Master Degree Project

UNIVERSITY OF SKÖVDE

NOVEL APPROACH TO STORAGE AND STORTING OF NEXT GENERATION SEQUENCING DATA FOR THE PURPOSE OF FUNCTIONAL ANNOTATION TRANSFER

Master Degree Project in Bioinformatics
One year Master 60 ECTS
Autumn term 2011

Tito Candelli

Supervisor: Angelica Lindlöf
Examiner: Zelmina Lubovac

Abstract

The problem of functional annotation of novel sequences has been a significant issue for many laboratories that decided to apply next generation sequencing techniques to less studied species. In particular experiments such as transcriptome analysis heavily suffer from this problem due to the impossibility of ascribing their results in a relevant biological context. Several tools have been proposed to solve this problem through homology annotation transfer. The principle behind this strategy is that homologous genes share common functions in different organisms, and therefore annotations are transferable between these genes.

Commonly, BLAST reports are used to identify a suitable homologous gene in a well annotated species and the annotation is then transferred from the homologue to the novel sequence. Not all homologues, however, possess valid functional annotations. The aim of this project was to devise an algorithm to process BLAST reports and provide a criterion to discriminate between homologues with a biologically informative and uninformative annotation, respectively. In addition, all data obtained from the BLAST report is to be stored in a relational database for ease of consultation and visualization.

In order to test the solidity of the system, we utilized 750 novel sequences obtained through application of next generation sequencing techniques to *Avena sativa* samples. This species particularly suits our needs as it represents the typical target for homology annotation transfer: lack of a reference genome and difficulty in attributing functional annotation. The system was able to perform all the required tasks. Comparisons between best hits as determined by BLAST and best hits as determined by the algorithm showed a significant increase in the biological significance of the results when the algorithm sorting system was applied.

Contents

1 Introduction **3**

1.1 Next Generation Sequencing 3

1.1.1 Technical Overview 3

1.1.2 Assembly of the Reads 5

1.2 Challenges of *de novo* Assembly: Functional Annotation . . . 6

1.3 Homology Search as a Tool to Annotate Novel Sequences . . . 7

1.4 Literature solutions 8

1.4.1 GeneQuiz 8

1.4.2 autoFACT 10

1.4.3 BLAST2GO 11

1.5 Relational Databases 12

2 Aim of the Project **14**

2.1 Aim of the Project 14

2.2 Objectives 15

2.2.1 BLAST Parsing 15

2.2.2 Informativeness Score Calculation 15

2.2.3 Relational Database 16

3 Results and Implementation **17**

3.1 The Relational Schema 17

3.1.1 Tables 17

3.1.2 Keys and Dependencies 18

3.1.3 Referential Integrity 20

3.2 Algorithm Structure 20

3.2.1 BLAST Output Parsing 20

3.2.2 Informativeness Score 22

3.2.3 Interface with MySQL 25

3.3 Testing 28

3.3.1 Origin of the Data 28

3.3.2 BLAST Results 28

3.4 Results and Validation 29

3.4.1 BLAST Comparison 29

3.4.2 Score Distribution 35

4 Discussion and Conclusions **38**

4.1 Discussion 38

4.2 Conclusion 41

5 Methods **43**

Bibliography **45**

Part 1

Introduction

1.1 Next Generation Sequencing

In these last years, new sequencing methods have progressed at an impressive rate. The most drastic improvement that next generation sequencing (NGS) techniques brought over Sanger sequencing, which has dominated the scene for the past 40 years as the only available method, consists of the huge amount of information that can be processed in a short amount of time [11, 12, 2]. This feature, combined with the ability to quantify the expression levels of the input sample, paved the way for a new approach to transcriptome analysis.

1.1.1 Technical Overview

NGS platforms abandon the notion of serial sequencing by electrophoretic separation, bringing forth the idea of massive parallel sequencing instead. This new approach focuses on the simultaneous acquisition of sequence data from an extremely large number of relatively small fragments (reads), which are then assembled into contigs and ultimately yield the complete starting sequence.

Currently, the world of massive parallel sequencing techniques is dominated by three platforms, all developed towards the end of the 1990's and

commercialised around 2005: Roche 454 Genome Sequencer, the Illumina Genome Analyser, and the Life Technologies SOLiD System [18]. Each one of these platforms, despite sharing substantial similarity in the approach to library preparation, template amplification and sequencing by chain extension, is based on different technologies and principles. The next sections will focus on giving an overview of the most significant steps in the next generation sequencing work-flow.

Step 1: DNA Extraction and Library Preparation

The first step in every sequencing experiment consists of the obtainment of DNA material [12]. A deluge of different protocols, all compatible with next generation sequencing analysis, exists nowadays for this specific purpose, each aiming to answer specific biological questions. Some examples of different techniques employed to obtain DNA for sequencing purposes include: total genomic extraction, used for genome sequencing or for genome-wide allelic variant screening; ChIP protocols, utilized to isolate the fraction of genomic DNA interacting with a specific protein; RNA extraction followed by a reverse-transcriptase reaction, employed for the study of different RNA species depending on the purification method used (poly(A) purification for mRNAs, size purification for miRNAs, etc.).

Ultimately, the outcome of this step is represented by a number of randomly generated fragments of appropriate length for the sequencing platform in use.

Step 2: DNA Amplification

In line of principle, every one of the fragments isolated during library preparation should give rise to a read, a sequence of known nucleotide composition representing a DNA fragment. Due to technical limitations in the detection process, however, an intermediate amplification step must be performed before the actual sequencing can take place [12]. This step ensures that each

fragment is sufficiently represented when it comes to the actual sequencing process, and that the signal emission is strong enough to be captured by the detector.

Different approaches exist for this amplification step. Two in particular are adopted by the major sequencing platforms mentioned above: emulsion PCR, used by both Roche 454 and SOLiD systems, and bridge PCR, employed by Illumina Genome Analyzer [18]. The result of this step consists of a clonal population of each fragment, which can be separated from the other populations either by being bound to a bead or by being in different physical locations on a glass slide.

Step 3: Sequencing

The act of sequencing represents the gathering of information concerning the nucleotide sequence of a fragment. The Illumina Genome Analyser and Roche 454 platforms are both based on the same principle of "sequencing by synthesis"; the SOLiD platform, on the contrary, uses a totally different method based on "sequencing by ligation" [18].

As the name suggests, technologies based on "sequencing by synthesis" acquire information about the sequence of the fragment by detecting specific signals emitted when a strand complementary to the fragment of interest is being synthesised. Contrarily, the technology employed by SOLiD sequencing does not rely on DNA synthesis, but employs a series of ligation reactions to help the deconvolution of the nucleotide sequence.

1.1.2 Assembly of the Reads

Regardless of the platform used to perform the sequencing, the output produced remains the same: a list of "reads", short sequences representing fragments of the starting material. Assembly refers to the process of piecing together overlapping reads into longer stretches called contigs. Depending on the type of experiment performed, contigs can represent different things,

such as mRNA population in case of RNA-seq or genomic sequences in case of whole genome sequencing or ChIP-seq. Regardless of the type of experiment, however, contigs provide an accurate representation of the starting material.

Depending on the availability of a reference genome, assemblies can be either mapping assemblies, when the reads are mapped against some reference sequence, or *de novo* assemblies, where the contigs are determined only by virtue of overlapping reads [8, p 655].

1.2 Challenges of *de novo* Assembly: Functional Annotation

As mentioned above, *de novo* assembly is performed when no reference genome is available for the mapping of the reads. This occurrence is becoming more and more frequent as NGS technologies grow cheaper and many laboratories around the world can afford to perform these kind of experiments on less known species. Many elaborate tools are available today for this kind of procedure and, despite the lack of a reference framework on which to assemble the reads, high-quality and accurate contigs can indeed be obtained [13, 6].

Difficulty in contig assembly, however, is not the only problem spawned by the lack of reference sequences. Another purpose of mapping against a reference genome, in fact, is that of acquiring functional information which may already be present for the subject organism. This procedure, however, becomes arduous when the genome is not yet sequenced, or does not have a sufficient level of annotation. Certain kinds of experiments, such as transcriptome analysis, especially the ones performed on less known species, heavily suffer from this problem. The limitedness of functional information obtainable about the sequences, in fact, denies any kind of insight about the biological mechanisms studied. While the standard statistical tools for NGS data analysis, such as differential expression, are still available, the re-

sults lack a biological context in which to be placed and are, therefore, of significantly lower quality.

1.3 Homology Search as a Tool to Annotate Novel Sequences

The most common solution to the problem of a missing genome or limited annotation is homology search [10, 9, 7, 1]. As a result of applying local alignment algorithms such as BLAST [17], the contigs can be compared to various databases of sequences, which originate from model organisms or other species known for their good annotation. This approach has the advantage of allowing the search for homologous sequences in a vast repository of DNA, RNA or protein sequences from numerous different species. This advantage, however, is offset by a well known problem: not every sequence in the database possesses an informative annotation.

The BLAST tool provides a great deal of information regarding each of the similarity hits it scores, such as mismatches, gaps, bitscore, etc. The most important values for each High Scoring Pair (HSP) are e-value and identity percentage. These values represent respectively the statistical significance of the hit and its similarity to the query sequence, and are used by BLAST to sort the results so that the most likely statistically significant homologous sequence is shown first. Similarity hits that would normally be considered to have all the hallmarks of a high quality homology, such as low e-value and high identity percentage, are not, however, guaranteed to have good annotation. This means that criteria which usually define a high quality similarity hit do not have the capability to determine how good the hit is for the purpose of annotation transfer.

This problem is greatly exacerbated when dealing with NGS data; because of the extremely high number of sequences. Manual choice of the best annotation from the top scoring similarity hits is not a viable option and algorithms for the (at least partial) automation of this process become a

necessity.

1.4 Literature solutions

Several solutions to this problems have been proposed in the form of automatic functional annotation tools. The most notable in this category include GeneQuiz [1], autoFACT [9] and BLAST2GO [7], which utilize different approaches to determine annotations for novel sequences.

1.4.1 GeneQuiz

GeneQuiz was the first tool for automatic functional annotation to be proposed to the public [14]. Although not specifically designed for NGS data, but rather as a tool to aid the researcher in the annotation of a moderate number of protein sequences, GeneQuiz was the first to introduce the concept of automated annotation analysis to identify informative annotations [1].

After having performed a BLAST search and applied a variety of sequence analyses tools, GeneQuiz relies on the "uninformative rule" (diagram in Fig. 1.1) to eliminate non-functionally informative annotations and keep those hits which posses a sufficiently informative description. In addition, this tool applies lexical analysis of keywords in the description to assign one of four functional categories to each hit: "energy" for sequences that have to do with aminoacid biosynthesis, energy metabolism, transport etc.; "communication" for sequences that have to do with regulatory functions, cellular process, cell envelopes, etc.; "information" for hits that have to do with replication, transcription, translation etc., and finally the "other" category for all the sequences that do not conform to any of the above.

GeneQuiz represents a very important step in the evolution of automatic functional annotation tools. While this system only works for protein sequences and is outdated and outperformed by newer programs, the idea of lexical analysis to determine the informativeness of a description has been

later used by other tools, as well as being a fundamental part of the algorithm proposed in this report.

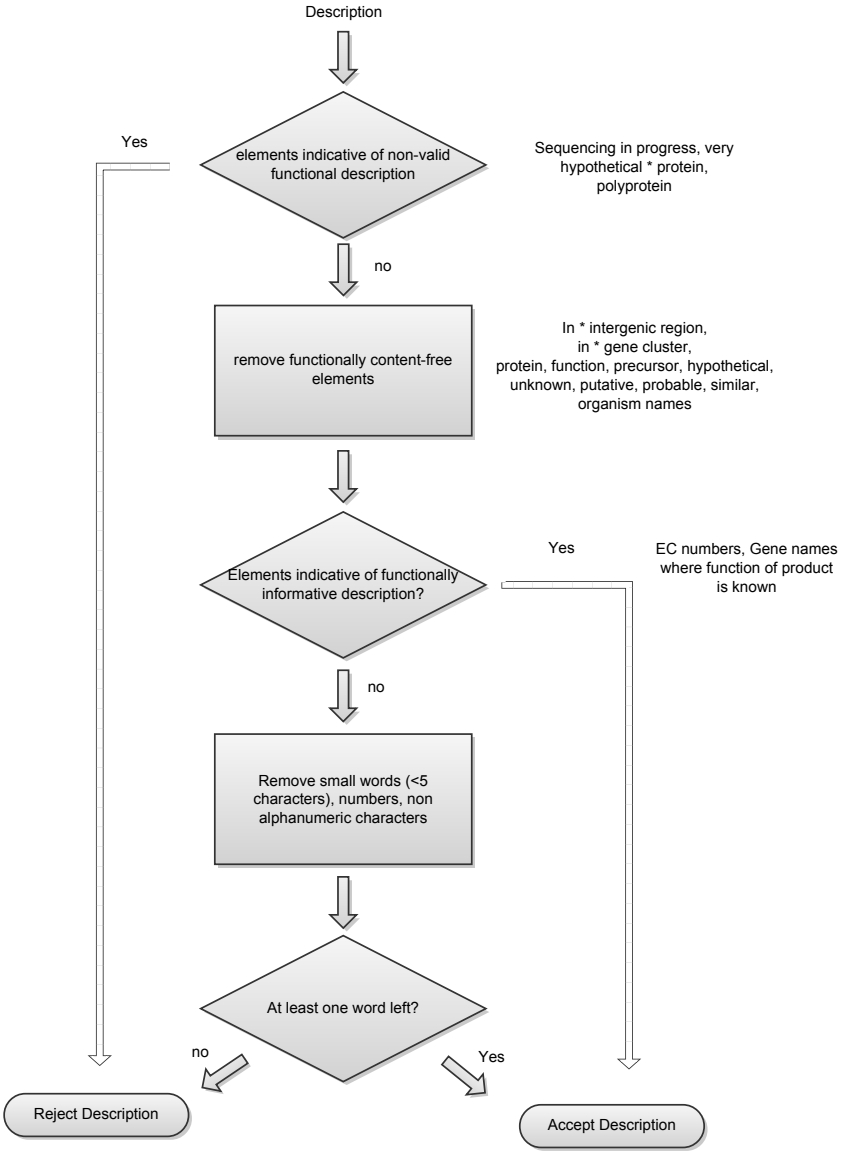

Figure 1.1: Flow chart explaining the "uninformative rule" to discriminate between uninformative and informative annotations, a similar approach is used by autoFACT.

1.4.2 autoFACT

autoFACT [9] is a much more recent tool developed for high throughput data. Conceptually similar to GeneQuiz, autoFACT extends automatic annotation to nucleotide sequences and makes use of different BLAST reports arranged in a hierarchical structure, which are analysed on a priority basis. Reports obtained by comparing the query sequence against databases of highly conserved and well annotated sequences are analysed first, as they are the most likely to give a positive annotation if a match is found. Subsequently, if no match is found, the algorithm proceeds to analyse less and less informative databases until a match is found or all the available reports are analysed. autoFACT uses the same "uninformative rule" used by the GeneQuiz tool to discriminate between informative and non-informative annotations [9].

The hierarchy of database analysis, as well as a diagram of autoFACT inner workings, is shown in Fig 1.2. We can see that the first analysed database is the one containing rRNA data; an extremely well conserved class of sequences that, in case of a positive match, can confer an excellent functional annotation. Subsequently, the algorithm searches for matches in "nr", "KEGG", "uniref" or "COG", according to user specification and nature of the sequence (protein or nucleotide). High scoring similarity hits in these databases are cross-referenced with each other and presence of Gene Ontology annotation is checked, providing further functional information. If still no hit is found, autoFACT looks for possible protein domains ("smart" and "pfam" databases) present in the query sequence and finally, if no positive match is found, the "EST" database, notoriously poor in functional annotations, is queried.

With its methodical parsing of several databases, autoFACT provides a very thorough search for the best annotation. It does, however, have several limitations. In particular, the high quantity of BLAST alignments required can be very computationally intensive and out of reach of those laboratories which do not have access to server clusters or high power distributed computing networks. In addition, use of customized databases to specifically

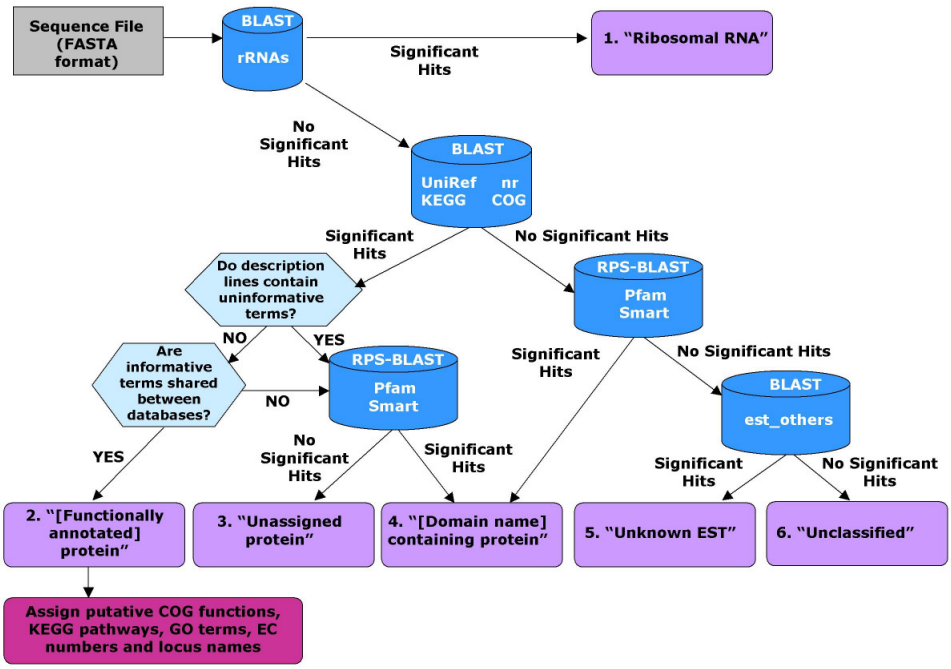

Figure 1.2: A diagram explaining the inner workings of autoFACT. The blue cylinders represent BLAST reports, the purple squares represent annotations. Picture taken from [9].

check for similarity in a particular subset of sequences is not possible.

1.4.3 BLAST2GO

Another tool for automatic functional annotation first proposed in 2008 is BLAST2GO (B2G) [7]. This program, although still based on BLAST output parsing, uses a different approach to determine the annotation of each similarity hit. B2G does not, in fact, require multiple BLAST reports obtained by querying specific databases, but accepts any XML BLAST output. For each query sequence, the tool screens the top hits' description for terms which denote an uninformative annotation such as "hypothetical" or "putative", discarding the hit if any are found. Subsequently, the algorithm uses an NCBI-provided file ("gene2accession") to associate each hit's GI number with the respective gene name, which can then be used to check for Gene

Ontology annotation.

The presence of Gene Ontology annotation [16] for significant similarity hits represents the cornerstone of B2G annotation method; by using a separate algorithm, all the terms associated with a particular gene are evaluated on the basis of the experimental evidence that originally led to their attribution to the hit. This way, only trustworthy terms which are sufficiently supported by hard evidence are included in the final annotation.

BLAST2GO represents an excellent tool for functional annotation. It does, however, suffer from too heavily relying on Gene Ontology. Although usually Gene Ontology provides trustworthy and complete annotations for many hits, it may miss some still valuable information. Some genes, in fact, still have decent functional annotation even without possessing an official Gene Ontology annotation.

1.5 Relational Databases

Despite not being used by any of the aforementioned tools, relational databases provide a fast and efficient way of storing and visualizing data. Given the importance of relational databases for the algorithm proposed in this report, a brief introduction to MySQL (the relational database management system chosen in this work) is provided below.

MySQL is a relational database management system based on the SQL language [3] used for storing the data extracted from the BLAST report. The relational model [4], on which all relational database management systems are based, is derived from first-order predicate logic and provides a declarative method to submit and retrieve data from the system; the user directly states which information is to be inserted or retrieved from the system thanks to a high level query language and lets the system take care of managing the actual storing and retrieval.

The relational model stores information in formally defined tables containing one or more attributes (columns) and zero or more rows (tuples).

Each table is usually associated with a key: one or more attributes which have unique values (that is the value, or combination of values in case of multiple attributes, is unique for each tuple) and define the rest of the attributes of the table. The key is often used to reference other tables, thus minimizing the amount of redundant information stored and optimising memory usage. A relational database's overall configuration of tables, attributes, keys and dependencies is called a schema. A database can be filled with data that complies with the structure and limitations imposed by the schema.

Perhaps the most useful feature of relational databases, however, is represented by the SQL query language. Based on relational algebra, this query language is very intuitive and high level. It allows to easily specify the required data through use of commands that define which table(s) and attribute(s) are to be retrieved, together with a set of conditions which must be true for a tuple to be displayed. It should be noted that the SQL language is capable of extremely complex queries and supports a number of devices to stratify the data on the basis of any attribute a tuple may possess.

Part 2

Aim of the Project

2.1 Aim of the Project

While existing tools are able to provide accurate annotations, they either require a substantial amount of computational power (autoFACT) or rely on external sources such as Gene Ontology to fetch the annotation (BLAST2GO), which restricts the range of well-annotated genes to those which posses Gene Ontology classification.

This Project aims at providing a computationally inexpensive tool to manage, sort and store information derived from BLAST reports. In particular, the algorithm introduces a new criterion, independent from external databases, to sort and rank similarity hits on the basis of the biological significance of their descriptions; providing a new tool to aid the process of functional annotation transfer.

While not providing the same type of "stand-alone" annotation provided by other tools (*e.g.* `Reverse Transcriptase` or `Threonin kinase`), this tool will be able to rank each similarity hit and provide a list of biologically significant words contained in said hit. In addition, by virtue of the structure of the relational database where the data will be stored, sequences that do not meet certain requirements (*e.g.* e-value too high, identity too low, not

enough functional informativeness) can be removed instantaneously without needing to apply external scripts to the data.

2.2 Objectives

The algorithm will be fully implemented in PERL and divided into three parts: 1) reading and parsing a BLAST output to gather the necessary data, 2) process this data and attribute an informativeness score to each hit, 3) interface with MySQL to store the information.

2.2.1 BLAST Parsing

The first step of the program will be to extract relevant data from a BLAST output. This process will store all the attributes of each HSP into PERL objects, so that they may be easily recalled and processed when necessary. In order to obtain this data and recover the description of each similarity hit, modules provided by Bioperl will be used. Bioperl is a suite of object oriented PERL modules that aims to simplify many typical bioinformatics tasks such as sequence retrieval, FASTA database creation and, as in this case, parsing of output files from on-line tools.

In particular, the modules that will be used are: `Bio::SearchIO` to parse an XML BLAST output and store the information inside specific objects, and `Bio::SeqIO` to obtain the query sequences from a fasta file [15].

2.2.2 Informativeness Score Calculation

When the parsing is done and all the data is stored inside PERL objects, the algorithm will proceed with the calculation of the informativeness score. The score is obtained by comparing the description of each similarity hit with a list of biologically significant words. Each match between the annotation of

a sequence and one of these words results in an increase of the informativeness score. In addition to positive matching words, negative ones such as "hypothetical" and "unknown" are screened for and determine a lowering of the score.

2.2.3 Relational Database

All the results included in the BLAST output, plus the informativeness score and additional information extracted by the algorithm itself (such as GI number and accession number), will be represented as attributes of a relational database. The relational database management system of choice in this case is MySQL [1], and the database schema is modelled on the three typical sections of a BLAST report: query, hit and HSP. In order to interface PERL with MySQL, the `DBI` PERL package will be used, which allows to access, both locally and remotely, SQL databases, execute queries and retrieve data.

[1] see www.mysql.com for additional information

Part 3

Results and Implementation

Implementing an algorithm that performs the tasks described in the above sections requires numerous steps of data gathering and processing. In this section, the structure of both the program and database schema will be described in detail, with emphasis on the procedures of testing with real data and validation.

3.1 The Relational Schema

3.1.1 Tables

Because the source of the vast majority of the information that will be stored in the relational database comes from BLAST, we decided to base the schema for the database on the structure of a BLAST result. Indeed, a BLAST result can be divided into a hierarchical structure containing three layers: Query, Hit and HSP. Each BLAST result can have multiple query sequences, which in turn can display several hits and each hit can be matched in different portions of its sequence, giving rise to different HSP. For this reason, we used the natural separation between query, hit and HSP to model our schema. The totality of the data is divided, in fact, into three tables representing these three categories. In addition to providing an easily understandable structure,

Table	Attribute	Description
Query	**queryID**	query sequence identifier
	sequence	actual sequence of the query
Hit	**hitID**	hit identifier
	annotation	description of the hit sequence
	GI	GI number of the hit
	accession	accession number of the hit
	informativeness	total informativeness score of the hit
	matches	positive words that match the description
HSP	**queryID**	query sequence identifier
	hitID	hit identifier
	hspID	rank of the HSP
	dbID	database used by BLAST
	algID	type of BLAST algorithm
	identity	identity percentage
	length	total length of the alignment
	gaps	number of gaps
	qstart	start position in query sequence
	qend	end position in query sequence
	sstart	start position in hit sequence
	send	end position in query sequence
	evalue	significance of the HSP
	bitscore	bitscore of the HSP

Table 3.1: List of all the attributes for each table with respective description. Attributes in bold represent the primary keys for the table.

this subdivision minimizes the amount of memory required by eliminating storage of redundant information, e.g. having the same information stored twice or more.

3.1.2 Keys and Dependencies

The three tables described in the above section contain each a different set of attributes (Table 3.1), these attributes are determined by the key of the table. As described in a previous section, a primary key is an attribute, or set of attributes, which uniquely identifies each tuple in the table.

The first two tables, Query and Hit, possess only one attribute as primary key: the identifier for the sequence. In the case of the Query table, the sequence identifier is represented by the arbitrary name assigned to the contig by the assembly algorithm e.g. `contig_44`. In the case of the Hit table, the identifier is represented by the combination of "gi" and "accession" numbers used by BLAST to identify a sequence, e.g. `gi|195614012|gb|ACG28836.1|`.

The primary key of the HSP table, contrarily, is composed of several attributes:

1. **Primary key of the Query table.** It identifies the query sequence for which the HSP was found, e.g. `contig_44`.

2. **Primary key of the Hit table.** It identifies the hit sequence in which the HSP was found, e.g. `gi|195614012|gb|ACG28836.1|`.

3. **Rank of the HSP.** An integer whose purpose is that of differentiating between HSPs when multiple HSPs are present for the same hit and query sequence, e.g. `2`.

4. **Database identifier.** discriminate between BLAST reports obtained by comparing different databases, e.g. `nr`.

5. **Algorithm Identifier.** It allows to differentiate between multiple BLAST algorithms if used against the same database, e.g. `BLASTX`.

These five attributes uniquely define each HSP even in case of multiple BLAST reports stored. In addition, the fact that the primary key of the Query and Hit table are also part of the key of the HSP table allows the merging of tables when the need arises (typically when a query is submitted), enabling a more complete visualization of the desired data without having to store redundant information.

Due to this particular structure of tables and keys, the schema is considered to be in "Boyce-Codd Normal Form" [5], a type of formal database normalization that ensures no redundant data storage.

3.1.3 Referential Integrity

The key attributes for the Query and Hit table, are also present as part of the key of the HSP table. This leaves the database open to inconsistencies if, for example, a query identifier were to be modified. The HSP table, in fact, would have as key a query identifier that is no longer present in the Query table, leading to several problems for the formulations of queries and visualization of data. By using tools provided by the SQL language, it is possible to configure a net of "referential integrity constrains", so that if a key that is referenced by another table (as the Query and Hit identifier are referenced by the HSP table) is modified, the change spreads to the referencing tables.

This makes the database more versatile and ensures consistency between its tables.

3.2 Algorithm Structure

The algorithm proposed in this thesis aims to achieve three primary objectives: (1) correctly parse BLAST results and store data in specific objects, (2) based on hit descriptions, calculate an informativeness score for each hit, (3) interface with MySQL and store all the information gathered inside a relational database. In this section, the structure of the algorithm and the tools used to achieve the three goals will be explored in detail.

3.2.1 BLAST Output Parsing

The parsing of the BLAST output is fundamental to recover all the necessary information about the sequences. To perform this procedure, the program makes use of the Bioperl module `SearchIO`. `SearchIO` is structured as an object oriented module, meaning that the information it recovers is stored in specific "objects", which contain both the data and a number of "methods"

(comparable to functions or subroutines) used to process this data or recover specific pieces of it.

BLAST outputs can be of many different types; for this program, the XML output was chosen by virtue of the wealth of information it manages to contain in a limited amount of space and for the speed at which it can be parsed. Once the XML file is parsed, the results are stored in a nested fashion: each query sequence's information is stored in an object that also contains all the hit objects that were matched with the query, in turn each hit object contains the hit information and all the HSP objects for that hit (Fig. 3.1).

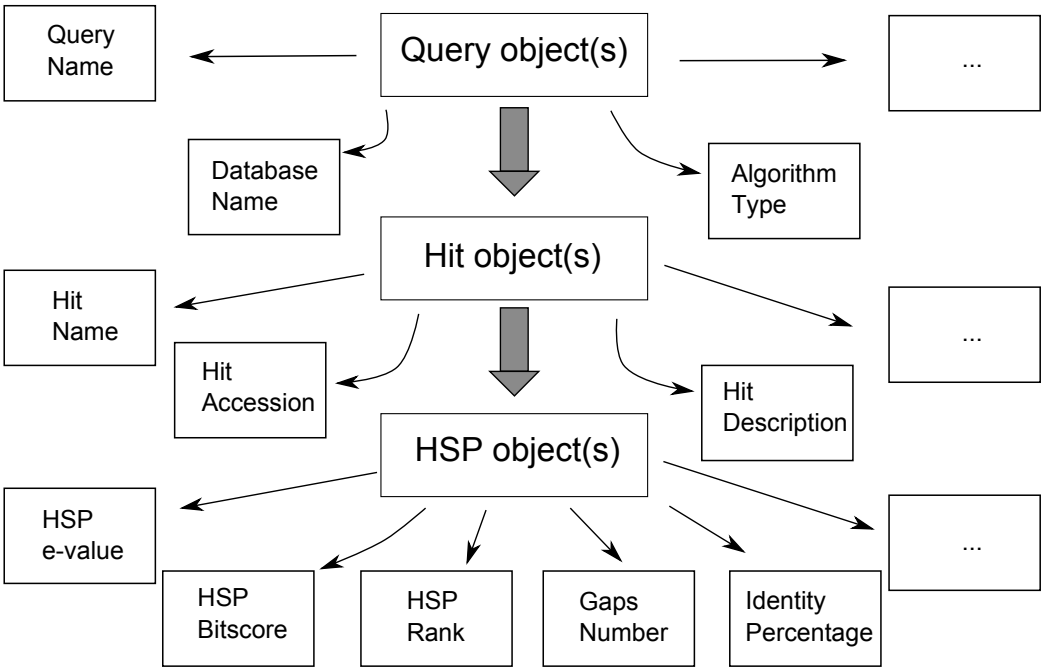

Figure 3.1: Schema representing the hierarchical system used by Bioperl to store Information derived by parsing BLAST.

With this structure, information about single query, hit or HSP objects can be easily extracted and processed inside a series of nested `while` loops.

3.2.2 Informativeness Score

The central part of this algorithm is the capability of providing a new criteria to rank sequences on the basis of the informativeness of their annotations. For this reason, the algorithm is equipped to process the description of each hit and provide a score (in the domain of relative numbers) to distinguish between informative and non informative description, the "Informativeness Score".

Words Obtainment

The calculation of the score is based on a lexical analysis of the description; the program makes use of two lists of words: (1) words that denote a functionally informative annotation (*e.g.* "high affinity nitrate transporter NRT2.6 [Hordeum vulgare subsp. vulgare]"), and (2) words that denote an uninformative or an imprecise annotation (*e.g.* "Sorghum Bicolor hypothetical protein"). These lists of words can be totally customized and new words can be added or taken away without having to tamper with the original code. It is very difficult, however, to obtain a list of sufficient length to cover the majority of positive annotations only through the manual insertion of words. For this reason, the program comes with a pre-made list of words obtained from the processing of Gene Ontology terms.

Gene Ontology possesses a wealth of information regarding good annotations. It is considered, in fact, one of the best ways to correctly annotate a sequence. For this reason, in order to generate a viable list of words with positive connotation, the totality of the Gene Ontology terms was processed. Each term was split into single words, and the obtained list was subsequently scanned for non-biologically relevant words (articles or conjunctions such as "the" or "within"), which were eliminated. To this already substantial list of biologically relevant words, a number of gene names coming from the "reviewed genes" list at UniprotKB was added. The list contains a total of 23,000 words.

Each of these words is stored in a file and possesses two attributes associated with them: a "class identifier" and a "score modifier" (Fig 3.2); the class identifier indicates the origin of the word (Gene Ontology, gene name etc.), while the score modifier represents how much the score will be increased or decreased if that word were to match a given description.

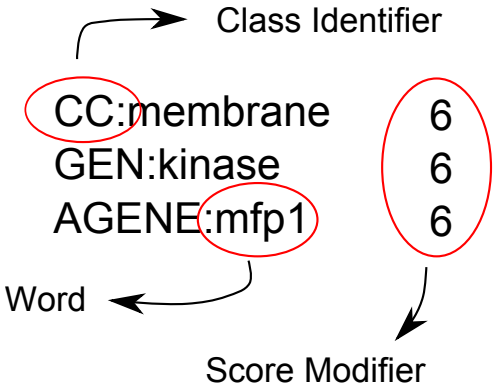

Figure 3.2: Example of how words are stored in a file. Portions highlighted in red mark the three main parts of each row: class identifier, word and score modifier.

An arbitrary number of class identifier can exist within a list of terms. In this particular case, the Gene Ontology-derived terms were divided up according to the functional class in which they were found (biological process, molecular function or cellular compartment). For example, a word that appeared only in Gene Ontology terms belonging to the cellular compartment class, would be labelled with an appropriate class identifier to reflect this. In the same manner, a word that appeared in Gene Ontology terms belonging to both the molecular function and cellular compartment, would have a different class identifier, reflecting the fact that the word was found in terms belonging to two different Gene Ontology functional classes. This process has yielded a total of 7 class identifiers, including all possible combinations; to this, three additional classes were added for gene names originating from different species. The complete description of all class identifier can be seen in table 3.2.

Contrarily to words with positive connotation, those with a negative connotation were manually added, including "hypothetical", "putative", "un-

Class Identifier	Description
CC	Words found only in terms belonging to the Cellular Compartment class
MF	Words found only in terms belonging to the Molecular Function class
BP	Words found only in terms belonging to the Biological Process class
MFBP	Words found in terms belonging to the Molecular Function *and* in terms belonging to the Biological process class
BPCC	Words found in terms belonging to the Cellular compartment *and* in terms belonging to the Biological process class
CCMF	Words found in terms belonging to the Molecular Function *and* in terms belonging to the Cellular compartment class
GEN	Words appearing in terms present throughout the three classes
PGENE	UniprotKB reviewed gene names belonging to the *Poaceae* family
AGENE	UniprotKB reviewed gene names belonging to *Arabidopsis thaliana*
ZGENE	UniprotKB reviewed gene names belonging to *Zea mays*

Table 3.2: List of all class identifiers with the description of their origin.

known", etc.

Score Calculation

When the score calculation takes place, the PERL program reads the words from the files and embeds them in a system of hashes. Due to the high efficiency of the sorting engine used by PERL, in fact, word matching is very efficient.

When the processing of the word-lists is done, the score calculation can begin. For each hit, the description is split into single words and each is

matched against all the words contained in the lists (only once, multiple instances of the same word do not count towards the calculation of the score). Each match increases or decreases the score depending on its connotation (i.e. if the word denotes a good or bad annotation). Ultimately, the sum of all the positive and negative scores obtained from the matches defines the overall informativeness score for the hit, a higher score denoting a better annotation (Fig 3.3).

3.2.3 Interface with MySQL

After all the data has been extracted and the informativeness score has been calculated, the data is ready to be embedded into the relational database. To achieve this goal, the program uses the "DBI" package, a module that allows the connection to a local or remote database, enabling the submission of queries and the retrieval of data through the program.

The insertion of data in each table is achieved by several subroutines working in concert. Three main subroutines insert the required data into the three tables, these subroutines are called each time a new object (either query, hit or HSP) is cycled through. The object is given as argument to the subroutine, which extracts the required data from it using specific methods, formulates a query in the SQL language and lets DBI execute it, inserting the data.

It should be noted that in order to avoid errors and increase the versatility of the program, each subroutine checks that the data it is trying to insert into the database is not already present. This allows to run the program multiple times with different BLAST reports without having to worry about overlapping results. In addition, this enables the program to skip a significant amount of calculations that would otherwise be required to process hits that are present more than once in a single BLAST report.

Finally, to facilitate the centralized storage of all the information, an optional subroutine is available to store the complete query sequences in the database. BLAST reports, in fact, do not contain the complete query

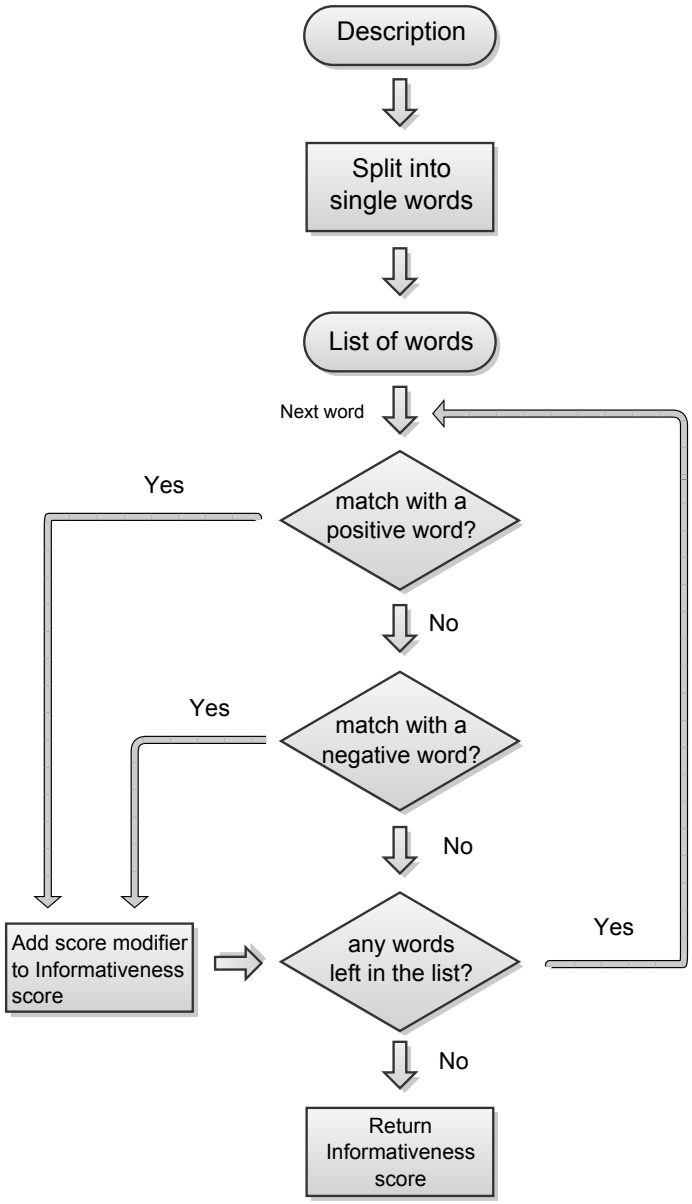

Figure 3.3: Flowchart detailing the steps for the calculation of the informativeness score.

sequence, but just the fragments involved in HSP alignments; for this reason, this optional query takes a FASTA formatted sequence list as argument and reads each sequence, updating the database only where the sequence name matches the queryID already present in the database. This option can be

activated from the command line through insertion of a special keyword, which is managed by the `Getopt::Long` package.

A representation of the general workflow of the program is available in Fig. 3.4.

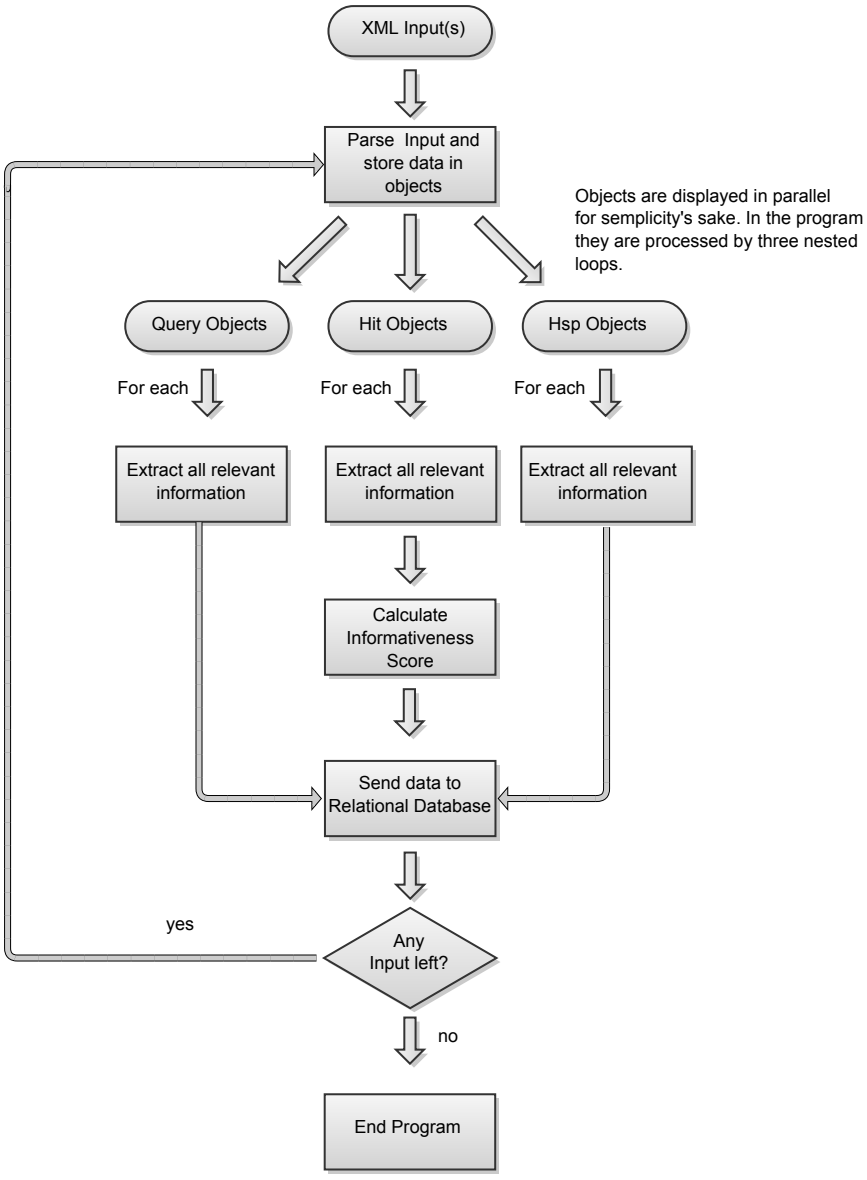

Figure 3.4: Flowchart representing the overall flow of the program. particulars about the calculation of the informativeness score can be seen in Fig. 3.3.

3.3 Testing

In order to determine the efficiency and accuracy in calculating the informativeness scores and interfacing with the database server, the algorithm was tested on a dataset containing 750 RNA sequences obtained by applying next generation sequencing on several *Avena sativa* samples.

3.3.1 Origin of the Data

The original experiments were aimed at studying cold stress response through the analysis of samples from different parts of the plant over several time points in a cold environment. The samples were sequenced with both the Roche 454 and Illumina solexa platforms and assembled using the algorithm available in the CLC cell assembly software[1]. The overall assembly was performed on a pool of reads obtained by merging the single data points, identifying about 25,000 total contigs; of these, 750 were used to perform BLAST searches and test the algorithm.

3.3.2 BLAST Results

The 750 contigs used for testing were compared to different databases using BLAST. In particular, two BLAST reports were generated in XML format: a BLASTn performed against the "nt" database, and a BLASTx performed against a custom protein database containing only data from the *Poaceae* family (also known as *Gramineae*, the family of cereals) and *Arabidopsis thaliana*.

The two BLAST searches were performed to test the robustness of the system with different algorithms and databases. Moreover, the presence of different BLAST reports in the database expanded the number of overall hits for each sequence, increasing the chance of positive annotation.

[1] http://www.clcbio.com/index.php?id=1332

3.4 Results and Validation

In this section, the results obtained by the sorting of the sequences by informativeness score will be explained in detail through the use of descriptive statistics and comparison between BLAST-determined top hits and algorithm-determined top hits. In addition, the frequency distribution of scores will be presented and analysed.

3.4.1 BLAST Comparison

To gauge the effectiveness of the algorithm, a series of comparisons have been run between the best hit determined by BLAST and those determined by the algorithm. For this and all the subsequent comparisons, a cut-off e-value of 1^{-10} was used. After application of this cut-off, the majority, but not all, of the original 750 sequences were left with at least one hit. In particular, a total of 567 sequences for the BLASTn report and 510 for the BLASTx report were left with at least one hit after the elimination of all hits with e-value higher than 1^{-10}.

Fig. 3.5 shows the core of the comparison between BLAST results and algorithm results. In this figure, the comparison between BLAST ranking and the algorithm ranking is done by comparing BLAST-determined best hits (the hit which is considered the best by BLAST for each query sequence) and the algorithm-determined best hits (the hit which is considered the best by the algorithm for each query sequence). These two populations of hits were intersected with the purpose of determining how well the algorithm performs with respect to BLAST. In other words, we wanted to determine for each query sequence, in how many cases the algorithm provided a more informative best hit than BLAST.

In the big pie charts of Fig. 3.5, the population of BLAST-determined best hit is represented. This set of hits is divided into two sub-populations: those that overlap with the algorithm-determined best hits (henceforth referred to as overlapping best hits), represented in blue; and those that do not overlap

with algorithm-determined best hits (non-overlapping best hits), represented in red.

As can be seen in the case of the BLASTn report, 38% of the sequences that were reported as best hits by BLAST were also reported as best hits by the algorithm. This means that for the remaining 62% (numbering 352 sequences out of a total of 567, see Table 3.3) the algorithm was able to provide a best hit that had a more informative description than the hit originally provided by BLAST. Ulterior statistics performed on these two sub-populations of best hits are represented in Fig. 3.5 as the two smaller pie charts.

In these smaller pie charts, the two populations of overlapping and non-overlapping best hits are analysed separately and divided according to their informativeness score. Hits that posses an informativeness score above zero are represented by the dark blue (or dark red) portion of the pie, while hits that posses an informativeness of zero or below are represented by the light blue (or light red) portion. This type of subdivision allows to visualize how BLAST results are distributed in term of informativeness when the data overlapping with the algorithm is taken away. The small red pie charts, in fact, represent the ripartition of the score of those results for which the algorithm could have determined a better best hit; showing the frequency of positive and negative results in those cases in which BLAST-determined best hits are not optimal. In the case of the BLASTn report, the population of overlapping best hits (represented as the blue small pie chart) shows a prevalence of hits with positive informativeness, indicator of a positive annotation. Contrarily, the population of non-overlapping best hits (red pie chart) shows a great prevalence of negative informativeness scores among its hits, denoting functionally uninformative descriptions. This further strengthen the notion that the criteria used by BLAST to choose the best hits are indeed not ideal to classify hits for the purpose of functional annotation transfer.

A slightly different trend can be observed when considering data from the BLASTx report. In this case, in fact, a larger percentage of sequences showed overlap between BLAST- and algorithm-determined best hits (58%).

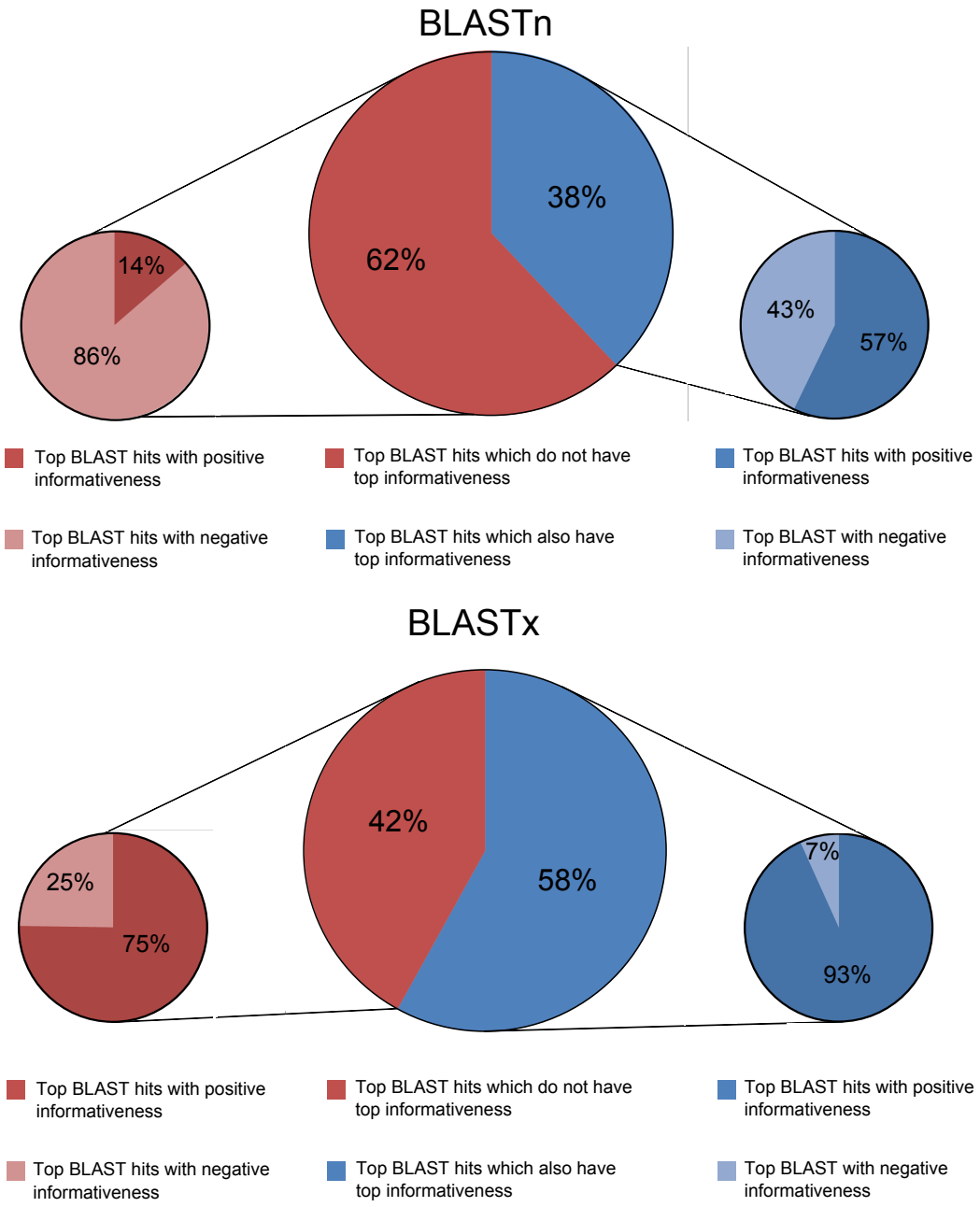

Figure 3.5: Pie chart representation of BLAST-defined best hits divided into overlapping and non-overlapping best hits. The smaller pies represent either the overlapping best hits (blue) or the non-overlapping best hits (red), dividing them up according to informativeness score. Actual numerical values are available in table 3.3.

Also, the additional statistics performed on the two sub-populations shows a significantly reduced rate of negative informativeness scores. This can be explained by the inherent better annotation of protein sequences with respect to nucleotide ones. Also, it should be remembered that the test sequences used for this validation are derived from an RNA-seq experiment, making it likely that a significant hit be found in a protein database. Despite this, the algorithm managed to provide better informativeness scores in 42% of the sequences (214 out of a total of 510, Table 3.3), possibly suggesting that the algorithm works best with sets of poorly annotated results.

Another comparison between BLAST-determined best hits and algorithm-determined best hits was performed in terms of overall informativeness. The aim of this comparison (Fig. 3.6) was to assess how many of the BLAST-defined best hits that have a non-informative description (zero or negative informativeness score) can be replaced by hits with informative descriptions when the algorithm is used instead of BLAST. It should be noted that in this comparison the population of BLAST-determined best hits or algorithm-determined best hits is only divided according to informativeness score, meaning that BLAST-determined best hits which show a positive informativeness may very well have an algorithm-determined counterpart that has a better one. For example, a query for which the BLAST-determined best hit has an informativeness of 12 will still be placed in the positive annotation part even if the algorithm-determined best hit for the same query sequence has an informativeness score of 18.

As Fig. 3.6 shows, BLAST-determined best hits for BLASTn and BLASTx reports have, respectively, 31% and 86% of their sequences with a positive annotation. This significantly changes when analysing the algorithm-determined best hit's informativeness. Indeed there was a significant increase (from 31% to 68%) in hits possessing positive informativeness score when considering the BLASTn report and a less marked but still substantial increase (from 86% to 95%) when considering the BLASTx report. This shows how application of this algorithm can significantly increase the amount of sequences which are paired with an informative best hit. Lastly, the overall

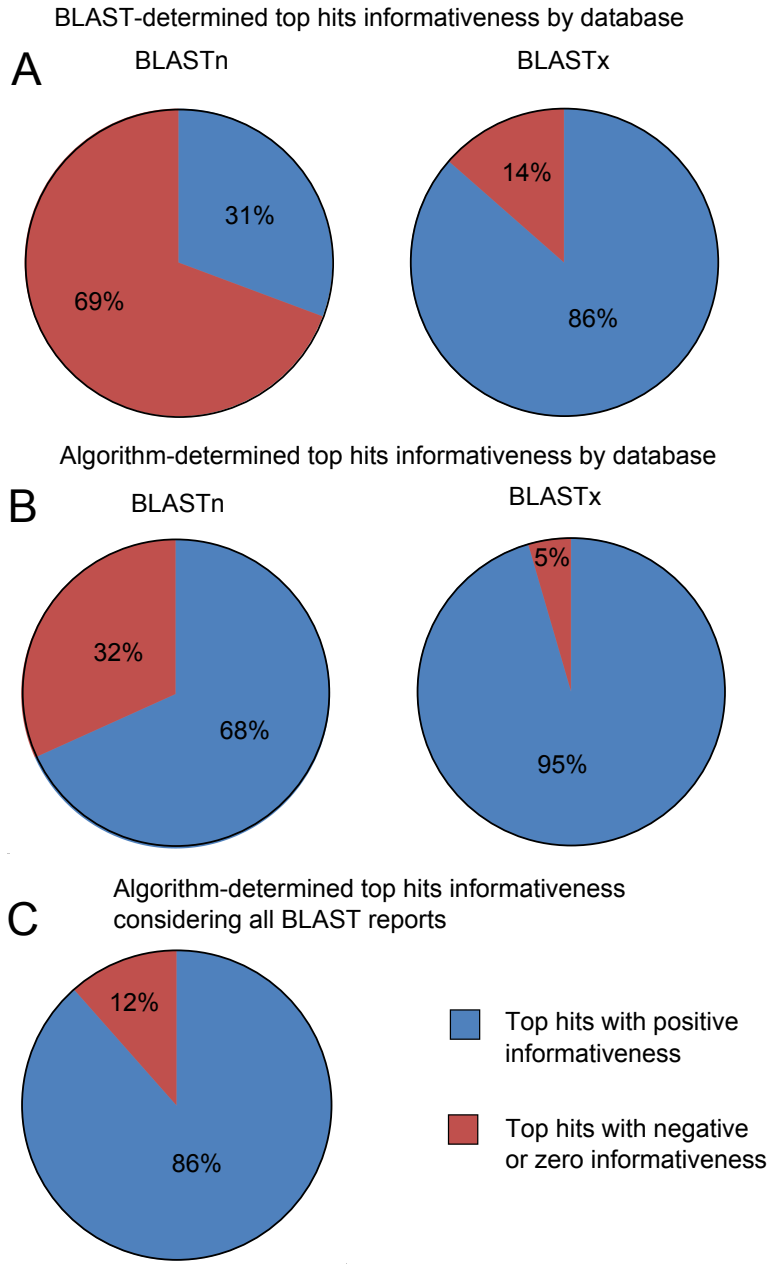

Figure 3.6: Pie charts depicting the percentage of hits with positive or negative informativeness in the following sets of hits. **A:** BLAST-determined best hits divided by BLAST report. **B:** Algorithm-determined best hits divided by BLAST report. **C:** Algorithm-determined best hits considering the combined BLAST reports. Actual numerical values are represented in table 3.3.

algorithm-determined best hits for all available sequences using all available BLAST reports is presented in figure 3.6C. 86% of the sequences (537 out of 607 total) have at least one hit with positive informativeness score, while the 12% (70 sequences) does not have any informative hit.

Statistic	BLASTn	BLASTx
Number of BLAST-determined top hits which are *also* top algorithm hits	215 (38%)	296 (58%)
Number of BLAST-determined top hits that are not top algorithm hits	352 (62%)	214 (42%)
Number of BLAST-determined top hits which are *also* top algorithm hits *and* have a positive informativeness	123 (57%)	276 (93%)
Number of BLAST-determined top hits which are *also* top algorithm hits *and* have a negative or zero informativeness	92 (43%)	20 (7%)
Number of BLAST-determined top hits which are not top algorithm hits *and* have a positive informativeness	48 (14%)	161 (75%)
Number of BLAST-determined top hits which are not top algorithm hits *and* have a negative or zero informativeness	304 (86%)	53 (25%)
BLAST-determined top hits that have a positive informativeness	174 (31%)	441 (86%)
BLAST-determined top hits that have a negative or zero informativeness	393 (69%)	69 (14%)
Algorithm-determined top hits that have a positive informativeness	387 (68%)	487 (95%)
Algorithm-determined top hits that have a negative or zero informativeness	180 (32%)	23 (5%)

Table 3.3: Actual numerical values for each of the categories represented in Fig 3.5 and 3.6. A "Top hit" is considered the first hit presented (either by the algorithm or by BLAST) for each query sequence. Percentages in brackets mirror the ones in Fig. 3.5 and 3.6.

3.4.2 Score Distribution

Another interesting statistic to understand and better optimize the mechanisms for score attribution is score frequency distribution. The distribution is represented as the histogram in figure 3.7.

Score Modifiers

The score assigned to each sequence is strictly dependent on the score modifier that each matching word has. In particular, during this validation a positive matching word would grant a +6 to the overall score, while a negative matching word would grant a -4 to the score; these score modifiers were empirically determined to still provide an overall positive score even if equal number of positive and negative words were matched in the same description. In addition to these two categories, a special PERL regular expression aimed at matching gene locuses or clone names (e.g. `J065163N20`, in this case a clone name) is present and awards a -2 to the overall score. This regular expression was introduced after noting that the vast majority of annotations which possessed an informativeness score of zero, but did not provide any kind of functional information also provided locuses or clone names. To avoid the problem of false positives (i.e. descriptions that possess a locus or clone name but are still informative) matching of this regular expression was set to provide a smaller negative contribution than a negative word, so that descriptions with no positive nor negative matching words (and therefore an informative score of zero) would get a negative score and be classified as bad annotations, while descriptions which had at least one positive matching words would still be positive. For example the description "Oryza sativa Japonica Group Os01g0101600 (Os01g0101600) mRNA, complete cds" does not match any positive nor negative words, and therefore, the algorithm cannot determine if this is a good or bad annotation; cases like this represent the vast majority of descriptions with no matching words. Due to the presence the locus-matching regular expression, this sequence can be attributed a negative score, more fit for its uninformativeness. Contrar-

ily, a description such as "Zea mays clone 223737 protein BRICK1 mRNA, complete cds" which is clearly informative in that it contains the name of a functionally known protein, but also contains a clone name, will still have a positive informativeness score of four (+6 for the matching positive word and -2 for the locus name, equalling a total informativeness score of +4).

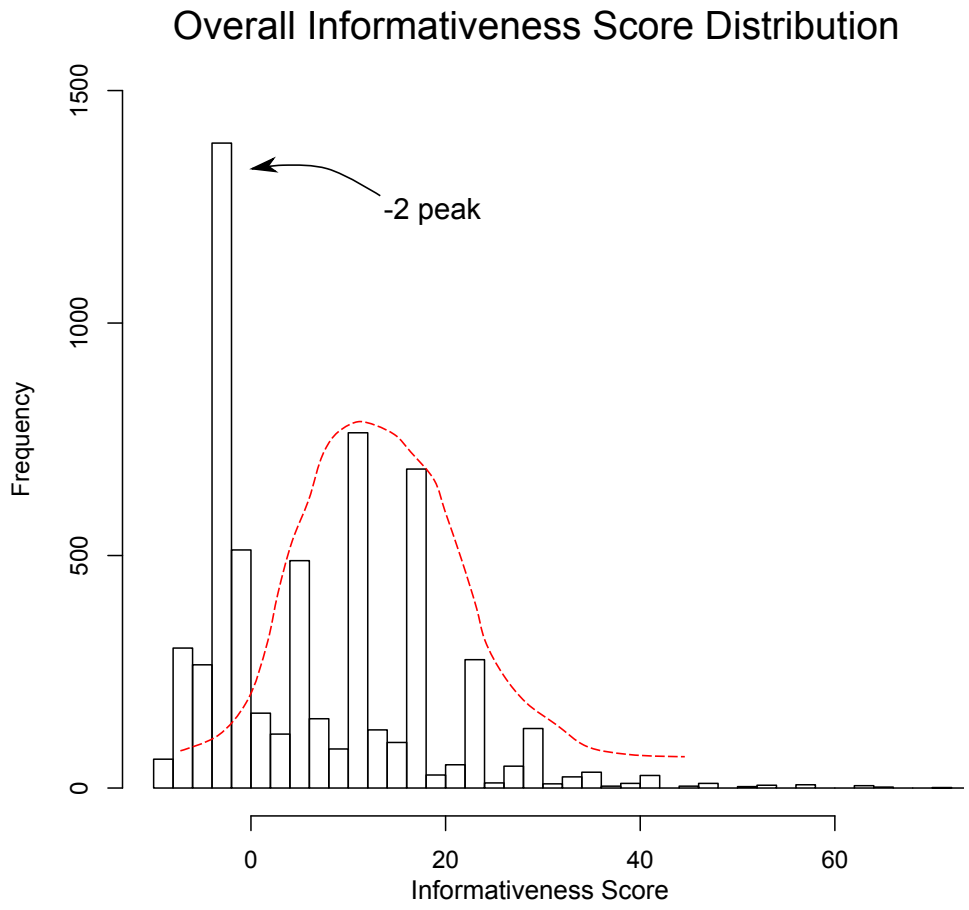

Figure 3.7: Histogram representing the frequency distribution of the informativeness score. Highlighted by the black arrow is the -2 peak generated by the clone name-matching regular expression. Highlighted in red is the Gaussian distribution outlined by the peaks in the positive side of the diagram.

Score Frequency Distribution

As can be seen from the histogram in figure 3.7, the most represented score is -2. This is a direct consequence of the clone name-matching regular expression, conferring a negative score to the huge number of descriptions containing nothing more than the clone name and the organism of origin. As for the overall distribution, peaks can be noted in the positive side of the histogram. These peaks represent the matching positive words alone, each word having a score modifier of +6, while the smaller values between the peaks represent descriptions that contain a combination of positive and negative words. The peaks in this positive section of the histogram also outline a Gaussian distribution with mean 12, suggesting that most frequent number of positive matches be 2, which is indeed the case (data not shown).

Lastly, it should be noted that the histogram does not represent the tail of the distribution in its entirety. The diagram was, in fact, cut to better display the most relevant section, but outliers are present up to an informativeness score of 250. These are extreme cases of very long descriptions containing a multitude of other sequence annotation and matching a huge quantity of positive words.

Part 4

Discussion and Conclusions

4.1 Discussion

The problem of functional annotation has been a constant thorn in the side of many laboratories that decided to apply next generation sequencing techniques to less studied species. In particular experiments such as transcriptome analysis heavily suffer from this problem due to the impossibility of ascribing their results in a relevant biological context. Several tools have been proposed to solve this problem through homology annotation transfer. The principle behind this strategy is that homologous genes share common functions in different organisms, and therefore annotations are transferable between these genes. The tool of choice to find homologous genes in different species is commonly BLAST. From several years this tool has dominated the scene of pairwise alignment tools, and all the tools that deal with functional annotation transfer are based on the analysis of its reports. In the literature, different tools use different approaches to annotate novel sequences. autoFACT, for example, compares the sequences with different databases on a priority basis, analysing the most stringent databases first in the hope of finding an appropriate match, and searching less and less informative databases, such as the EST database, if no significant are found. Another popular tool for annotation transfer is BLAST2GO; this tool is based on determining the

Gene Ontology terms associated with each similarity hit that BLAST provides, and subsequently determines the most relevant terms, annotating with them the novel sequences.

In this report, we tried to propose a novel approach to annotation transfer by devising an algorithm able to provide a score to single similarity hit descriptions on the basis of their biological and functional informativeness. In addition to this, we decided to embed the data in a relational system to facilitate the visualization of the data. In its current state of development, our program is not able to compete with currently available tools for functional annotation. This is mainly due to the lack of a feature able to convert the informativeness score and the positive matching words into fully fledged functional annotation. Indeed, as of now, our scoring system is more aimed at providing a red flag for sequences which contain functional information and is not designed to describe the function of each sequence. Several improvements, however, are in the realm of possibility and could significantly push further the quality of the results.

The first improvement, that would results in a clearer and more accurate attribution of the informativeness score, is a review of the words used and their score modifiers. As of now, the positive matching words are obtained from Gene Ontology together with a list of gene names obtained from UniprotKB. Despite the very good coverage that Gene Ontology terms provide in term of biological terminology, alternative sources of biologically relevant words would increase the number of relevant information that the program would be able to match. A possible endeavour would be to analyse the annotation of a sizeable quantity of well annotated sequences, eliminating trivial or non-informative words and maintaining those with biological relevance. By means of a statistical analysis of sequences of known annotation, tweaks could be made to the score modifiers of each word, attributing more weight to words that are significantly more present in good annotation with respect to bad annotation. Such analysis could also lead to the attribution of high score modifiers to words which are strong indicators of function such as “Kinase” or “ATPase” and less strong score modifiers to words which do not indicate

a function by themselves, but hint to the presence of one, such as "Alanine" or "Cell Wall". This would enable the informativeness score to be a very precise tool to gauge how much functional information a given description contains.

In addition to a more curated version of the word lists, the algorithm could surely benefit from the use of more complex lexical recognition features. The refinement of lexical recognition to go beyond the simple checking of whether a word is present or not in the description would open up a lot of possibilities. Instead of having a fixed score modifier per word, the contribution of each match could vary depending on which other words match. For example, the presence of the word "Arginine" could provide a moderate increase in informativeness score due to its biological relevance, but it would not provide a good annotation by itself; however, the word "Arginine" paired with "Biosynthesis" would significantly change the relevance of the word, warranting the attribution of more points. Another example could be the word "Kinase", usually very important as it defines a very well defined functional role. When this word, however, is paired with "like" in a statement as "Kinase-like protein" the significance of the word significantly decreases.

Another improvement that would immensely benefit this program is the capability of interfacing with Gene Ontology. Gene Ontology has become a force to be reckoned with in the world of sequence annotation, its formally defined categories and terms allow precise function definitions that are clear and easy to read. Adding the gene ontology attribute beside the informativeness score could greatly help the determination of an overall functional annotation, while still retaining the ability to rely on the informativeness score in case Gene Ontology is not present.

A fourth improvement could consist of a feature that allows comparison of results coming from different BLAST outputs. As shown in this very report, the use of different BLAST algorithms such as BLASTn and BLASTx on the same sequences can significantly increase the overall number of well annotated sequences; this is due to an increase in the average number of hits

per sequences and, as a consequence, the increase of the likelihood that one of this hits will be a positive one. While the integration of different BLAST reports and their combined use for the determination of the overall most informative hit is already possible, combination of information from several best hits from different BLAST reports with the aim of cross referencing and confirming one final functional annotation is not currently supported. Cross-referencing a functional annotation in several species or databases could confirm the obtained data, increasing the significance of the result, or point out discrepancies that would have otherwise been missed.

Finally, although not strictly related to the efficiency of the algorithm, a requisite for the popularisation of this program would be the presence of a graphical user interface and the re-development of the algorithm in a platform-independent programming language. This would grant the most visibility and usability to the program, especially to those people who are not computational biologists and might be intimidated by many of the apparently advanced features that this program provides.

4.2 Conclusion

In conclusion, we can say that the program proposed in this report is a net advantage over the use of raw BLAST results. The main feats that the program successfully manages to accomplish affect two different sections of BLAST result analysis: storage and consultation of the data, and annotation transfer. The storage of the totality of the information contained in one or more BLAST reports in a relational database is a great leap forward in how the data are store and consulted. By using Bioperl and other dedicated modules such as `DBI`, the transfer of the data can be accomplished in a fast and efficient manner, without loss of information with respect to the flat file version. In addition, the structure of the database itself provides a very memory-efficient way of storing the data, recurring to generation of very large tables only when specifically required by a user query. Moreover, by providing its own query language, the use of this kind of databases removes

the need for *ad hoc* scripts written with the purpose of extracting data for stratification and visualization.

The second achievement of this program is that of providing the user with an additional criteria for sorting sequences over those already provided by BLAST. The informativeness score, in fact, provides a way to gauge the biological and functional information provided by a sequence description; a fundamental requirement for homology annotation transfer. By applying the informativeness score, a substantial increase in annotation informativeness can be achieved with respect to verbatim BLAST results. Moreover, the usage of this score as a sorting criteria does not limit the manual interaction from the user, which can merely use the score as a visual aid in the choice of the best annotation. Another point in favour of the informativeness score is its full customizability; each word, be it positive or negative, can be give a higher or lower score modifier, therefore increasing or diminishing the impact of the word for the determination of the final score. In addition, words or sets of words can be added or removed without effort from the lists.

In order to be competitive with the tools for functional annotation already present in the literature, this algorithm requires several additional features. However, in the current state of development, it represents a good aid to the determination of sequence informativeness and a significant improvement over flat file information storage.

Part 5

Methods

PERL

The language used for the development of this program was PERL. In particular, Activestate[1] PERL version 5.14.2 for 64 bit windows was used. The build used was 1402.

In addition, the modules `Get::Optlong` and `DBI` were used. The first was included with Activestate PERL installation, the second was version 1.621 and was downloaded through Activestate Perl Package Manager.

BioPerl

The version of bioperl used was 1.6.9, and it was downloaded through the Activestate Perl Package Manager. the modules used in the program are: `Bio::SearchIO`, and `Bio::SeqIO`.

MySQL

The SQL server used to test the algorithm was "MySQL community server" version 5.5.24 downloaded from www.MySQL.com. In addition the "MySQL

[1] http://www.activestate.com/activeperl/downloads

workbench" tool version 5.2.40 was extensively used to manage the connections and design the relational schema.

BLAST

For the purpose of obtaining the BLAST reports, NCBI BLASTn and BLASTx version 2.2.26+ was used.

Bibliography

[1] ANDRADE, M. A., BROWN, N. P., LEROY, C., HOERSCH, S., DE DARUVAR, A., REICH, C., FRANCHINI, A., TAMAMES, J., VALENCIA, A., OUZOUNIS, C., AND SANDLER, C. Automated genome sequence analysis and annotation. *Bioinformatics 5* (1999), 391–412.

[2] CASALS, F., IDAGHDOUR, Y., HUSSIN, J., AND AWADALLA, P. Next-generation sequencing approaches for genetic mapping of complex diseases. *J. Neuroimmunol. doi:10.1016/j.jneuroim.2011.* (2012), 12.017.

[3] CHAMBERLIN, BOYCE, D. D., AND F, R. SEQUEL: A Structured English Query Language. In *Proceedings of the 1974 ACM SIGFIDET Workshop on Data Description, Access and Control* (1974).

[4] CODD, E. F. A relational model of data for large shared data banks. *Communications of the ACM 13* (1970), 377–387.

[5] CODD, E. F. Recent investigations into relational data base systems. Tech. rep., IBM, 1974.

[6] GRABHERR, M. G., HAAS, B. J., YASSOUR, M., LEVIN, J. Z., THOMPSON, D. A., AMIT, I., ADICONIS, X., FAN, L., RAYCHOWDHURY, R., ZENG, Q., CHEN, Z., MAUCELI, E., HACOHEN, N., GNIRKE, A., RHIND, N., DI PALMA, F., BIRREN, B. W., NUSBAUM, C., LINDBLAD-TOH, K., FRIEDMAN, N., AND REGEV, A. Full-length transcriptome assembly from rna-seq data without a reference genome. *Nat Biotechnology 29* (2011), 644–652.

[7] Götz, S., García-Gómez, J. M., Terol, J., Williams, T. D., Nagaraj, S. H., Nueda, M. J., Robles, M., Talón, M., Dopazo, J., and Conesa, A. High-throughput functional annotation and data mining with the Blast2GO suite. *Nucleic Acids Res. 30* (2008), 3420–3435.

[8] Isakov, O., and Shomron, N. *Bioinformatics - Trends and Methodologies*. IN-TECH, 2011.

[9] Koski, L. B., Gray, M. W., Lang, B. F., and Burger, G. AutoFACT: An Automatic Functional Annotation and Classification Tool. *Bioinformatics 25* (2005), 151.

[10] Loewenstein, Y., Raimondo, D., Redfern, O. C., Watson, J., Frishman, D., Linial, M., Orengo, C., Thornton, J., and Tramontano, A. Protein Function Annotation based by homology-based inference. *Genome Biology 10* (2009), doi:10.1186/gb–2009–10–2–207.

[11] Mardis, E. R. Next-Generation DNA Sequencing Methods. *Annu. Rev. Genom. Human Genet 9* (2008), 387–402.

[12] Metzker, M. L. Sequencing technologies - the next generation. *Nature Reviews 11* (2010), 31:46.

[13] Miller, J. R., Koren, S., and Sutton, G. Assembly algorithms for next-generation sequencing data. *Genomics 95* (2010), 315–327.

[14] Scharf, M., Schneider, R., Casari, G., Bork, P., Valencia, A., Ouzounis, C., and Sander, C. Genequiz: a workbench for sequence analysis. In *ISMB-94: proceedings, Second International Conference on Intelligent Systems for Molecular Biology* (1994).

[15] Stajich, J., Block, D., Boulez, K., Brenner, S., Chervitz, S., Dagdigian, C., Fuellen, G., Gilbert, J., Korf, I., Lapp, H., Lehväslaiho, H., Matsalla, C., Mungall, C., Osborne, B., Pocock, M., Schattner, P., Senger, M., Stein, L., Stupka,

E., Wilkinson, M., and E., B. The Bioperl toolkit: Perl modules for the life sciences. *Genome Res 12* (2002), 1611–1618.

[16] The Gene Ontology Consortium, Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., Harris, M. A., Hill, D. P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J. C., Richardson, J. E., Ringwald, M., Rubin, G. M., , and Sherlock, G. Gene ontology: tool for the unification of biology. *Nature Genetics 25* (2000), 25–29.

[17] Zhang, Z., Schwartz, S., Wagner, L., and Miller, W. A greedy algorithm for aligning dna sequences. *Journal of Computational Biology 7* (2000), 203–214.

[18] Zhou, X., Ren, L., Meng, Q., Li, Y., Yu, Y., and Yu, J. The next-generation sequencing technology and application. *Protein Cell 6* (2010), 520:536.

Document typeset with LaTeX