FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Information and Data Analysis System for Gene Expression

Diogo André Rocha Teixeira

DISSERTATION PLANNING

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Rui Camacho

Second Supervisor: Nuno Fonseca (EMBL-EBI, Cambridge, UK)

February 10, 2014

Information and Data Analysis System for Gene Expression

Diogo André Rocha Teixeira

Mestrado Integrado em Engenharia Informática e Computação

February 10, 2014

Abstract

Resumo

Contents

1 Introduction **1**

1.1 Context . . . 1

1.2 Motivation and Objectives . . . 2

1.3 Project . . . 2

1.4 Structure of the Report . . . 3

2 State-of-the-Art **5**

2.1 Biological Base Concepts . . . 5

2.2 RNA Sequencing and Transcriptome Assembly . . . 7

2.2.1 RNA Sequencing Tools . . . 7

2.2.2 Relevant Standard File Formats . . . 9

2.3 Data Mining . . . 10

2.3.1 Data Mining Algorithms . . . 11

2.3.2 Model Evaluation Procedures and Measures . . . 14

2.3.3 Data Mining Tools . . . 17

2.4 Chapter Conclusions . . . 19

3 Work Plan **21**

3.1 Planning . . . 21

3.2 Experimental Data . . . 23

3.3 Thesis Work Evaluation . . . 23

4 Conclusions **25**

References **27**

CONTENTS

List of Figures

2.1 Overall structure of a gene 6
2.2 Removal of introns from precursor mRNA 6
2.3 Example of a decision tree 12
2.4 Example of linearly separable training data 13
2.5 Example of ROC curve 16
2.6 RapidMiner user interface 18
2.7 Weka interface selection 18

3.1 Work distribution planning 22
3.2 Specimen of *Drosophila melanogaster* 24

LIST OF FIGURES

Abbreviations

API	Application Programming Interface
AUC	Area Under the Curve
BLAST	Basic Local Alignment Search Tool
cDNA	Complementary DNA
CSS	Cascading Style Sheets
DBMS	Database Management System
DNA	Deoxyribonucleic Acid
GUI	Graphical User Interface
HTML	HyperText Markup Language
IBMC	Institute for Molecular and Cell Biology *(Instituto de Biologia Molecular e Celular)*
ILP	Inductive Logic Programming
K-NN	K-Nearest-Neighbors
mRNA	Messenger RNA
NCBI	National Center for Biotechnology Information
NGS	Next Generation Sequencing
RNA	Ribonucleic Acid
RNA-Seq	RNA Sequencing
ROC	Receiver Operating Characteristic
SAM	Sequence Alignment/Map
SVM	Support Vector Machine
rRNA	Ribosomal RNA
tRNA	Transfer RNA
WTSS	Whole Transcriptome Shotgun Sequencing

Chapter 1

Introduction

This chapter aims at giving a general overview about the themes addressed by this thesis. We will present the problem we are addressing, as well as the motivation that led to its choice. Furthermore there will be a brief description of the thesis main objectives and the methods that will be used to achieve those objectives.

1.1 Context

Molecular biology is a branch of biology that studies biological activities of living beings, at a molecular level. The early grounds for this field of study were set in the early 1930's, although it only emerged in its modern form in the 1960's, with the discovery of the structure of DNA. Among the processes studied by this branch of biology is gene expression. Gene expression (further explained in Chapter 2) is the process by which DNA molecules are transformed into useful genetic products, typically proteins, which are essential for living organisms. This knowledge is not only important in fields like evolutionary or molecular biology, but may have crucial applications in fields such as medicine. One example of such an application is the usage of gene expression analysis in the diagnosis and treatment of cancer patients [PASH03].

With the advent of NGS (Next Generation Sequencing) techniques, researchers have at their disposal huge amounts of sequencing data, that is not only cheaper and faster to produce, but also more commonly available. This data can then be used to obtain relevant information about organisms' gene expression. But, as the cost of sequencing genomes was reduced, the cost of processing such information was increased. NGS techniques tend to produce much smaller reads[1] than previously used techniques, presenting a more complicated problem, from a computational standpoint [Wol13].

[1]A *read* is a single fragment of a genome/transcriptome, obtained through sequencing techniques.

1.2 Motivation and Objectives

Despite its great advancements in the past decades, molecular biology is still a relatively new subject and, as such, there are still some unknowns and partial knowledge in this area. In respect to gene expression, some mechanisms of this intricate process are yet to be fully understood. One such mechanism is the one that regulates the transcription speed of RNA. The objective of this thesis is to understand how the final segments of the genome's exons are responsible for the speed at which the exons themselves are transcribed. This is, however, a complex task, that can be further decomposed in the two main problems that will be address in the thesis, namely:

- Assembly of the study transcriptome, using experimental sequencing reads and a reference genome. This is effectively one of the most complex problems addressed in this thesis. In order to assemble the genome a method called RNA Sequencing[2] will be used. Further insight about this method will be given in Chapter 2, with particular emphasis for RNA Sequencing tools (Section 2.2.1).

- Further analysis of the assembled transcriptomes, using machine learning algorithms applied to data mining. These techniques will be used in an effort to try to understand the already mentioned transcription mechanisms. This topic will be developed in Section 2.3.

Solving these problems requires the use of computational tools. As such, the development of a computer system to address these problems emerges as a secondary objective of the thesis. Some details of this system will be presented in Section 1.3, along with its overall structure, main components and possible technologies to be used.

1.3 Project

The project itself will revolve around the development of a prototype computer system. The first objective of this prototype is to solve the aforementioned thesis problems, namely the transcriptome assembly and analysis. Beyond this objective, the prototype should become an easy to use and useful tool for any researcher investigating this or other similar problems. To fulfill these objectives, we will need to develop a complex system, composed by several smaller systems. Therefore, the envisioned system architecture is divided into three major components, to wit:

Information system is responsible for storing and managing genetic data, coordinating interaction between the other components of the system and providing a web interface for user interaction. This component will be based mainly on typical web technologies, that is, relational databases for data storage (SQL DBMS's), web frameworks for business logic implementation (Ruby on Rails, Padrino, NodeJS[3]) and web markup and styling languages for interface implementation (HTML, CSS).

[2]RNA Sequencing is also referred to as "Whole Transcriptome Shotgun Sequencing", or WTSS.

[3]Ruby on Rails and Padrino are Ruby based web frameworks, while NodeJS is a Javascript based web framework.

Assembly pipeline will use genetic data stored in the information system in order to produce assembled transcriptomes. This pipeline will be composed by several tools, corresponding to each phase of the RNA Sequencing process, possibly intercalated with data format conversion programs. The tools to be used in this component will be further discussed in Section 2.2.1.

Transcriptome analysis will be responsible for the data mining analysis of the assembled transcriptomes, in the context of the problem of the thesis. It is expected that this component integrates with the rest of the system. Further information about the tools that will be used in this component is given in Section 2.3.3.

From here, this document will not dwell in the details of the implementation of such a system, focusing instead the specificities of the problem's solution, from the molecular biology and data mining perspectives. This is due to the fact that the development of the system itself is not the focus of the thesis, but rather a natural consequence of the project's work process.

1.4 Structure of the Report

Besides the introduction chapter, this document is composed three additional chapters. These chapters have the following structure:

Chapter 2 introduces some basic Biology and RNA Sequencing concepts, that are essential to understand the problems with which this document deals. Furthermore, we describe the main techniques used for genome/transcriptomesequencing and assembly, their differences and applications and the tools and data formats typically used on those areas. Lastly, we give some insight about data mining algorithms and how they will be applied to this work.

Chapter 3 outlines the main steps in the development of this thesis (and the respective software prototype) and attempts to provide a feasible schedule for the work's execution. It also presents the datasets that will be studied and used in this work, their origins and features, as well as the validation methods that will be used to ascertain the quality of our results.

Chapter 4 sums up the what has been defined in the report, emphasizing the problem that the thesis addresses and the work that will be executed towards solving that problem. It will also give a brief idea of what are the expected results at the end of the project.

Chapter 2

State-of-the-Art

In this chapter we will begin by making a more in depth presentation of the process of gene expression. This will be followed by a literature and state of the art review in the fields of genome/-transcriptome assembly and data mining. Lastly, we will present some of the tools used in each of those areas, as well as some relevant data representation formats for genetic data.

2.1 Biological Base Concepts

Before dwelling in the details of the state of the art that are on the foundation of this thesis, it is important to explain some concepts of the domain of molecular biology. As explained in Section 1.1, gene expression is the mechanism by which an organism's DNA can be expressed into functional genetic products, like proteins, rRNA and tRNA. This process starts with the genetic code, or nucleotide sequence, of each gene. Different genes in an organism's DNA are responsible for the creation of different genetic products. The process of gene expression itself is composed by two main stages, transcription and translation [GEN].

Transcription is the stage at which genetic data in the form of DNA is used to synthesize RNA, being this the process that concerns the thesis main question. Several different types of RNA are produced by this process, including mRNA (which specifies the sequences of amino acids that form a protein), rRNA and tRNA, both later used in the translation stage. Simplifying a gene's structure, it can be seen as composed by two types of areas, introns and exons, as seen in Figure 2.1.

Only the exons are useful in the gene expression process, being also known as coding regions. Introns, on the other hand, are not used in the process. They are present in an early stage mRNA molecule, the precursor RNA, but are later removed (or spliced) in the final molecule before the translation stage [GEN]. Figure 2.2 illustrates the removal of introns from the mRNA molecule, during the splicing process. As stated before, the main goal of this thesis is to explain how the final nucleotide sequence of each exon affects the transcription speed of the exon itself.

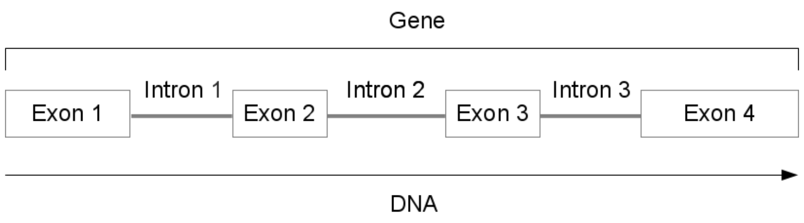


Figure 2.1: Overall structure of a gene, with its different areas (simplified).

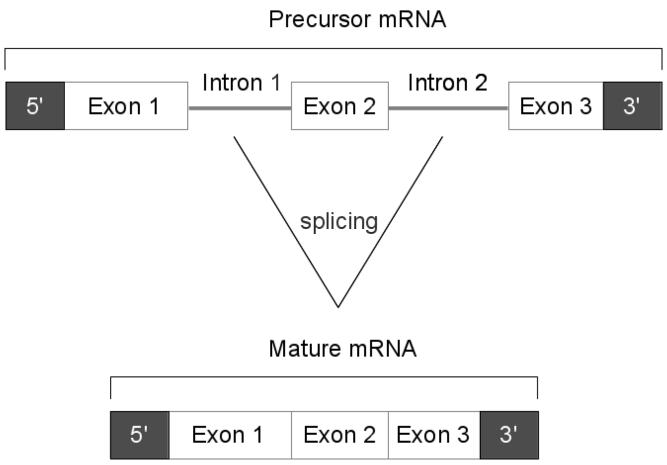


Figure 2.2: The removal (splicing) of introns from the precursor mRNA, during the transcription process.

After the conclusion of the transcription process comes the translation process. In this process, the synthesized mRNA is used to specify the sequence of amino acids that constitute the particular protein being produced. The other types of RNA molecules (rRNA and tRNA) are also used in this stage of the gene expression process.

Obtaining this genetic information is done experimentally, by employing a genome sequencing technique. For quite some time this process was carried out using the Sanger sequencing method and other similar methods [RF09]. Though effective, such methods we're notably slow and costly, with large projects like the Human Genome Project (HGP) consuming roughly thirteen years and US$ 3 billion. Other than the realm of human genetics, this kind of study was restricted to model organisms, such as the fruit fly and mouse genomes [Wol13]. The past few years have seen the appearance and rise in popularity of the NGS techniques. These techniques differ from the more classical ones by producing larger amounts of information, in less time. They are also typically more cost effective than previous techniques and can be easily employed by single laboratories, which has greatly contributed to their popularity. As a disadvantage, NGS techniques produce shorter reads than their older counterparts, being that "*(...) transcriptome assembly from billions of RNA-Seq reads (...) poses a significant informatics challenge*" [MW11, p. 671] Although this thesis will not deal with the problems of sequencing techniques, it is important to indicate that the read dataset that will be used is a result of NGS techniques. As such, we will use assembly techniques more suited to situations where short reads are available.

2.2 RNA Sequencing and Transcriptome Assembly

Transcriptome assembly is the process by which experimentally obtained genetic data reads can be organized and merged together in a partial or complete genome expression profile. As stated above, the advent of next generation sequencing techniques, with their reduced costs, greatly increased the availability of genome sequencing data.

For years, microarrays were the standard tool available for examining features of the transcriptome and global patterns of gene expression [Wol13]. However, microarrays typically produce data more oriented towards assembly against existing reference data, hence limiting its application to species with well known reference genomes. This is impractical, as NGS techniques allow to cheaply obtain genetic information of previously not studied species. This is one of the reasons that led to the inception of RNA-Seq. Contrary to microarrays, RNA-Seq techniques are able to wield results that are suitable for both reference guided assembly and *de novo* assembly approaches [WL09]. *De novo* or exploratory assembly has captured the interest of researchers in the past few years, leading to the appearance of multiple RNA-Seq tools that are capable of making this type of assembly without a reference genome [EBS+11]. Despite this amazing capability, we will restrict this project to reference genome guided problems, which are simpler and more oriented to a Masters level thesis.

2.2.1 RNA Sequencing Tools

Below we will present some bioinformatic tools, used to support the multiple steps of the RNA-Seq process. Although there are several tools capable of executing all steps of the RNA-Seq process, it has been decided that in this project we will create our own assembly pipeline, using specialized tools for every step. We will also present some of the most popular file formats used in this context, along with some tools used to manipulate these files. We will focus in open source, Unix command line based tools.

Tuxedo Suite

The Tuxedo suite is a free, open-source collection of applications that has been widely adopted as analysis toolset for fast alignment of short reads. It is composed by four separate tools, TopHat, Bowtie, Cufflinks and CummRbund, briefly reviewed below. These tools are extensively used for RNA Sequencing analysis. Although the applications are made for command line execution, there are several workflow managers, like Galaxy[1], that easily integrates with the suite, providing a web interface for its use.

[1] http://galaxyproject.org/

Bowtie. Bowtie[2] is an ultrafast, memory-efficient short read aligner. Bowtie is typically used to build a reference index for the genome of the organism being studied, for posterior use by other tools, like TopHat. It can also output alignments in the standard SAM format, allowing Bowtie to interoperate with tools like SAM Tools. However, it shouldn't be used as a general purpose alignment tool, as it was created and is more effective when aligning short read sequences against large reference genomes.

TopHat. TopHat[3] is a fast splice junction mapper for RNA Sequencing reads. It uses Bowtie as the underlying aligning tool, using its results and a FASTA formated reference genome to identify splice junctions between exons.

Cufflinks. Cufflinks[4] assembles transcripts, estimates their abundances, and tests for differential expression and regulation in RNA Sequencing samples. It uses the SAM or BAM formatted files as input, typically the ones produced by TopHat, outputting GTF files as a result.

CummeRbund. Lastly, CummeRbund[5] is an R package (see Section 2.3.3) designed to help the visualization and analysis of Cufflinks' RNA Sequencing output. As such, it is not directly involved in the trascriptome alignment process. It takes the various output files from Cufflinks and uses them to build a SQLite database describing appropriate relationships between genes, transcripts, etc. It implements several plotting functions, as well as commonly used data visualizations.

SAM Tools

SAM Tools[6] is a library and package designed for parsing and manipulating alignment files in the SAM/BAM format [LHW+09] (see Section 2.2.2). SAM Tools has two separate implementations, one in C and the other in Java, with slightly different functionality. Beyond manipulation of SAM and BAM files, this package is able to convert between other read alignment formats, sort and merge alignments and show them in a text-based viewer.

BLAST

BLAST[7] is a tool, implemented in C++, that is used to find regions of local similarity between biological sequences. It uses FASTA sequences (see Section 2.2.2) as search input and outputs the results reports in XML, HTML or plain text. There are several different BLAST programs available at the moment, that can be used depending on our objective and type of data. BLAST

[2] http://bowtie-bio.sourceforge.net/index.shtml
[3] http://tophat.cbcb.umd.edu/
[4] http://cufflinks.cbcb.umd.edu/
[5] http://compbio.mit.edu/cummeRbund/
[6] http://samtools.sourceforge.net/
[7] http://blast.ncbi.nlm.nih.gov/Blast.cgi

is particularly useful to search biologic sequence databases, but can be used in for other purposes, like identifying an unknown species or comparing common genes in two related species.

2.2.2 Relevant Standard File Formats

As expected, the great diversity of RNA Sequencing tools brings with it a wealth of file formats. Some of these formats are developed from the ground up to satisfy a specific need, while other are mere contextual adaptations or specializations of already established formats. Below we will present a few of the most popular and wide spread file formats, talking about their basic structure, the types of data they represent and their applications.

FASTA

FASTA is the standard line and character sequence format used by NCBI [NCB], using this last organization's character code conventions. It is a simple format, that can be used to easily store data represented by character sequences, like nucleotide (DNA, RNA) or amino acid (protein) sequences. This file format is widely use to store sequencing reads, DNA/RNA sequences and other character sequences in database systems. Its simplicity makes it extremely easy to manipulate and parse, presenting also an attractive solution for data transfer between different tools.

FASTQ

FASTQ is used to store store character sequences, typically nucleotide sequences [CFG+10]. It's quite similar to the standard FASTA format, in respect to the manner in which character sequences are represented. However, for every sequence, there is a second sequence of equal length, representing the quality scores of the original sequence. These quality scores are also represented as single characters, taking values between and including ASCII-33 to ASCII-126. It's typically used in the same situations as the FASTA format, when quality scores are available/relevant.

SAM and BAM

The SAM format is a text format for storing sequence alignment data [Lab]. It is widely used to store mapping information between sequencing reads and a given reference genome. This sort of information is typically the product of sequencing alignment tools, that consume sequencing reads from FASTQ files and align them with a reference genome.

The BAM format contains exactly the same information as the SAM format and the same rules apply for both formats. The difference between both formats lies in their encoding. While SAM is a text based format, BAM is a binary format. This means that BAM sacrifices human readability for increased machine processing performance, as it is more efficient to work with compressed and indexed binary data.

VCF

VCF is a text file format used to store gene sequence variants [Smi13]. In the past few years, as larger and larger DNA sequencing projects became more common (like the 1000 Genomes Project[8]), storing such large amounts of information became a serious concern. To address these concerns the VCF format was created. Instead of storing the complete genome, VCF stores only the variations (and their respective positions) of newly sequenced genomes relatively to a known reference genome, typically in a compressed text file. As such, it is a format often used when building genome databases.

GFF and GTF

GFF is a text based file format to store gene features [San11]. Many genome assembly tools execute this process in two seperate steps: feature detection for identification of specific regions (exons, introns, etc.) and genome assembly, using those features as reference. However, often times it is beneficial to decouple these two steps, using different and more efficient tools for each. As such, the GFF format emerged as a protocol for feature information transfer between tools.

The GTF format is similar to the GFF format, in which it is based. It is also used in similar situations. However, GTF builds on top of GFF, defining additional conventions, specific to the domain of genetic information. Despite their initial relation, both formats are developed individually.

2.3 Data Mining

Data mining is the process of "*extracting or "mining" knowledge from large amounts of data*" [HKP06, p. 5]. As such, it consists in a set of techniques that can be used to find interesting patterns in large data sets, that translate in newfound knowledge. Data mining borrows techniques from multiple fields, such as artificial intelligence, machine learning, statistics, and database systems [CEF$^+$12]. Its ultimate goal is to combine all those techniques and transform large and (apparently) meaningless sets of data into understandable and useful information. Thus, data mining was motivated by the perspective of harnessing the abundance of data, that characterizes today's information systems, to produce meaningful knowledge.

Because of their large quantities of input data, data mining tasks are usually totally, or at least partially, automated. As such, there are several algorithms for these tasks and tools that implements such algorithms, as presented in Section 2.3.1 and Section 2.3.3, respectively.

We can divide data mining into main types: descriptive data mining and predictive data mining [FPSS96]. Descriptive data mining is focused on finding the underlying structure of a given set of data. Instead of predicting future values, it concerns the intrinsic structure, relations and interconnectedness of the data being analysed, presenting its interesting characteristics without having

[8]The 1000 Genomes Project, started back in 2008, is an international effort to establish the most comprehensive catalogue to date of human genetic variations.

any predefined target. On the other hand, predictive data mining is used to predict explicit values, based on patterns determined from the dataset. With predictive data mining we try to build models using known data and use those models as a base to predict future behaviour.

As we're seeing, data mining does not represent a single type problem. In fact there are several different types of problems that can be addressed by data mining techniques. Each of these problems may require a different data mining method. A brief review of the most common methods is given below.

Classification is a method that tries to generalize the already known structure, so that it applies to new datasets. In other words, with classification we try to learn a function that is capable of mapping our data into predefined classes.

Regression tries to learn a function that models relationships between variables in the dataset. That function can latter be used to find real value predictions of future behaviour of the same or similar datasets.

Clustering consists in identifying a finite set of categories or clusters of similar values, to describe the dataset. As such, it is used without prior knowledge about data structure.

Summarization provides a more compact representation of a subset of data, in a way that the summarized data retains the central points of the original data. This can be accomplished in several different ways, like using report generation or multivariate visualization techniques.

Dependency modelling finds a model which describes relationships between variables, revealing their dependencies.

Change and deviation detection tries to discover the most significant changes in the data, when compared with previously measured data. This method is useful to find interesting data variations or data errors.

2.3.1 Data Mining Algorithms

In this project we will be concerned with the classification side of data mining. Below, we will review some algorithms that can be used in classification problems.

Decision Trees

"A decision tree is a flowchart-like tree structure, where each internal node (nonleaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (or terminal node) holds a class label" [HKP06, p. 291], as seen in Figure 2.3. Decision tree learning algorithms use a decision tree as a predictive model, that maps observed data about an individual to conclusions about the expected value for that individual. From a classification problem standpoint, this means means creating a decision tree structure that is able to predict the class of an individual based on its attributes.

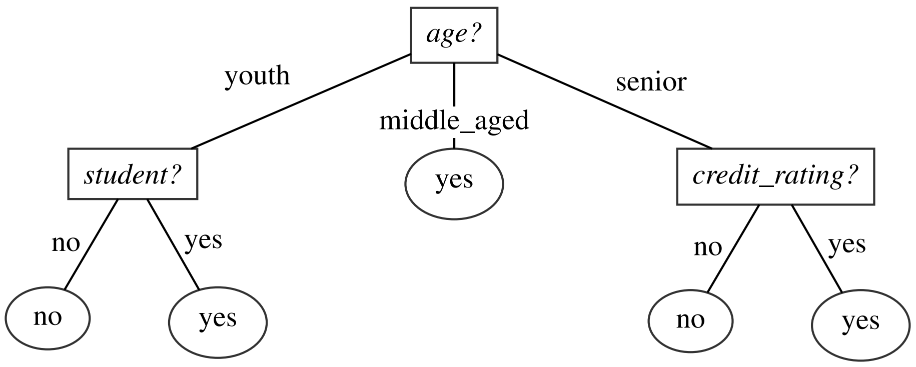


Figure 2.3: Example of a decision tree for the concept *buys_computer*, indicating whether a customer is likely to purchase a computer. Each internal (nonleaf) node represents a test on an attribute. Each leaf node represents a class (either *buys_computer* = *yes* or *buys_computer* = *no*) [HKP06, p. 291].

Decision tree classifiers are among the most popular used nowadays. They are able to handle high dimensional data and their learning and classification steps are simple and fast. Other than that, the construction of decision tree classifiers does not not require domain knowledge or parameter setting. These classifiers usually have a good level of accuracy, though it may vary with the type of data available.

That are several algorithms that use the decision tree principle, most notably *ID3, C4.5* and *CART*. As the problem of learning an optimal decision tree is NP-complete, most algorithms for decision tree construction are based in some sort of heuristics (the aforementioned algorithms adopt a greedy approach).

Random Forest

Random forest [Bre01] is an ensemble approach for classification and regression problems. Ensembles use a divide-and-conquer approach to classification problems, in order to increase performance. The base principle behind ensembles is simple: a group of “weak” learners can come together to become a “strong” learner, where their individual shortcomings are amortized by their combined results. As such, random forest algorithms use several individual decision trees, in order to mitigate the problems with variance and bias in single decision trees.

Each individual tree in the forest in trained used a random subset of the original dataset, where the subsets distribution is the same across the forest. Then, at each node we choose the predictor variable that provides the best split (as per an objective function) and use it to do a binary split at that node. This continues until all the trees are grown to the largest extent possible, as no pruning is made.

Random forest is a fairly fast method and is able to deal with unbalanced and missing data. It as few parameters to tune and can be easily and effectively used with default parameters. As a limitation, when used for regression random forests cannot predict beyond the range in the training data, and that they may overfit datasets that are particularly noisy.

Support Vector Machines

Support vector machines are a set of related supervised learning methods used for pattern recognition, that can be applied to classification and regression problems [CV95]. SVMs typically fall under a simple premise, the spacial division of classes. As seen in Figure 2.4, we can define an infinite number of possible separating hyperplanes or "decision boundaries" (in this case straight lines). The objective of an SVM is to find the optimal hyperplane or set of hyperplanes to separate the classes. To classify a new set of data each value is represented in the space and classified according to the side of the hyperplane in which it falls.

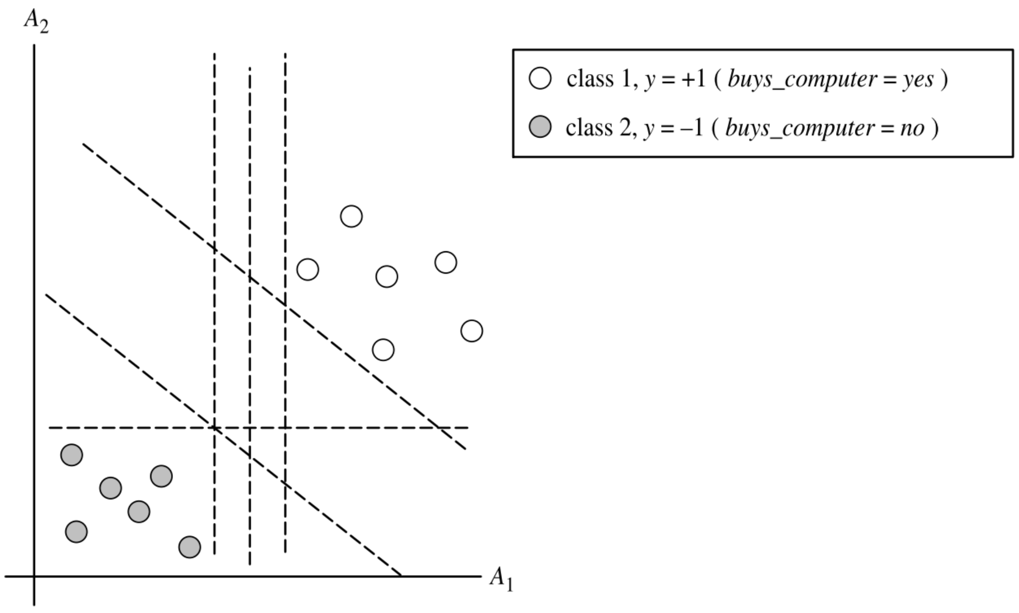

A_2
A_1
○ class 1, $y = +1$ (*buys_computer* = *yes*)
● class 2, $y = -1$ (*buys_computer* = *no*)

Figure 2.4: Example of linearly separable training data in a two dimensional space. There are an infinite number of (possible) separating hyperplanes or "decision boundaries" [HKP06, p. 338].

SVMs are widely used nowadays, and have been successfully applied in areas such handwritten digit recognition, object recognition, and speaker identification, as well as benchmark time-series prediction tests [HKP06]. They are typically regarded. They are highly accurate, due in part to their ability to model complex nonlinear decision boundaries.

However, even the fastest SVMs can be extremely slow in both the training and testing steps. It is also directly limited to two-class tasks, requiring the use of other algorithms to reduce multi-class tasks to binary ones.

K-NN

K-NN is a non-parametric method for classification and regression problems. First described in the early 1950s [HKP06], it's one of the simplest machine learning algorithms and is widely used in the area of pattern recognition.

Nearest neighbor classifiers are based on the premise of learning by analogy, in other words, by comparing a given test tuple with similar training tuples. This classification of tuples as “similar” is typically based on simple rules, like the *Euclidian distance* between tuples.

In k-NN, each training tuple is described by n attributes and is represented as a point in an n-dimensional space. For classification problems, the test tuple is represented in this space, along with the training tuples. The unknown tuple is then assigned to the most common class among its k nearest neighbors.

K-NN as several advantages when compared to other classification algorithms. For one it has a very simple implementation. It's also very robust in terms of search space (the classes don't have to be linearly separable) and has few parameters, making it easy to tune. On the other side, it's an expensive method in terms of testing, as we need to calculate the distance between a testing tuple and every other known tuple. Furthermore, it's sensitive to noisy or irrelevant attributes and unbalanced datasets.

Inductive Logic Programming

TODO

2.3.2 Model Evaluation Procedures and Measures

Creating a data model using a suitable classification algorithms is not the last step in the data mining process. At this point our model works well with the original training data, but we need to verify its power to generalize to other sets of data. Without this evaluation process, our models may be susceptible to problems like overfitting, in which the model wrongly describes a random error or data noise as a significant pattern [HKP06]. As such, we need methods that allow us to test our models before deployment, as well as standard measures, to determine their quality.

Evaluation Procedures

As stated, evaluation procedures are essential to verify the generalization capabilities of a given model. These procedures typically consist of dividing the test dataset in two or more subsets, using some of them for training and the others for testing. We will present a brief overview of some of these procedures below.

Hold-Out. The *hold-out* method reserves a part of the dataset for training purposes and uses the remaining data for testing [WFH05]. Typically we separate one third of the original dataset for testing, using the other two thirds for training.

However, this arbitrary division of the dataset might be problematic if the subsets aren't representative of the population. For example, if the test subset is missing a class our results might be erratic. One way to attenuate this problem is to use stratification. Using stratification of the dataset we ensure that both subsets are representative, with approximately equal proportions for each class. We can lessen error rates even further and make the *hold-out* estimate more reliable using the *repeated hold-out* method. This is an iterative method, where in each iteration a subset is randomly selected to use as training (possibly using set stratification), using the remaining subset for testing. After all the iterations, the error rates of each one are averaged to yield an overall error rate.

Cross-Validation. *Repeated hold-out* methods pose a problem: the different test subsets will eventually overlap. This may cause that some examples never appear in the training subsets. The overlapping problem can be solved using the *cross-validation* procedure, also called *k-fold cross-validation*. This method consists in splitting the original dataset into *k* subsets, using each subset in turn for testing and the remainder for training.

The standard method for evaluation is *10-fold cross validation*, where the dataset is divided into ten subsets. The subsets are typically stratified to reduce result variance. In each iteration one of the ten folds is picked as test and the other nine are used for training. Sometimes, to further reduce variance, *repeated stratified cross-validation* is used, repeating normal *10-fold cross-validation* ten times, then averaging the results.

Leave-One-Out. *Leave-one-out* method is a form of *cross-validation*, taken to extreme lengths. In this particular method, the original dataset is divided into *n* folds, where *n* is the number is the number of individual training instances. It has some benefits, like allowing for a better use of the dataset and involving no random set sampling. However, the sheer number of folds and tests makes it a very computationally expensive method. Another disadvantage is that no stratification is possible, as there is always only one instance in the test subset.

Common Measures

Applying evaluation procedures is not sufficient by itself. In order to appraise the quality of our models, we need to use standard quality measures, that give meaning to the obtained results. Below we will review some common measures, used in the context of classification problems.

Precision. Precision determines the fraction of positive cases that are correctly identified as such. Low precision values mean that the model identifies many cases as positive when they are in reality false positives. Precision is calculated as shown in Equation 2.1.

$$Precision = \frac{true\ positives}{true\ positives + false\ positives} \tag{2.1}$$

Recall. Recall represents the portion of actual positive results in the dataset that were identified as such. From a statistical standpoint, it can be viewed as the average probability of find all the positive cases in the dataset. The value is computed using Equation 2.2.

$$Recall = \frac{true\ positives}{true\ positives + false\ negatives} \tag{2.2}$$

Accuracy. Accuracy represents the percentage of predictions that are in fact correct. This measure relates to the ability of our model to correctly predict the class of new data. It can be calculated with Equation 2.3. For a given problem, better accuracy might not always mean that a model is better than another. As such, other measures like precision and recall are also taken into account.

$$Accuracy = \frac{true\ positives + true\ negatives}{true\ positives + false\ positives + true\ negatives + false\ negatives} \tag{2.3}$$

F-Measure. F-measure is the harmonic mean of precision and recall. The reason behind the usage of an harmonic mean instead of an arithmetic mean is that the first is more intuitive, when computing a mean of ratios [Sas07]. F-measure is calculated as shown in Equation 2.4. It is particularly useful to differentiate between cases where one of the variables (precision or recall) has a very high value and the other has a very low value. For example, if in a particular situation we have a precision of 1.0 and a recall of 0.2, the arithmetic mean would be 0.6. However, a system with a recall as low as 0.2 might not be very useful. On the other hand, in the same situation the harmonic mean would be $0.333(3)$, giving a much more realistic measure of the quality of the model.

$$Fmeasure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.4}$$

AUC. Sometimes in order to evaluate the quality of a certain model we might plot a ROC (or *receiver operating characteristic*) curve. ROC curves depict the performance of a classifier without regard to class distribution or error costs. The curve is build by plotting false positive rates in the *x* axis and true positive rates (recalls) in the *y* axis (Figure 2.5).

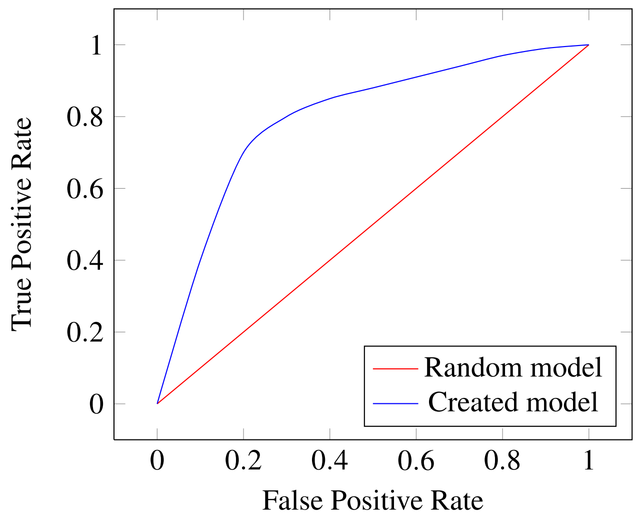


Figure 2.5: Example of ROC curve.

However, the curve can be hard to interpret at times. In this situation we can use the AUC measure (or *area under the curve*), in order to condensate the curve's information in a single value. The AUC of a random classification system is 0.5. This means that any model that as an AUC of 0.5 is not able to distinguish between two groups. As the AUC increases, the model's quality also increases and at a value of AUC equal to 1.0 the model is able to perfectly separate both groups.

2.3.3 Data Mining Tools

Except in rare cases of very specific problems, it typically makes no sense for someone to implement any data mining algorithm that they might need. In fact, today we have lots of data mining tools (many of which are free), that already implement many of those algorithms. These tools are usually customizable, making it easy to adapt them to most problems. Below we'll briefly review some of the most popular data mining tools, that apply to the specific needs of this thesis.

RapidMiner

RapidMiner[9] is a complete solution for data mining problems. It's available as a standalone GUI based application, as seen in Figure 2.6. It is a commercial application, although its core and earlier versions are distributed under an open source license and it offers a free version, beyond its multiple paid versions. Being one of the most popular data mining tools used today, its applications span several domains, including education, training, industrial and personal applications, among others. Its functionality can also be easily extended through the use of plugins[10], reflecting in an increased value for this tool. One such example in the area of bioinformatics is the integration plugin between RapidMiner and the Taverna[11] open source workflow management system [JEF11].

Weka

Weka[12] is an open source tool that collects several machine learning algorithms and allows its user to easily apply those algorithms to data mining tasks [HNF⁺09]. Created at the University of Waikato, New Zeland in 1997 (the current version was completely rewritten in 1997, despite the first iteration of the tool being developed as early as 1993), it's still in active development to date. Weka supports several common data mining tasks, like data preprocessing, classification, clustering, regression and data visualization. It's core libraries are written in Java and allow for an easy integration of its data mining algorithms in pre existing code and applications. Other than that, Weka can be used directly through a command line/terminal or through one of its multiple GUI's (Figure 2.7). Its simple API and well structure architecture allow it to be easily extended by users, should they need new functionalities.

[9] http://www.rapidminer.com/

[10] Plugin is a software module that adds new functionality to an existing software application. Plugins are typically dependent on the platform they extend and can't be used as standalone tools.

[11] http://www.taverna.org.uk/

[12] http://www.cs.waikato.ac.nz/ml/weka/

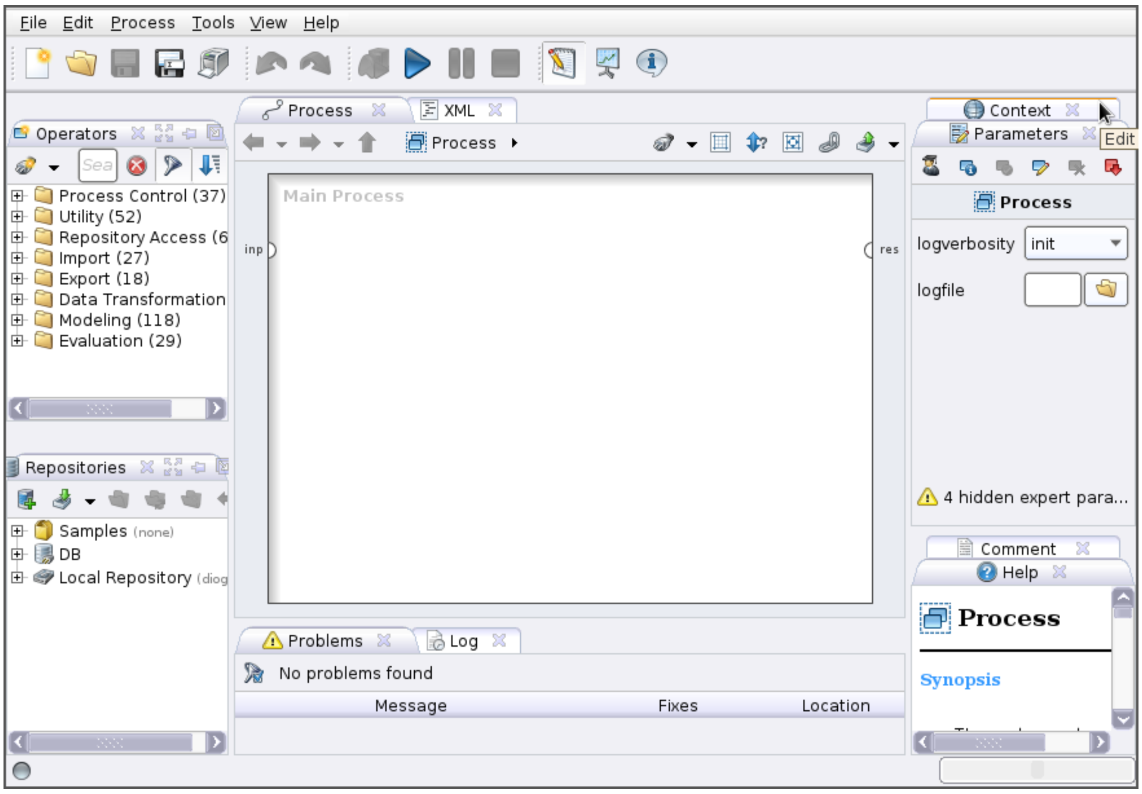
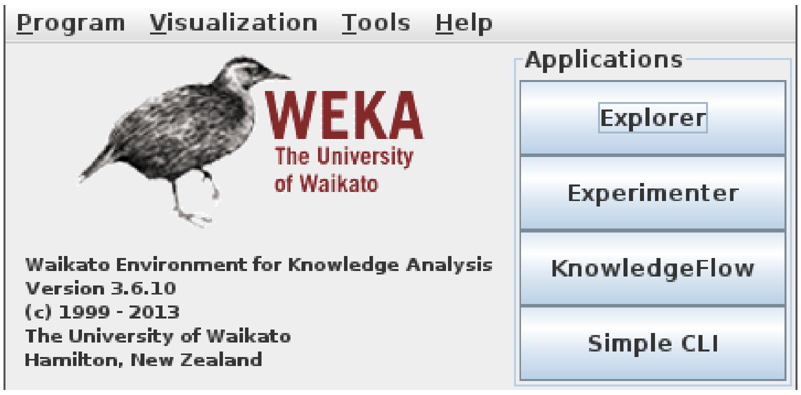


Figure 2.6: RapidMiner user interface.



Waikato Environment for Knowledge Analysis
Version 3.6.10
(c) 1999 - 2013
The University of Waikato
Hamilton, New Zealand

Figure 2.7: Weka interface selection.

R Language

R[13] is a free programming language and software environment for statistical computing and graphics generation. Originally developed by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand in 1993 [Iha98], it's still under active development. R is typically used by statisticians and data miners, either for direct data analysis or for developing new statistical

[13] http://www.r-project.org/

software [FA05].

R is an implementation of the S programming language[14], borrowing some characteristics from the Scheme programming language. It's core is written in a combination of C, Fortran and R itself. It is possible directly manipulate R objects in languages like C, C++ and Java. R can be used directly through the command line or through several third party graphical user interfaces like Deducer[15]. There are also R wrappers for several scripting languages.

R provides several different statistical and graphical techniques, including linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering, among others. It can also be used to produce publication-quality static graphics. Tools like Sweave [Lei02] allow users to embed R code in $\LaTeX$ documents, for complete data analysis.

Bioconductor Package. Bioconductor is a free and open source set of tools for genomic data analysis, in the context of molecular biology [Lei02]. It is primarily based on R. It is under active development, with two stable releases each year. Counting with more than seven hundred different packages, it's the most comprehensive set of genomic data analysis tools available for the R programming language. It also provides a set of tools to read and manipulate several of the most common file formats used in molecular biology oriented applications, including FASTA, FASTQ, BAM and GFF.

2.4 Chapter Conclusions

- in this chapter we talked about...
- we don't know where classification algorithms will be sufficient or if we'll need ILP...
- what else?

[14]S is an object oriented statistical programming language, appearing in 1976 at Bell Laboratories.
[15]`http://www.deducer.org/pmwiki/index.php`

Chapter 3

Work Plan

This chapter describes the general work plan for the thesis, in terms of activities and their respective timings. Furthermore, we will discuss the datasets that will be used in the work, their characteristics and provenience. Lastly, we will address the subject of work evaluation and validation, explaining how it will be conducted, both during and at the end of the project.

3.1 Planning

Aside the preparation phase (already completed), the time available for the thesis will span from February to July, 2014, roughly totalling twenty weeks. It is essential to define a top level schedule beforehand, to ensure that sufficient time will be allotted for every phase of the project and that the timings of those phases are feasible. As such, Figure 3.1 represents the division of the six main phases of the project, during the available period of twenty weeks. Although each phase comprises several smaller tasks, we believe that such a small granularity planning is not needed in this phase and will be defined during the work's execution, as needed. Each main phase of the project is composed as follows:

Information system development comprises the design and development of the data management component of the project and will take roughly six weeks. Despite not being the most critical component, making it the first in the development timeline facilitates later data intensive phases like the transcriptome assembly and, at the same time, allows extensive testing and performance evaluation through usage. A substantial time allotted for this phase will be spent tackling the performance aspects of implementing a system for such large quantities of data, both in terms of database size and response times.

Assembly pipeline development consists of the construction of the tool pipeline responsible for assembling the transcriptomes. At first, several of the already mentioned tools will be studied and tested against small datasets, in an effort to ascertain which are best suited to our

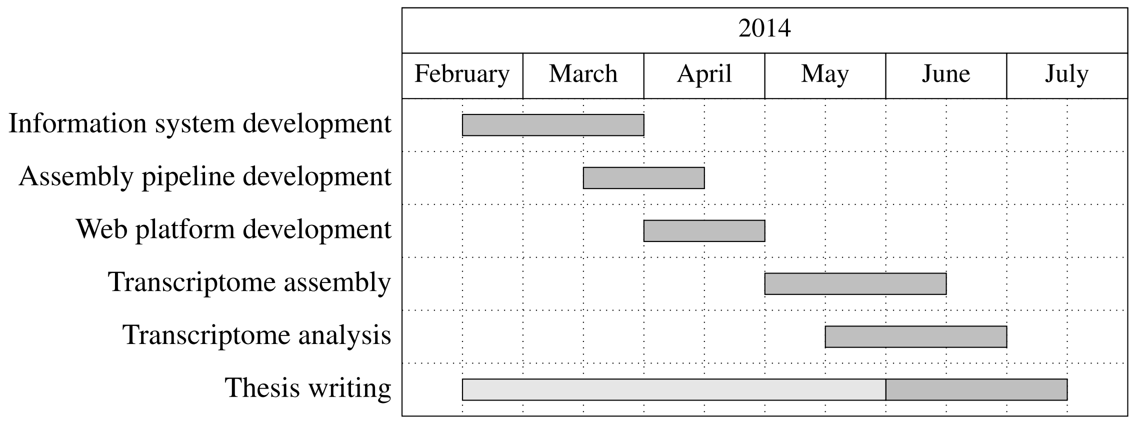


Figure 3.1: Work distribution planning.

particular problem. As the tools are selected, the pipeline itself will take shape, integrating the tools in sequence. Any actual development effort in this phase will be in the form of simple data format conversion scripts, since we will use existing assembly tools. Because of this, the estimated duration of this phase is only one month or four weeks, despite its critical importance to the project.

Web platform development will take four weeks and comprises the design and implementation of the system's web front-end. The web platform will integrate the information and assembly systems, providing a user friendly interface for genetic data storage, management and assembly. From a technical standpoint it's a fairly trivial system, which explains why only four weeks were reserved to this phase.

Transcriptome assembly is the first phase after concluding the development of the main components of the system. In this phase the developed system will be used to produce the assembled transcriptome, employing the given production dataset. This phase will take about six weeks, despite no implementation work taking place (saving some small system tweaks). This is because genome and, in this case, transcriptome assembly are resource and time intensive processes that can take several days, making the extra time necessary for both new and repeat experiments.

Transcriptome analysis will consist in the usage of several data mining tools in order to try to explain the already mentioned RNA transcription mechanisms. This phase is expected to last about six weeks. Although not as resource demanding as the transcriptome assembly phase, this will require choosing and testing a new set of tools and possibly integrate them with the developed system.

Thesis writing is the last phase of the project, with an expected six weeks allocated time. These last six weeks refer to a period to collect and report the obtained results and to make the final reviews to the produced content. However, it is expected that the thesis report will be worked on continuously from the start of the project, in parallel with the other project phases.

3.2 Experimental Data

During this project there will be essentially three types of datasets used: read data, genome data and test data. Each type of dataset has its own nature, origin and purpose. We will use real genetic data from a fly species called *Drosophila melanogaster*, commonly known as fruit fly, which can be seen in Figure 3.2. It is one of the most frequently used organisms to provide its genetic data for these kind of studies and work.

The read data will be made available during the project through IBMC. As stated, this data consists of several short sequencing reads of the *Drosophila melanogaster* genome. It is this dataset that will be ultimately used for assembly and posterior data mining analysis. It should be noted that this will be real data, experimentally obtained in a laboratory for this project.

The genome data will consist of already assembled *Drosophila melanogaster* genome(s), that will be used as a reference in our own assembly process. This data will be obtained through FlyBase[1]. FlyBase is an online and publicly accessible database of *Drosophila* genes and genomes. This database allows its data to be downloaded in several formats, that can be either directly used in our assembly pipeline, or be automatically converted by one of the created conversion tools.

Lastly, we will use some small scale datasets for the test and calibration of the assembly pipeline. Such datasets are usually shipped with the assembly tools themselves. If needed, a combination of the two previous datasets can be used to produce small scale test data for this purpose.

3.3 Thesis Work Evaluation

- rephrase a bit and reference the data mining evaluation section...

In the second part of the project, that is the transcriptome assembly and analysis phases, results evaluation and validation is essential. Even more so when such results are typically evaluated from a molecular biology standpoint and therefore are out of the scope of knowledge of the thesis itself. In such cases we will have two evaluation methods at our disposal.

The first method is based on relevant metrics for the problems at hand, from both the transcriptome assembly and data mining parts. Such metrics are usually produced by the tools themselves. As for these metrics there is usually a well defined range of expected results, which makes them a very important method of early result evaluation, in the sense that they can be interpreted without a profound knowledge about molecular biology.

[1] `www.flybase.org`

Figure 3.2: Specimen of *Drosophila melanogaster*, viewed from above[2].

The second method available is the evaluation by IMBC's technicians, that will assist us whenever expert biology knowledge is required. This will ultimately be the method that will provide a real measure of the success of the project.

Furthermore, IMBC's technicians will be essential during the entirety of the project. They will help steer the project into its intended direction, giving some insight about their expectations towards the system. Project phases like the implementation of the information system or the transcriptome analysis will be driven by their feedback, giving us a sense about what should be done. Lastly, they will also be present throughout the project to help with any biology related questions that arise.

[2]Image taken from `http://pt.wikipedia.org/wiki/Drosophila_melanogaster`.

Chapter 4

Conclusions

References

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[CEF$^+$12] Soumen Chakrabarti, Martin Ester, Usama Fayyad, Johannes Gehrke, Jiawei Han, Shinichi Morishita, Gregory Piatetsky-Shapiro, and Wei Wang. Data mining curriculum: a proposal, version 1.0. Available at www.kdd.org/curriculum/CURMay06.pdf, last access on February 2014, April 2012.

[CFG$^+$10] Peter J a Cock, Christopher J Fields, Naohisa Goto, Michael L Heuer, and Peter M Rice. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic acids research*, 38(6):1767–71, April 2010.

[CV95] C Cortes and V Vapnik. Support-vector networks. *Machine learning*, 297:273–297, 1995.

[EBS$^+$11] Dent Earl, Keith Bradnam, John St John, Aaron Darling, Dawei Lin, Joseph Fass, Hung On Ken Yu, Vince Buffalo, Daniel R Zerbino, Mark Diekhans, Ngan Nguyen, Pramila Nuwantha Ariyaratne, Wing-Kin Sung, Zemin Ning, Matthias Haimel, Jared T Simpson, Nuno a Fonseca, İnanç Birol, T Roderick Docking, Isaac Y Ho, Daniel S Rokhsar, Rayan Chikhi, Dominique Lavenier, Guillaume Chapuis, Delphine Naquin, Nicolas Maillet, Michael C Schatz, David R Kelley, Adam M Phillippy, Sergey Koren, Shiaw-Pyng Yang, Wei Wu, Wen-Chi Chou, Anuj Srivastava, Timothy I Shaw, J Graham Ruby, Peter Skewes-Cox, Miguel Betegon, Michelle T Dimon, Victor Solovyev, Igor Seledtsov, Petr Kosarev, Denis Vorobyev, Ricardo Ramirez-Gonzalez, Richard Leggett, Dan MacLean, Fangfang Xia, Ruibang Luo, Zhenyu Li, Yinlong Xie, Binghang Liu, Sante Gnerre, Iain MacCallum, Dariusz Przybylski, Filipe J Ribeiro, Shuangye Yin, Ted Sharpe, Giles Hall, Paul J Kersey, Richard Durbin, Shaun D Jackman, Jarrod a Chapman, Xiaoqiu Huang, Joseph L DeRisi, Mario Caccamo, Yingrui Li, David B Jaffe, Richard E Green, David Haussler, Ian Korf, Benedict Paten, John St. John, Pramila Nuwantha, *Nuno A. Fonseca*, and Ynanc Birol. Assemblathon 1: A competitive assessment of *de novo* short read assembly methods. *Genome Research*, 21(12):2224–2241, December 2011.

[FA05] John Fox and Robert Andersen. Using the R statistical computing environment to teach social statistics courses. *Department of Sociology, McMaster University*, (January), 2005.

[FPSS96] UM Fayyad, G Piatetsky-Shapiro, and P Smyth. Knowledge Discovery and Data Mining: Towards a Unifying Framework. *KDD*, 1996.

[GEN] GENIE. Gene expression and regulation. Available at http://www2.le.ac.uk/departments/genetics/vgec/schoolscolleges/topics/geneexpression-regulation, last access on February 2014.

[HKP06] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques*. Morgan kaufmann, 2006.

[HNF+09] Mark Hall, Hazeltine National, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The WEKA Data Mining Software : An Update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[Iha98] Ross Ihaka. R: Past and future history. *COMPUTING SCIENCE AND STATISTICS*, 1998.

[JEF11] Simon Jupp, James Eales, and Simon Fischer. Combining RapidMiner operators with bioinformatics services–a powerful combination. In *RapidMiner Community Meeting and Conference, (RCOMM)*, 2011.

[Lab] Abecasis Lab. Sam - genome analysis wiki. Available at `http://genome.sph.umich.edu/wiki/SAM`, last access on February 2014.

[Lei02] Friedrich Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In Wolfgang Härdle and Bernd Rönz, editors, *Compstat 2002 — Proceedings in Computational Statistics*, pages 575–580. Physica Verlag, Heidelberg, 2002. ISBN 3-7908-1517-9.

[LHW+09] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, and Richard Durbin. The Sequence Alignment/Map format and SAMtools. *Bioinformatics (Oxford, England)*, 25(16):2078–9, August 2009.

[MW11] Jeffrey a Martin and Zhong Wang. Next-generation transcriptome assembly. *Nature reviews. Genetics*, 12(10):671–82, October 2011.

[NCB] NCBI. Web blast page options. Available at `https://www.ncbi.nlm.nih.gov/BLAST/blastcgihelp.shtml`, last access on February 2014.

[PASH03] Lajos Pusztai, Mark Ayers, James Stec, and Gabriel N Hortobágyi. Clinical Application of cDNA Microarrays in Oncology. *The Oncologist*, 8(3):252–258, January 2003.

[RF09] Jorge S. Reis-Filho. Next-generation sequencing. *Breast cancer research : BCR*, 11 Suppl 3:S12, January 2009.

[San11] Sanger Institute. Gff (general feature format). Available at `http://www.sanger.ac.uk/resources/software/gff/spec.html`, last access on February 2014, 2011.

[Sas07] Yutaka Sasaki. The truth of the F-measure. *Teach Tutor mater*, pages 1–5, 2007.

[Smi13] Smith, Steven and Browning, Brian. Introduction to variant call format. Available at `http://faculty.washington.edu/browning/beagle/intro-to-vcf.html`, last access on February 2014, 2013.

[WFH05] Ian H IH Witten, Eibe Frank, and Mark A Hall. *Data Mining: Practical Machine Learning Tools and Techniques: Practical Machine Learning Tools and Techniques*. Elsevier, second edition, 2005.

[WL09] Brian T Wilhelm and Josette-Renée Landry. RNA-Seq-quantitative measurement of expression through massively parallel RNA-sequencing. *Methods (San Diego, Calif.)*, 48(3):249–57, July 2009.

[Wol13] Jochen B W Wolf. Principles of transcriptome analysis and gene expression quantification: an RNA-seq tutorial. *Molecular ecology resources*, 13(4):559–72, July 2013.