the different types of epidemiological studies used in the research communities today, including the prospective cohort NOWAC study. It has given an introduction to molecular biology by introducing the building blocks of an organism and its genetic code, in order to understand different diseases studied in epidemiology, such as cancer. The report has presented the modern technologies used by researchers to measure and collect information about an individuals genetic material, and describe why these produce immensely large datasets. Continuing it has described state of the art big data systems for managing large datasets, and how visualization systems handle the drive towards big data exploration. It describes both state-of-the-art commercial and biology visualization tools and the challenges tackled by these. The report also contain a review of a few systems to meet future demands, such as interaction on high-resolution tiled display walls. Finally it has described the first prototype of the NOWAC Explorer, a simple prototype of a biology data exploration tool. It was created to understand the data structures and requirements of visualizing NOWAC data. The NOWAC data is visualized using 2D heatmap visualizations and 3D metabolic pathway graphs. It interacts with the KEGG database and utilizes technologies for visualizing data in the web browser.

The main lesson learned in this project is that recent advances in epediomolgy, biology instruments, and big data systems are making it possible to conduct new studies using massive data sets. However, we found that current biology visualization systems have not fully reached their potentional to support novel biological discoveries.

References

[1] 10gen Inc. MongoDB homepage. http://www.mongodb.org/, 2012.

[2] 10gen Inc. The MongoDB NoSQL Database Blog, 32-bit limitations. http://blog.mongodb.org/post/137788967/32-bit-limitations, 2013.

[3] Atul Adya, Gregory Cooper, Daniel Myers, and Michael Piatek. Thialfi: A client notification service for internet-scale applications. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pages 129–142. ACM, 2011.

[4] ALoopingIcon. Porin.qutemol.ao.png - Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/File:Porin.qutemol.ao.png.

[5] ALoopingIcon. Porin.qutemol.dl.png - Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/File:Porin.qutemol.dl.png.

[6] Miguel Andrade. Heatmap.png - Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/File:Heatmap.png, 2013.

[7] Otto J Anshus, Daniel Stødle, Tor-Magne Stien Hagen, Bård Fjukstad, John Markus Bjørndalen, Lars Ailo Bongo, Yong Liu, and Lars Tiede. Nine Years of the Tromsø Display Wall. *CHI 2013 Extended Abstracts*, 2013.

[8] Aza and Toth. Myoglobin.png - Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/File:Myoglobin.png, 2013.

[9] Doug Beaver, Sanjeev Kumar, Harry C Li, Jason Sobel, Peter Vajgel, and Others. Finding a needle in Haystack: Facebooks photo storage. *Proc. 9th USENIX OSDI*, 2010.

[10] BioConductor. Screenshots and Graphics -bioconductor.org. `https://phssec1.fhcrc.org/secureplone/bioconductor.org/overview/screenshots/`, 2013.

[11] Dhruba Borthakur. The hadoop distributed file system: Architecture and design. *Hadoop Project Website*, 11:21, 2007.

[12] Michael Bostock. D3.js - Data-Driven Documents. `http://d3js.org`, 2013.

[13] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3: Data-Driven Documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2011.

[14] Mike Burrows. The Chubby lock service for loosely-coupled distributed systems. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 335–350. USENIX Association, 2006.

[15] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. BigTable: A Distributed Storage System for Structured Data. *ACM Transactions on Computer Systems*, 26(2):1–26, June 2008.

[16] Kai Chang. Render Queue. `http://bl.ocks.org/syntagmatic/raw/3341641/`.

[17] John Cook. Tableau making name for itself - seattlepi.com. `http://www.seattlepi.com/news/article/Tableau-making-name-for-itself-1152891.php`.

[18] Valerio Costa, Claudia Angelini, Italia De Feis, and Alfredo Ciccodicola. Uncovering the complexity of transcriptomes with RNA-Seq. *Journal of biomedicine and biotechnology*, 2010:853916, January 2010.

[19] M Courtney. Puzzling out big data [Information Technology Analytics]. *Engineering & Technology*, 7(12):56–60, 2013.

[20] Jeffrey Dean and Sanjay Ghemawat. MapReduce: a flexible data processing tool. *Communications of the ACM*, 53(1):72, January 2010.

[21] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: amazon's highly

available key-value store. In *ACM Symposium on Operating Systems Principles: Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, volume 14, pages 205–220, 2007.

[22] Vanessa Dumeaux, Karina S Olsen, Gregory Nuel, Ruth H Paulssen, Anne-Lise Bø rresen Dale, and Eiliv Lund. Deciphering normal blood gene expression variation–The NOWAC postgenome study. *PLoS genetics*, 6(3):e1000873, March 2010.

[23] Greg Elgar and Tanya Vavouri. Tuning in to the signals: noncoding sequence conservation in vertebrate genomes. *Trends in genetics : TIG*, 24(7):344–52, July 2008.

[24] Marcel Kornacker & Justin Erickson. Cloudera Impala: Real-Time Queries in Apache Hadoop, For Real. `http://blog.cloudera.com/blog/2012/10/cloudera-impala-real-time-queries-in-apache-hadoop-for-real/`, 2013.

[25] W. Fiers, R. Contreras, F. Duerinck, G. Haegeman, D. Iserentant, J. Merregaert, W. Min Jou, F. Molemans, A. Raeymaekers, A. Van den Berghe, G. Volckaert, and M. Ysebaert. Complete nucleotide sequence of bacteriophage MS2 RNA: primary and secondary structure of the replicase gene. *Nature*, 260(5551):500–507, April 1976.

[26] Apache Software Foundation. Hadoop Wiki: Project Description. `http://wiki.apache.org/hadoop/ProjectDescription`, 2013.

[27] Nils Gehlenborg, Seán I O'Donoghue, Nitin S Baliga, Alexander Goesmann, Matthew A Hibbs, Hiroaki Kitano, Oliver Kohlbacher, Heiko Neuweger, Reinhard Schneider, Dan Tenenbaum, and Anne-Claude Gavin. Visualization of omics data for systems biology. *Nature methods*, 7(3 Suppl):S56–68, March 2010.

[28] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google file system. *ACM SIGOPS Operating Systems Review*, 37(5):29, December 2003.

[29] Gladstone Institues - and University of California at San Francisco. GenMAPP - Home Page. `http://www.genmapp.org/default.html`.

[30] Google. Google Earth. `http://www.google.com/earth/index.html`.

[31] Google. Chrome Experiments - WebGL Globe. http://www.chromeexperiments.com/globe, 2013.

[32] Tor-Magne Stien Hagen. Interactive Visualization on High-Resolution Tiled Display Walls with Network Accessible Compute-and Display-Resources. 2011.

[33] IBM. IBM What is big data? - Bringing big data to the enterprise. http://www-01.ibm.com/software/data/bigdata/, May 2013.

[34] Ahmedin Jemal, Freddie Bray, Melissa M Center, Jacques Ferlay, Elizabeth Ward, and David Forman. Global cancer statistics. *CA: a cancer journal for clinicians*, 61(2):69–90, 2011.

[35] S. D. Kahn. On the Future of Genomic Data. *Science*, 331(6018):728–729, February 2011.

[36] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Wrangler: interactive visual specification of data transformation scripts. In *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*, page 3363, New York, New York, USA, May 2011. ACM Press.

[37] Kanehisa Laboratories. KEGG: Kyoto Encyclopedia of Genes and Genomes.

[38] Johannes Kehrer and Helwig Hauser. Visualization and Visual Analysis of Multi-faceted Scientific Data: A Survey. *IEEE transactions on visualization and computer graphics*, April 2012.

[39] Khronos Group. WebGL - OpenGL ES 2.0 for the Web. http://www.khronos.org/webgl/, 2013.

[40] Avinash Lakshman and Prashant Malik. Cassandra. *ACM SIGOPS Operating Systems Review*, 44(2):35, April 2010.

[41] Lodish. *Molecular Cell Biology,6e.* W.H.Freeman and Company, 2007.

[42] Eiliv Lund. The Norwegian Women and Cancer study, NOWAC. http://site.uit.no/nowac/.

[43] Michael L Metzker. Sequencing technologies - the next generation. *Nature reviews. Genetics*, 11(1):31–46, January 2010.

[44] National Institute of Health. 1000 Genomes Project data available on Amazon Cloud, March 29, 2012 News Release - National Institutes of Health (NIH). `http://www.nih.gov/news/health/mar2012/nhgri-29.htm`, 2013.

[45] Seán I O'Donoghue, Anne-Claude Gavin, Nils Gehlenborg, David S Goodsell, Jean-Karim Hériché, Cydney B Nielsen, Chris North, Arthur J Olson, James B Procter, David W Shattuck, Thomas Walter, and Bang Wong. Visualizing biological data-now and in the future. *Nature methods*, 7(3 Suppl):S2–4, March 2010.

[46] Christopher Olston, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, and Andrew Tomkins. Pig latin: a not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1099–1110. ACM, 2008.

[47] Laura Wegener Parfrey, Daniel J G Lahr, and Laura A Katz. The dynamic nature of eukaryotic genomes. *Molecular biology and evolution*, 25(4):787–94, April 2008.

[48] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[49] Nataniel Rothman, Pierre Hainaut, Paul Schulte, Martyn Smith, Paolo Bofetta, and Frederica Perera. *Molecular epidemiology: principles and practices*. International Agency for Research on Cancer, 2011.

[50] Saqibali. Google File System (GFS) - Google Drive. `https://docs.google.com/drawings/d/1OQXFUhMxxoYcfDqbBHBKizdzVO1st-XyIrZSpTwu3uU/edit?hl=en_US&pli=1`.

[51] Yair Sovran, Russell Power, Marcos K Aguilera, and Jinyang Li. Transactional storage for geo-replicated systems. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pages 385–400. ACM, 2011.

[52] Daniel Stødle. *Device-Free Interaction and Cross-Platform Pixel Based Output to Display Walls*. PhD thesis, University of Tromsø, 2009.

[53] Daniel Stødle. Interaction Spaces: A camera-based system for scalable multi-touch 2D and 3D interaction on wall-sized, high-resolution displays. `http://www.scsc.no/daniel/multi-touch/index.html`, May 2013.

[54] Eliza Strickland. The gene machine and me. *IEEE Xplore*, 50(3):30–59, 2013.

[55] Tableau Software. Tableau Desktop. `http://www.tableausoftware.com/products/desktop`, 2012.

[56] Ronald C Taylor. An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. *BMC bioinformatics*, 11(Suppl 12):S1, 2010.

[57] The Apache Software Foundation. Apache CouchDB. `http://couchdb.apache.org/`, 2013.

[58] Threejs.org. three.js - JavaScript 3D library. `http://threejs.org/`, 2013.

[59] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning Zhang, Suresh Antony, Hao Liu, and Raghotham Murthy. Hive-a petabyte scale data warehouse using hadoop. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 996–1005. IEEE, 2010.

[60] TIBCO Software Inc. TIBCO ActiveSpaces. `http://www.tibco.com/multimedia/ds-activespaces_tcm8-9043.pdf`, 2012.

[61] TransControl. RNA-codons.png - Wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/File:RNA-codons.png`, 2013.

[62] Arthur Vander, James Sherman, and Dorothy Luciano. *Human Physiology: The Mechanisms of Body Function.* McGraw-Hill, 8th edition, 2001.

[63] J C Venter *et. al.* The sequence of the human genome. *Science (New York, N.Y.)*, 291(5507):1304–51, February 2001.

[64] Visual Computing Laboratory at ISTI - CNR. QuteMol. `http://qutemol.sourceforge.net/`, 2013.

[65] W3C. HTML 5.1 Nightly. `http://www.w3.org/html/wg/drafts/html/master/`, 2013.

[66] Ted Wasserman. Announcing Integration with Cloudera Impala: Fast Analytics for Hadoop. `http://www.tableausoftware.com/about/blog/2012/10/cloudera-impala-19726`.

[67] Richard Wesley, Matthew Eldridge, and Pawel T. Terlecki. An analytic data engine for visualization in tableau. In *Proceedings of the 2011 international conference on Management of data - SIGMOD '11*, page 1185, New York, New York, USA, June 2011. ACM Press.

[68] Leishi Zhang, A Stoffel, M Behrisch, and S Mittelstädt. Visual Analytics for the Big Data EraA Comparative Review of State-of-the-Art Commercial Systems. In *Proceedings of IEEE Symposium on Visual Analytics Science and Technology*, pages 173–182, 2012.

Appendix A

Source Code

The source code for the prototype of the NOWAC Explorer

The source code is attached on a CD-ROM, and contains two folders:

- **Heatmaps and histograms**
- **Kegg Pathways**

Since the NOWAC dataset is of both commercial and scientific interest it cannot be shared without permission. Because of this the heatmap and histogram visualizations will not function on a computer without the dataset. If required, temporary access can be granted to anyone requesting to see the visualizations, just contact `bfj001@post.uit.no`. The graph visualziation should however work fine without the NOWAC dataset.

The visualizations can be run starting

```
    python -m SimpleHTTPServer
```

from the root folder and accessing `localhost:8000` in the local web browser

Heatmaps and Histograms

- `average_diff_per_gene.html` contains the html page for creating the avg. difference per gene.
- `colorbrewer.css` contains some css that can be used in the visualizations that are using d3. this is not currently used
- the `d3` folder contains the source code for the d3 library
- `expression_avg.py` contains the python code required to generate the histogram labels for the two histograms.
- `expression_d3_datagen.py` empty
- `exprs.js` javascript code for the average difference in gene expression
- `gl-matrix` matrix library. not used.
- `heatmap.html` html page for the heatmap visualization using DOM manipulation
- `heatmap.js` javascript for generating the heatmap using DOM manipulation
- `http_server.py` simple HTTP server. used when testing the different visualizations.
- `index.html` overview of the possible heatmap and histogram visualizations
- `playground.html` html page that contains the code for the heatmap using a render queue.
- `playground.js` javascript for generating the heatmap using render queue.
- `style.css` some styles
- `three` three.js library
- `three.html` threejs test
- `three.js` threejs test
- `webgl.html` WebGL testing, not complete.
- `webgl.js` WebGL testing, not complete.
- `years_hist.py` python script for generating the years to diagnosis histograms.
- `years_to_diag_histogram.html` html page for years to diagnosis histograms.
- `years_to_diag.js` javascript code for visualizing the years to diagnosis histograms.

Kegg Pathway

- `circle.png` image of a circle. can be used to represent a node in the different graphs. not used.
- `csvGraph.html` example graph generated from a set of csv-files.
- `forceDirectedGraph.html` the kegg pathway visualization
- `graph.csv` input to csv graph
- `index.html` overview of possible visualizations
- `js` javascript files:
    - `csvGraph.js` javascript for generating graph from csv files
    - `forceDirectedGraph.js` kegg graph visualization
    - `forceDirectedGraph.js.old` old implementation of the kegg graph. not used.
    - `randomParticleGraph.js` javascript for generating a large random graph.
    - `randomSphereGraph.js` javascript for generating a random graph using spheres as nodes.
- `kgml_nodes.csv` nodes in the kegg pathway. used in visualization.
- `kgml_paths.csv` paths in the kegg pathway. used in visualization.
- `lib` different libraries used
    - `d3` d3 library
    - `gl-matrix` matrix library
    - `jquery` jquery library. for fetching csv files.
    - `three` threejs library.
- `nodes.csv` node list for csv graph
- `particle.png` image that can be used to represent a node. not used.
- `paths.csv` path list for csv graph
- `python` various python scripts
    - `data_generator.py` for generating random graphs
    - `kgml` folder containing kgml files downloaded from the kegg web site
    - `kgmlparser.py` scipt for parsing kgml files and writing a set of nodes & paths csv files.
- `randomParticleGraph.html` html page for random particle graph.
- `randomSphereGraph.html` html page for random sphere graph.
- `stats.csv` node list. not used.