DOI:10.15774/PPKE.ITK.2014.011

LARGE SCALE ANNOTATION OF BIOMOLECULAR DATA USING INTEGRATED DATABASE MANAGEMENT TOOLS

Ph.D. dissertation

Pázmány Péter Catholic University

Faculty of Information Technology and Bionics

Roberto Vera Alvarez

Supervisor: Prof. Sándor Pongor

2014

DOI:10.15774/PPKE.ITK.2014.011

DOI:10.15774/PPKE.ITK.2014.011

Abstract

Bioinformatics experiments usually require efficient computational systems that streamline the data processing. Recent advances in high-throughput technologies have been expanding the experimental scenario. This fact is producing an avalanche of unmanageable data converting the biological sciences from a poor data discipline to a rich one. Furthermore, next-generation sequencing (NGS) technologies created to sequence very long DNA pieces at low cost, are widely used to generate biological data. Unfortunately, the bioinformatics' tools haven't changed their algorithms and computational techniques to deal with this data explosion. Therefore, the integration of biological data, as a product of those technological advances, is far from being a solved task although it is one of the most important and basic element inside the bioinformatics research and/or System Biology projects.

Hence, in this thesis, we developed a biological data integration framework (**JBioWH)** that has a modular design for the integration of the most important biological databases. The framework is comprised of a Java API for external use, a desktop client and a webservices application. This system has been supplying integrative data for many bioinformatics projects. Also, a program (**Taxoner)** was developed to identify taxonomies by mapping NGS reads to a comprehensive sequence database. As a result of alterations to the indexing used, this pipeline is fast enough to run evaluations on a single PC, and is highly sensitive; as a result, it can be adapted to the analysis problems such as detecting pathogens in human samples. Finally, a workflow for DNA sequence comparison is presented. This workflow is applied either to create a marker database for taxonomy binning or just to obtain unique DNA segments among a group of targets sequences. It is based on a set of *in-house* developed programs that includes the **JBioWH** and **Taxoner**. All the programs developed are freely available through the Google Code Platform.

DOI:10.15774/PPKE.ITK.2014.011

DOI:10.15774/PPKE.ITK.2014.011

List of Publications

1. Pongor L S, Vera R, Ligeti B, **Fast and sensitive alignment of microbial whole genome sequencing reads to large sequence datasets on a desktop PC: application to metagenomic datasets and pathogen identification.** *PloS one 2014; 9(7):e103441.*

2. Dogsa I, Choudhary KS, Marsetic Z, Hudaiberdiev S, Vera R, Pongor S, Mandic-Mulec I: **ComQXPA Quorum Sensing Systems May Not Be Unique to Bacillus subtilis: A Census in Prokaryotic Genomes.** *PloS one* 2014, 9(5):e96122.

3. Vera R, Perez-Riverol Y, Perez S, Ligeti B, Kertész-Farkas A, Pongor S: **JBioWH: an open-source Java framework for bioinformatics data integration.** *Database: the journal of biological databases and curation* 2013, **2013**:bat051.

4. Kertész-Farkas A, Reiz B, Vera R, Myers MP, Pongor S: **PTMTreeSearch: a novel two-stage tree-search algorithm with pruning rules for the identification of post-translational modification of proteins in MS/MS spectra.** *Bioinformatics (Oxford, England)* 2013, **30**:234-241.

5. Ligeti B, Vera R, Lukacs G, Gyorffy B, Pongor S: **Predicting effective drug combinations via network propagation**. In: *2013 IEEE Biomedical Circuits and Systems Conference (BioCAS).* IEEE; 2013: 378-381.

6. Perez-Riverol Y, Vera R, Mazola Y, Musacchio A: **A parallel systematic-Monte Carlo algorithm for exploring conformational space.** *Current topics in medicinal chemistry* 2012, **12**:1790-1796.

7. Perez-Riverol Y, Sánchez A, Ramos Y, Schmidt A, Müller M, Betancourt L, González LJ, Vera R, Padron G, Besada V: **In silico analysis of accurate proteomics, complemented by selective isolation of peptides.** *Journal of proteomics* 2011, **74**:2071-2082.

8. Sanchez A , Perez-Riverol Y , González LJ , Noda J, Betancourt L, Ramos Y, Gil J, Vera R, Padrón G , Besada V: **Evaluation of Phenylthiocarbamoyl-Derivatized Peptides by Electrospray Ionization Mass Spectrometry: Selective Isolation and Analysis of Modified Multiply Charged Peptides for Liquid Chromatography-Tandem Mass Spectrometry Experiments.** *Analytical chemistry* 2010, **xxx**:552-559.

9. Mazola Reyes Y, Chinea Santiago G, Guirola Cruz O, Vera Alvarez R, Huerta Galindo V, Fleitas Salazar N, Musacchio Lasa A: **Chemical compounds having antiviral activity against dengue virus and other flaviviruses.** In.: WO/2009/106019; 2009.

10. Rodriguez Fernandez RE, Vera Alvarez R, de la Nuez Veulens A, Mazola Reyes Y, Perea Rodriguez SE, Acevedo Castro BE, Musacchio Lasa A, Ubieta Gomez R: **Antineoplastic compounds and pharmaceutical compositions thereof.** In.: WO/2006/119713; 2006.

DOI:10.15774/PPKE.ITK.2014.011

DOI:10.15774/PPKE.ITK.2014.011

List of Abbreviations

Bp	Base pair
BWT	Burrows-Wheeler transforms
DBG	Bruijn Graph
LC	Liquid chromatography
mRNA	messenger RNA
MS	Mass Spectrometry
NGS	Next-generation sequencing
OLC	Overlap/Layout/Consensus
PCR	polymerase chain reaction
RAM	Random-access memory
rRNA	ribosomal RNA
tRNA	transfer RNA
WGS	Whole-genome shotgun
XML	eXtensible Markup Language

DOI:10.15774/PPKE.ITK.2014.011

DOI:10.15774/PPKE.ITK.2014.011

List of Figures

DOI:10.15774/PPKE.ITK.2014.011

DOI:10.15774/PPKE.ITK.2014.011

List of Tables

DOI:10.15774/PPKE.ITK.2014.011

DOI:10.15774/PPKE.ITK.2014.011

Acknowledgements

This work is the result of an exceptionally fruitful collaboration between institutions from Cuba, Italy and Hungary. The project started during my first visit to the ICGEB in 2008, when I was invited by Prof. Sándor Pongor. His support, kindness and, especially, his scientific vision has made possible all these results. Special thanks are due to my friends and close collaborators, Yasset Perez-Riverol (European Bioinformatics Institute), Sonal K. Choudhary and Sanjar Hudaiberdiev (ICGEB, Trieste), Lőrinc S. Pongor (Semmelweis University, Budapest) and Balázs Ligeti (Pázmány University, Budapest). Much of this work is the result of our common efforts so a large part of the credit goes to them.

I wish to express my gratitude to ICGEB-Trieste, Italy, to LNCIB, Trieste, Italy, and to Pázmány Péter Catholic University, Budapest, Hungary for the fellowships and the financial support. In particular, I wish to thank the help and assistance of Profs. Francisco E. Baralle (ICGEB) and Claudio Schneider (LNCIB)

I want to thank my labmates at ICGEB, Sonal and Sanjar for being family and for the many English corrections to my thesis and my presentations. I also want thank my colleagues at LNCIB, Vanessa Florit and Raffaella Florit for their trust and support.

My special hanks are due to all my friends in Trieste, especially to Piero, Roberta, Giacomo, Elena, Giorgia, Giampaolo, Simona, Sorrentino, Roberto and Daniela for their friendship and support.

And to my catholic community at the San Giusto Cathedral of Trieste: *Grazie mille per la preghiera e per ricordarci che non dobbiamo avere paura: Dio ci ama e sempre provvede.*

I would like to thank my wife and children for being there for me always and letting me work despite of their needs. And to my parents and my sister for their sacrifices which made me the man that I am: *Gracias mis amores por todo. Este también es un logro de todos ustedes. Los amo infinitamente.*

DOI:10.15774/PPKE.ITK.2014.011

DOI:10.15774/PPKE.ITK.2014.011

Contents

DOI:10.15774/PPKE.ITK.2014.011

DOI:10.15774/PPKE.ITK.2014.011

1 Introduction

The introduction of information technology to manage the biological data has been changing the biological research. Also, a technological revolution on labs, like low-cost genomic DNA sequencing technologies and biological assays made by robots, have been contributing to this new era of the biological sciences.

The field of Bioinformatics is a product of this new era. The **National Center for Biotechnology Information (NCBI)** defines bioinformatics in 2001 as:

“Bioinformatics is the field of science in which biology, computer science, and information technology merges into a single discipline. There are three important sub-disciplines within bioinformatics: the development of new algorithms and statistics with which to assess relationships among members of large data sets; the analysis and interpretation of various types of data including nucleotide and amino acid sequences, protein domains, and protein structures; and the development and implementation of tools that enable efficient access and management of different types of information.”

This discipline have been used for decades as a basic support for biological research projects, its use provides a better understanding of complex biological process.

Additionally, the Systems Biology, known as the next big intellectual challenge in biology, has been integrating the Bioinformatics tools in its research projects producing high quality results. This synergy is not a method but a paradigm, a general approach of thinking [2]. They form an inter-discipline that makes use of principles, knowledge and tools coming from biology, computer sciences, medicine, physics, chemistry and engineering, bridging the gaps between them [3].

This paradigm is helping to develop new research archetypes that shifts focus from traditional studies of single gene or protein to unified systems view on biological processes. It combines data-mining from large-scale, technology-driven projects, such as human

DOI:10.15774/PPKE.ITK.2014.011

genome or structural genomics projects, with traditional hypothesis-driven experimental work.

The aforementioned way of thinking tend to improve the understanding of the biology and its mechanisms, particularly, the interactions between its key elements (DNA, RNA, proteins, chemical compounds, cells, etc.). The bottom up approach builds on its key elements, which are strictly related to the OMICS data sets, and the proposed models of biological systems.

This integration through novel methodologies is not limited to the description of the existing knowledge using a new syntax. It combines old and new models to develop new approaches for the characterization of the biological systems.

Based on an integrationist approach, this paradigm fills the empty spaces left by the reductionist approaches where the complex system is nothing more than the sum of its parts. These reductionist approaches have played an important role in the development of the biology sciences until now. They allowed the identification of the majority of the biological elements independently. Unfortunately, they offer no convincing concepts and methods to comprehend how system properties emerge [4].

The technical challenges of the Systems Biology [5] are mainly along four lines: (i) system-wide component identification and quantification ("OMICS" data sets); (ii) experimental identification of physical component interactions, primarily for information processing networks; (iii) computational inference of structure, type, and quantity of component interactions from data; and (iv) rigorous integration of heterogeneous data [41]. Being this last step significantly behind, such that many more data are generated than possibly can be analyzed or interpreted [6].

DOI:10.15774/PPKE.ITK.2014.011

1.1 Next-generation sequencing

Knowing the DNA sequence is essential for all kind of projects in biological research. Scientists extracting the genetic information from the biological systems produced a limitless insight to the "OMICS" disciplines, especially to the Genomics and Transcriptomics. The DNA sequencing techniques have been evolving continuously, and consequently, they have been incorporating new challenges to the Bioinformatics.

On the other hand, major developments of bioinformatics usually result from parallel development in data processing. "Traditional bioinformatics" provides data processing and storage technologies that served protein and DNA sequencing technologies developed from the 70's and 80's. However, next generation sequencing **(NGS),** that is an umbrella name for DNA sequencing technologies that followed after 2000, is a complete new approach that cannot be afforded using the traditional bioinformatics tools. The goal of this chapter is to give an overview of the data processing of NGS, I briefly review this field based on Homer and Li [7].

These new sequencing technologies often aims at sequencing very long DNA pieces, such as whole chromosomes, although, large-scale sequencing can also be used to generate very large numbers of short sequences. Also, they are designed for low-cost sequencing producing high-throughput DNA sequences data of the order of giga base-pairs (Gbp) per machine day [8].

There are two groups of DNA sequencing technologies that fall into the NGS umbrella. The first group (second generation) follows chemical synthesis techniques, similar in part to conventional Sanger sequencing, another group (third generation) uses single molecular techniques that, on one hand, do not require amplification, but on the other hand, report single molecule variants, see Table 1.

From the perspective of bioinformatics, the length and the accuracy of the reads is the most important (two right columns), as this will determine the probability of chance identities, i.e. the background noise of sequence alignment.

DOI:10.15774/PPKE.ITK.2014.011

NGS applications fall into a few characteristic categories, listed in Table 2. It is interesting to realize the parallelism between the data processing perspectives of high throughput proteomics and NGS. Nevertheless, the two fields were developed in vastly different ways, see Table 3.

Table 1: Types of NGS technologies

"Generation"	Type of sequencing	Instrument	Max read length	Accuracy
Second generation sequencers	sequencing-by-synthesis	Roche/454	700	99.9%
		Illumina	150	98%
	sequencing-by-ligation	ABI/ SOLiD	75-100	99.9%
	sequencing-by-synthesis	Ion Torrent/Life Technologies	200	98%
Third generation sequencers (single molecule sequencing)	single-molecule sequencing	Helicos	50	99.5%
	single-molecule sequencing	Pacific Biosciences	1500	87-99
	nanospore single-molecule sequencing	Oxford Nanospore	n/a	n/a

The Bioinformatics approaches used by the NGS applications to process the data can be described by two main groups. The first group includes the approaches to manipulate, store and share the experimental results, namely reads or segments of DNA sequences, including the experimental description. The second group includes computational applications to process the experimental results. These computational tools can be generally classified in tools to align the DNA reads and tools to assemble the final DNA sequence.

DOI:10.15774/PPKE.ITK.2014.011

Table 2: Goals of next generation sequencing

Category	**Application example**
De novo sequencing of complete genomes	Description of new species, identification of unique genes.
Complete genome resequencing	Identification of mutations and polymorphisms; structural variants
Paired-end sequencings	Inherited and acquired structural variations (e.g. CNV)
Metagenomic sequencing	Study of microbial communities, detection of pathogens
Transcriptome sequencing	Quantifying/comparing gene expression
MicroRNA sequencing	Gene regulation studies
Exome sequencing	Concomitant study of all exons in all genes
Molecular barcode sequencing	Parallel identification of several species

Table 3: The parallelism between NGS and high throughput proteomics

	NGS	**High throughput proteomics**
Macromolecular fragments	Random, overlapping	Enzyme generated, disjunction
Analysis of single items	Genomics, assembly Goal: single genome sequencing	*De novo* sequencing by proteomics Goal: Single protein sequencing
Analysis of complete mixtures	Metagenomics; Goal: Identification of taxa in complex communities	High throughput MS/MS; Goal: identification of proteins in complex mixtures

DOI:10.15774/PPKE.ITK.2014.011

1.1.1 Sequence alignment algorithms

The crucial part of NGS data processing is the alignment of the DNA reads. Most current alignment algorithms index structures either for the read sequences or for the reference database sequence, or sometimes both. Based on the index properties, alignment algorithms can be largely grouped into two large categories: a) algorithms based on hash tables, b) algorithms based on suffix trees and algorithms based on merge sorting. (An additional group consists of Slider [9] and its descendant SliderII [10], we concentrate on the first two categories).

1.1.2 Hash table based approaches

All hash table based algorithms essentially follow the idea of BLAST of Steve Altschul and associates [11, 12] which can be called a seed-and-extend paradigm. The philosophy was based on protein sequencing, but the main corollaries keep also for DNA sequencing. The fundamental supposition is that homologous sequences contain conserved segments that can be located by the position of k-mer words shared by two sequences. BLAST keeps the position of each k-mer (k = 11 by default) subsequence of the query in a hash table with the k-mer sequence being the key, and scans the database sequences for k-mer exact matches, by looking up the hash table. A sufficient (user selected) number of k-mers is called a seed. BLAST extends and joins the seeds first without gaps and then refines them by a Smith-Waterman alignment [13, 14]. It outputs statistically significant local alignments as the final results. This basic approach has been improved and adapted to alignments of different types, but here we focus on mapping a set of short query sequences against a long reference genome of the same species.

1.1.3 Improved seeding techniques

Ma and associates realized that seeds of k non-consecutive identities are more sensitive than seeds of k consecutive identities, used in the original BLAST algorithm [15, 16]. A seed allowing internal mismatches is called a *spaced seed*; the number of matches in the

DOI:10.15774/PPKE.ITK.2014.011

seed is its weight. The time complexity of spaced seed alignment is approximately proportional to $mnL/4^q$ where q is the weight, m the number of templates, n the number of reads and L the genome size. The memory required by hashing genome is usually log2 L max(4q, L/s) bytes where s is the sampling frequency [17]. It is memory demanding to hold in RAM a hash table with q larger than 15. Homer and associates proposed a two-level indexing scheme for any large q [18]. They build a hash table for j-long ($j < q$, typically 14) bases. To find a q-long key, they look up the hash table from the first j bases and then perform a binary search among elements stored in the resulting bucket. Looking up a q-long ($q > \log4 L$) key takes O(max(1, log4 L – j)) time, only slightly worse than the optimal speed O(1). And, as a consequence, the peak memory becomes independent of q.

The idea can be implemented in several ways, sometimes linked to the peculiarities of individual sequencing technologies; we only cite a few here. The program Eland builds k-mer templates for the reads. For an Illumina read it builds 6 templates and allows for two mismatches during the alignment. SOAP [19] adopts almost the same strategy except that it indexes the genome rather than reads. SeqMap [20] and MAQ [21] allow k mismatches, this requires an exponentially high number of templates for the same sensitivity which is inefficient for larger k values. To improve the speed, MAQ only takes advantage of the fact that the first half of the reads is more reliable so it accepts only 2 mismatches in the first 28bp *i.e.* the most reliable part of an Illumina read. MAQ will extend the partial match when a seed match is found. There are several other programs that capitalize on the same or similar ideas, for a further review see [7].

A potential problem with consecutive seed and spaced seed is they disallow gaps within the seed. Gaps are usually found afterwards in the extension step by dynamic programming, or by attempting small gaps at each read positions [19, 20]. The q-gram filter [22-24] is based on the observation that at the occurrence of a w-long query string with at most k differences (mismatches and gaps), the query and the w-long database substring share at least $(w+1)-(k+1)q$ common substrings of length q [25-27]. Methods based on spaced seeds and the q-gram filters are similar in that they both rely on fast lookup in a hash table. They are mainly different in that the former category initiates seed extension from one long seed

DOI:10.15774/PPKE.ITK.2014.011

match, while the latter initiates extension usually with multiple relatively short seed matches. In fact, the idea of requiring multiple seed matches is more frequently seen in capillary read aligners such as SSAHA2 and BLAT; it is a major technique to accelerate long-read alignment.

1.1.4 Improved seed extension

Due to the use of long spaced seeds, many aligners do not need to perform seed extension or only extend a seed match without gaps, which is much faster than dynamic programming. Nonetheless, several improvements over BLAST have been made regarding seed extension. A major improvement comes from accelerating the standard Smith-Waterman with vectorization. The basic idea is to parallelize alignment with the CPU SIMD instructions such that multiple parts of a query sequence can be processed in one CPU cycle. Using the SSE2 CPU instructions implemented in most latest x86 CPUs, [28]results in a revised Smith-Waterman algorithm that is over 10 times faster than the standard algorithm. Novoalign (http://novocraft.com), CLC Genomics Workbench (http://clcbio.com) and SHRiMP are known to make use of vectorization.

Another improvement is achieved by constraining dynamic programming around seeds already found in the seeding step [18, 29, 30]. Thus, unnecessary visits to cells far away from seed hits in iteration are greatly reduced. In addition, [31] found that a query can be aligned in full length to an L-long target sequence with up to k mismatches and gaps in O(kL) time, independent of the length of the query. These techniques also help to accelerate the alignment when dynamic programming is the bottleneck.

1.1.5 Fast aligners based on suffix/prefix try

The use of hash tables is getting impractical for larger input data sizes. As a consequence, current aligners seek to the inexact matching problem to the exact matching problem and involve two steps: 1) identifying exact matches and 2) building inexact alignments supported by exact matches. To find exact matches, these algorithms rely on a certain data

DOI:10.15774/PPKE.ITK.2014.011

representations, such as suffix tree, enhanced suffix array [32] and FM-index [33]. The advantage of using a try as the basic data structure is that alignment to multiple identical copies of a substring in the reference is only needed to be done once because these identical copies collapse on a single path in the try, whereas with a typical hash table index, an alignment must be performed for each copy. It should be noted that the choice of these data structures is independent of methods of step 2 for finding inexact matches *i.e.* an algorithm built upon FM-index, for example, would also work with suffix tree index in principle.

Algorithms in this class make use of key methods, such as the Burrows-Wheeler transform (BWT) [34], the Ferragina-Manzini index (FM) [33] and the Huffman coding (HC) [35]. Briefly, BWT is a lossless compression algorithm used among others in bzip2, it allows one to compress and decompress data without loss of information. The compression of biological data, such as a genome becomes practical if one combines BWT with HC or other coding technique. A BWT encoded dataset is not suitable for searching in itself, but Ferragina and Manzini discovered in 2005 that a new index, now called the FM index, which can search a BWT dataset without decompression. With this combination we have a new index structure that can replace the hash table with a concomitant gain in speed. Further, speedup is possible by avoiding dynamic programming, even though some of the current programs allow dynamic programming as a more sensitive option.

There are two current programs that are widely used today in the bioinformatics community. BWA [36] is the program of the Durbin group that was the first in time, it is relatively sensitive but somewhat less fast than Bowtie [37] and Bowtie2, developed at the Salzberg group which is considered somewhat faster but less sensitive.

1.1.6 Sequence assembly

The assembly process is out of the scope of this thesis, however, we would like to introduce briefly some concepts and algorithms used to assemble DNA segments up to chromosome length.

DOI:10.15774/PPKE.ITK.2014.011

An assembly is a hierarchical data structure that maps the sequence data to a putative reconstruction of the target. It groups reads into contigs and contigs into scaffolds. Contigs provide a multiple sequence alignment of reads plus the consensus sequence. The scaffolds, sometimes called supercontigs or metacontigs, define the contig order and orientation and the sizes of the gaps between contigs. Scaffold topology may be a simple path or a network [38].

Sequence assembly is the reconstruction of sequence up to chromosome length. The assembly task is relegated to computer software [39]. Assembly is possible when the target is over-sampled by the shotgun reads, such that reads overlap. De novo Whole-genome shotgun (**WGS**) assembly refers to reconstruction in its pure form, without consultation to previously resolved sequence including from genomes, transcripts, and proteins [38]. DNA sequencing technologies share the fundamental limitation that read lengths are much shorter than even the smallest genomes. WGS overcomes this limitation by over-sampling the target genome with short reads from random positions. Assembly software reconstructs the target sequence.

The NGS assemblers can be group into three categories, all based on graphs. The Overlap/Layout/Consensus (**OLC**) methods rely on an overlap graph. The de Bruijn Graph (**DBG**) methods use some form of K-mer graph. The greedy graph algorithms may use OLC or DBG.

The OLC approach was typical of the Sanger-data assemblers. It was optimized for large genomes in software including Celera Assembler [40], Arachne [41, 42], and CAP and PCAP [43]. The OLC approach has been reviewed elsewhere [44]. The most representatives assembler programs using this approach are Newbler [45], the Celera Assembler [40] and Edena [46]

The DBG approach is most widely applied to the short reads from the Solexa and SOLiD platforms. It relies on K-mer graphs, whose attributes make it attractive for vast quantities of short reads, see [47] for review. The most representatives assembler programs using this

DOI:10.15774/PPKE.ITK.2014.011

approach are Euler developed for Sanger reads [48-50], Velvet [51, 52], ABySS [53], AllPaths [54] and SOAPdenovo [55].

1.2 OMICS disciplines and biological databases

The OMICS suffix has been added to the names of many kinds of biological studies undertaken on a large or genome-wide scale. Today, there are numerous derivatives of the basic concept of large-scale biological analysis, with the common denominator of aiming to study the complete repertoire of particular biological entities [56]. It includes several disciplines that are growing within this new biological era. They are mainly represented by Genomics, Transcriptomics, Metagenomics and Metatranscriptomics, Proteomics, Metabolomics/metabonomics, Localizomics, etc.

The advent of whole-genome sequencing and other high-throughput experimental technologies transform the biological research from a relatively data poor discipline into one that is data rich [5]. This exponential increment of the data produced by the OMICS disciplines is associated to the number of available biological databases. A synergy between the OMICS and the Bioinformatics tools is used to produce, store, process and validate the biological data. They use high-throughput screening experiments for identification and validation of biological entities; computational tool and databases to manage the data generated in the previous stage; and algorithms for computational predictions of biological properties and interactions [57-59].

One of the biggest problems of this synergy is that the high-throughput experimental technologies used for querying the biological system have an inherent high rate of false positive results and they are able to produce an unmanageable volume of data [60, 61]. Although, each experiment result obtained is not useful by itself. They have a limited utility unless efficient computational systems are used to manage, integrate and process them.

Indeed, a biological database as any kind of database is a collection of data that is organized so that its contents can easily be accessed, managed, and updated [59, 62]. A

DOI:10.15774/PPKE.ITK.2014.011

simple database might be a single file containing many records, each of which includes the same set of information.

The Nucleic Acids Research (NAR) journal online Molecular Biology Database Collection [63] published a collection of 1552 databases that are sorted into 14 categories and 41 subcategories. This updated collection includes only active databases at the beginning of 2014, see Figure 1.

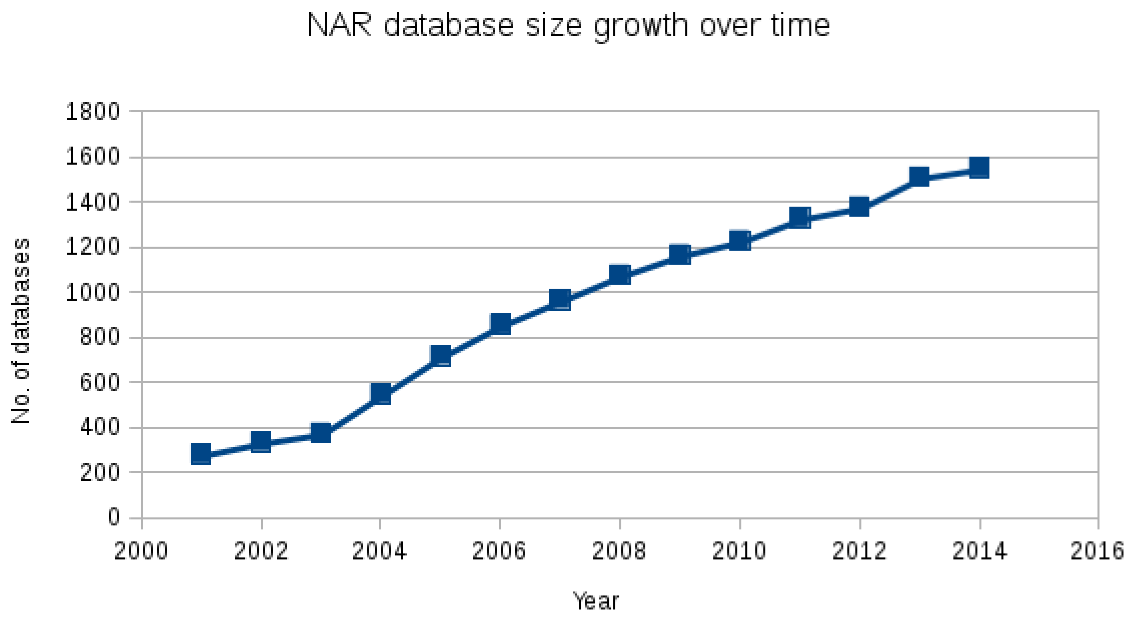


Figure 1: Biological database growth during the last decade

These set of categories and subcategories are not the only way to classify the biological databases. Some authors grouped them in three general groups for better understanding [4]. The first group includes the primary databases. These databases contain information of the sequence or structure alone. It includes nucleotide, RNA, protein sequences and structures databases. A second group includes databases which are generated by a computational processing of the primary databases and they are named secondary databases. This group includes databases of genes, genomes, protein domains and families, etc. Finally, the last group, named composite databases or metabases [64], includes databases generated from

DOI:10.15774/PPKE.ITK.2014.011

the integration of the primary and secondary databases. This group has been growing according with the level of acceptance of the Systems Biology and its use to solve problems in research projects. Also, it is closely related to one of the most important problems faced today which is the integration of biological data.

We accept and respect these classifications; however, we will present and discuss the biological databases in more detail through the organization of the OMICS disciplines. This structure follows the Systems Biology point of view and offers a better overview of all its elements and their integration. Despite of that, the databases will be located in one of these three groups once they are presented and analyzed. Furthermore, there are 1552 active databases, so only the most important and useful databases will be presented and discussed here.

1.2.1 Genomics

This OMICS discipline is defined as the study of the whole genome sequence and the information contained therein, is clearly the most mature of all OMICS [5]. Since 1995, when the first bacterial genome was sequenced [65], a huge explosion on genomic data has occurred. More than thousand organisms have been sequenced producing more than 169 million of sequences available according to the NCBI statistics [66].

In the last years, this discipline has received a fresh support with the arrival of new technologies for sequencing at low costs. These high-throughput sequencing (or next-generation sequencing) technologies are able to produce thousands or millions of sequences concurrently [67].

Additionally, the genomics databases represent the most widely used databases and they are the best established. With terabytes of data, these databases cover the three main groups aforementioned.

The primary databases related to Genomics are: GenBank [68], DDBJ [69] and ENA [70]. These databases provide public repositories for the nucleotide sequences data. They daily

DOI:10.15774/PPKE.ITK.2014.011

exchange data between them to ensure worldwide coverage. These databases contain sequences for almost 260 000 formally described species [68]. They offer to the user web portals and desktop tools to submit and update entries creating a direct channel between the database administrators and the data suppliers.

The secondary databases are mainly represented by Entrez Gene [71], Entrez Genome [72], KEGG Genes and KEGG Genome [73], GOLD [74] and Ensembl [75].

The Entrez Gene database is a gene-specific database which establishes a gene-to-sequence relationship used by other NCBI resources. It provides tracked, unique identifiers for genes and to report information associated with those identifiers for unrestricted public use [71].

The Entrez Genome database provides access to more than six thousand complete genomes. The database offers a graphical overview of an entire genome to the level of a single gene. At the level of a genome or a chromosome, a Coding Regions display gives the locations coding regions, and the lengths, names and GenBank identification numbers of the protein products [72].

The KEGG is an integrated database of 15 main databases. KEGG Genes and KEGG Genome are the KEGG resources related to the Genomics discipline. KEGG Genes is a collection of gene catalogs for all complete genomes generated from publicly available resources. The KEGG Genome is a collection of organisms with known complete genome sequences. Similar to NCBI databases, the KEGG database offers multiple tools for submitting the data to their resources. It also provides an avowed group of biocurators involved in the analysis, interpretation and integration of the biological information into the data repositories.

The GOLD database provides an online centralized portal for genomic and metagenomic projects. It includes the implementation of GOLD-specific controlled vocabularies for representation of the associated data, in coordination with the Genomics Standards Consortium (GSC) [74, 76].

DOI:10.15774/PPKE.ITK.2014.011

Finally, the Ensembl project creates and distributes genome annotations and provides integrated views of other valuable genomic data for supported genomes. Ensembl provides unique tools, datasets and user support compared to similar projects such as the UCSC Genome Browser. It offers an open software infrastructure with diverse analysis pipelines supporting a variety of genome analysis methods [75].

1.2.2 Transcriptomics

Transcriptomics is defined as the study of Transcriptome and its interactions. Transcriptome is set of all RNA molecules including the messenger RNA (mRNA), ribosomal RNA (rRNA) and transfer RNA (tRNA). All this important molecules perform multiple vital roles in coding, decoding, regulation, and expression of genes.

There are several databases related with this OMICS discipline. Both the primary and the secondary databases are included in this particular discipline.

The most important primary databases in this discipline are Ribosomal Database Project (RDP) [77] and miRBase database [78]. These databases provide a computational framework for management of Transcriptomics primary data.

The RDP database has expanded its resources to handle high-throughput data. Also, it provides a set of Open Source tools for custom analysis. Whereas, the miRBase has focused on micro RNA, which play an important role in cellular physiology, development and disease using a negatively regulating gene expression approach [79].

The secondary databases are led by the HMDD [80], DIANA-LncBase [81] and NCIR [82]. The HMDD database is a collection of experimentally supported human microRNA (miRNA) and disease associations. It provides a web interface for users to browse, search and download data sets. Also, user friendly tools are available for submission.

The DIANA-LncBase database provides experimentally verified and computationally predicted microRNA targets on long non-coding RNAs. The miRNA-lncRNA interactions supported by experimental data for both human and mouse species are also available.

DOI:10.15774/PPKE.ITK.2014.011

The NCIR database provides a rapid access to all RNA structures with emphasis in those with base-base interactions reported. Moreover, the database offers a collection of important properties associated to RNA molecules and to their interactions [82].

1.2.3 Metagenomics and Metatranscriptomics

Metagenomics and Metatranscriptomics describe the functional and sequence-based analysis of the collective genomes contained in a sample [83], see Figure 2. They provide a unique opportunity to explore earth's limitless environments harboring scores of yet unknown and mostly unculturable microbes and other organisms [84]. Whereas WGS targets one genome, metagenomics usually targets several. They refer to culture-independent studies of the collective set of genomes of mixed microbial communities and apply to explorations of all microbial genomes in consortia that reside in environmental niches, in plants, or in animal hosts [85].

Metagenomics is a powerful approach for exploring the ecology of complex microbial communities. Its power will be realized when it is integrated with classical ecological approaches and efforts to culture previously unculturable microorganisms, which will likely be facilitated by clues about the physiology of the uncultured microorganisms derived from metagenomic analysis. Microscopy and stable isotope analysis are two approaches that will be particularly informative when linked to metagenomics [86].

Metagenomics and associated meta-strategies have arrived at the forefront of biology primarily because of 2 major developments, the deployment of next-generation sequencing technologies and the emerging appreciation for the importance of complex microbial communities in mammalian biology and in human health and disease.

However, each stage of the analysis suffers heavily due to inherent problems of the metagenomic data generated, including incomplete coverage, massive volumes of raw sequence data produced by the next-generation sequencers, generally short read-lengths, species abundance and diversity and so on [87, 88].

DOI:10.15774/PPKE.ITK.2014.011

The metagenomics assembly problem is confounded by genomic diversity and variable abundance within populations. Assembly reconstructs the most abundant sequences [89]. Simulations indicate high rates of chimera, especially in short contigs assembled from complex mixtures [90]. Studies that rely on characterization of individual reads prefer long reads [91]. The role for de novo genomic assembly from NGS metagenomics data should grow as NGS read lengths and NGS paired-end options increase.

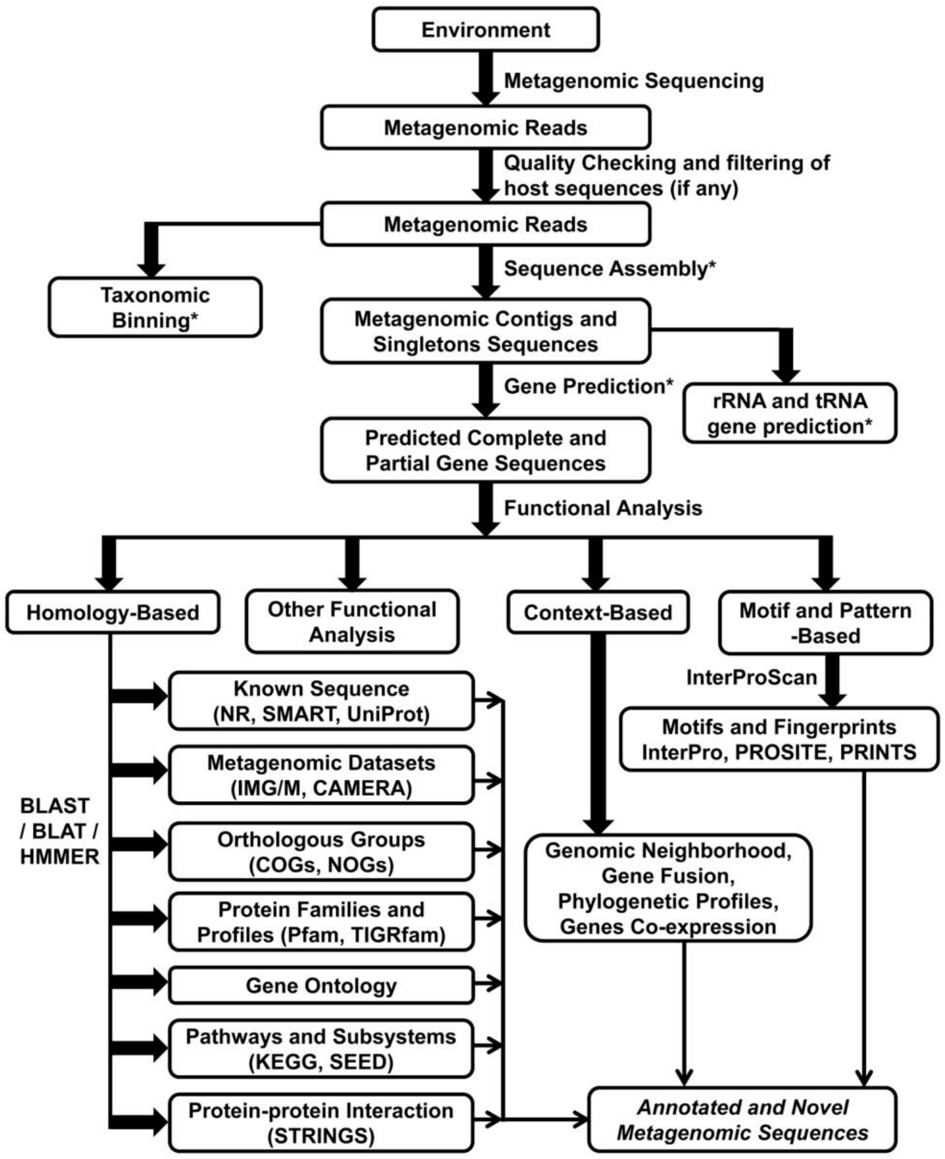

Environment
Metagenomic Sequencing
Metagenomic Reads
Quality Checking and filtering of host sequences (if any)
Metagenomic Reads
Taxonomic Binning*
Sequence Assembly*
Metagenomic Contigs and Singletons Sequences
Gene Prediction*
rRNA and tRNA gene prediction*
Predicted Complete and Partial Gene Sequences
Functional Analysis
Homology-Based
Other Functional Analysis
Context-Based
Motif and Pattern -Based
InterProScan
Motifs and Fingerprints InterPro, PROSITE, PRINTS
BLAST / BLAT / HMMER
Known Sequence (NR, SMART, UniProt)
Metagenomic Datasets (IMG/M, CAMERA)
Orthologous Groups (COGs, NOGs)
Protein Families and Profiles (Pfam, TIGRfam)
Gene Ontology
Pathways and Subsystems (KEGG, SEED)
Protein-protein Interaction (STRINGS)
Genomic Neighborhood, Gene Fusion, Phylogenetic Profiles, Genes Co-expression
Annotated and Novel Metagenomic Sequences

Figure 2: Flow chart for the analysis of a metagenome from sequencing to functional annotation. Only the basic flow of data is shown up to the gene prediction step. For the context-based annotation approach, only the gene neighborhood method has been implemented thus far on metagenomic data sets; although in principal, other approaches which have been used for whole genome analysis can also be implemented and tested [84].

DOI:10.15774/PPKE.ITK.2014.011

1.2.3.1 Taxonomy identification in Metagenomics

One direct application of metagenomics using NGS technologies is the taxonomy identification in unknown samples [92]. This is crucial in various fields, such as detection of human or animal pathogens [93], detection of bacterial contamination in food samples [94] etc. Additionally, the majority of these organisms in environmental samples belong to hitherto unknown taxonomic groups, the challenges is not only just to catalog the known organisms, but also to identify and characterize new organisms belonging to known or unknown taxonomic groups. These organisms could belong to an entirely new species or genus or family or order or class or even a new phylum [95].

This process is known as "Taxonomic binning" and corresponds to the process of assigning a taxonomic identifier to sequence fragments, based on information such as sequence similarity, sequence composition or read coverage [96].

Next generation sequencing is increasingly becoming the method of choice in many areas because of the richness of data it can provide. Moreover, metagenomics produces massive volumes of raw sequence data were NGS technologies are used, so, the processing of NGS data is problematic in many respects. Namely, current sequence alignment problems were developed with genome sequencing in mind; they are optimized for handling a single reference genome (the human genome) on which they work very efficiently.

Current computational approaches for taxonomic binning fall into two broad categories. The first group, marker-based methods seek to bypass the bottleneck via search space reduction, using dedicated, small datasets. A typical example is 16S RNA analysis wherein, a dataset of short sequence items is searched with sensitive alignment techniques, such as BLAST [12]. While this is the traditional standard for taxonomic identification, it has well known limitations, including the need for PCR amplification that introduces extra overhead as well as experimental bias. Alternatively, word-based techniques combined with artificial intelligence can be used to construct a database of clade-specific recognizers that make it possible to use rapid string matching techniques for species identification [97].

DOI:10.15774/PPKE.ITK.2014.011

The MetaPhlAn [98] program uses a small clade-specific sequence marker database using the genome sequences of the known taxa that can be searched with general purpose aligners. This search is extremely fast and accurate for determining taxa and their approximate proportions within large microbial communities. A potential common drawback of marker-based approaches is the frequent lack of lower (e.g. strain-level) taxon identification, as the markers are often identical to many strains. This may cause problems in identifying pathogenic strains of common commonly occurring bacteria such as *E. coli*.

A recently developed program uses a radically different approach, that of compressing sensing [99]. This methodology goes back to a “mixing problem” used in various fields of signal processing and analytical chemistry. Briefly, if pure signals (pure chemical materials) can be described in terms of a vector, than the mixed signal can be described as a linear combination of these vectors, where the coefficients of each vector are proportional to the % of each signal/chemical material in the mixture. For n vectors of m components, we have (n-1) percentage values (coefficients). This problem defines a set of n linear equations, each of them containing m members (vector components).

In the technical life, there are many, sometimes over a hundred years old methods for such problems, least squares fitting is perhaps the best known. The problem requires that the number of measured vector components should be greater than the number of coefficients,. Compressed sensing [100] simply relies to the relatively recent discovery that such equation systems can be solved for problems where the number of number of measured components are low, but the number of equations is also below a certain limit. Metagenome identification is such a problem: From all the possible species, only a few or a few hundred are present.

As vector description, WGSQUIKR uses 7-mer word composition vectors (16 000 components), calculated for entire genomes. The reads of a metagenomics experiments are directly translated into a 7 mer vector, which can be considered as a mixture of pure genomic vectors and the system of equations solved via the methods of compressed sensing.

DOI:10.15774/PPKE.ITK.2014.011

In the second group of metagenome sequencing approaches, whole genome shotgun sequencing reads are directly aligned against a comprehensive sequence database. In this group of approaches database search is a critical step since aligning a large set of reads against a comprehensive database using high quality aligners such as BLAST is either too time consuming, or requires computational resources that that are not readily available for all research groups. A good alternative to BLAST style alignment are the dedicated aligners developed for next generation sequencing such as bowtie2 [101], BWA [102], mrFAST [103] (for a review sew see [7]). These aligners are extremely fast but often require an excessive amount of memory for storing the indexed database, especially when comprehensive sequence databases are used.

A crucial step in all approaches is taxon assignment [96] which is often carried out via various flavors of lowest common ancestor search within a taxonomic hierarchy. Briefly, alignment programs assign reads either to one taxon (say, an *E. coli* strain), or to several taxa (say 100% identity with an *E. coli* strain and an *E. fergusoni* strain), and in the latter case the lowest common taxonomic ancestor (the genus Escherichia) is reported. This principle is used in such popular programs as MEGAN [97, 104], Mothur [105] and SOrt-ITEMS [106].

The variety of computational approaches indicates that there is a need for further computational improvements. The need for dedicated tools is a crucial problem, since most of the current software tools are developed for general research purposes. In research settings, the qualitative and quantitative answers are not always clearly separated. For instance, the presence of *E. coli* reads in an output may be a safe indication for *E. coli* being present, but the number of the identified reads is not necessarily a quantitative measure of the abundance of the species. Currently, MetaPhlAn is considered a reliable quantitative indicator for species abundance in metagenome analysis [98]. Diagnostic settings pose a separate problem: here one has to precisely detect whether or not a pathogen is present above a certain threshold level, but the knowledge of the exact quantity is not necessarily important.

DOI:10.15774/PPKE.ITK.2014.011

1.2.4 Proteomics

Proteomics is the discipline to study the large-scale set of proteins, particularly their structures and functions. It is based on the study of the Proteome which describe the cellular stage or the external conditions of the cell. The Proteome analysis is an essential tool in the understanding of regulated biological systems [107]. It can be used to compare cellular stages in order to determine the molecular mechanism that are involves in a specific cellular process. Additionally, there is a great interest in the Proteomics due to the fact that the majority of the pharmacological targets are proteins [108].

In current science, proteomics is almost exclusively used for a well defined field where mass spectrometry is used for the analysis of complex protein or peptide mixtures. Namely, mass spectrometry coupled with high performance liquid chromatography has become the de facto experimental standard for the proteomic analysis of complex biological materials such as tissue samples, biofluids, immunoprecipitates etc [109]. Each sample produces many thousand spectra, so the interpretation of LC-MS/MS relies entirely on computational tools [110]. The field of mass spectrometry is very complex, so, an overview of the topics closely related to bioinformatics analysis used in routine analysis of biological samples will be outlined.

Proteomics databases are widely spread on internet and include the most heterogeneous biological data. Moreover, they have more tools associated covering a huge range of applications than any other biological database.

The most important databases for this discipline are UniProt [111], RefSeq [112], RCSB PDB [113] and the protein DB from the Ensembl project, already described [75]. The first two databases are primary databases and include the protein sequences and its primary descriptions. The last one is a metabase that integrates several kinds of biological data and computational tools.

The UniProt database provides freely accessible resource of protein sequences and functional annotations. This database has two sections: a reviewed section containing manually annotated records with information extracted from literature and curator-

DOI:10.15774/PPKE.ITK.2014.011

evaluated computational analysis (UniProtKB/Swiss-Prot), and an unreviewed section with automatically annotated records (UniProtKB/TrEMBL) [68]. The proportion of reviewed entries varies between proteomes, and is obviously greater for the proteomes of intensively curated model organisms.

The RefSeq database integrates an organism’s genomic, transcript and protein sequence with descriptive feature annotation and bibliographic information. It is build from sequence data available in public archival sequence databases of the International Nucleotide Sequence Database Collaboration. Unique features of the RefSeq collection include its broad taxonomic scope, reduced redundancy, informative cross-links between nucleic acid and protein records [112].

The Research Collaboratory for Structural Bioinformatics Protein Data Bank (RCSB PDB) provides a structural view of biology for research and education. The online PDB archive is a repository for the coordinates and related information for more than 84 000 entries, including proteins, nucleic acids and large macromolecular complexes that have been determined using X-ray crystallography, NMR and electron microscopy techniques [113, 114]. This database includes cross-references with UniProt databases for a close relationship between the protein sequence and the structure.

The next group of databases is secondary databases which provide proteins properties, domains, families and motifs.

The first group of databases to be mentioned is the protein clusters. These databases provide data sets of proteins clustered or grouped by sequence similarity or any other feature. The most representative database of this group is UniRef [115]. It provides clustered sets of sequences from the UniProt database sequences. Currently covering 44 million source sequences, the UniRef100 combines identical sequences and subfragments from any source organism into a single UniRef entry. UniRef90 and UniRef50 are built by clustering UniRef100 sequences at the 90 or 50% sequence identity levels.

DOI:10.15774/PPKE.ITK.2014.011

Alike, the protein families databases are led by the Pfam database [116]. This database includes a curated set of protein families, each of which is defined by two alignments and a profile hidden Markov. It contains 14 831 manually curated entries.

The next group of databases is protein orthologs. Orthology refers to a homologous relationship resulting from a speciation event, as opposed to paralogy, which is the result of a gene duplication event [117]. The first database in this group is the COG [118]. This database includes one-to-many and many-to-many orthologous relationships in form of clusters. Each COG consists of individual orthologous genes or orthologous groups of paralogs from three or more phylogenetic lineages. The COG database lacks phylogenetic resolution and is not regularly updated due to the manual labor required. However, its groups are still used by other databases to classify proteins. As an extension of COG, the eggNOG database was created [119]. It can be updated without the requirement for manual curation, covers more genes and genomes than COG, contains a hierarchy of orthologous groups to balance phylogenetic coverage and resolution and provides automatic function annotation of similar quality to that obtained through manual inspection [120].

1.2.5 Metabolomics/metabonomics

These two OMICS disciplines are one of the essential parts of the Systems Biology approach. They are used for the study of metabolism which makes life possible and is one of the most complex processes in nature [121].

The discipline of metabolomics seeks to identify the complete set of metabolites, or the Metabolome, of the cell. The related metabonomics field specifically studies the dynamic metabolic response of living systems to environmental stimuli or genetic perturbation [122]. The Metabolome represents the output that results from the cellular integration of the Transcriptome, Proteome and Interactome [5]. The Interactome is the whole set of molecular interaction. It includes the protein-DNA, protein-protein and protein-metabolites interactions dictating many cellular processes [123].

DOI:10.15774/PPKE.ITK.2014.011

There are several secondary databases and metabase systems related with the two OMICS disciplines. The first group is related to the Interactome and includes the databases for molecular interactions. These databases are led by BioGrid [124, 125], IntAct [126, 127], MINT [128, 129] and DIP [130]. All these databases provide protein-DNA, protein-protein and protein-metabolites interactions with a high redundancy between them.

The metabase systems are led by KEGG [73], Reactome [131], BioCyc and MetaCyc [132, 133]. All these systems provide collections of pathway databases with multiple tools for data analysis. The tools are online resources specialized on pathway analysis and visualization. The main differences between them are related with the level of curated and predicted data included into the database. The KEGG and Reactome resources offer a more manually curated set of data than the other systems but, at the same time, the BioCyc and MetaCyc predicted data are really useful in multiple scenarios.

1.2.6 General databases

The general databases are resources which are not generated by the OMICS disciplines but they are used by them. The taxonomy and ontology databases are included in this extra category. These are useful databases that are used to standardize the species names and terms used by the biological databases.

The main taxonomy database is the NCBI Taxonomy [134]. This database provides nomenclature and classification for the source organisms in the biological databases. Its taxonomy identifier (taxid) is widely used by all other databases to identify the source organism of the biological data.

Finally, the Gene Ontology (GO) database is a community-based bioinformatics resource that classifies gene product function through the use of structured, controlled vocabularies. It serves as a comprehensive source of functional information on gene products and descriptions of functions through the use of domain-specific ontologies. The GO terms follow a hierarchical organization that is widely used by other biological databases [135].

DOI:10.15774/PPKE.ITK.2014.011

1.3 Integration of biological data

The computer assisted data integration is an important field of Bioinformatics. It can be defined as the problem of combining data residing at different sources, and providing the user with a unified view of these data [136, 137]. The integration of biological data is essential to understand and analyze the function of a biological entity; therefore, it is essential for the Systems Biology. In addition, it can be used as an effective mechanism of data validation reducing the false positives produced by the experimental technologies [138, 139] and to know the extended biological context of a biological entity, including its relationships with other elements [140]. However, this is one of the most open subjects on Bioinformatics [141] because there is not a definitive solution for the integration of the biological databases [142].

Data integration is one of the oldest themes in computer science that takes another face when it is dealing with biological data. It is perhaps one of the most challenging tasks today due to the nature of the data itself. Even though, it is well known its importance for the Systems Biology and, in general, for the biological sciences. This integration faces a lot of complexities that go beyond of a simple integration process. The fast accumulation of data and its eventual modification are only one side of the problem. The heterogeneity and redundancy are also an important part of this complexity.

The problem of integrating biological data sources has several aspects none of which have been solved with the available tools [143-146]. The most important aspects are: (i) a large collection of interrelated heterogeneous data that are connected through internet, (ii) they are distributed using various file formats and delivery systems [147, 148], and finally, (iii) there is not a unique solution for data integration because it depends on the biological or application context in which it will be used.

There are several active biological databases which share a high level of redundancy or semantic equivalent data. Each database has its own Web-based interface and its own schema and access formats. Therefore, the number of available biological databases may befog the integration process. This number can be reduced if the biological or application

DOI:10.15774/PPKE.ITK.2014.011

context is used to address the task [149]. Additionally, the interoperability between them, which depends on the use of the unique ID keys, can make the integration process difficult due to inconsistencies between databases version and updates.

Moreover, the biological data are often reformulated in some fashion or worst; they may have a short lifetime and can be neglected after new experimental evidence. Therefore, the integration process has to be an ongoing process closely related to the data generation process.

This volatile property of the data may have terrible consequences if the context is not taken into account. For example, the simple way to model the data about two proteins which interact in a metabolic pathway can be just the proteins' IDs and the metabolic pathway's name. However, the protein-interacting region can be added to the model if more details are needed. After new experimental evidences, the protein-interacting region changes, and the database have to be updated. This update would affect the solution where the protein-interacting region is included and not the simple solution aforementioned. Both solutions are practical but their use will depend on the context.

The simple solution would be enough for metabolic network construction and the extended solution would be useful for virtual experiments on drug design. In the last case, changing the protein-interacting region could affect seriously the experiment result. This example shows how the application context can reduce the complexity of the integration process.

Hence, the general task of data source integration can be seen to consist of five conceptual tasks: (i) transformation from heterogeneous data model to a global model, (ii) semantic schema matching, (iii) schema integration, (iv) data transformation to the global schema and (v) comparison and identification of semantic equivalent data [150].

Traditionally, biological database integration efforts are classified into three main classes [151]: federated, mediated and warehouse-style integration. Federated integration, (sometimes termed portal, navigational or link integration) provides hyperlinks to join data; early examples include SRS [152] and Entrez [71]. The Semantic Web and linked data are a

DOI:10.15774/PPKE.ITK.2014.011

more recent approaches that use the World Wide Web to create typed links between data from different resources [153]. With the federated approach, it is relatively easy to provide up to date information but extra care is required to maintain the links. On the other hand, mediated integration, also called view integration provides a unified query interface and collects the results from various data sources. DiscoveryLink [154], BioMediator [155], BioMoby [156] are good examples of this approach.

Finally, warehouse databases integrate data sources in one place include Biowarehouse [157], Biozon [158], Atlas [159], EnsMart [160] and IGD [161] [157, 159, 162-164]. This approach provides fast querying over joined data sets, but also requires continuous updating.

1.3.1 Heterogeneity and redundancy in biological data

The inherent heterogeneity of the biological data, which arises in many forms, ranging from the hardware and software platform that a database system is based on, to the data model and schema used to provide logical structure for the stored data, to the various kinds of data and information that are being stored. This heterogeneity of the data is a problem which one faces during the data integration process [136].

The heterogeneity can be found at different levels of the biological data organization.

Data heterogeneity

- Naming differences: Two objects that represent the same concept are named differently, e.g. ProteinID or UniProtID.
- Semantic differences: Two objects that represent the same concept are described differently, e.g. proteins in UniProt [111] and proteins in DrugBank [165] represent the same concept but they have a complete different set of properties and attributes.
- Content differences: Two databases both contain gene data, but one also contains gene functions using the COG [166] and the other does not.

DOI:10.15774/PPKE.ITK.2014.011

- Format differences: Data elements use different formats to represent the same concept, using numeric id in Entrez Gene [71] for the Gene ID vs. strings on KEGG Genes [167].

Database heterogeneity

- Schematic dissimilarity: The relationships among the entities are defined differently.
- Query language dissimilarity. Two databases with similar content do not share a common query language. Individual sources provide their own user-access interface, all of which a user must learn in order to retrieve information that is likely spread across several sources. Additionally, the sources often allow for only certain types of queries to be asked, thereby protecting and preventing direct access to their data.
- Format dissimilarity: The databases are distributed using different file format and release systems.

The heterogeneity has been faced using methods that seek to unify the biological databases through the imposition of external structures. The uses of ontologies and vocabulary standardization schemes have been also used.

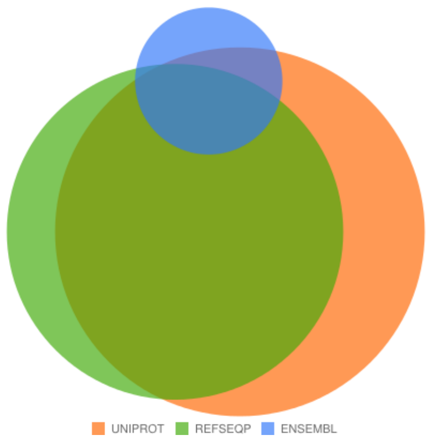


Figure 3: Overlap between protein databases. Figure from ProgMap [1] server.

The redundancy is another problem to face. For example, the UniProt, RefSeq and Ensembl databases are highly redundant among them [1] but, at the same time, they include sets of unique proteins, see Figure 3.

This redundancy increases when large-scale automatic processes are being used for the data

DOI:10.15774/PPKE.ITK.2014.011

generation and, as a result, the database management becomes more difficult and time-consuming. Although, there have been several attempts to remove excessive redundancy inside a database [168], the redundancy between different databases remain uncontrollable.

1.3.2 The format of biological data

The biological data comes in multiple kinds of formats. The majorities of the biological databases provide flat files in text with their specific formats or non-standardized tabular text files. Also, the new experimental technologies are providing a huge number of different files formats [169]. The lack of standardization in the biological data obstructs the data exchange and process by external researchers. Multiple parsers and compilers have been written by decades to extract the biological data from the flat files representing a waste of resources and time.

Inconsistency on key definitions, internal errors, lack of standardization and computer readable file structures, files with gigabytes of size in text format without any methods to verify the file integrity are the most common problems faced during the work with the biological data.

In the last years, multiple initiatives have introduced community standards to the biological data files. They are trying to provide standard computer libraries, in several computer languages, for data manipulation. The eXtensible Markup Language (XML) [170] file format have been adopted by many databases [171] due to its benefits on data definition and integrity, also, because it is widely used by multiple computer systems. This file format allows the definitions of your own markup language keeping a standard file structure. Furthermore, multiple international initiatives have been developed to offer standard definitions for the biological data files. As an example, the Proteomics Standards Initiative (PSI) [172] have been working to define community standards for data representation in proteomics to facilitate data comparison, exchange and verification.

DOI:10.15774/PPKE.ITK.2014.011

1.4 Scope

Several Bioinformatics problems have been discussed during this introduction. Although, they may seem independents there is a strong relationship among them. First, all of them need an organized and structured data supplier system. Also, optimized, fast and reliable computers programs to process the data are required.

This thesis concentrates on developing three approaches to solve some of the aforementioned problems. First, a system named **Taxoner**, designed for prediction of bacterial taxa and gene function from NGS results is presented. Then, a workflow, based on the Taxoner programs and the **JBioWH** framework, is used as a DNA marker databases generator or just for DNA sequence comparison. Finally, **JBioWH,** a framework for biological data integration is presented.

All the software presented in this thesis are open-source and freely available through their respective web sites.

DOI:10.15774/PPKE.ITK.2014.011

2 Materials and Methods

During the course of these projects, many tools, pipelines and computational environments have been developed using various programming languages and third party libraries.

A workstation computer (two Intel Core CPU Q6850 (3.00 GHz and 4.0 MB of cache) and 8.0 GB of RAM) with GNU/Linux operating system (kernel version 3.11.10-11-desktop), specifically OpenSuSE distribution version 13.1, was used for developing and tests the programs.

The Google Cloud Platform [173] was used for computations which require a high performance computing using virtual machines *in-house* modified to create a virtual Beowulf like Cluster [174] inside the cloud platform.

2.1 Database designs

Relational schemas were developed using standard SQL language. The Database Management System used was MySQL community server [175] version 5.6.

The MySQL Workbench Tool [176] was used for design and visualization of the relational schemas. All databases developed have an associated MySQL Workbench project freely available for users.

2.2 Programming languages and libraries

2.2.1 Java

Oracle Java Standard Edition (SE) [177] version 7 was used for the Java programs. Netbeans IDE [178] was used as integrated development environment for writing the Java codes.

All the source code generated using Java languages are freely available through the projects web sites using their respective version control systems. The source code building process is executed by the Maven tool [179]. This is a software project management and comprehension tool. It is used to manage a projects building, reporting and documentation from a central location.

Several Java technologies and libraries have been used in the projects. The most important technologies used are EclipseLink [180] for the Java Persistence Model (JPA), Red Hat JBoss Middleware [181] for the webservices development, JGraph [182] for graph computing and visualization, Mojarra JavaServer Faces [183] using the Java Server Faces (JSF) technology and Primefaces [184] as JSF component for web interfaces.

2.2.2 C

The ANSI C language was used for programs with a high computing demand. The GCC compiler for GNU/Linux operating systems was used as compiler for the C code. The projects use the Make program to build the executables and libraries using the well-known Makefile.

The parallelism in the programs was implemented using the POSIX Thread [185] library. This library allows our programs to take advantage of the multicores and multiprocessors architectures available today.

Additionally, the MPICH [186] library, an implementation of the Message Passing Interface (MPI) paradigms [187], was used to extend the parallelism of our programs from the multicores and multiprocessor architectures to the cluster platforms where the communication between the nodes is preformed through computer networks.

2.2.2.1 The BioC library and tools

Nowadays, working with “mountains” of data is a common task in Bioinformatics. Frequently, high level programming languages like Perl, Python or even Java are not fast enough to process data efficiently. For instance, a simply random access to entries of a fasta of size 50 GB becomes prohibitively resource consuming, even in high performance computational environments like the Google Cloud Platform. For facing this problem, there arose a necessity to develop some utilities on low-level programming language.

As a result, highly optimized and fast programs were developed to for dealing with demanding tasks in terms of computational power and highly loaded data access. The **BioC** library is a freely available C project comprised of eight modules. These modules are structured in the form

DOI:10.15774/PPKE.ITK.2014.011

of two major groups; the first group contains the basic data structures and functions to process error, memory, time and string objects. Full documentation is freely available in https://code.google.com/p/bioc/wiki/bioc.

The main idea of boosting access time is loading the data into custom data structures and indexing them using B+ Trees [188].

B+ tree ("bee plus tree") is a data structure used as an index to facilitate fast access to the elements of a larger body of data, such as the entries in a database. Each target object (entry) is associated with an index key. The B+ tree is laid out like a family tree, where each node has some number of keys that is between some predetermined maximum limit and half that limit (inclusive). Each node also has one more pointer than the number of its keys. (A "pointer" is the address of a location in memory.) You can picture the node as having alternating pointers and keys, starting and ending with pointers.

At the bottom level of the B+ tree are the leaves. Each pointer on the leaf except the last (rightmost) one points to the data object whose key stands immediately to the right of that pointer. The rightmost pointer points to the next leaf over to the right. Then, each bunch of leaves has its own parent node. If there are enough of these parents, then they, in turn, are divided into bunches, each of which shares but one parent — and this one-parent-many-children family tree goes all the way up to a single ancestor at the top, the root. The internal nodes, which are the parents, grandparents, etc., of the leaf nodes, also have keys, which are (initially) duplicates of some of the keys on the leaves. A given internal node's keys are "representative" copies of a few of the keys to be found on the leaves that are the (ultimate) descendants of that node. The pointers on the internal nodes point to nodes at the next level down on the tree, which may be leaves or other internal nodes.

Our B+ Tree implementation exploits two kinds of keys, integer based keys and string based keys. Both approaches can index any kind of object without data duplication. These kinds of indexes can be used to create an index of offset positions in a fasta file using the entries’ GI as keys, see full description of data structures and functions in https://code.google.com/p/bioc/wiki/btreeString. Index catalogue is created from a fast sequential

DOI:10.15774/PPKE.ITK.2014.011

reading of the file where the sequences are excluded and only the headers are processed. The entry's offset position is used to create the B+ Tree which takes the GI as keys.

The second group has two Bioinformatics modules, the **fasta** and the **taxonomy** modules. These modules were implemented in Object Oriented Design fashion, in order to encapsulate the functions/methods available for each data structure and hide internal functions. This is a useful approach that increases the readability and reusability of the code offering a safe environment for developing, see Table 4. All the data structures developed using this approach have two standard functions, the first one named ***free*** used to release the memory occupied by the object and the second named ***toString*** to print the object.

Table 4: Object oriented design for the C structures.

Data structure definition in the .h file	Function access from the object in .c file

```
struct fasta_s {
        /**
         * Set the fasta header
         *
         * @param self the container object
         * @param string the header
         */
        void (*setHeader)(void *self, char *string);
};
typedef struct fasta_s *fasta_l;
```
	```
	// Header definition
	char *header = "the fasta header";

	// Creation of the fasta_l object
	fasta_l myObj = CreateFasta();

	// Setting the fasta header
	myObj->setHeader(myObj, header);
	```

The **fasta** module is designed to process fasta files. It has a ***fasta_l*** data structure with methods to manipulate the sequence and the header. Also, the module has independent functions for creating, reading and writing B+ Tree indices from fasta files allowing a fast and random access to the entries. See https://code.google.com/p/bioc/wiki/fasta for full description of the fasta module.

The second module of this group is named taxonomy and it is designed to load the **NCBI Taxonomy** database using B+ Tree index for fast retrieval of the taxonomy entries. Several Bioinformatics experiments need a taxa organization and/or classification according to their lineage or rank. This module provides a fast way of using the taxa information in C programs. See https://code.google.com/p/bioc/wiki/taxonomy for full description of the taxonomy module.

BioC project provides a set of tools for making use of the concepts and paradigms described above. They are briefly summarized in Table 5.

DOI:10.15774/PPKE.ITK.2014.011

Table 5: Tools available in the BioC Project.

Tool	**Description**	**BioC modules used**
BuildBtreeIndexFasta	Build a binary index file for a fasta file	B+ Tree, Fasta
SplitFastaFile	Split a fasta file by reads (overlapped or not)	B+ Tree, Fasta
TaxLineageFromGi	Print the taxonomy lineage from GenBank Gi	B+ Tree, Taxonomy
TaxLineageFromTaxId	Print the taxonomy lineage from TaxId	B+ Tree, Taxonomy
TaxonerAssamblerMarkerDB	A basic assembler program for the Taxoner output using overlapped reads	B+ Tree, Fasta, Taxonomy

2.2.3 Bash shell scripting

Bash shell is a command processor script language widely used by the Linux users to create small programs that can be used to integrate complex workflows and pipeline processes. This scripting language was used in our projects to create automatics scripts for retrieval, manipulations and transformations of data files.

Also, several scripts were developed for the Google Cloud administration and configuration offering an automatic workflow for running computational experiments on the Cloud.

2.3 Google Code Projects

The Google Code Platform [189] is a free project hosting service that provides a free collaborative development environment for open source projects. The source codes developed are included in Projects that are freely available using the Google Code hosting services.

This platform offers to each project a version control system. In our projects we are using the Apache Subversion System (SVN) [190], to manages files and directories, and the changes made to them, over time. Also, an easy Wiki pages maker is available for publishing the project documentation developed by the programmers.

Table 6: List of Google Code Projects developed.

Name	**URL**	**Language**
Taxoner	http://code.google.com/p/taxoner/	C
JBioWH	http://code.google.com/p/jbiowh/	Java
BioC	http://code.google.com/p/bioc/	C

The Table 6 shows the list of projects published on the Google Code Platform and the programming languages used on them.

DOI:10.15774/PPKE.ITK.2014.011

2.4 Cloud Platforms

Google Cloud Platform provides both a fully managed platform and flexible virtual machines, allowing the users to choose a system that meets the needs of their own projects. This platform is built on the same infrastructure that allows Google to return billions of search results in milliseconds.

For our projects, a virtual Beowulf like cluster was implemented on the Google Cloud Platform. The virtual machines provides by Google were modified to have a server virtual machine (*head*) and multiple computing clients (*nodes*).

The *head* virtual machine runs a network file system (NFS) server. The NFS server is a distributed file system protocol that allows a client computer to access files over a network much like local storage is accessed. Also, a network information system (NIS) is installed on the *head*. The NIS is a client-server directory service protocol for distributing system configuration data such as user names and host names, between computers on a computer network.

Additionally, the head includes the Torque Resource Manager that is a distributed resource manager providing control over batch jobs and distributed computer *nodes*. This server allows the creation of queues to organize and manage the jobs executed by the users.

Virtual cluster also includes a directory named *progs* distributed by the NFS server with the most important programs for Bioinformatics installed in it. This directory allows the usage of the virtual cluster for any Bioinformatics projects.

Two scripts where developed to start the *head* virtual machine remotely from our workstations and to create, start and delete the client *nodes*. These scripts, developed in bash, use the Google Cloud ***gcutil*** utility to interact with the cloud platform. Using this platform we can easily expand or reduce our virtual cluster depending on our needs. The *nodes* are created from a snapshot image and they are inserted in the cluster environment automatically.

DOI:10.15774/PPKE.ITK.2014.011

3 Results and Summary

3.1 Biological Database integration

3.1.1 General approach

Biological database integration is one of the most challenging topics on Bioinformatics. As we mentioned above, the integration of biological data has to be an ongoing project due to the nature of the data itself. The data heterogeneity and the redundancy are important problems to be solved during an integration process. Also, the biological context has to be taken into account in order to reduce the integration domain. Multiple the systems have been developed to integrate biological data using different kinds of approaches. However, as we specified in the Introduction, a number of specific challenges still remain.

We present here a data integration framework developed using Java environment, usage of which is not restricted to it. **J**ava **BioW**are**h**ouse (**JBioWH**) is an open-source framework developed in Java which offers to the Bioinformatics community a set of computational tools designed for a wide range of users. Users with advanced programming skills in Java, SQL and Webservices can find **JBioWH** a useful framework to work with biological data. At the same time, users without programming skills can use the **JBioWH** multiplatform desktop client to perform complex queries to the integrated data included in the system.

The framework is freely available in https://code.google.com/p/jbiowh/. This site also offers full documentation and multiple examples, figures and tables. Also, a Google Group called ***jbiowh-discuss*** (https://groups.google.com/forum/#!forum/jbiowh-discuss) is active for posting questions and ideas; it keeps an open channel between the system's users and our developer group. Additionally, a MySQL server, designed for demonstration purposes, is available at: hydra.icgeb.trieste.it:3307 and a Webservices site available at: http://net.icgeb.org/jbiowh-webservices/.

Our goal was to design a flexible system which would be used in multiple scenarios. Its modular design allows the creation of context dependent integrated databases where the database size and

DOI:10.15774/PPKE.ITK.2014.011

the databases to integrate can be fully manipulated by the user. This made the **JBioWH** completely different from the available solutions due to its versatility. It can be used in personal computers with a low level of resources, in big servers for high demanding tasks or simply in the Cloud through the Google Cloud Platform [191].

Five interrelated components sketch out the **JBioWH** architecture: (i) the data sources, (ii) the relational schema and database, (iii) Java API, (iv) Desktop client and (v) Webservices. This architecture and the relationship between the components can be seen on Figure 4, further description of these components will be outline in the next subchapters.

3.1.2 Data sources

JBioWH contains data retrieved from 24 databases, see Table 7. The framework provides a Java command line tool for fetching the data from their own providers and insertion of data into the JBioWH relational database (top of the Figure 4). This process includes the data transformation from their own file formats to internals TSV files which are inserted into MySQL database.

This command line tool is able to work with the data locally or remote. Also, compressed files can be handled in order to reduce network transfer and data storage.

The loading times depend on the size of the data to be inserted. Small databases like NCBI Taxonomy [134], Gene Ontology [135], DrugBank [165] and OMIM [192] can be loaded in a few minutes. However, big databases like Gene [193], UniRef [115] and KEGG [73] may take multiple hours. The worst case is the UniProt TREMBL which can take until 3 days on a personal computer. The Table 8 shows the database original file size, the final MySQL file size, the loading times and the number of elements inserted.

Figure 5 shows loading times against the total number of elements inserted into the MySQL database. For small and medium databases the loading time scales linearly. However, three databases do not obey this tendency: Draft Genomes (48 084 s), KEGG (50 416 s) and UniProt TREMBL (253 304 s). Draft Genomes and the KEGG databases include multiple files, so the process of opening and closing of these files create a long delay which is not the case for other databases. The case of the UniProt TREMBL is the other exception. This database includes only

DOI:10.15774/PPKE.ITK.2014.011

one file of 21 **GB** but it contains 1x10^9 elements to insert. Leaving out the outliers and fitting a linear equation to the rest we can find a correlation coefficient of **0.98**. The Figure 6 shows the graph for this linear correlation and the linear equation itself.

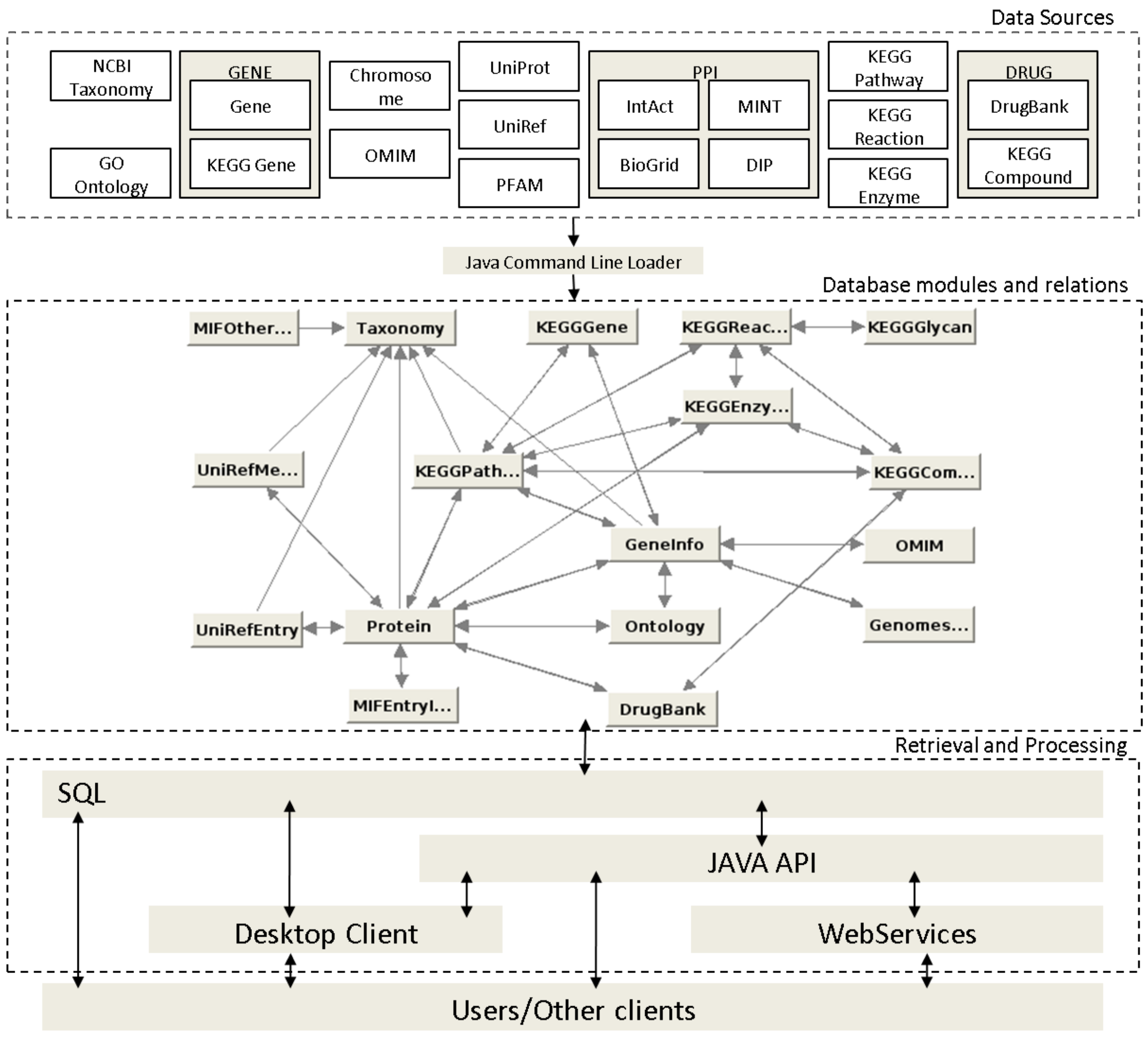


Figure 4: The JBioWH architecture and the relationship between the components

DOI:10.15774/PPKE.ITK.2014.011

Table 7: Data sources included in JBioWH

Data Type	Data Source	URL	Data Format
Taxonomy	NCBI Taxonomy	ftp://ftp.ncbi.nih.gov/pub/taxonomy/taxdump.tar.gz	Delim. Text
Ontology	GO	ftp://ftp.geneontology.org/pub/go/godatabase/archive/latest-full/	OBO XML
Gene	Gene	ftp://ftp.ncbi.nlm.nih.gov/gene/DATA/	Delim. Text
Gene	KEGG Gene	http://www.bioinformatics.jp/en/keggftp.html	Text
Gene	GenBank	ftp://ftp.ncbi.nih.gov/genbank	Text
Gene	RefSeq	ftp://ftp.ncbi.nih.gov/refseq/release/	Text
Chromosome	Genomes	ftp://ftp.ncbi.nih.gov/genomes/	Delim. Text
Protein	UniProt	ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/complete/	XML
Enzyme	KEGG Enzyme	http://www.bioinformatics.jp/en/keggftp.html	Text
PPI	IntAct	ftp://ftp.ebi.ac.uk/pub/databases/intact/current/psi25/pmidMIF25.zip	PSI 25 XML
PPI	MINT	ftp://mint.bio.uniroma2.it/pub/release/psi/current/psi25/pmids/	PSI 25 XML
PPI	DIP	http://dip.doe-mbi.ucla.edu/dip/	PSI 25 XML
PPI	BioGrid	http://thebiogrid.org/	PSI 25 XML
Prot. Cluster	UniRef	ftp://ftp.uniprot.org/pub/databases/uniprot/uniref/uniref100/	XML
Drug	DrugBank	http://www.drugbank.ca/system/downloads/current/drugbank.xml.zip	XML
Drug	KEGG Comp.	http://www.bioinformatics.jp/en/keggftp.html	Text
Pathway	KEGG Pathway	http://www.bioinformatics.jp/en/keggftp.html	Text
Reaction	KEGG Reaction	http://www.bioinformatics.jp/en/keggftp.html	Text
Disease	OMIM	http://www.omim.org/downloads	Text
Prot. Domain	PFAM	ftp://ftp.sanger.ac.uk/pub/databases/Pfam/releases/Pfam26.0/database_files/	SQL
Prot. Groups	COG	ftp://ftp.ncbi.nih.gov/pub/COG/	Text
Prot. Groups	eggNOG	http://eggnog.embl.de/version_4.0.beta/downloads.v4.html	Text
Prot. Groups	NCBI Prot Cluster	ftp://ftp.ncbi.nih.gov/genomes/CLUSTERS/	Text
Prot. Groups	PirSF	ftp://ftp.pir.georgetown.edu/databases/pirsf/	Text

The databases were accessed in May 2014.

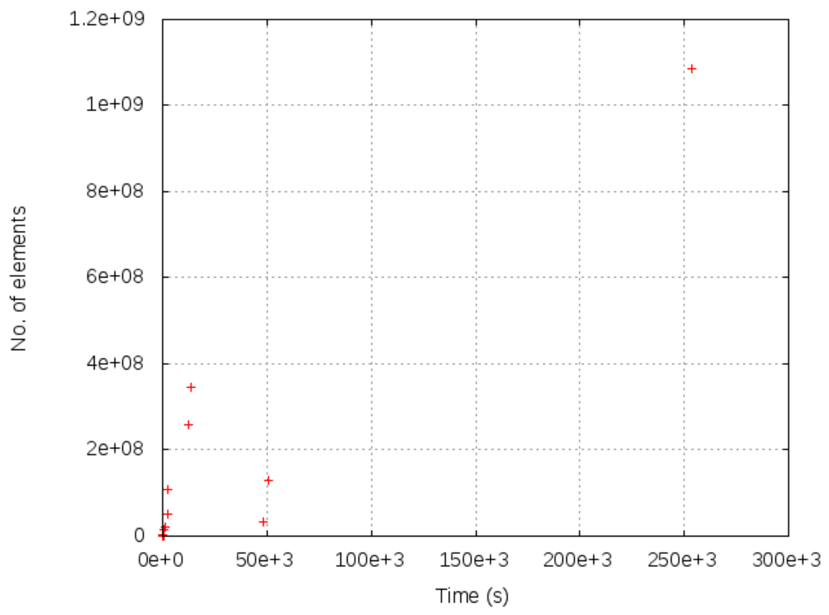


Figure 5: Loading times against the number of elements inserted. The outliners are the Draft Genomes (48 084 s), KEGG (50 416) and UniProt TREMBL (253 304 s).

DOI:10.15774/PPKE.ITK.2014.011

Table 8: Loading times, final data sizes and number of elements inserted. This example was executed in a computer with two Intel Core CPU Q6850 (3.00 GHz and 4.0 MB of cache) and 8.0 GB of RAM running OpenSuSE 13.1 (Linux version 3.11.10-11-desktop.

DB	Download Date	File size (MB)	MySQL size (MB)	Load time (s)	No. main elements	No. elements	Total elements
GO Ontology	3/22/2013	4	70	16	39 118	473 599	512 717
DrugBank	3/23/2013	90	143	24	6 711	397 847	404 558
DIP	3/22/2013	164	280	90	1	1 833 668	1 833 669
NCBI Taxonomy	3/22/2013	159	547	143	1 002 774	2 778 537	3 781 311
MINT	3/22/2013	990	15 466	308	1 400	4 394 018	4 395 418
OMIM	3/23/2013	163	332	381	22 811	879 509	902 320
IntAct	3/22/2013	2 757	2 579	494	6 234	14 851 480	14 857 714
BioGrid	3/22/2013	2 598	2 053	652	1	15 583 825	15 583 826
Complete Genomes	3/25/2013	577	2 055	1 132	7 301 020	14 603 906	21 904 926
UniProt (SwissProt)	3/22/2013	737	4 228	2 185	539 616	51 168 667	51 708 283
Gene	3/22/2013	5 619	14 801	2 484	11 402 702	97 450 383	108 853 085
PFAM	3/28/2013	31 000	20 501	12 668	14 831	259 648 666	259 663 497
UniRef 90	3/25/2013	4 379	25 787	13 606	13 613 286	332 379 407	345 992 693
Draft Genomes	3/11/2013	3 130	6 143	48 084	17 166 166	17 168 429	34 334 595
KEGG	6/11/2011	40 668	14 784	50 416	126 478	130 469 935	130 596 413
UniProt (TREMBL)	3/5/2013	21 953	131 352	253 304	29 266 939	1 056 770 378	1 086 037 317
Total (GB and Hours)		112.29	235.47	107.22	80 510 088	2 000 852 254	2 081 362 342

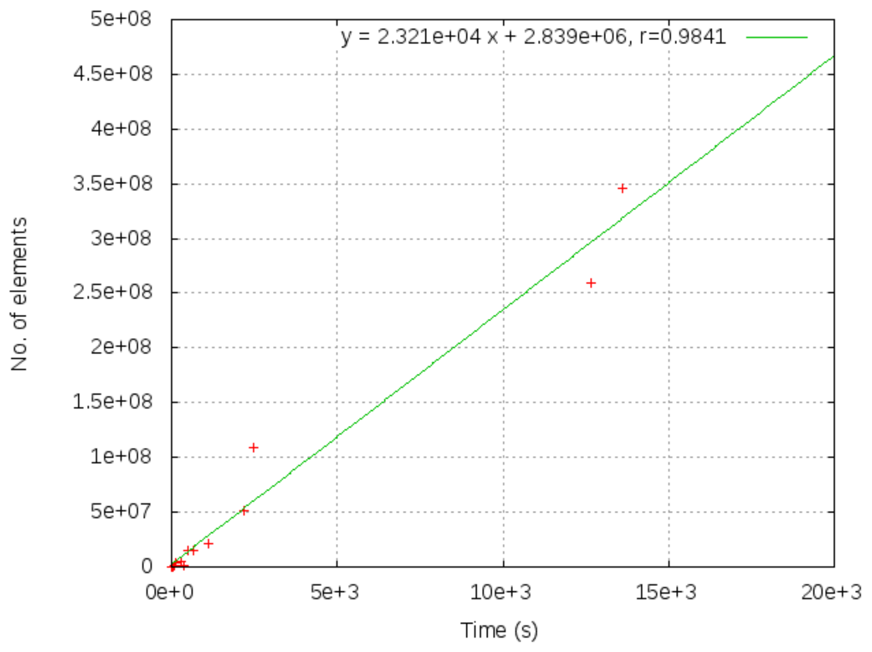


Figure 6: Linear correlation between the loading times and the number of elements inserted. The outliers are left out

DOI:10.15774/PPKE.ITK.2014.011

3.1.3 Relational schema

For the integrated database, we designed a relational database scheme in SQL. The relational scheme contains 414 tables including auxiliary tables for cross-references and for speeding up the ‘join’ operations. Figure 7 shows the main tables and their relationships.

Additionally, the **JBioWH** relational schema is designed to store **all** the information that comes from biological databases. This means that JBioWH can be treated as a relational representation of each of biological databases, without any missing features. This is possible because the original sequential data representation in terms of large flat files was transformed into representation in terms of object relations, which we call as biological objects.

While the storage of all the information that comes from the original sources might seem as a waste of space and resources, it gives **JBioWH** framework extra features. First, it can be used as a data supplier to other applications, allowing a fast data retrieval process to integrated data. Second, context dedicated views of the relational schema can be generated without adding new data to the relational database.

The schema can be described through databases groups. These groups organize the **JBioWH’s** database modules taking into account the level of cross-references of the biological databases. There are three groups: (i) the basement group which includes the aforementioned general databases (NCBI Taxonomy and Gene Ontology) that does not included references from any other database, (ii) the primary group which includes primary databases like GenBank and UniProt, and (iii) the secondary group which include secondary databases and metabases.

On the other hand, two informative tables are included into the relational schema which does not hold biological data. The first one is the **DataSet** table which is used to store the biological data sources information. This table stores information such as the data source name, version, release date, the user who performed the insertion etc. Also, a status field is included which is used to store to current status of the database, *e. g.* status: ***created*** that is used when the database is created for the first time and is under the insertion process, ***updating*** that is used when the database is being updated and ***inserted*** that is used when the ***created*** or ***updating*** processes ended successfully. The identifier of this table is used as a foreign key in the main tables of the different modules to keep the track of the biological objects.

DOI:10.15774/PPKE.ITK.2014.011

The second informative table is named **WIDTable**. This table is used to keep the last global identifier used by the system. The WID identifier, that is explained bellow, is a global unique number that is used as a surrogate primary key to identify every database entry.

3.1.3.1 Naming convention and global surrogate primary key

Additionally, the naming convention is used to organize the table names offering a direct way to identify which module the table belongs. Each database module uses the name of its main table as a prefix for the other module's tables. For instance, in the Taxonomy module the main table is called **Taxonomy** and all other tables of this module are named using the prefix Taxonomy (the table which stores the synonyms is named: **TaxonomySynonym**).

A surrogate primary key named **WID** is generated and is used to identify the biological objects inserted into the database. This is a global unique number that can be used to identify all the entries inserted into the database. Also, it is used to create cross-references between the entries. The next available number (which is the last number plus one) is stored in the **WIDTable** aforementioned.

Many biological databases use strings as unique identifiers instead of integers. As it is well known, the usage of strings as identifiers in the SQL databases increase the index size and introduce some delay during the 'join' operations. The **JBioWH**'s relational schema uses name conventions which include the creation of **WID**, which is not a descriptive value, in each table as internal identifier and surrogate primary key in order to reach the best possible performance. This internal identifier will be hidden from users keeping the original identifier (the one that come with the biological database) as the main reference for users. The WID field will be only available if the **JBioWH** is used from a computer client application or directly in the SQL environment.

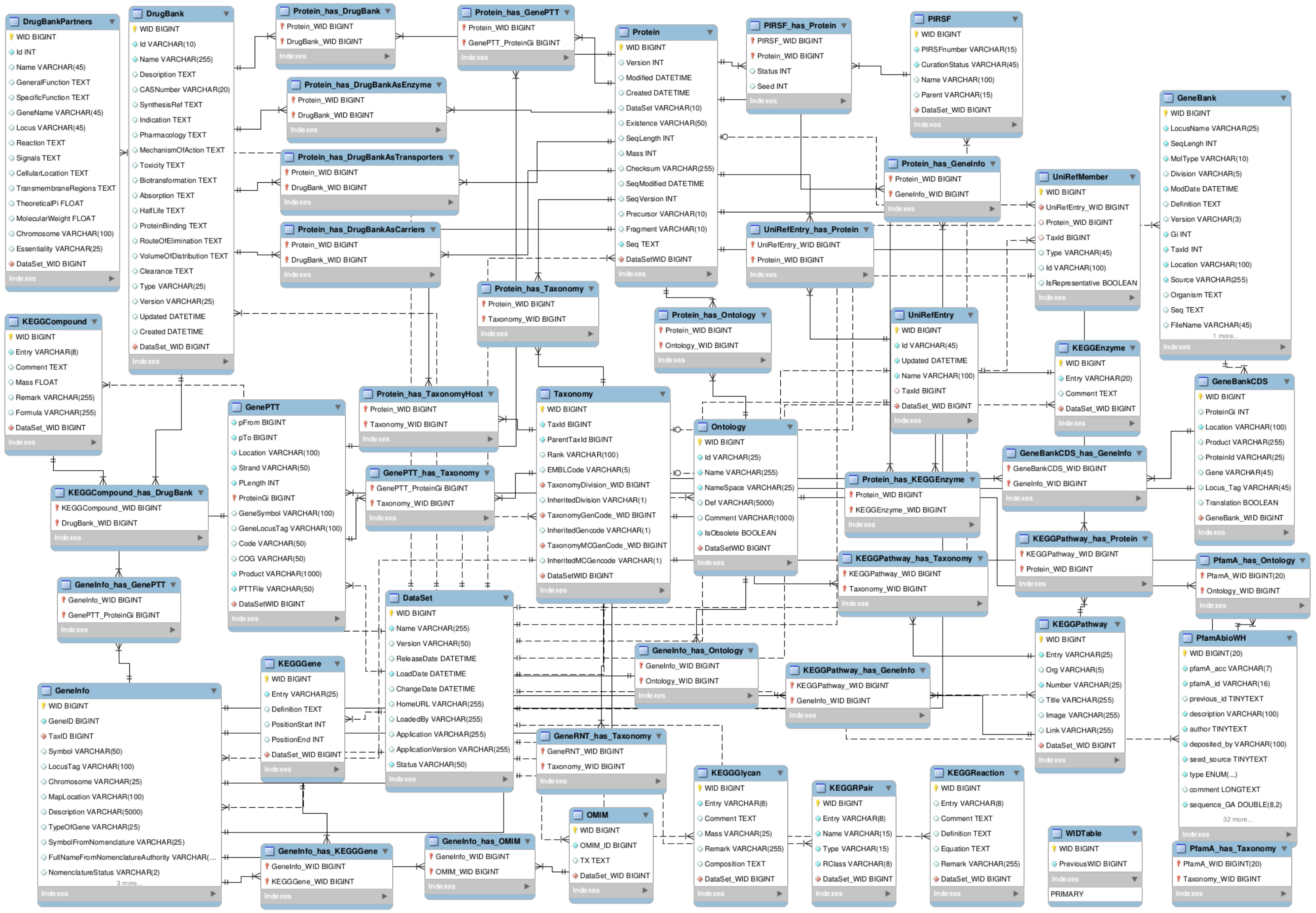


Figure 7: JBioWH relational schema with the main tables and their relationship.

DOI:10.15774/PPKE.ITK.2014.011

The fields which are foreign keys follow a similar convention; they use the foreign table's name as a prefix followed by the underscore "_" character and the foreign field's name, *e. g.* the **Taxonomy** table has a field named **WID** as surrogate primary key and the **TaxonomySynonym** table has a field named **Taxonomy_WID** as a foreign key between the **Taxonomy** and the **TaxonomySynonym** tables.

The **many-to-many** relationship tables are also included in the naming conventions. The tables use as prefix the name of the table which owns the relationship and at the same time defines to which module the table belongs to. For instance, the table named **Protein_has_Ontology** represents the relationship between the proteins and the ontology terms, *i. e.* one protein can *have* multiple ontology terms and one ontology term can be *used* by multiple proteins. This table belongs to the Protein module and will remain empty if the proteins are not inserted into the database.

3.1.3.2 Basement group

The basement group is formed by the NCBI Taxonomy and Gene Ontology databases. These databases do not include cross-references to any other biological database but they are cited by all of them.

The Taxonomy module includes the NCBI Taxonomy database. This database is used to organize the biological data through a curated taxonomy classification and nomenclature of all biological organisms. It is cited by almost all of the biological databases through the **TaxId** identifier. This is a unique numerical field used to identify all biological organisms. The module includes eight tables plus six temporal tables that are used during the insertion process.

The Ontology module includes the Gene Ontology database. The module includes eighteen tables in total with 5 temporal tables. This database offers the possibility to use well defined terms to classify the biological objects. The GO database use a string identifier for its terms, therefore the **WID** is inserted in the main module (**Ontology** table) as surrogate primary key. This key is used in the cross-references as the naming convention specify.

DOI:10.15774/PPKE.ITK.2014.011

3.1.3.3 Primary group

The primary group includes modules for primary databases. These modules are Protein, GenBank, and the Gene.

The Protein module follows the UniProt XML file format structure and contains all database data including the proteins sequences. It includes sixty tables in total; eleven are used for many-to-many relations and fourteen are temporary tables. This is a highly cross-reference module but also it contains multiple cross-references to other databases. One important characteristic of the UniProt database is that it includes a wide and well curated list of cross-references. This module splits the Unipart's external database references table (**ProteinDBReference**) into multiple tables creating small tables for each external database that is cited by UniProt, *e. g.* the modules includes tables like ProteinGO, ProteinGene, ProteinDrugBank, etc. Those tables are used to store the original cross-reference data that comes in UniProt. Additionally, the relational schema includes relationship tables like **Protein_has_Ontology**, **Protein_has_GeneInfo** and **Protein_has_DrugBank** which are used to store the cross-reference between the databases using the **JBioWH**'s surrogate global primary key and are created from the before mentioned tables.

The GenBank module is based on the GBK file format from the NCBI. This module includes nine tables in total; one of them is used to store the relation between the GenBank CDS and the Gene information. The relational schema stores the nucleotide sequences offering a fast delivery system for multiple applications. This module can be used not only for the GenBank database but also it can be used to manage any database using the GBK file format. For instance, this module can be used for the RefSeq database offering a relational schema for a more concise database than the GenBank, although their usage will depend on the biological context.

The last database included in this group is not a primary database but it contains the gene data. This database is the NCBI Gene database that is cited by multiple databases. It acts as the bridge between GenBank and UniProt data. This module includes thirty nine tables in total with five many-to-many relation tables and eighteen temporary tables.

DOI:10.15774/PPKE.ITK.2014.011

The relationships between these modules are through the Gene module. The GenBank CDS are linked to the genes and the genes are linked to the proteins. Additionally, they are linked to the basement group through bilateral relations with the tables on that group. The Figure 8 shows the relations between these modules.

As can be seen in the figure, the modules are designed in a circular reference way. This design is useful because it allows join queries that can start in any table of the circle but it may introduce extra complications from the programming point of view. Extra care has to be taken on the programming side to handle this circular reference and the data integrity during insertion or update processes.

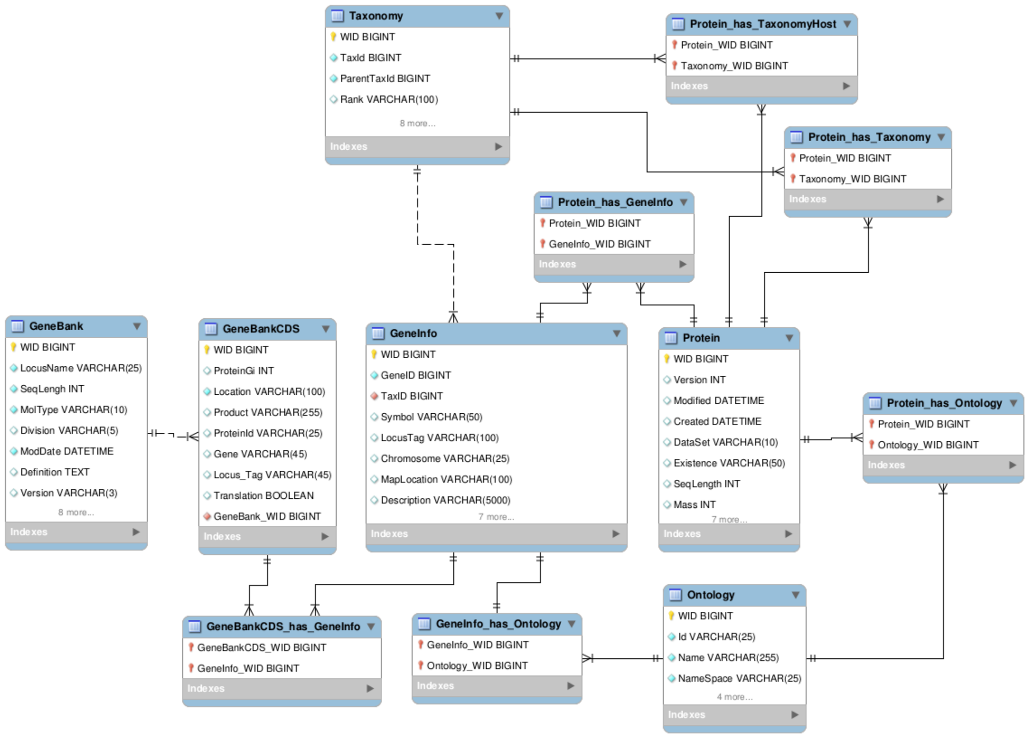


Figure 8: The relations between the Basement and first group of modules

DOI:10.15774/PPKE.ITK.2014.011

3.1.3.4 Secondary group

The secondary group is formed by the rest of modules included into the **JBioWH** relational schema. This group includes the Chromosome, Protein-Protein interaction, Protein clusters, Drugs, Pathway, Disease, Protein families and Protein Groups modules.

The chromosome module is based on the Genomes database of the NCBI and it is closely related to the Gene database. The modules is based on the data of the PTT and RNT file formats offering the possibility to work with genes' positions in the chromosomes. The module includes five tables and it is linked to the Gene, Protein and Taxonomy modules.

The Protein-Protein interaction module is based on the MIF25 file format. The module includes forty two tables with one temporary table. It is linked to the Protein module. This module can store the data provided by BioGrid, IntAct, MINT, DIP and any other database providing MIF25 file format. This module can be used for protein network reconstruction.

Protein cluster module is based on the UniRef database and it is connected to Protein and Taxonomy modules. It includes eight tables with one temporary table. The UniRef database includes 3 kinds of protein clusters, 50, 90 and 100 % of identity. The module can handle all three files but they will be inserted as independent datasets.

The Drugs module is based on the DrugBank database and contains forty seven tables. It is connected to the Protein and Pathway modules. This module has the particularity of included proteins which have different biological meanings than the proteins in the UniProt database. Consequently, there are some redundancy between the Drug and the Protein modules. The **Target** object in DrugBank is a protein but with an associated biological role. Therefore, the biological object is not the protein sequence and its attributes but also, its biological role. These biological roles are mapped to the Protein module using many-to-many relationship tables like: **Protein_has_Drugbank** for the proteins who act as targets, **Protein_has_DrugBankAsCarrier**s for the proteins which act as carriers, the **Protein_has_DrugBankAsEnzyme** for proteins which act as enzymes and **Protein_has_DrugBankAsTransporters** for proteins which act as transporters. This approach allows joining extra information that comes with DrugBank to all the useful information included in UniProt.

The Pathway module is based on the KEGG database and includes 84 tables. This is one of the most cross-reference modules in the relational schema; it is linked to Gene, Protein, DrugBank and Taxonomy. It has a hierarchical design that starts with **KEGGCompounds**, **KEGGGenes**, **KEGGEnzymes** and **KEGGReactions** blocks. Then, all of them are linked through the **KEGGPathway** block. The **KEGGPathway** is graph based structure where the nodes are the **KEGGEnzymes** and the edges are the **KEGGReactions**. This module can be used for metabolic pathway reconstruction.

Disease module is based on the OMIM database. The module includes 22 tables with 3 temporary tables. This module is linked to the Gene module.

The next module is the Protein domain. This module is based on PFAM database and includes 39 tables with 3 temporal tables. The tables of this module are based on the SQL files provided by PFAM.

Finally, the last module is the Protein Groups module. It contains four sub-modules based on the following databases: PirSF, COG/eggNOG, NCBI Protein Cluster and the OrthoXML [194] file format.

All these sub-modules provide orthologous groups for genes and proteins. We designed the Protein Groups module as the addition of the tables included by the four sub-modules mentioned. The approaches followed by these databases to generate the orthologous groups are completely different and they can be used according to the biological context of the experiments.

The PirSF based sub-module includes 3 tables. The module is linked to the Protein module and it is based on the “pirsfinfo.dat” file. The Orthologs group’s members provided by PirSF use the UniProt Accession Number for the protein identification. The cross-reference in **JBioWH** is recreated using the internal **WID** in order to avoid the usage of strings for cross-references.

DOI:10.15774/PPKE.ITK.2014.011

The COG/eggNOG sub-module includes 10 tables with 2 temporary tables. This module is linked to the modules: Taxonomy, Gene and Protein. Although the COG database is no longer updated, it is included into the **JBioWH** because several databases and bioinformatics tools still use the COG classification of functional groups. Also, the module was designed to store the eggNOG database using the same **JBioWH** relational schema as the COG database. As we aforementioned, the eggNOG database is a continuity of the COG database keeping the same organizational schema of functional categories and groups.

The Protein Cluster module includes 9 tables and it is linked to the Taxonomy, Gene and Protein.

The OrthoXML file format based sub-module includes 10 tables. This sub-module is not based in a particular database; it can handle any database which provides the data using the OrthoXML file format. This file format is designed broadly to allow the storage and comparison of orthology data from any ortholog database. It establishes a structure for describing orthology relationships while still allowing flexibility for database-specific information to be encapsulated in the same format. This file format is used by the OMA [195, 196] and ProGMap [1] among others.

3.1.3.5 Cross-references

The cross-references in the **JBioWH** relational schema are **one-to-one**, **one-to-many** and **many-to-many** relations. All of them are created using the cross-references that come with the biological databases.

There are two cases of cross-reference between the databases. The first one is when the cross-reference is in one direction, *e. g.* one database includes a cross-reference of another database that doesn't have any reference to the first one. In this case the owner database of the relationship is the one that includes the cross-reference and the relation table will be included into its **JBioWH** module. Namely, we use the Gene Ontologies references included in the UniProt entries. However, the Gene Ontology database does not contain any

DOI:10.15774/PPKE.ITK.2014.011

reference; so, the Protein module has a relation table named **Protein_has_Ontology** that is the **many-to-many** relationship between the Proteins and the ontology terms.

The second case of cross-reference is when both databases are cited among them. In this case the module which will be the relationship owner is selected according to the biological meaning of the data. As an example we can see the cross-references between UniProt and Gene databases. The final relationships are generated using the data provided by two databases. They are converted to the internal **JBioWH**’s **WID** and then, the redundant ones are eliminated. This procedure allows us to create multiple relationships that are not available using the databases independently.

3.1.4 Java API

The **Java API** has been designed for maintaining the relational database and querying the data. It is implemented using the Oracle Java SE 7 [177] and the classes were designed according to the standard Java design patterns [197]. The usage of commonly known design patterns from the field of object oriented software design in software development leads to standardization of the code, making it easier to understand and use for other programmers or users.

The API has three principal libraries, **jbiowh-core**, **jbiowh-dbms** and **jbiowh-persistence** and five Java applications, **jbiowh-parser**, **jbiowh-desktop**, **jbiowh-webservices**, **jbiowh-webservices-client** and **jbiowh-tools**.

This is an ongoing work that is used by several external projects to our lab. Also, the framework is indexed by multiple search engines like the Ohloh metasite (http://www.ohloh.net/p/jbiowh/).

The Figure 9 shows the number of lines committed per time. This graph shows that **JBioWH** have been growing during the time and is in continues development.

DOI:10.15774/PPKE.ITK.2014.011

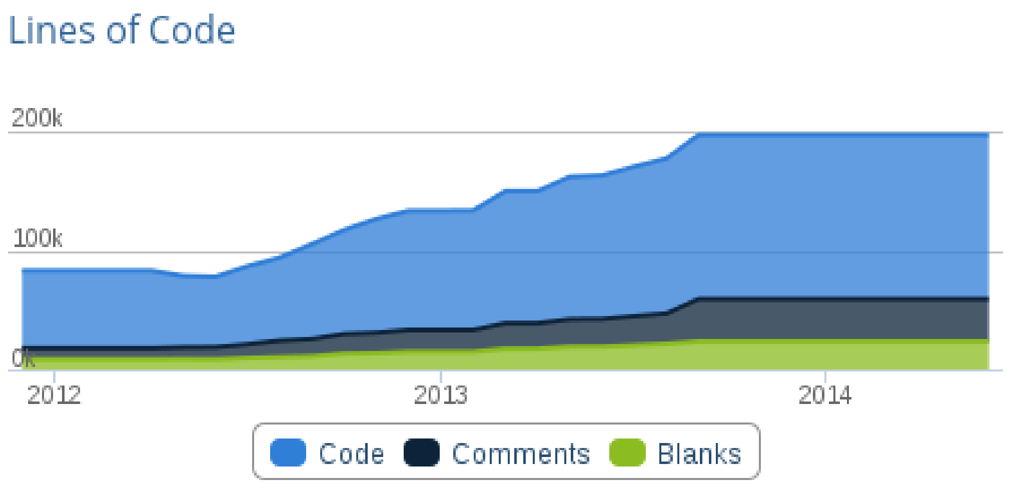
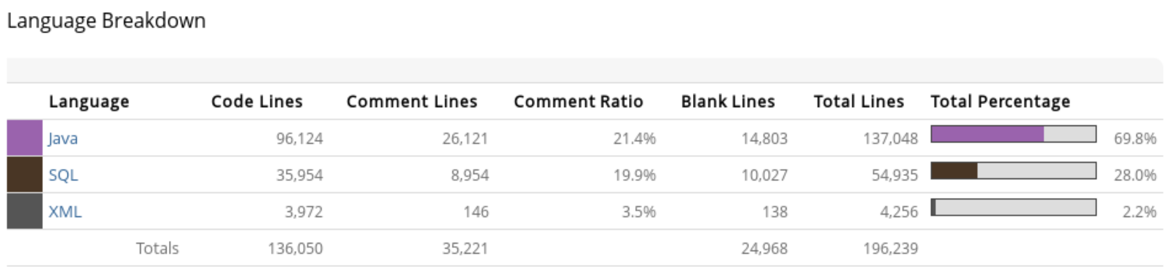


Figure 9: The number of code lines included into the JBioWH framework. Statistics taken from http://www.ohloh.net/p/jbiowh/

The next Figure 10 shows the statistics for the **JBioWH** source code.



Language Breakdown

	Language	Code Lines	Comment Lines	Comment Ratio	Blank Lines	Total Lines	Total Percentage
	Java	96,124	26,121	21.4%	14,803	137,048	69.8%
	SQL	35,954	8,954	19.9%	10,027	54,935	28.0%
	XML	3,972	146	3.5%	138	4,256	2.2%
	Totals	136,050	35,221		24,968	196,239	

Figure 10: Number of code lines in the JBioWH framework. Statistics taken from http://www.ohloh.net/p/jbiowh/

3.1.4.1 The Core library

The first library includes all basic classes of the **JBioWH** framework. It also has other useful classes like **JBioWHUserData** class which is used to store the User information, the **VerbLogger** class used as the logger system and the **fileformats** package.

In addition, this package provides the classes for parsing some files used by **JBioWH** framework. The formats supported are the BIOpolyer Markup Language (bioml) [198], the NCBI XML Blast [12], fasta [199] and the Structured Query Language (SQL).

DOI:10.15774/PPKE.ITK.2014.011

3.1.4.2 The DBMS library

The second library comprises of classes which are used to manage the Database Management Systems (DBMS). This library includes an interface named **JBioWHDBMS** which is used to specify the available methods to interact with the DBMS. This class is implemented by the **WHMySQL** class that extends the aforementioned **JBioWHUserData** class, see Figure 11. The **WHMySQL** class is used to communicate the JBioWH framework with the MySQL DBMS [175].

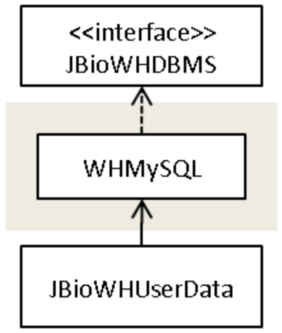


Figure 11: The functionality to manage the DBMS.

3.1.4.3 The Persistence library

The last library applies the Java Persistence Model (JPA) for the **JBioWH** relational schema. This library operates the classes to handle the database entities in accordance with the object-oriented model described by the JPA approach. Each entity maps the relationship between the biological objects giving total interconnectivity between the objects within the Java code. The core classes use the EclipseLink [180] library to map object-oriented models onto the relational database tables in the back-end through the **JBioWHPersistence** singleton class.

Additionally, this library employs the **JBioWHSearch** interface to provide the methods to search over the JBioWH relational schema. Also, an abstract class named **SearchFactory** is provided here. This abstract class includes the basic modules that will be used by the class that implement the JBioWHSearch class, see Figure 12.

DOI:10.15774/PPKE.ITK.2014.011

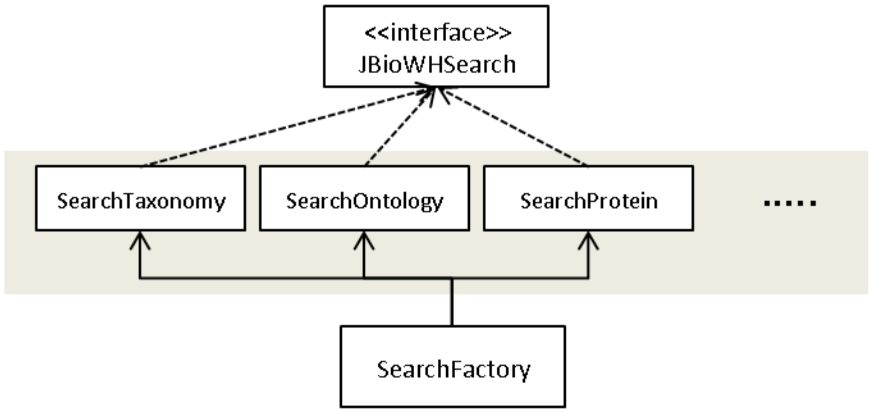


Figure 12: The structure of the Search functionality. The interface JBioWHSearch is implemented by the modules search classes that extends the SearchFactory abstract class.

3.1.4.4 The Parser application

The **jbiowh-parser** application comprises of classes which are used for reading the data from the data sources (parsers) and inserting into the relational schema. This application can process the data sources locally (previously copied by the user) or remotely through an HTTP or FTP server (the data sources servers providers).

A **JBioWHParser** interface is used by the parsers which also extend the **ParserFactory** class as shown in the Figure 13. This application includes java packages for each module with classes that implement the **JBioWHParser** interface and also extends the **ParserFactory** class.

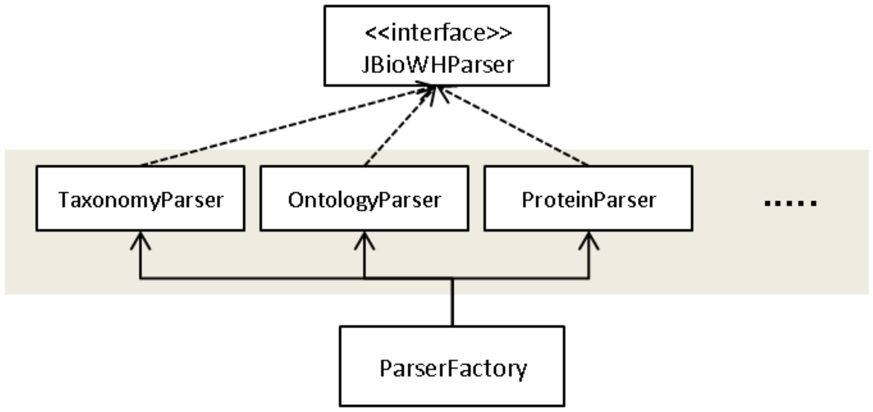


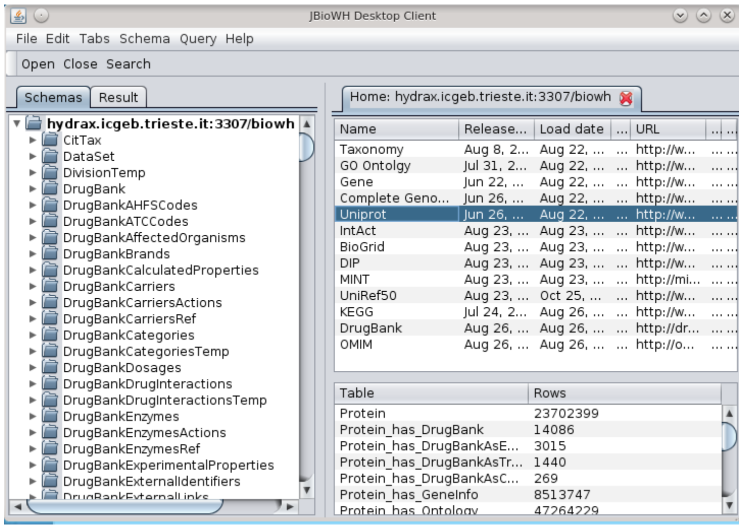
Figure 13: The JBioWH parser structure.

DOI:10.15774/PPKE.ITK.2014.011

Each module implements their own methods to connect to the source databases, retrieve data in flat file format, parse and put the data into the relational database using the **jbiowh-persistence** and **jbiowh-dbms** libraries. Modules can be loaded or unloaded independently from each other; hence, unnecessary data can be easily omitted to save storage space and to speed up query execution. The biological databases are often large and their syntax is often poorly defined. This problem frequently causes failures while loading the data in the available integration frameworks, or worse, it can corrupt the biological data themselves. The loader functions of **JBioWH** were designed to preserve the biological data so that the loader process stops in case of errors in the database format or structure.

3.1.4.5 The Desktop application

The **JBioWH** Desktop Client application (**jbiowh-desktop**) has been developed for users who are not familiar with SQL scripting or the Java programming languages. The client application provides a graphical interface to access, manipulate and execute complex queries by simple mouse clicking from the integrated database (the desktop client is illustrated in Figure 14).


JBioWH Desktop Client
File Edit Tabs Schema Query Help
Open Close Search
Schemas Result
Home: hydrax.icgeb.trieste.it:3307/biowh

hydrax.icgeb.trieste.it:3307/biowh
- CitTax
- DataSet
- DivisionTemp
- DrugBank
- DrugBankAHFSCodes
- DrugBankATCCodes
- DrugBankAffectedOrganisms
- DrugBankBrands
- DrugBankCalculatedProperties
- DrugBankCarriers
- DrugBankCarriersActions
- DrugBankCarriersRef
- DrugBankCategories
- DrugBankCategoriesTemp
- DrugBankDosages
- DrugBankDrugInteractions
- DrugBankDrugInteractionsTemp
- DrugBankEnzymes
- DrugBankEnzymesActions
- DrugBankEnzymesRef
- DrugBankExperimentalProperties
- DrugBankExternalIdentifiers
- DrugBankExternalLinks

Name	Release...	Load date	...	URL	...	...
Taxonomy	Aug 8, 2...	Aug 22, ...	...	http://w...	...	...
GO Ontolgy	Jul 31, 2...	Aug 22, ...	...	http://w...	...	...
Gene	Jun 22, ...	Aug 22, ...	...	http://w...	...	...
Complete Geno...	Jun 26, ...	Aug 22, ...	...	http://w...	...	...
Uniprot	Jun 26, ...	Aug 22, ...	...	http://w...	...	...
IntAct	Aug 23, ...	Aug 23, ...	...	http://w...	...	...
BioGrid	Aug 23, ...	Aug 23, ...	...	http://w...	...	...
DIP	Aug 23, ...	Aug 23, ...	...	http://w...	...	...
MINT	Aug 23, ...	Aug 23, ...	...	http://mi...	...	...
UniRef50	Aug 23, ...	Oct 25, ...	...	http://w...	...	...
KEGG	Jul 24, 2...	Aug 26, ...	...	http://w...	...	...
DrugBank	Aug 26, ...	Aug 26, ...	...	http://dr...	...	...
OMIM	Aug 26, ...	Aug 26, ...	...	http://o...	...	...

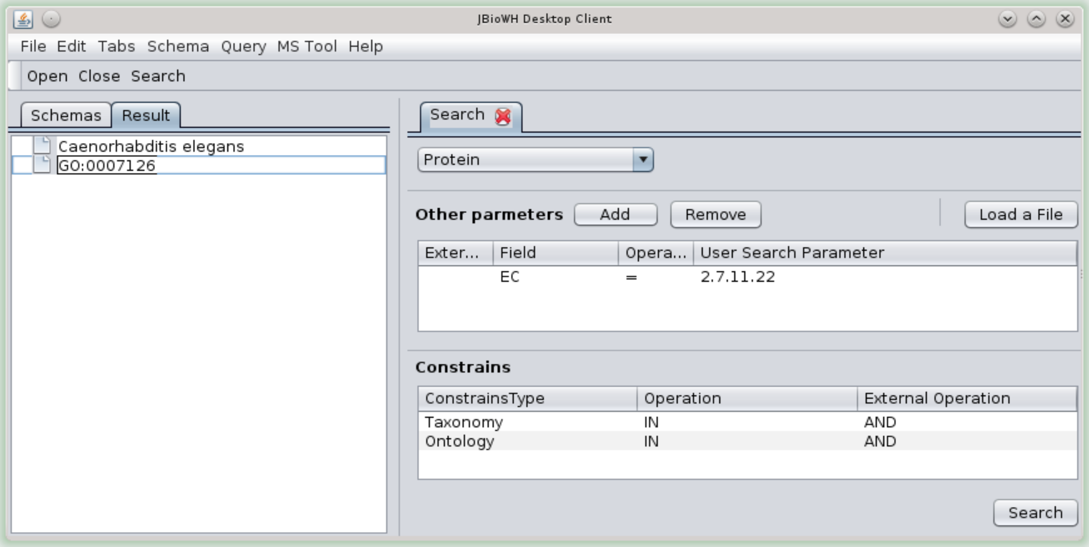
Table	Rows
Protein	23702399
Protein_has_DrugBank	14086
Protein_has_DrugBankAsE...	3015
Protein_has_DrugBankAsTr...	1440
Protein_has_DrugBankAsC...	269
Protein_has_GeneInfo	8513747
Protein_has_Ontology	47264229

Figure 14: A screenshot of JBioWH Desktop Client. The left panel shows the relational schemes opened. The top right panel shows the list of the database inserted in the relational scheme, while on the bottom left panel one can see the tables in the selected database.

DOI:10.15774/PPKE.ITK.2014.011

Next example shows a query over the Protein, Taxonomy and Ontology modules performed by the user. The query is to find the proteins which have the EC (Enzyme Id) equal to 2.7.11.22 belonging to the Taxonomy ***Caenorhabditis elegans*** species and to the Ontology term: ***GO:0007126*** (***Meiosis***). The result is shown with a tab named ***CDK1_CAEEL*** which is the name of the protein.

The application has a temporary working space, the tab ***Result***, where the user can store intermediate results that will be used in further queries as shown in the Figure 15. In this example, two queries are executed to retrieve the Taxonomy and Ontology objects that are used in the final query as Constrains.



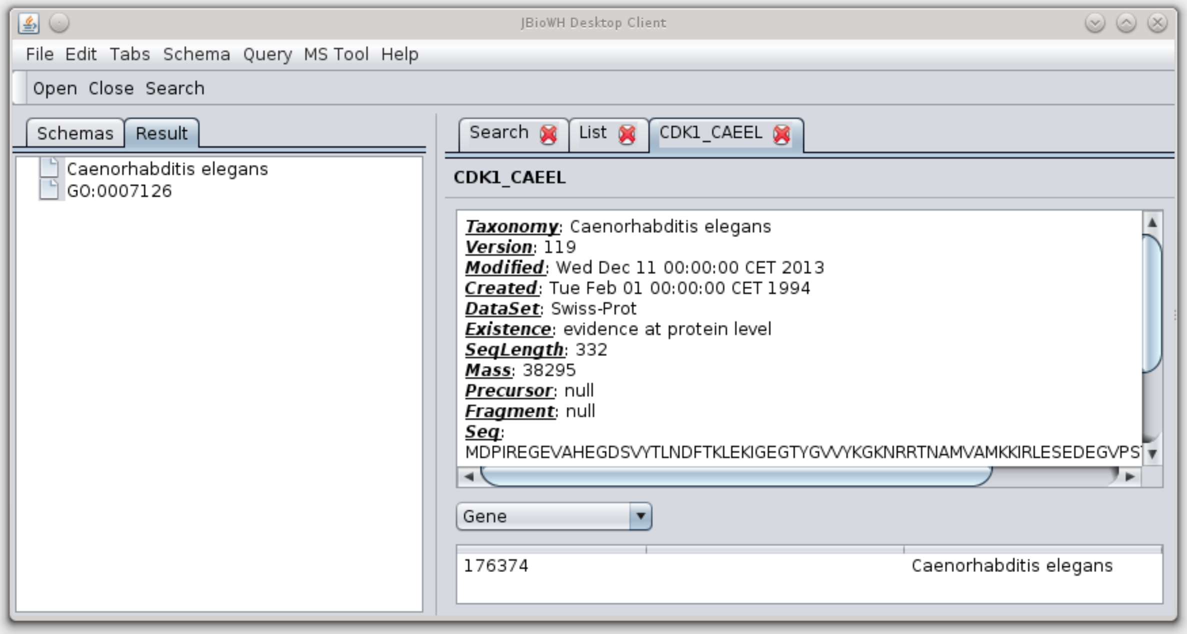
Exter...	Field	Opera...	User Search Parameter
	EC	=	2.7.11.22

Constrains

ConstrainsType	Operation	External Operation
Taxonomy	IN	AND
Ontology	IN	AND

Figure 15: The search interface with constrains.

DOI:10.15774/PPKE.ITK.2014.011



Caenorhabditis elegans
GO:0007126

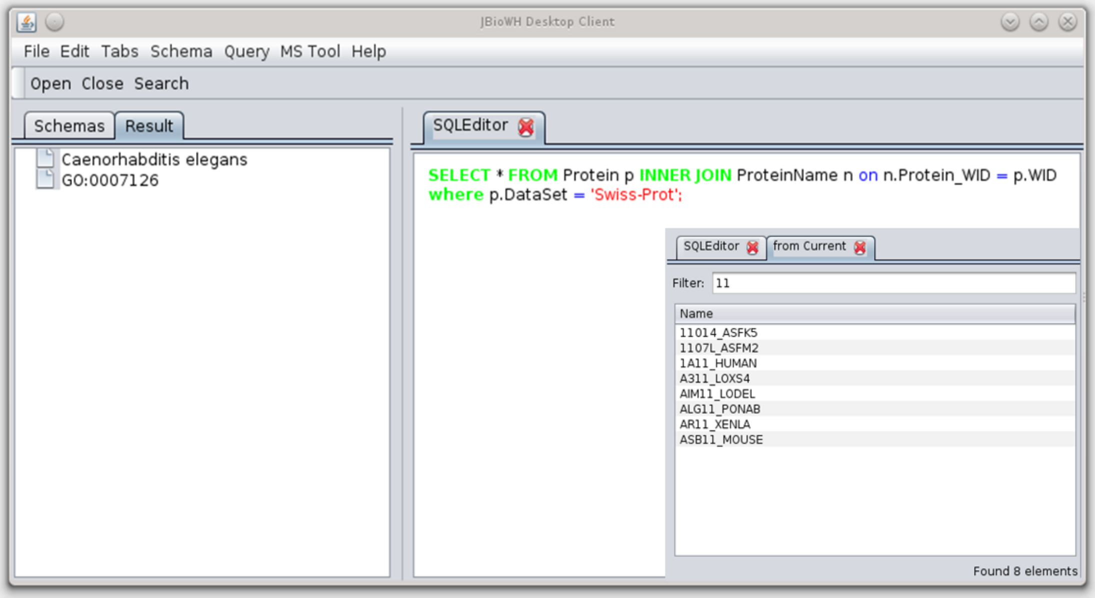
CDK1_CAEEL

Taxonomy: Caenorhabditis elegans
Version: 119
Modified: Wed Dec 11 00:00:00 CET 2013
Created: Tue Feb 01 00:00:00 CET 1994
DataSet: Swiss-Prot
Existence: evidence at protein level
SeqLength: 332
Mass: 38295
Precursor: null
Fragment: null
Seq:
MDPIREGEVAHEGDSVYTLNDFTKLEKIGEGTYGVVYKGKNRRTNAMVAMKKIRLESEDEGVPS

176374		Caenorhabditis elegans

Figure 17: The result interface showing the gene linked to the protein.

The search interface is designed to allow complex queries using the Constrains box as shown in the Figure 15. The result interface is shown in the Figure 17



Caenorhabditis elegans
GO:0007126

```
SELECT * FROM Protein p INNER JOIN ProteinName n on n.Protein_WID = p.WID
where p.DataSet = 'Swiss-Prot';
```

Filter: 11

Name
11014_ASFK5
1107L_ASFM2
1A11_HUMAN
A311_LOXS4
AIM11_LODEL
ALG11_PONAB
AR11_XENLA
ASB11_MOUSE

Found 8 elements

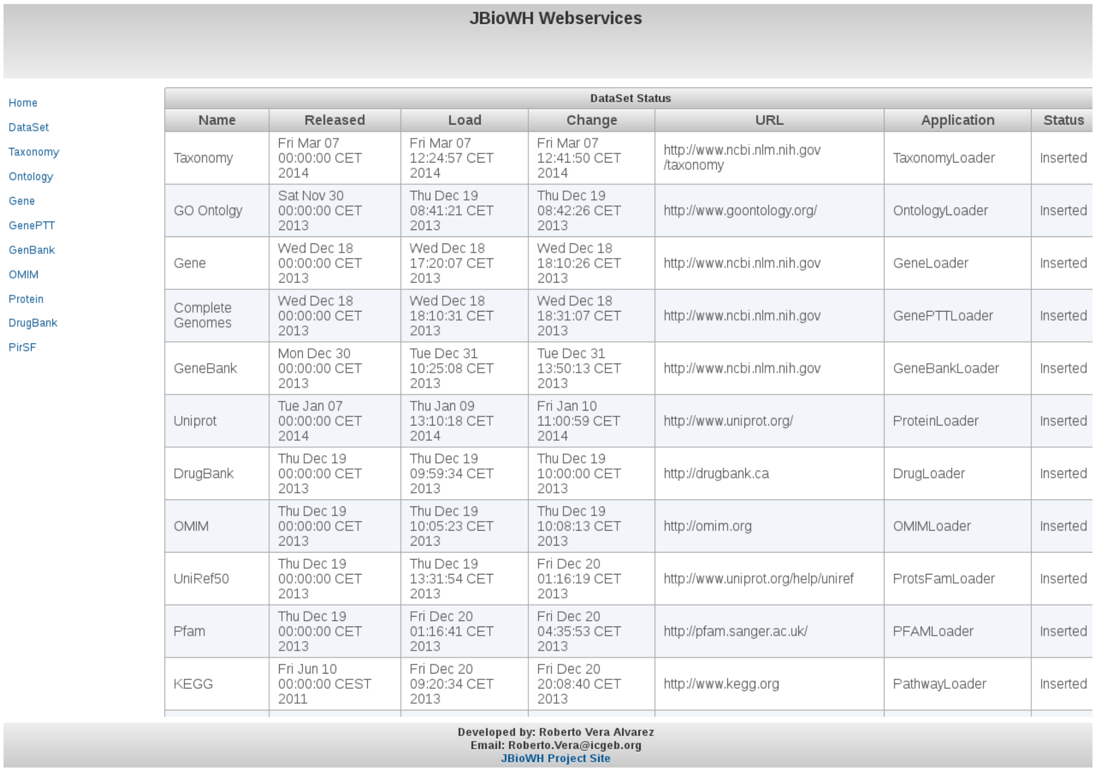
Figure 16: The SQL query interface with the result list.

DOI:10.15774/PPKE.ITK.2014.011

Additionally, the application provides a SQL editor that allows the execution of queries on the relational schema. The editor is highlighted to help the typing and the results are shown in a different tab that also includes a simple filter as shown in Figure 16.

3.1.5 Web services

The **JBioWH** webservices (***jbiowh-webservices*** and ***jbiowh-webservices-client***) are applications designed to provide the integrated data inserted into the **JBioWH** relational schema access over a ***HTTP***. The webservices have an associated web site that is published using the Apache Tomcat server [200], see a demo server: http://net.icgeb.org/jbiowh-webservices/. The associated web site offers a table with the data inserted into the relational schema as shown in the Figure 18. Also, tutorial pages are published for each module in order to describe the available webservices paths (methods) for each module.



JBioWH Webservices

Home
DataSet
Taxonomy
Ontology
Gene
GenePTT
GenBank
OMIM
Protein
DrugBank
PirSF

DataSet Status

Name	Released	Load	Change	URL	Application	Status
Taxonomy	Fri Mar 07 00:00:00 CET 2014	Fri Mar 07 12:24:57 CET 2014	Fri Mar 07 12:41:50 CET 2014	http://www.ncbi.nlm.nih.gov/taxonomy	TaxonomyLoader	Inserted
GO Ontolgy	Sat Nov 30 00:00:00 CET 2013	Thu Dec 19 08:41:21 CET 2013	Thu Dec 19 08:42:26 CET 2013	http://www.goontology.org/	OntologyLoader	Inserted
Gene	Wed Dec 18 00:00:00 CET 2013	Wed Dec 18 17:20:07 CET 2013	Wed Dec 18 18:10:26 CET 2013	http://www.ncbi.nlm.nih.gov	GeneLoader	Inserted
Complete Genomes	Wed Dec 18 00:00:00 CET 2013	Wed Dec 18 18:10:31 CET 2013	Wed Dec 18 18:31:07 CET 2013	http://www.ncbi.nlm.nih.gov	GenePTTLoader	Inserted
GeneBank	Mon Dec 30 00:00:00 CET 2013	Tue Dec 31 10:25:08 CET 2013	Tue Dec 31 13:50:13 CET 2013	http://www.ncbi.nlm.nih.gov	GeneBankLoader	Inserted
Uniprot	Tue Jan 07 00:00:00 CET 2014	Thu Jan 09 13:10:18 CET 2014	Fri Jan 10 11:00:59 CET 2014	http://www.uniprot.org/	ProteinLoader	Inserted
DrugBank	Thu Dec 19 00:00:00 CET 2013	Thu Dec 19 09:59:34 CET 2013	Thu Dec 19 10:00:00 CET 2013	http://drugbank.ca	DrugLoader	Inserted
OMIM	Thu Dec 19 00:00:00 CET 2013	Thu Dec 19 10:05:23 CET 2013	Thu Dec 19 10:08:13 CET 2013	http://omim.org	OMIMLoader	Inserted
UniRef50	Thu Dec 19 00:00:00 CET 2013	Thu Dec 19 13:31:54 CET 2013	Fri Dec 20 01:16:19 CET 2013	http://www.uniprot.org/help/uniref	ProtsFamLoader	Inserted
Pfam	Thu Dec 19 00:00:00 CET 2013	Fri Dec 20 01:16:41 CET 2013	Fri Dec 20 04:35:53 CET 2013	http://pfam.sanger.ac.uk/	PFAMLoader	Inserted
KEGG	Fri Jun 10 00:00:00 CEST 2011	Fri Dec 20 09:20:34 CET 2013	Fri Dec 20 20:08:40 CET 2013	http://www.kegg.org	PathwayLoader	Inserted

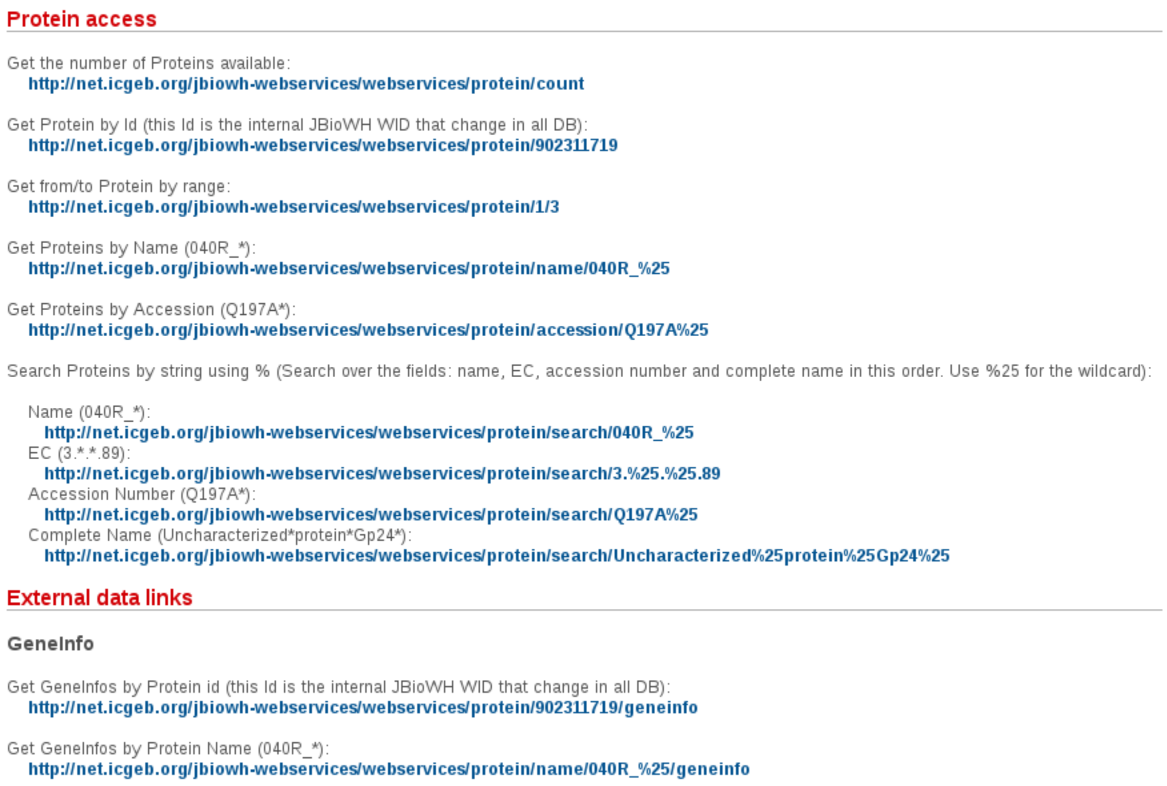
Developed by: Roberto Vera Alvarez
Email: Roberto.Vera@icgeb.org
JBioWH Project Site

Figure 18: The webservices associated web site showing the available Datasets.

DOI:10.15774/PPKE.ITK.2014.011

The *URL* path is structured using the global path ***webservices*** (common for all modules), the module's name (***protein***), the method's name (**accession**) and the parameters (***040R_%25***). Additionally, all modules included the ***count*** and the ***search*** methods. The ***search*** method is implemented using the **JBioWHSearch** interface. Each module will have a ***search*** path that acts as a wrapper to the search interface. This allows to include automatically all the search options available for the module directly in the webservices. For example, for the **Protein** module the paths are showed in the Figure 19.

Finally, the parameters passed to the webservices are designed to use regular expressions in the ***MySQL*** format. The regular expressions have to be encoded using the HTTP encoding system in order to be correctly passed through the webservices to the DBMS.


Protein access

Get the number of Proteins available:
http://net.icgeb.org/jbiowh-webservices/webservices/protein/count

Get Protein by Id (this Id is the internal JBioWH WID that change in all DB):
http://net.icgeb.org/jbiowh-webservices/webservices/protein/902311719

Get from/to Protein by range:
http://net.icgeb.org/jbiowh-webservices/webservices/protein/1/3

Get Proteins by Name (040R_*):
http://net.icgeb.org/jbiowh-webservices/webservices/protein/name/040R_%25

Get Proteins by Accession (Q197A*):
http://net.icgeb.org/jbiowh-webservices/webservices/protein/accession/Q197A%25

Search Proteins by string using % (Search over the fields: name, EC, accession number and complete name in this order. Use %25 for the wildcard):

Name (040R_*):
http://net.icgeb.org/jbiowh-webservices/webservices/protein/search/040R_%25
EC (3.*.*.89):
http://net.icgeb.org/jbiowh-webservices/webservices/protein/search/3.%25.%25.89
Accession Number (Q197A*):
http://net.icgeb.org/jbiowh-webservices/webservices/protein/search/Q197A%25
Complete Name (Uncharacterized*protein*Gp24*):
http://net.icgeb.org/jbiowh-webservices/webservices/protein/search/Uncharacterized%25protein%25Gp24%25

External data links

GeneInfo

Get GeneInfos by Protein id (this Id is the internal JBioWH WID that change in all DB):
http://net.icgeb.org/jbiowh-webservices/webservices/protein/902311719/geneinfo

Get GeneInfos by Protein Name (040R_*):
http://net.icgeb.org/jbiowh-webservices/webservices/protein/name/040R_%25/geneinfo

Figure 19: The available webservices methods for the Protein module.

DOI:10.15774/PPKE.ITK.2014.011

3.1.6 Examples

To show the multiple applications that **JBioWH** framework can have we would like to describe some experiments. The experiments are designed to use all the available features of **JBioWH** from the Java API to the Desktop client.

Additionally, there are multiple examples published on the project's Web site http://code.google.com/p/jbiowh/wiki/Examples.

Examples of simple data retrieval (retrieving one protein sequence, or all human sequences) are shown in

The Table 9 shows two solutions for each problem, *i.e.* the SQL command for retrieving the data, and the Java code used in conjunction with the API.

Table 9: Two simple examples and their solutions using SQL language and the Java API code

Task	SQL solution	Java solution
Retrieve the protein sequence for the protein Q8DR59 from UniProt.	`SELECT p.seq FROM Protein p INNER JOIN ProteinAccessionNumber a ON a.Protein_WID = p.WID WHERE a.AccessionNumber = 'Q8DR59';`	`JBioWHSearch sProt = new SearchProtein();` `List prots = sProt.search(“Q8DR59”, null);` `for(Protein p : (List<Protein>) prots)` `   System.out.println(p.getSeq());`
Retrieve the protein sequence of all human proteins	`SELECT p.seq FROM Protein p INNER JOIN Protein_has_Taxonomy pt ON pt.Protein_WID = p.WID INNER JOIN TaxonomySynonym ts ON ts.Taxonomy_WID = pt.Taxonomy_WID WHERE ts.Synonym like 'human';`	`JBioWHSearch sTax = new SearchTaxonomy();` `JBioWHSearch sProt = new SearchProtein();` `List taxs = sTax.search(‘human’,null);` `List c = new ArrayList();` `List o = new ArrayList();` `c.add(taxs);` `o.add("IN");` `JPLConstrains constrain = new JPLConstrains(c,o,null);` `List prots = sProt.search(“”, constrain);` `for(Protein p : (List<Protein>) prots)` `   System.out.println(p.getSeq());`

More complex questions can be solved using the **JBioWH** Java API. These kinds of questions cannot be solved directly using SQL language (neither the Desktop Client). The API can be used to answer queries that cannot be easily handled by the SQL language.

DOI:10.15774/PPKE.ITK.2014.011

Namely, recursive queries such as finding nearest neighbors in terms of metabolism, taxonomy or chromosomal locations are typical examples of this kind of a problem.

For instance, we want to find out if there are antibiotics that target a certain chromosomal region. As an example, we take the ±10 genes in the chromosomal neighborhood of the gene **spr0328** of ***Streptococcus pneumoniae R6***, which encodes the protein ***Endo-alpha N-acetylgalactosaminidase*** (**Q8DR60**). The **JBioWH** will find gene **spr0329** (GeneId: 934791). This gene encodes protein **Q8DR59**, a penicillin-binding protein that is the target of *Oxacillin*, *Hetacillin*, *Nafcillin*, *Ampicillin*, *Cefalotin*, *Azidocillin*, *Cefotaxime*, *Cefoxitin* and *Cephalexin*.

To answer this question one needs to retrieve gene and chromosomal position information from the Gene and Genome databases, respectively, then use the cross-references to identify the corresponding proteins in the UniProt database. Subsequently, **JBioWH** retrieves antibiotic information for these proteins from DrugBank. The total execution time for this operation is 10 s. The complete description and the source code are available at http://code.google.com/p/jbiowh/wiki/Example7.

We can extend the scope of this question for an entire taxonomic subgroup. The question now is whether the orthologs of gene **spr0328** in a certain taxonomic group have chromosomal neighbors that encode for antibiotic targets. We will use the ±10 genes neighborhoods in two genera, ***Streptococcus*** and ***Burkholderia***.

To answer this question, we first have to retrieve the taxonomic groups. For this purpose, **JBioWH** uses graph structures that can be created by extending the **JBioWHGraph** class in Java. For instance, the **TaxonomyGraph** class represents the hierarchical structure of a Taxonomy family. Table 10 shows the data of three example taxonomies that can be created by a code shown in http://code.google.com/p/jbiowh/wiki/Example5.

DOI:10.15774/PPKE.ITK.2014.011

Table 10: This table shows the use of the TaxonomyGraph class to create the hierarchical structure of a Taxonomy family.

Family	**Tax Id**	**Graph**		**Time (s)**
		Vertex	**Edges**	
Bacteria	2	283 371	283 370	121
Burkholderia	32008	3790	3789	4
Streptococcus pneumoniae	1313	303	302	3

Then, we want to get the orthologs of **spr0328** from all ***Streptococci***, located within a 10-gene neighborhood of **spr0328** and thereafter, locate the antibiotic target genes. For the location of the orthologs **JBioWH** uses the eggNOG databases. In ***Streptococcus*, JBioWH** finds two genes encoding antibiotic target proteins, gene 934791 of ***S. pneumoniae R6***, which was already found in our first example, and gene 930269 ***S. pneumoniae TIGR4***. The total execution time for this query was 15 s. The complete description and the source code are available in http://code.google.com/p/jbiowh/wiki/Example8.

Now we further generalize the query: Given a taxonomic group find all chromosomal regions (say maximum 5000bp in length) that harbor at least two genes encoding antibiotic targets.

Again, we will use the genera ***Streptococcus*** and ***Burkholderia*** as the examples. **JBioWH** will use **TaxonomyGraph** class to retrieve the genus members. The NCBI PTT table of the genomes will be used for a step-by-step search. Genes that encode an antibiotic target will be identified through links to UniProt, and from UniProt to DrugBank. The results in Table 11 show that one gene-pair in ***S. pneumoniae TIGR4***, and two gene- pairs in ***Burkholderia xenovorans LB400*** are retrieved. The execution time for ***Streptococcus*** is 112 s and for the ***Burkholderia*** 249s. The complete description and the source code are available at http://code.google.com/p/jbiowh/wiki/Example6.

DOI:10.15774/PPKE.ITK.2014.011

Table 11: This table shows the genes encoding for drug's target protein that are in the same chromosome at a distance less than a specific number of pair bases.

Family	Genes	Found gene ID	Species	Time(s)
Streptococcus pneumoniae	41 576	930805-930802	*Streptococcus pneumoniae TIGR4*	112
Burkholderia	224 568	4010698 -4010703 4010703-4010704	*Burkholderia xenovorans LB400*	249

Finally, we show examples related to drugs that act on similar targets. In the database, the drugs are linked to proteins, and proteins are members of a network of metabolic pathways. In this system, two drugs can be (i) target neighbors, if they act on the same protein (ii) pathway neighbors, if they act on proteins that belong to the same metabolic pathway or (iii) distant neighbors, if they act on different pathways, and in the latter case it is important to know, in addition, how far apart in the metabolic network the two drugs are because distant relationships can be biologically meaningless.

Questions related to (i) and (ii) can be answered by SQL queries but they need multiple joints, which makes the search time-consuming, especially in the case of (ii). Questions of type (iii), however, involve a prohibitively large number of multiple SQL join operations. **JBioWH** can handle these complex queries because of the graph structures implemented using the **DrugPathwayGraph** class.

Hetacillin (DrugBank id: **DB00739**) is a beta-lactam that does not have intrinsic antibacterial activity, but is converted in the human body to *Ampicillin*, which is active against a variety of organisms. In DrugBank, *Hetacillin* is reported to act only on ***Penicillin-binding protein 1A*** (UniProt Id: **PBPA_STRR6**) and ***Penicillin-binding protein 2B*** (UniProt Id: **PBP2_STRR6**) both of which are parts of the pathway ***Peptidoglycan biosynthesis*** (KEGG: **spr00550**) of the strain ***S. pneumoniae R6*** (TaxId: 171101).

Other links of *Hetacillin* are not reported even though its metabolite *Ampicillin* is well-known to act on various organisms. If we use a simple SQL query, only the links to ***S. pneumoniae R6*** will be found. However, the **DrugPathwayGraph** class can be used to find all target neighbors, pathway neighbors and also distant neighbors of the *Hetacillin*.

DOI:10.15774/PPKE.ITK.2014.011

The answer provided by **JBioWH** is that (i) the drug has 19 target neighbors that would act on the same protein target, (ii) has 38 pathway neighbors and (iii) the drug has 35 nearest distant neighbors that are identified as antibiotics. The execution time was 300-400 s.

In addition, the drug-pathway graph can be useful for identifying antibiotic drugs that target the same pathway in other organisms. For instance, *Ceftazidime* (DrugBank Id: **DB00438**) and *Cyclacillin* (DrugBank Id: **DB01000**) are antibiotic drugs that target the same pathway as *Hettacillin*, but in three different organisms, ***S. pneumoniae R6*** (TaxId: **171101**), ***Escherichia coli str. K-12 substr. MG1655*** (Taxid: **511145**) and ***Clostridium perfringens str. 13*** (Taxid: **195102**). This answer can be obtained using the **DrugPathwayGraph** class as described in http://code.google.com/p/jbiowh/wiki/Example10.

We point out that graph-based queries cannot be easily answered by SQL-based systems such as relational databases that do not have graph extensions, and these queries are practically impossible to answer using the traditional central resources or federated databases such as Biomart [201, 202]. Although the data sources involved are well-known and sufficiently cross-referenced, it would require multiple visits from one database to another, which would make the process too time-consuming and complex for human operators, and also, too vulnerable to network failure. On the other hand, such complex questions may arise in data mining projects where the queries need to be answered many times within a loop.

3.1.7 Applications

The **JBioWH** has been used as integrated data supplier system to different OMICS disciplines like genomics, proteomics and drug design. A detailed explanation of these applications is out of the scope of this Thesis. Therefore, we simply mentioned them and the scientific papers and patents related to those works.

Specifically, the integrated database was used in drug design experiments to characterize the metabolic pathways and the protein targets. The Drug module was used to provide the drug-target relationship used to test the docking programs and the three-dimensional

DOI:10.15774/PPKE.ITK.2014.011

location of the drug inside the protein’s active site. See the patents [203, 204] and paper [205].

Also, proteomics databases designed for protein identification using the Mascot Server [206] was created from **JBioWH**. Specifics criterions for filtering the proteins by sequences composition was used in order to create training datasets for testing proteomics experiments. See the papers [207, 208].

Finally, **JBioWH** is the data supplier of the applications used to study Quorum Sensing Systems. This application requires classified taxonomic data integrated with the bacterial genomes. See the paper [209].

3.1.8 Summary

The **JBioWH** framework provides an open-source Java API for integrating biological data from various public databases in a data warehouse manner. The aim of **JBioWH** is to allow users to construct application-specific databases taking into account the biological context; in this chapter, we present a demo example of integrating 24 data sources. The integrated database is hosted on a local computer so it can be used for data-intensive calculations. This feature is especially important for queries that are not, or not easily, accommodated on central data resources.

We note also that this feature could be important in environments with slow, or limited, Internet access. The relational schema of **JBioWH** is defined in a MySQL DBMS, and contains Java classes and parsers that load the modules of the **JBioWH** with data from public databases.

Finally, **JBioWH** includes an API interface for programmatic access and a Desktop Client that lets users easily manipulate and query data via a graphical interface. Future work will aim to integrate further biological data sources and the implementation of other DBMS such as PostgreSQL and Oracle.

DOI:10.15774/PPKE.ITK.2014.011

The webservices application can be used by external programs to retrieve data from the integrated database over **HTTP**. This kind of communications between the integrated database and the application clients is very important for mobile applications due to the actual limitation of the mobile platforms to use the Java Persistence models.

DOI:10.15774/PPKE.ITK.2014.011

3.2 Prediction of bacterial taxa and gene function from NGS results

Identification of bacteria in unknown samples is crucial in various fields, such as detection of human or animal pathogens, detection of bacterial contamination in food samples etc. Next generation sequencing is increasingly becoming the method of choice in many areas because of the richness of data it can provide, however the processing of NGS data is problematic in many respects. Namely, current sequence alignment problems were developed with genome sequencing in mind; they are optimized for handling a single reference genome (the human genome) on which they work very efficiently.

From the computational point of view, the problem lies in the indexing process. All database search programs, starting from BLAST, gain efficiency by indexing the database in a form that can be rapidly searched. Current alignment programs (BWA, Bowtie2) use the Burrows Wheeler Transform combined with the Ferragina index for preprocessing the database.

An indexed database can be quite large, but searching such a database with a query is not proportional to the database size, it rather depends on the length and number of the queries. Current aligners are optimized for indexing one “human-size” genome and running a large number of reads against the indexed database. Identification of bacterial taxa is not such a task, here we have many thousands of genomes, draft genomes and individual DNA sequences and in an optimal case, we should use all these data for the identification of bacteria. Running searches on these genomes separately may require several thousand of separate indexing and alignment procedure which is computationally not tractable.

3.2.1 The Taxoner principle

The idea we proposed to solve this problem is the construction of a number of artificial genomes from many bacterial sequences, and running alignments on the artificial chromosome. In this way, we can use the advantages of the genome-mapping programs. When aligning a complex metagenomic dataset against the artificial chromosomes, in the ideal case, each read will map to its own genome, so the mapping will provide a way to

DOI:10.15774/PPKE.ITK.2014.011

identify the taxa present in the sample. For this aim, we need to know the location of genomes within the artificial chromosomes.

From the logical point of view, our database consists of segments of a large artificial chromosome sequence which is nothing but a large annotated sequence. Each annotated segment represents a taxon, more precisely a bacterial strain which is a leaf in the taxonomic hierarchy, so whenever a sequencing read maps to a segment, in can be traced back to a taxon, see Figure 20.

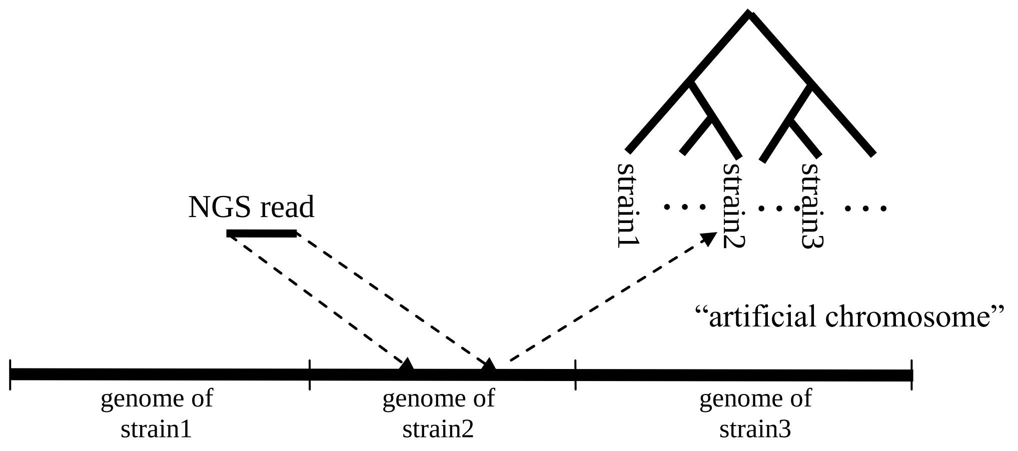


Figure 20: Mapping of reads to bacterial strains using artificial chromosomes. A strain is a segment of the artificial chromosome that is named by a label in the taxonomical hierarchy.

Importantly, this process is analogous with the mapping of reads to an annotated genome. In an annotated genome, the segments are the genes which are named according to the function they carry. The functions are defined as part of a classification scheme, such as the COG, eggNOG or GO databases.

The analogy between taxon assignment and function assignment allows us to develop a common system which will interpret metagenomic sequencing reads in terms of both taxa and functions. The central data structure is the artificial chromosome - which is – same as the sequences annotated in GenBank or UniProt – contains annotated segments. The annotation is particular in this case. The top level corresponds to a genome which is named by a taxon. The next level corresponds to segments annotated within the genome segments; these are the genes, named by the functional hierarchies.

DOI:10.15774/PPKE.ITK.2014.011

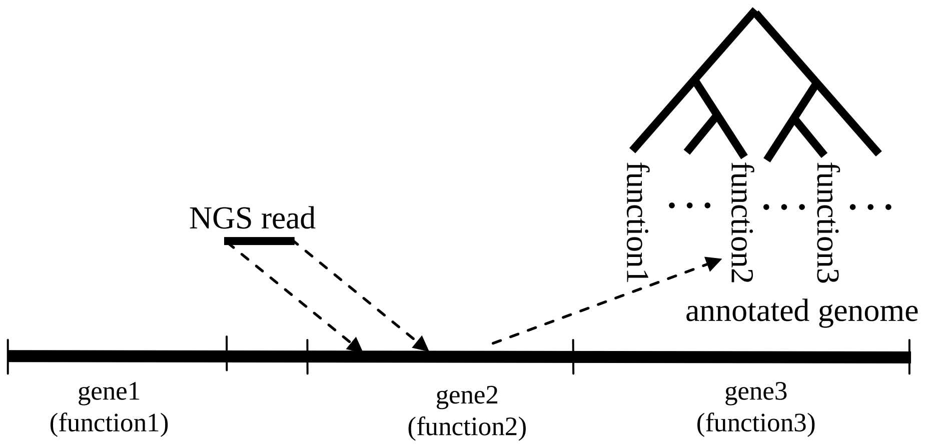


Figure 21: Mapping of reads to gene functions within an annotated genome. A gene is a segment of the genome that is named by a label in the functional hierarchy, such as the COG/EggNOG system or the GO databases.

The database is a set of artificial chromosomes which have to be built from the public databases, given as concatenated FASTA files. A database can be built from the finished (complete) bacterial genomes, and then it will allow the identification of taxa and gene functions. Or it can also include draft genomes, or simply the entire ***nt*** database of **NCBI**. The number of chromosomes that need to be built depends on the capacity of the computer. Table 12 shows the current size of a few databases and the number of artificial chromosomes necessary for the analysis. The artificial chromosomes are then indexed using the indexing facility of Bowtie2 which is the current aligner used by the Taxoner program.

Table 12: Current size of a few databases subsets and the number of artificial chromosomes necessary for the analysis.

The original dataset was the NCBI *nt*

Database	Size (GB)	No. of Chromosomes	Bowtie Index size (GB)
Bacteria	13.4	4	18.8
Archaea	0.58	1	0.83
Fungi	2.6	1	3.7
Virus	2.0	1	2.8

In addition to the artificial chromosome data, we also need the taxonomy tree, the gene sequences and their positions in the chromosome and COG or GO data. These data are extracted using an SQL script over the **JBioWH** integrated database. This script uses the

DOI:10.15774/PPKE.ITK.2014.011

JBioWH modules GenBank, COG, Protein Cluster and Taxonomy (see Section 3.1 for further explanation of the **JBioWH** framework). Then, the text file obtained is converted to a binary B+ Tree index using an *in-house* program developed in C (see Section 2.2.2.1). This program creates a B+ Tree index to store the data in a form that allow fast retrieval.

We mention that the design of the index files is not a straightforward task, since genomes and genes are named with a number of different ids that all have to be correctly mapped before we can establish an unequivocal mapping between the taxon, the gene and the function (See Section 3.2.2.4).

An important problem is the *handling of uncertainties* which is solved by empirical rules. First, a read can map to several taxa. In this case, the lowest common ancestor will be used as the resulting taxon, so, if a read maps both the *Escherichia coli* and to *Escherichia fergusoni* strains, it will be assigned to the genus *Escherichia*, but if it maps to two *E. coli* strains, it will be assigned to the species *E. coli*. But if a read maps to two taxa in such a manner that it overlaps their endpoints within the artificial chromosome, the read will be discarded as an artifact.

Mapping of reads to functions represent a different problem of uncertainties. In our approach, we used the functions annotated for known genes instead of performing a function prediction. For instance, if a read is assigned to a region who fall inside an annotated gene that gene and its annotated functions will be assigned to the read and reported by our programs.

Additionally, it is assumed that functions are constant within the members of species and not necessarily within higher taxonomies levels. Accordingly, if a read maps to several species, it will be not used for function assignment. If a read overlaps with two functions annotated in a genome, both functions will be reported.

Further sources of uncertainties are genes to which functions are not assigned. It is well known that most bacterial genomes contain a large number of such genes. When building the function assignment database for Taxoner, extra effort is made to assign a function to

DOI:10.15774/PPKE.ITK.2014.011

the hypothetical genes if reliable data are found in other databases, such as COG and eggNOG. In other words, the hypothetical proteins are re-annotated before insertion into the Taxoner database. Reads mapping to the remaining hypothetical proteins that are without function will be reported separately during the analysis.

3.2.2 The Taxoner algorithm

Taxoner is a program that identifies taxa, primarily bacteria, by mapping NGS reads to a comprehensive sequence database such as the **NCBI *nt*** database or its predefined subsets. The program is developed in such a way that I can run both on standard desktop/laptop computers under the Linux operating system or in high performance system like the Google Cloud Platform.

The algorithm consists of 3 phases. i) In the preprocessing phase, the database is divided into partitions and indexed with the bowtie2-build program of the Bowtie2 package. Alternatively, pre-built indices can be downloaded from the project site. ii) Alignment is carried out with bowtie2 and the taxa are identified with a lowest common ancestor search algorithm. The standard output of this phase is a summary of the found taxa and alignments in the SAMtools format. iii) Unlike other metagenome analysis programs, Taxoner can optionally provide a list of genes identified at the species level, along with their predicted functions. It also contains a utility that can produce a summary of the found functions, based on the COG-EggNOG scheme of functional descriptors [166, 210], using a B+ tree index. In addition, the read alignments provided in the SAMtools format can be further processed with other taxon assignment programs such as MEGAN.

3.2.2.1 The Taxoner pipeline

Taxoner is a pipeline written in C, which currently uses *Bowtie2* [101] (2.0.0-beta5) for the sequence alignments.

The input is a reference nucleotide database in the form of concatenated FASTA file, and a set of nucleotide sequence reads typically 40-500bp in length, in fastq format. The Taxoner

DOI:10.15774/PPKE.ITK.2014.011

database is freely available from http://pongor.itk.ppke.hu/taxoner/databases/bowtie2/. The **NCBI Taxonomy** database retrieved on 11-07-2013.was used for taxon assignment. For function assignment, Taxoner uses a preformatted dataset that contains two binary files which can be easily created from a pipeline that uses a SQL script and the JBioWH framework [211].

The binary files can be downloaded from http://pongor.itk.ppke.hu/taxoner/databases/geneassignment/ and the pipeline description can be found in the Taxoner Google Code web site.

The source code is freely available from the Taxoner Project published on the Google Code web site (http://code.google.com/p/taxoner/). The program runs on Linux computers and includes a simple html graphical interface for local use. The program can be operated in the command line mode and allows evaluation of large read datasets on personal computers or laptops with at least 8GB RAM. A demo web server, with test cases and a capacity to process datasets up to 100 thousand reads can be found at http://pongor.itk.ppke.hu/taxoner.

3.2.2.2 Preprocessing

The database used for alignment is created using the **NCBI *nt*** fasta file. The standard database creation process is done by splitting the ***nt*** fasta file into ~4Gb fasta files (sub-databases), where the headers of each fasta sequence is replaced with the **GI** identifier and the organism taxon **ID**. The fasta files cannot be larger than ~ 4.0 Gb, since Bowtie2 has a limit on the reference genome size. When the database is created, the final step is then to index each reference fasta file with the bowtie2-build program. Since the ***nt*** database is not restricted to microbial organisms, an alternative database can be created by extracting the subset organisms of interest (e.g. bacteria, fungi, archaea, see Table 12).This greatly reduces the analysis time since reads do not have to be aligned to non-microbial entries. For this reason we made a pre-parsed and indexed database for each major microbial superkingdom available that can be downloaded via our website at

DOI:10.15774/PPKE.ITK.2014.011

https://code.google.com/p/taxoner/wiki/07_Databases. We also included a database creator program that can create a database with any taxon ID(s) specified by the user.

3.2.2.3 Analysis

For the analysis the user must provide the input parameters (listed and explained below, under command line usage). During analysis, Taxoner first runs Bowtie2 using the default or user specified parameters and writes the alignments into a SAM (Sequence Alignment/Map)[212] file.

The sequence alignment with Taxoner is done using the Bowtie2 aligner and the pre-indexed databases. Since the ***nt*** database is too large to fit in a single fasta file, the input sequencing reads have to be aligned separately against each sub-database. Fortunately, Bowtie2 is a multithreaded aligner, which enables faster alignments using multithreaded processors. After all reads are aligned against each database, the taxonomic evaluation can be done. The main difficulty here is that a read can align with the same alignment quality to each fasta file. For this reason, Taxoner calculates a “local” common ancestor (LCA) for each read in each alignment file and then merges the results into one file. In the final step, the merged file is sorted by read’s name and a final LCA is calculated for each read using their “local” lowest common ancestors.

The output is a file that contains read names, alignment information, nearest neighbor taxon **ID**, start and ending positions of the alignment against the best hit and the genome accession number of the best hit. An optional output is a MEGAN compatible output, which enables the post analysis and visualization of the results.

3.2.2.4 Gene Assignment

In the Taxoner framework, detection of genes and assignment of functions is a problem analogous to function assignment (see Figure 21). This is a standalone utility that take the alignment output in the Taxoner format and maps the hits to genes specified in GenBank [213].

Function assignment is carried out by a scheme that is analogous to taxon assignment and it is performed on reads assigned to strains or to species. Briefly, a read mapping a (protein or RNA coding) gene within a strain contributes one count to that function. Reads assigned at species level are assigned to the highest-ranking gene's function in the top-list. It is assumed that genes within a species have identical functions, so if there are several hits within the same species the read is assigned to the highest-ranking annotated gene. The result of function assignment is a list of functions with the corresponding read counts. The COG-EggNOG scheme of functional descriptors [166, 210] is used for functional assignments, in conjunction with a pre-calculated database file that uses the B+ Tree index [214] for fast function retrieval. The B+ Tree index algorithm was implemented in C. Its implementation is part of a C library developed by our group and is freely available at https://code.google.com/p/bioc/, see Section 2.2.2.1.

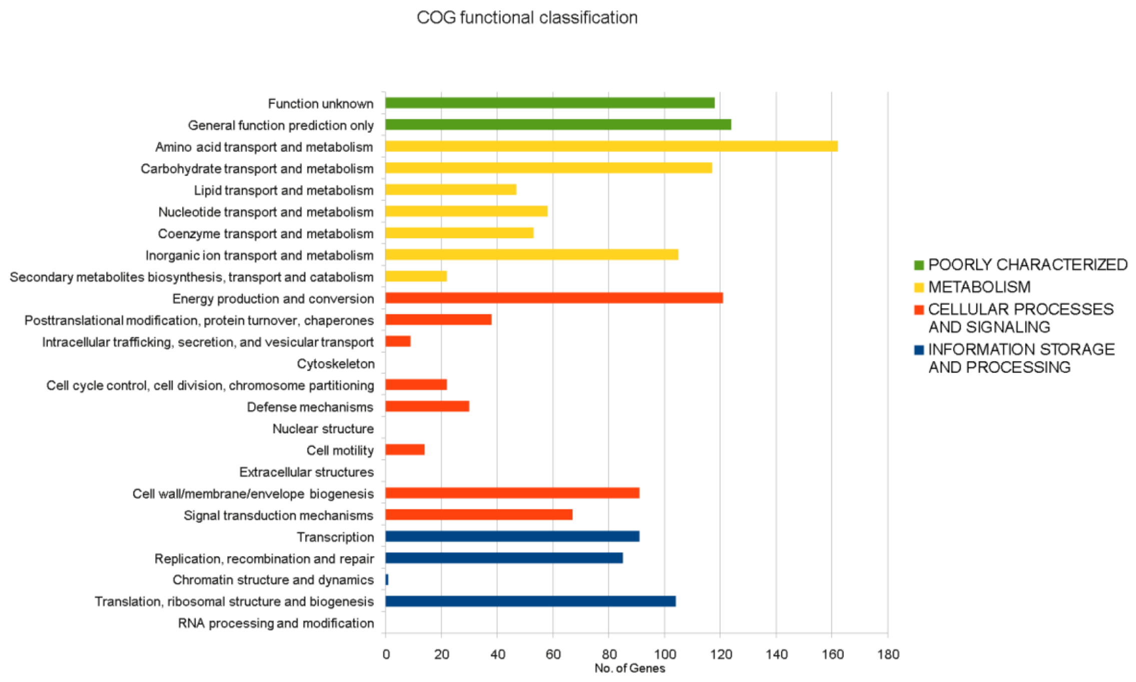


Figure 22: COG Functional classification made by the Gene Assignment tool for the genes identified by Taxoner.

The identifiers of the genes are stored either in a relational database, prepared with the **JBioWH** [211] framework, or are stored in the form of a binary file with a B+ Tree index. This stored form contains from-to location of each gene, the identifier and pointers to the COG-eggNOG [166, 210, 215] functional classification terms. If a read is mapped to one single genome, and the hit overlaps with one or more genes, each of the concerned genes will receive one hit. If the read is 100% identical with several genomes, then the first genomes in the Bowtie2 alignment will receive the hit. After evaluating all hits in this manner, a potentially large number of genes will have hits assigned. The utility can simply list the genes with the number of hits, or can combine the genes into functional categories using the COG-eggNOG scheme. As result, the hits collected by single genes will be added up to higher categories, and graphical statistics can be made. Examples are shown in Figure 22.

3.2.2.5 Output files generation.

The output of this phase is a file with a list of GenBank accession number(s), taxonomy id, NCBI protein id, NCBI gene symbol, the number of reads hitting the gene, a list of COG/eggNOG ids and NCBI Protein Cluster ids. Examples are deposited at http://pongor.itk.ppke.hu/taxoner/examples/.

3.2.3 Desktop and server versions

Taxoner includes a web based graphical interface that helps users to run the programs and parse the results. This interface is a JavaScript based set of web services developed using AngularJS library [216] and running over on the Nodes.JS platform [217].

This graphical interface is designed to be used locally on a PC. It offers input forms for the different components of the Taxoner system. The starting script will open a web site running on http://127.0.0.1:8081 (the local computer) and creates a series of web services running in the same IP address but in different ports.

DOI:10.15774/PPKE.ITK.2014.011

It is well know that modern web browsers do not have access to the local file system due to a security issue. As such a set of different web services must be created to allow access to the local files from the web browser.

The Taxoner web services are running on ports from 8081 to the 8084. The port 8081 is for the HTTP web site (the user interface). The rest are ports for internal uses: port 8082 offer a web service to read and parse the log files of Taxoner, 8083 is a web service to run system commands from the web browser and 8084 is a web service to create the Taxoner summaries from the output files.

Three web forms are available on the graphical interface to run the Taxoner program, the gene assignment program and to parse the outputs of both programs and show a summary of them.

Full description and images of the graphical interface can be seen at https://code.google.com/p/taxoner/wiki/06_Graphical.

In addition, a demo server was developed for demonstration purposes using the same look and feel of the local graphical interface. This web server offers the same features as the graphical interface but has a file size limitation for the input files. The demo server is available from http://pongor.itk.ppke.hu/taxoner.

3.2.4 Run times and examples

We evaluate the performance of Taxoner in comparison with Metaphlan [98], BLASTall [218] and MegaBLAST [219], both in combination with the MEGAN taxon assignment program [97, 104]. Metaphlan was selected because of its speed and accuracy in estimating taxon composition, BLAST was selected because of its reputation in alignment.

It is noted that comparison is a complex task since, for instance, Metaphlan compares reads to its own small taxon-marker database of about 367 million nucleotides that includes only bacteria. BLAST and Taxoner, on the other hand can run on comprehensive databases such as **NCBI *nt*** (52 billion nucleotides), which includes all species, or on a bacterial subset

DOI:10.15774/PPKE.ITK.2014.011

(typically 15.5 billion nucleotides). The search of the database thus impacts the speed and the accuracy of the results.

The actual alignment times for Taxoner, Metaphlan and BLAST depend on the size of the database and the number of threads used for the calculation, and naturally on the length and the number of the reads to be evaluated. The input read datasets used for testing are listed Table 13.

Table 13: Typical running times for the alignments.

		Running time[1]		
	Dbase	**1 thread**	**4 threads**	**12 threads**
MetaPhlAn[5]	own bacterial marker dbase[2]	14 sec	7 sec	6 sec
Taxoner[5]	NCBI nt Bacteria[3]	165 sec	105 sec	90 sec
Taxoner[5]	NCBI nt full dbase[4]	2446 sec	2031 sec	1866 sec
MEGABLAST[6]	NCBI nt bacteria[3]	8.3 h	n/a	3.9 h
MEGABLAST[6]	NCBI nt full dbase[4]	37.6 h	n/a	9.4h

[1]Read dataset: Dataset A, Table 1. Processor: Intel(R) Xeon(R) CPU E5-2640; [2]The built-in dataset is 366,988,039 nucleotides (367 MB) and contains only bacterial sequences; [3] 15,400,949,699 nucleotides (15 GB), downloaded on 11/07/2013; [4] 52,380,339,934 nucleotides (54 GB), downloaded on 11/07/ 2013; [5]Times include taxon assignment; [6] time of taxon assignment by MEGAN is not included.

3.2.5 Analyzing metagenomic datasets

The classification performance of Taxoner is compared with that of two programs in Table 14. We carried out an analysis of a metagenomic dataset published by the Human Microbiome Project that consisted of 6.5 and 1.4 million reads (Dataset G and H, respectively) and consisted of equal amounts of 22 strains representing 22 species. The data presented in Table 14 show that Taxoner can detect taxa at the strain level, which is in contrast to MetaPhlAn (and other programs, such as WGSQUIKR). The accuracy of MetaPhlAn and Taxoner are comparable in this task, but it has to be mentioned that Taxoner can achieve this accuracy only if one sets a minimum threshold to the number of reads necessary to identify a taxon (strain, species, etc). MetaPhlAn uses a similar thresholding for improving the accuracy. Without setting this threshold, Taxoner will report all spurious similarities, which would result in a very high number of false positives. In this analysis we also included WGSQUIKR, an extremely fast and innovative program that uses compressed sensing principles for finding a number of taxa that can identify the presence of the reads [99]. This analysis is extremely fast, the number of taxa present is apparently underestimated at all taxonomic levels.

Table 14: Detection of species in a metagenomic datasets

A) Illumina sequenced HMP Mock Community sample[1] (dataset G)														
		MetaPhlAn				**Taxoner[2]**				**WGSQUICKR**				
	No of positives (taxa present)	**TP[3]**	**FN[4]**	**FP[5]**	**F-measure**	**TP**	**FN**	**FP**	**F-measure**	**TP**	**FN**	**FP**	**F-measure**
strain	22	NA[6]	NA	NA	NA	14	7	8	0,65	1	20	79	0,02
species	22	21	1	7	0,84	20	2	0	0,95	9	13	67	0,18
genus	19	18	1	5	0,86	17	2	0	0,94	13	6	45	0,34
family	18	18	0	6	0,86	17	1	0	0,97	13	5	29	0,43
B) 454 sequenced HMP Mock Community sample[1] (dataset H)													
		MetaPhlAn				**Taxoner[7]**				**WGSQUICKR**			
taxa assigned	**No of positives (taxa present)**	**TP**	**FN**	**FP**	**F-measure**	**TP**	**FN**	**FP**	**F-measure**	**TP**	**FN**	**FP**	**F-measure**
strain	22	NA	NA	NA	NA	9	12	19	0,37	1	20	58	0,03
species	22	19	3	0	0,93	19	3	0	0,93	5	17	52	0,13
genus	19	16	3	0	0,91	16	3	0	0,91	8	11	37	0,25
family	18	16	2	0	0,94	16	2	0	0,94	9	9	23	0,36

The data was a mock community dataset provided by the Human Microbiome Project, consisting of 22 strains. [2]Only those hits (read-taxon assignments) were considered where the worst alignment score was at least 0.9. Positive taxa predicted by Taxoner are those that received at least 1000 hits (dataset G). [3]True positives. [4]False negatives. [5]False positives. [6]Not available. [7]Only those hits (read-taxon assignments) were considered where the worst alignment score was at least 0.9. Positive taxa predicted by Taxoner are those that received at least 100 hits (dataset H).

3.2.6 Analyzing known and unknown strains

Given the very high sequence variability of bacterial genomes, it is crucial to know whether or not the genome of a bacterium to be detected by NGS is included in the database. This is a very important question, since unknown strains represent the majority of environmental samples. In the context of data analysis, a strain is known when its genome or draft genome is included in the database. With this in mind we compared the detection probability of two *Bacillus anthracis* strains (Table 15). The "known" strain was the Sterne strain, used for vaccination, and the dataset consisted of 100bp long overlapping segments ("artificial reads") of the genome, offset by 50bp. This is thus a perfect dataset on which no mistakes are expected. The unknown strain was a Japanese isolate which was not included in the database at the time of this analysis, and the dataset contained 7.7 million Illumina reads.

DOI:10.15774/PPKE.ITK.2014.011

In fact, the number of errors (false negatives) is substantially higher for the strain not included in the database. The synthetic reads generated from a genome included in the database are perfectly detected by Taxoner, which is not surprising (they are perfectly detected also by MegaBLAST, data not shown). What is somewhat unexpected is that Metaphlan detects a substantial % of species not present in the samples, even in the synthetic reads. Even though we cannot explain this finding, it is good to remember, that *B. anthracis* belongs to the *B. cereus* group which includes three highly related species, *B. cereus*, *B. thurigiensis* and *B. anthracis*. This is reflected by the fact that a large percentage, about 75% of the synthetic reads generated from the *Sterne* strain is 100% identical in all three species (data not shown). Metaphlan assigns these reads to the *Bacillus* genus, while Taxoner (and Megan) assigns the reads to the *B. cereus* group. This illustrates the fact that species % reported by the various programs highly depend on the database as well as on the taxonomy definitions used by the given programs.

Table 15: Analysis of known and unknown *B. anthracis* strains.

Taxa assigned		**Metaphlan**[1]	**Taxoner**[2]
A) Strain included in the database (*B. anthraci*s strain Sterne (NCBI taxon id: 260799), 104574 synthetic reads, Dataset F)			
all		991 reads	104,573 reads
genus	*Bacillus*	100.00	100.00%
species	*Bacillus anthracis*	76.60%	100.00%
species	*Bacillus cereus*	15.40%	0.00%
species	*Bacillus thuringiensis*	8.00%	0.00%
species	other	0.00%	0.00%
	False negative %[3]	**23.40%**	**0.00%**
B) Strain not included in the database (*B. anthraci*s strain BA104 (NCBI taxon id: Not Available) , 7.7million Illumina reads, Dataset E)			
all		96,045 reads	7,379,118 reads
genus	*Bacillus*	99.58%	100.00%
species	*Bacillus anthracis*	65.30%	96.50%
species	*Bacillus cereus*	18.70%	0.80%
species	*Bacillus thuringiensis*	15.60%	0.40%
species	other	0.00%	2.30%
	False negative %	**34.44%**	**3.50%**

[1]The values are taken from the standard output of the program. [2]Values indicate the number of reads expressed as % of the total. [3]False negative% is the % of taxa (Metaphlan) and reads (Taxoner) detected but not present

DOI:10.15774/PPKE.ITK.2014.011

3.2.7 Summary

Here we described a pipeline of programs, called Taxoner that uses a fast aligner and a comprehensive database for analyzing metagenomic datasets. As a result of alterations to the indexing used, this pipeline is fast enough to run evaluations on a single PC, and it is highly sensitive so it can be adapted to analysis problems such as detecting pathogens in human samples.

Taxoner is much faster and at times more accurate than BLAST based evaluation schemes. In our case we tested BLAST in conjunction with the MEGAN program.

Detection of unknown strains is a problem to most aligners. It is important to remember that strains of the same species isolated from different natural environments can differ in a very large portion of their genome. As such analyzing the metagenome of soil bacteria may require the identification of strains that are largely novel as compared to the current databases. In this sense, approaches based on a comprehensive database, such as Taxoner, are at an advantage in comparison to approaches based on marker databases. This is because new strains do not necessarily contain the unique sequences included in a marker database. On the other hand, this is an important problem since detection of hazardous pathogens requires strain level identification. This feature is included in Taxoner, but not in many other programs designed for metagenome analyses.

We note that Taxoner uses bowtie2, and not BLAST, still its sensitivity is at times better than that of BLAST-based methodologies. This shows that fast alignment techniques may provide a useful alternative for sensitive analysis of metagenomic samples.

DOI:10.15774/PPKE.ITK.2014.011

3.3 The Genome Specific Marker Database

Modeling the alignment of reads to a genome is only seemingly simple. Namely, sequence reads are of varying length and quality, various segments of genomic DNA may not be equally amenable to sequencing reaction.

However, as a first approximation, we can neglect these differences and model a simpler problem (Figure 23, left). Let's suppose that the reads are of roughly equal length, *R*, and they are evenly distributed along the genome.

For diagnostic use, we need to distinguish those segments of the genome that are unique *i.e.* they can serve as diagnostic identifiers of the genome. Let's denote the cumulative length of such unique parts is *U*. And finally, if one is to build a marker database, one has to identify suitable markers which, by definition will be a subset of the unique segments. Let's denote the cumulative length of the markers as *M*.

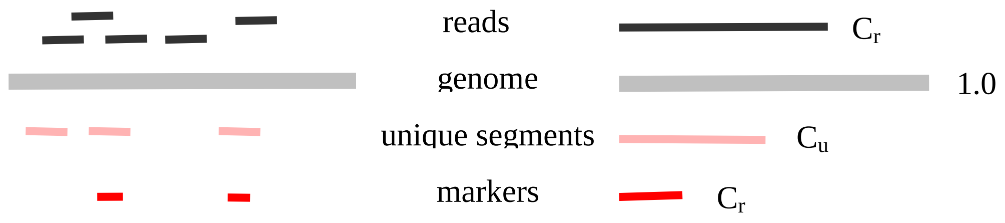


Figure 23: Reads, unique segments and markers (left), cumulative coverage values (right). Note that reads can overlap with each other but unique ("diagnostic") segments ad markers are disjunct. Also note that markers must fully overlap with the unique segments.

To describe the problem in quantitative terms, we define the coverage values. For N sequencing reads of length R and a total DNA length D, the read coverage C_r is defined as:

$$C_r = \frac{NR}{D} \quad (1)$$

For genome sequencing, the only DNA in the sequencing reaction is the genome being sequenced G. In this case,

DOI:10.15774/PPKE.ITK.2014.011

$$C_r = \frac{NR}{G} \quad (2)$$

For genome sequencing, C_r has to be greater than one, for medical diagnostics read coverage of several thousands are necessary.

When bacterial communities are sequenced such as in metagenomics experiments, the read coverage may fall much below one. Namely, for rare species we may only get a few reads. We can denote the proportion of a given genome within the total DNA being sequenced is a_G ($0 \leq a_G \leq 1$) we can express the expected coverage of genome in a mixture of genomes as:

$$C_r = \frac{NR}{a_G G} \quad (3)$$

In other words, a_G is 1 for genome sequencing, and can be close to zero for metagenomics experiments which, at the same time, indicates the importance of high sequence coverage in metagenomics experiments. We note the read coverage values defined in this manner do not correspond to the real coverage of the genome by reads. Namely, real life reads are not disjunct but can be highly overlapping, so C_r can be >>1. However, as we take that the reads are distributed evenly along the genome and among the various genomes, this proportion is considered constant.

The proportion of the unique diagnostic regions within a genome is denoted by the coverage value C_u as

$$C_u = \frac{U}{G} \quad (4)$$

Here U denotes the unique portion of the genome. In principle, $0 \leq C_u \leq 1$, in practice the proportion of the unique parts is much smaller, e.g. the proportion of the markers, selected for identification purposes within a given genome is denoted by the coverage value C_m as

DOI:10.15774/PPKE.ITK.2014.011

$$C_m = \frac{M}{G} \quad (5)$$

In this equation M denotes the unique portion of the genome. In principle, $0 \leq C_m \leq C_u$, in practice the proportion used by a particular marker database is often defined empirically with statistics, for instance a sufficient number of marker regions are collected that allow safe identification of genomes in a given dataset.

We can now estimate p_{gi}, the probability of genome identification by aligning a single read to the genome or to a marker database. Note, that the read is considered perfect if its sequence is part of the DNA to be sequenced. If we compare the read with the entire genome, we would expect that p_{gi} will be 1 since the read is contained within the genome. This may hold for long reads, however, we know that sequences of 40nt can occur by chance. So the probability of genome identification is better for long reads and decays for shorter reads. This property can be simply captured by assigning a significance value to the hit, which is calculated by aligner programs such as BLAST or Bowtie.

Let's now consider a marker database which is, by definition shorter than the genome itself. Since the reads are evenly distributed along the genome, only C_m fraction of them will hit the marker database and result in positive genome identification. In other words, the size of the marker dataset of a given genome should be close to C_mL so as to ensure a high probability identification.

Another approach is to use the entire genome as the database which, by definition includes the unique regions necessary for identification. Even this simple overview shows that there are several compromises possible when we design a marker database, as summarized in the Table 16.

As a specific example I mention Metaphlan, which is a small database, C_m~0.01. The search is extremely fast, but the sensitivity is quite low, we need over 100 reads per genome in order to notice one or two of them.

DOI:10.15774/PPKE.ITK.2014.011

Table 16: Summary pros and cons for the marker database design,

Marker database	**Pros**	**Cons**
Small, C_m<< C_u	Fast searches	Low sensitivity, need for high read coverage, some genomes contain few unique regions, dbase building takes time
Large, C_m ~ C_u	Medium to Fast searches	Higher sensitivity, some genomes contain few unique regions, dbase building takes time
Full genome	Full coverage, high sensitivity	Long searches

Taxoner, described in the previous section, uses full genomes as the database, so it's searching times is about 10 times that of Metaphlan. However the sensitivity is much better, genomes can be detected from a few reads.

GSMer [220] has a larger marker database than Metaphlan, but smaller than a full genome. The sensitivity and the search times are thus between Metaphlan and Taxoner.

One problem which is not mentioned so far explicitly is the taxonomic depth of identification. Taxon are usually identified by versions of the lowest common ancestor algorithm, i.e. reads common to two strains within one species and not present elsewhere, are assigned to the species. If a large part of the genomes of two strains are common, much more reads will be identified at the species level than at the strain level. On the other hand, we can define marker sequences also for the species level (or for higher taxonomic levels). These higher level marker datasets can be substantially bigger that those defined at the strain level. As a result, the searching time can increase above the practical level.

The selection of the marker dataset is not easy even if the markers are ideally specific for a given strain or species. Small marker datasets allows for fast searching, but a sufficiently sensitive detection of low abundance species may require prohibitively high read coverage. The loss of sensitivity also makes the detection of lower taxonomic levels more difficult. Full genomic databases, on the other hand, allow a sufficiently sensitive detection at lower read coverage values, but the search time may become prohibitively long because a large number of identical sequences are present in the database (redundancy caused by common sequence parts).

DOI:10.15774/PPKE.ITK.2014.011

Parallel to identification by NGS, the methods of in vitro identifications also need unique markers, especially PCR primers, for various taxa. In addition to the condition to ubiquity, efficient primers need to have certain physicochemical properties, like stability, melting point etc., which are required for the PCR reaction. The summary of these conditions is beyond the scope of this thesis; we can find good descriptions in [221, 222]. Briefly, when the goal is to design efficient primers for PCR identification, unique sequences of a marker dataset are processed further by primer design program that will select a subset of potential sequences.

From this brief introduction it is conspicuous that one needs to find the right balance between database size, sensitivity, taxonomic depth and search time. We will use these principles for designing a) a taxon identification system that uses NGS reads for metagenomic communities and b) a nearly non-redundant, comprehensive bacterial marker database, suitable for marker selection and primer design.

3.3.1 The marker database approach

There are several ways to construct specific marker databases, but the underlying workflows are not always described clearly in the literature. We can roughly picture the problem as a k-mer tiling: we cover a selected genome with overlapping words. As we increase the size of the words, we find words that are “unique” in sense that only the selected genome contains it and no other genome or genome segment in the entire database. Then, we simply concatenate our unique words into marker regions. We need to note that “uniqueness” means that a certain word does not have an exact match, but more precisely, we may be better of looking for words whose nearest neighbors are less than, say, 80% identical. In order to find unique words, we need, as an example, a) to check the occurrence of all words in the database and preselect the unique ones, b) as we build up the tiling we need to check if the actual word is in the preselected list of unique words.

This workflow is not practical since the number of potentially useful words (say 25 to 50 mers) is too large; it is not practical to search all of them in the database. The GSMer

DOI:10.15774/PPKE.ITK.2014.011

database uses an interesting workaround: It first establishes a set of redundant words that occur only in more than one genome. Then it makes a dense 50mer tiling for the target genome, and discards those words that contain one of the redundant words. This means that the retained 50mer will have few close neighbors since all of their subsequences are “unique” (*i.e.* have neighbors max 80% identical with them). This is thus a two step procedure and generates a set of overlapping words that are not concatenated into longer regions or “contigs”. In many cases, the number of unique sequences is too low, so the procedure has to be repeated with longer unique words. In any case, this procedure approaches the “unique regions” from a below, *i.e.* finding relatively short words with very high resolutions (dense tiling).

The approach we propose is to find the unique regions from above, *i.e.* make a sparse tiling with longer words, say 100mers, offset by 20-50 nucleotides. There are relatively few such words in a genome, *i.e.* the resulting set of words (in the range of a few hundred thousand k-mers) is small as compared to the capacity of current aligners such as Bowtie. The solution is then straightforward, we make a Bowtie search against a large comprehensive database, using Taxoner, and simply identify the k-mers that have no sequence neighbors above the level of, say 80% identity. Henceforward, we simply concatenate the words into regions wherever possible. This analysis does not take prohibitively long times, it gives long regions for strains, species or genera, *i.e.* for any desired taxonomic level.

It is useful to overview this seemingly trivial problem from the perspective of string matching. According to a naive definition, a unique region of a genome is a substring that does not occur in other database entries.

While this subsequence can be useful for finding the only database entry which it contains, we immediately have to add it as a biological marker. This “unique segment” should be common to all individual members of a strain (or species), so we cannot use simplistic terms like identity or non-identity, and we have to consider small local mutations, sequencing errors, etc.

DOI:10.15774/PPKE.ITK.2014.011

So we are better off if we use approximate string matching terms, *i.e.* a marker substring has to be at least x% identical with the genome of other organisms within its strain (or positive group) and has to be less than y% identical with the genome of other organisms (or negative group). It is easy to realize that this is a nearest neighbor definition where *distance/similarity-like* measures and thresholds are used to determine group membership (Figure 24).

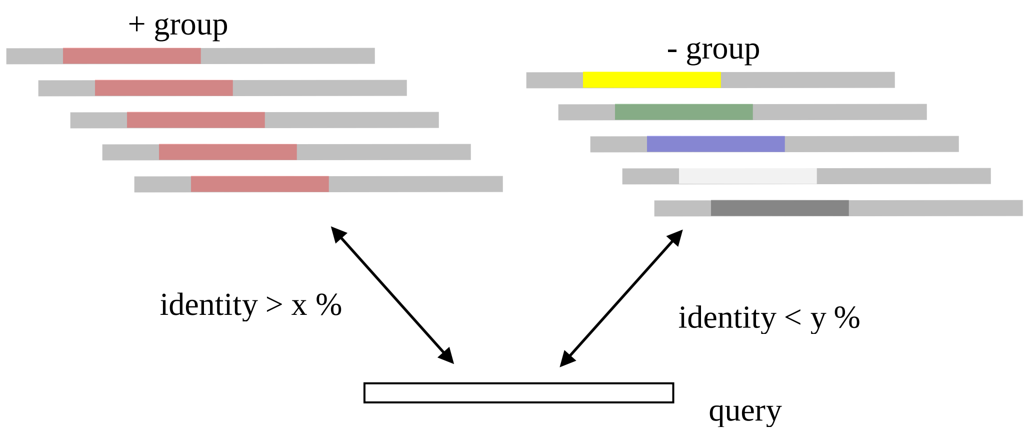


Figure 24: Logical scheme of identifying a query sequence as a marker using a nearest neighbor paradigm.

The second problem is how we define the boundaries of a unique segment. Imagine we have a unique region to start with – can we extend it and check whether or not the extended segment still obeys our marker criteria? This is certainly possible; this is essentially how BLAST finds HSPs between two sequences. For optimizing markers in an entire database this may not be computationally feasible, but it is not even important. Namely, the identity criterion is by and large arbitrary, and we can easily find regions that approximately match this criterion, using a tiling approach.

For the construction of the GSM (genome specific marker) database, a dense tiling was applied to the target genome using 50 nucleotide words, shifted by one nucleotide. Each 50mer was checked against a precomputed list of 18mer redundant words, that occurred in at least two genomes. The 50mer was considered a marker (“GSM”) if it did not contain such a redundant word, in other words one would naively expect that it was less than 18/50

= 36 % identical with anything else in the database. This is not true, however. Since a 50mer can contain two nonadjacent polymorphisms with respect to another genome, say at positions 17 and 42, respectively, which will ensure that it is still 96% identical with another sequence, but this 50mer still will be included by the GSMer procedure since it does not contain any contiguous 18mers known as redundant words.

In order to exclude such cases, the GSMer database uses a second validation step, a MEGABLAST screening that discards all 50mers that are more than 85% identical with anything else in the database. Note that this is a three-step procedure (1: Build the database of redundant words, 2: perform dense tiling of the each genome, exclude those 50mers that contain redundant words, and 3: filter the results for 85% identity using BLAST). Steps 2 and 3 have to be performed for ALL genomes, and if the number of markers is below a certain threshold (a minimum of 50 pieces of 50mer markers), they are repeated again for a redundant database of larger words.

Because of the BLAST, searches and the iterative multistep filtering are a very time consuming processes that are not particularly accurate in terms of the boundaries. Let’s imagine a single unique region of 100 residues within a genome. This region would be covered by 100 – 50 +1 = 51 completely unique markers with zero identity to other genomes, and further 7 markers would pass the 85% percent limit of the BLAST search on either side of the regions (Figure 25). This is a total of 114 GSMs for a single region (meaning 114 BLAST searches for this single region.

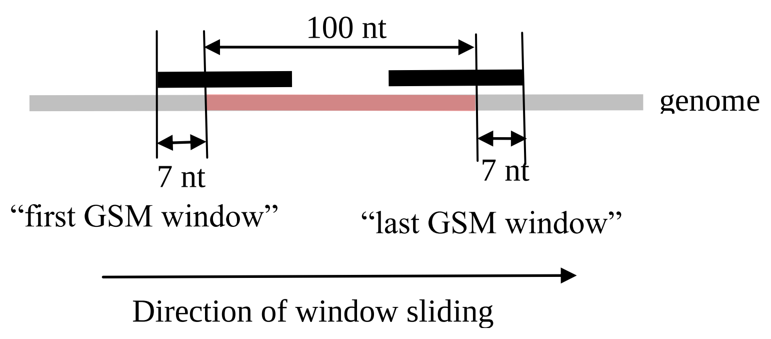


Figure 25: The outline of the GSMer procedure. A unique region of 100 nt is indicated shown in pink. Imagine sliding a window of 50nt along the sequence. The first and last windows passing the < 85% threshold are indicated by black lines.

DOI:10.15774/PPKE.ITK.2014.011

Because of the high computational load and the low accuracy of the boundaries, let's consider an alternative procedure. We apply a sparse tiling to a genome; we cover it by windows of 100 nt, offset by 20. It is easy to show (Figure 26), that wherever we place a unique region within this roster, it will be covered by 2 consecutive windows. In other words, it will be located within a region of 150nt, which is larger ("less accurate") than the one noted in Figure 25. Nevertheless, the uncertainty of the boundaries is proportional with the offset (average equals half of the offset), so the accuracy can be linearly improved as we decrease the offset.

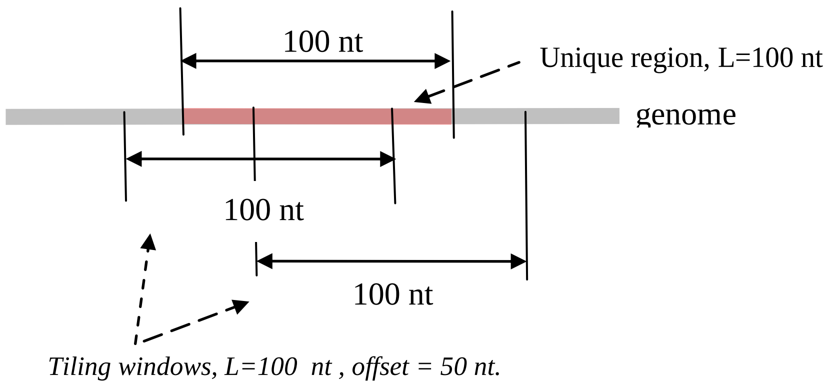


Figure 26: The outline of a sparse tiling procedure for marker identification. Note that the unique region (pink) can overlap with a minimum of one, or a maximum of two tiling windows.

Based on this consideration we suggest the following, non-iterative procedure for building a database of genome-specific markers.

1. Cover the target genome with a sparse tiling (say windows of 100nt, offset 50nt).
2. Do a similarity search using a fast aligner, such as Bowtie2, against the entire DNA dbase. Exclude windows that have neighbors above a similarity threshold (say 70-80%).
3. Concatenate overlapping windows.

The differences between this outline and the GSMer procedures are conspicuous: a) similarity search is the only identification step; there is no need for building a pre-computed database for filtering. b) A different similarity search process is used which is optimal for the purpose. Namely, genome aligners look for minimal differences (as is

DOI:10.15774/PPKE.ITK.2014.011

necessary here), while the much more time consuming BLAST programs used in GSMer look for maximal similarities. c) More similarity searches are used, but this is not a problem for the fast genome aligners. For instance, an average bacterial genome needs 100 000 Bowtie searches which allows one to process the database of complete genomes in 10 hours on a Google Cloud virtual Machine server (**n1-standard-8**, 8 virtual CPUs (2.75 GCEUs) and 30 GB of memory). In comparison, the search of testing the GSMer database on the same computer would take a minimum of 30 days.

3.3.2 Production of markers for higher level taxa or other sequence groups

It is important to realize that the identification of marker segments follows the simplified logics presented in Figure 24. In this scheme any two disjunct groups of genomes can be selected as a pair of positive/negative groups. So we can generate markers for one strain (positive group), vs. the rest of the database (negative group). But we can also generate markers for one species vs. the rest of the database – these will be species level markers. Using this definition, a species level marker is a sequence that appears in any of the genomes belonging to the species. This type of definition is a set-union (Figure 27). But conversely, we can collect marker windows that appear in all strains of a species, which corresponds to the intersection of sets.

It is important to realize that the simple picture in Figure 27 does not hold for practical situations. Namely, the union-type definition cannot be made easily non-redundant since the marker regions never completely overlap, *i.e.* the marker dataset for a species may not be much smaller than the sum of all strain-markers.

DOI:10.15774/PPKE.ITK.2014.011

On the other hand, Figure 27 illustrates two important points: i) we can define marker regions, which appear in the genome of one strain but not in the other strains (the +group in Figure 24 will be one strain, all others will be the –group). ii) The problem of strain-specific identification is not an easy one. Namely, strains within a bacterial species can differ in a large part of their genomes, so every isolate of a species can be roughly regarded as a “new strain”.

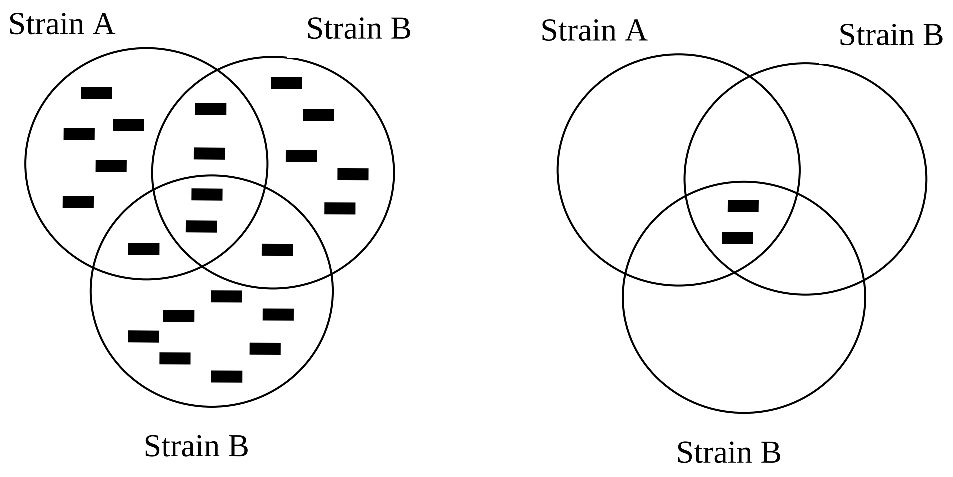


Figure 27: Two types of marker datasets, illustrated on a hypothetical species of 3 strains. Left: A set union type definition, k-mers appearing in any of the 3 strains of the species. This is a non-redundant set, i.e. k-mers appearing in more than one strain appear only once. Right: A set-intersection type definition, k-mers appearing in all 3 strains of the species. Note that this is a much smaller set.

3.3.3 Overview of the database production workflow

The marker database was obtained following the workflow shown in Figure 28. This approach can be used either to create a marker database for taxonomy binning or just to obtain unique segments of a sequence.

DOI:10.15774/PPKE.ITK.2014.011

The initial targets are the sequences in fasta format that will be processed for the unique marker extraction. The sparse tilling over these sequences is executed using the **SplitFastaFile** program included in the BioC library. This program cover the targets sequences producing overlapped reads with length and offset entered as parameters.

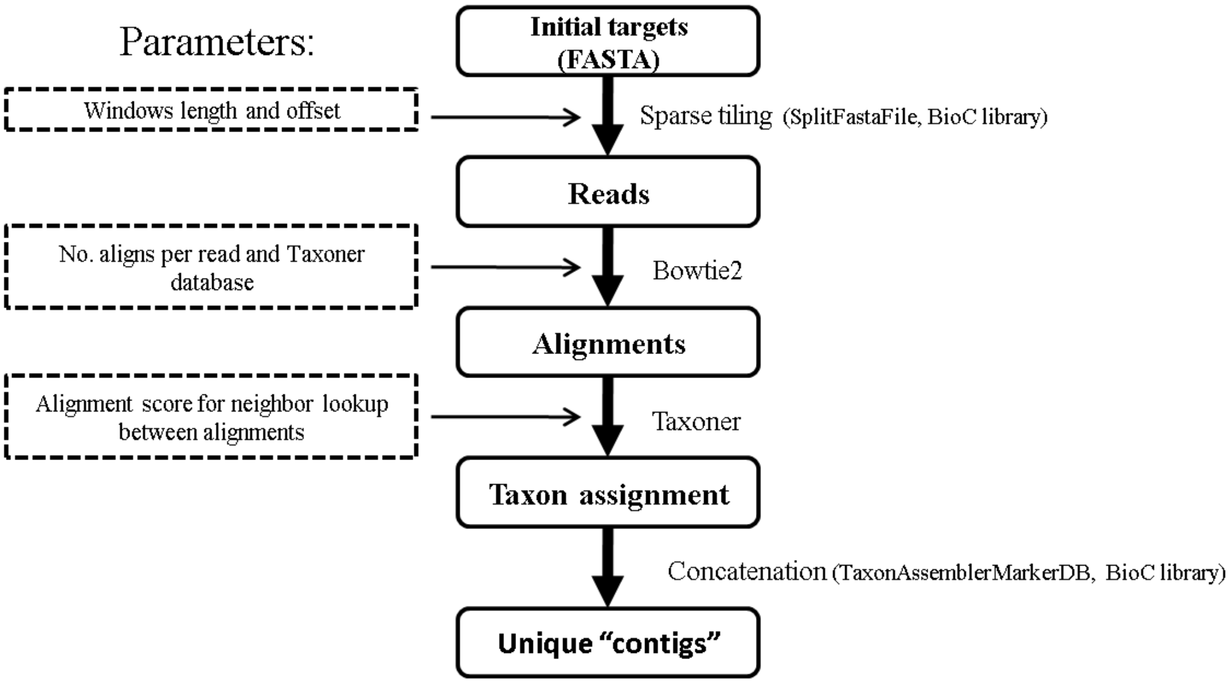

Parameters:
Windows length and offset
No. aligns per read and Taxoner database
Alignment score for neighbor lookup between alignments
Initial targets (FASTA)
Sparse tiling (SplitFastaFile, BioC library)
Reads
Bowtie2
Alignments
Taxoner
Taxon assignment
Concatenation (TaxonAssemblerMarkerDB, BioC library)
Unique "contigs"

Figure 28: Workflow for unique segments identification

The obtained reads are aligned with Bowtie2 using the Taxoner database. This procedure will create the SAM alignments of the targets reads with the database. Bowtie2 can report a number of alignments per reads (option -k) or just all of them (option -a). Reporting all alignments per reads is a time consuming process that can delay the Bowtie2 runs from a few hours to days. In our cases, we realized that results reporting all alignments and those using just the first 2000 alignments are completely equal differing just in the runtime. Therefore, the marker database was generated using the first 2000 alignments from Bowtie.

Then, the SAM files are processed by Taxoner for the taxon assignments using the score parameter as input. This score parameter is used to set the maximum percent of identities allowed between the target read and the database reads. Subsequently, from the taxon assignments point of view, having a read assigned to the strain level means that the next

DOI:10.15774/PPKE.ITK.2014.011

aligned database segment to that reads fall below the 80% of identity. Following this idea, the concatenated reads that will be assigned to any taxonomic level can be considered as unique segments for that level.

Finally, the **TaxonerAssemblerMarkerDB** program, included in the BioC library, is used to concatenate the reads using the Taxoner output creating the marker databases for the different taxonomic ranks.

It should be noted that this approach can be used to compare sequences in order to know how similar they are. For example, if the Taxoner database is created with the same targets sequences, then, the workflow align the targets sequences again themselves. Consequently, the concatenated reads that will be assigned to the different taxonomic level can be considered as unique segments for that level. This is a useful procedure that can be used to compare families, species or even genus for DNA studies.

3.3.4 Sequence comparison for the Burkholderia genus

The aforementioned workflow was used to create a marker database for the *Burkholderia* genus. Accordingly with the previous section, using the same targets sequences as Taxoner database the workflow will return the unique segments of those targets for each taxonomic level. This kind of study shows how similar the strain and species inside a genus are.

Then, 39 *Burkholderia* sequences were used from the Complete Genomes database with a total of 145,993,599 bp. We executed our workflow varying the Taxoner score from 0.05 to 0.99. As we commented previously, lower Taxoner score turn out in more specificity of the unique segments assigned to each taxonomic level.

DOI:10.15774/PPKE.ITK.2014.011

The Figure 29 shows the variation of the bp assigned to each taxonomic level against the Taxoner score. It can be seen that lower scores reduce the bp assigned to the strain (no rank) and species level, but at the same time, it increases the bp assigned to the genus and species group. Moreover, reducing the Taxoner score allows the program to collect more alignments per read. Then, for those collected alignments, the lowest common ancestor algorithm is executed assigning a taxa to the read. Therefore, we can say that the reads assigned to a taxonomic level are unique for that level if more alignments per read are collected.

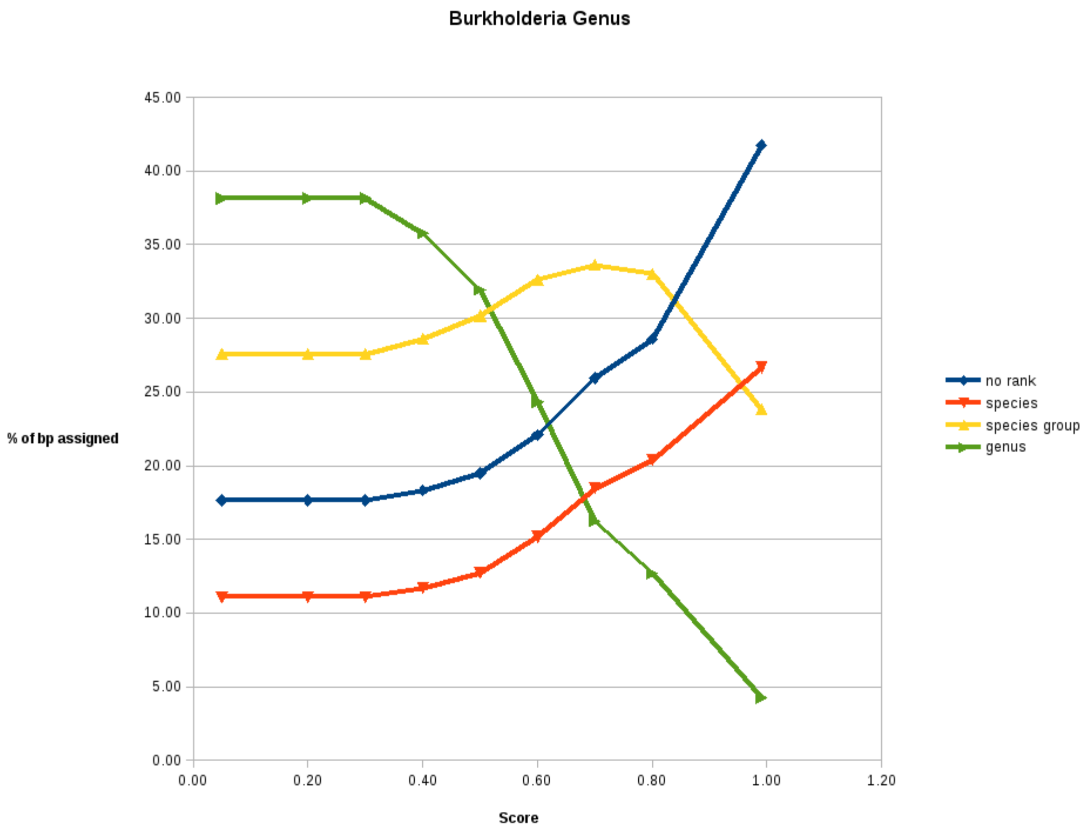


Figure 29: The figure shows the percent of bp assigned against the Taxoner score for the Burkholderia genus. After the score of 0.30 the Taxoner returns the same result.

Finally, the Table 17 shows the full distribution of bp for the Taxoner score 0.50. Here we can see the variability in the sequences among the taxonomic groups. There are special cases like *Burkholderia mallei* species which does not have any unique segment between

DOI:10.15774/PPKE.ITK.2014.011

the strains whereas, *Burkholderia rhizoxinica HKI 454* has 97.78% of unique sequences. Additionally, we can say the 32.72% of the sequences is similar for the whole genus.

DOI:10.15774/PPKE.ITK.2014.011

Table 17: The table shows the percent of bp assigned to the taxonomic levels for the *Burkholderia* genus using a Taxoner score of 0.50. Note that the table continues in the next page. They should be seen in parallel.

GENUS				SPECIES GROUP			
Taxonomy	**Length**	**bp assigned**	**% of total bp**	**Taxonomy**	**Length**	**bp assigned**	**% of species group bp**
Burkholderia	142545178	46642275	32.72	*cepacia* complex	43447291	6232175	14.34
				pseudomallei Group	58085018	37817075	65.11

DOI:10.15774/PPKE.ITK.2014.011

SPECIES				STRAIN			
Taxonomy	**Length**	**bp assigned**	**% of species bp**	**Taxonomy**	**Length**	**bp assigned**	**% of strain bp**
lata	3694126	1084400	29.35				
ambifaria	7000128	944125	13.49	ambifaria AMMD	3556545	451,675	12.70
				ambifaria MC40-6	3443583	358,000	10.40
cenocepacia	14181430	2764525	19.49	cenocepacia AU 1054	3294563	427,400	12.97
				cenocepacia HI2424	3483902	81,700	2.35
				cenocepacia J2315	3870082	747,975	19.33
				cenocepacia MC0-3	3532883	305,125	8.64
				sp. CCGE1001	4063449	566175	13.93
				sp. CCGE1002	3518940	2643600	75.12
				sp. CCGE1003	4077097	2707675	66.41
				cepacia GG4	3463655	866,475	25.02
				multivorans ATCC 17616	3448466	1,523,800	44.19
mallei	13961522	21575	0.15	mallei ATCC 23344	3510148	0	0.00
				mallei NCTC 10229	3458208	0	0.00
				mallei NCTC 10247	3495687	0	0.00
				mallei SAVP1	3497479	0	0.00
thailandensis	7776995	309250	3.98	thailandensis E264	3809201	948,500	24.90
				thailandensis MSMB121	3967794	1,127,700	28.42
				pseudomallei 668	3912947	106,875	2.73
				pseudomallei BPC006	4001777	30,075	0.75
				pseudomallei K96243	4074542	133,275	3.27
				pseudomallei MSHR305	4054155	113,300	2.79
				pseudomallei MSHR346	4098576	106,125	2.59
				pseudomallei NCTC 13179	3997089	103,300	2.58
				pseudomallei 1026b	4092668	36250	0.89
				pseudomallei 1106a	3988455	31950	0.80
				pseudomallei 1710b	4126292	120050	2.91
				sp. RPE64	3013410	2198250	72.95
				sp. YI23	3131280	2300525	73.47
				sp. KJ006	3145156	140225	4.46
				phenoliruptrix BR3459a	4152217	651,300	15.69
				phymatum STM815	3479187	3,019,225	86.78
				phytofirmans PsJN	4467537	2,771,075	62.03
				gladioli BSR3	4413616	3,439,525	77.93
				glumae BGR1	3906507	2,912,000	74.54
				vietnamiensis G4	3652814	612,075	16.76
				xenovorans LB400	4895836	3,189,000	65.14
				rhizoxinica HKI 454	2755309	2694050	97.78

DOI:10.15774/PPKE.ITK.2014.011

3.3.5 Marker database for the Complete Genomes

The Complete Genomes from the NCBI was used to generate a marker list using as Taxoner database the bacteria subset of the **nt** file. The Table 18 shows the data of the target sequences used as input in our workflow. The reads are 100 bp long with an offset of 25 bp.

Table 18: Complete genomes data downloaded from the NCBI FTP site 03/24/2014.

Target Group	Sequences	bp	No. reads
Complete genomes	5,190	9,572,092,555	127,628,722

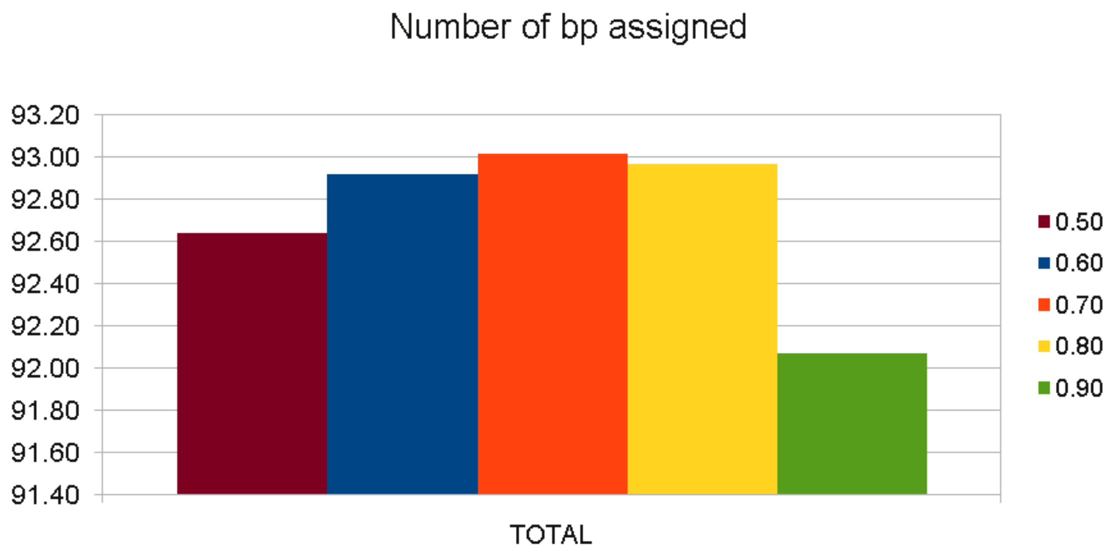


Figure 30: Percent of bp assigned per Taxoner score for the Complete Genomes using the nt file as Taxoner database.

The Figure 30 shows the percent of bp assigned using different Taxoner score values. Varying this score does not produce a considerable change in the number of bp assigned. However, as we can see in the Figure 31, the score redistribute the reads among the Taxonomic ranks depending of the percent of identity between the targets reads and the

Taxoner database. The reads are switched from the strain and species ranks to the genus when the score change from 0.90 to 0.50. As was explained before, the reduction of the Taxoner score increase the percent of identity of the reads that are assigned to any Taxonomic rank.

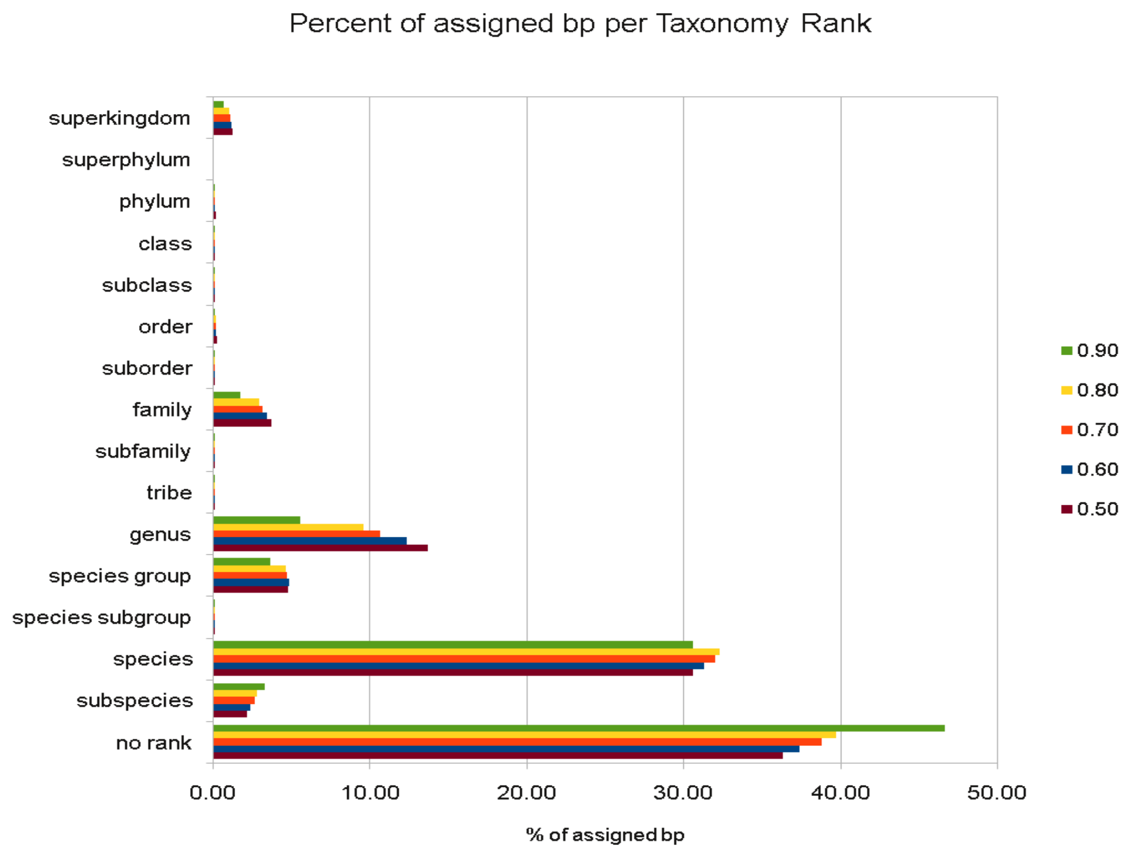


Figure 31: Distribution of the assigned bp against the Taxonomic ranks using different Taxoner scores.

The potential uses of a marker database fall in two main categories: i) Taxon identification by computational analysis. This is the approach initiated by Metaphlan [98] and adopted by other programs such as GSMer. ii) Designing diagnostic tools such as PCR primers of microarray tests. This is an important practical application since the identification of potential pathogens, microbial contaminants is crucial in many areas. With these areas in mind, we developed a server application http://pongor.itk.ppke.hu/markerdb that has the following functionalities:

DOI:10.15774/PPKE.ITK.2014.011

a) Defining marker sequences unique for bacterial groups, *i.e.* finding markers that are diagnostic for a strain, or a group of strains. This task can include running primer selection algorithms on various subgroups of the marker database.
b) Defining differential markers that can distinguish (group of) sequences from another group of sequences. This is important when we try to detect one particular (group of) strain(s) in the present of closely related strains.

3.3.6 Run times and space complexity

Creating a marker database from genomes sequences can be time consuming even if powerful computers are used. Table 19 shows the total time for different set of input data. It should be notice that in the case of general databases like NCBI NT a pre-filter step is needed in order to extract the target taxonomies.

The workflow's space complexity depends on the different stages. The biggest space, both RAM and hard disk is required by Bowtie when it is doing the alignments. It requires 8.0 GB of RAM minimum and 500 GB of disk space.

Table 19: Run times for the creation of the Marker database for different set of input data

Input DB	Initial Size (GB)	Selecting microbial entries (Hour)	Creation of tiling segments (Hour)	Creation of Taxoner-Bowtie DB (Hour)	Identifying unique segments, Taxoner (Hour)	Concatenation of unique segments (Hour)	Total Time (Hour)
NCBI nt	**60**	4	6	4	18	6	**42**
Complete Bacterial Genomes	**9.4**	-	1.5	2	9	3	**15.5**
Burkholderia genus	**0.14**	-	0.1	0.1	0.002	0.1	**0.3**

3.3.7 Summary

During this section we presented a workflow for the generation of a DNA marker database from (group of) sequences and a Taxoner database. This workflow allows the identification of unique DNA segments among the (group of) sequences targets. Also, it can be use for

DOI:10.15774/PPKE.ITK.2014.011

sequence comparison from taxonomic groups providing a taxa classification of DNA segments.

The workflow uses a group of *in-house* developed programs and the Bowtie2 aligner. The programs developed by our group are freely available through the Google Code Platform.

We show a complete comparison for the Burkholderia genus using the workflow and the percentages for each taxonomic level.

Additionally, a marker database was created from the Complete Genomes from the NCBI. This database is freely available through the project web site.

Finally, a web server was developed to compare groups of sequences using an automatic pipeline based on the workflow aforementioned.

DOI:10.15774/PPKE.ITK.2014.011

4 <u>Conclusion</u>

The integration and fast processing of biomolecular data are crucial topics for the Life Sciences. An avalanche of data is generated continuously by the new experimental technologies producing new kinds of data or simply modifying the existent one. Next-generation sequencing (NGS) technologies can sequence the complete genome of isolated organisms or complex mix of them at a very low cost. It is becoming a standard approach to detect individual species or pathogenic strains of microorganisms. Computer programs used in the NGS community have to balance between speed and sensitivity and as a result, species or strain level identification is often inaccurate and low abundance pathogens can sometimes be missed. Parallel to that, Bioinformatics' tools are changing slowly its way of access, manage and integrate the upcoming data but this evolution is not fast enough.

In this thesis, we presented a) an open-source framework for biological data integration (**JBioWH**), b) a pipeline of programs (**Taxoner**) for taxonomic binning or metagenomics analysis of complex mix of NGS data and c) a workflow for DNA sequencing comparison that can be used for the generation of marker databases or just for identification of unique DNA segments from a group of target sequences.

The **JBioWH** framework is a mature computational system freely available that can be used to answer complex biological questions, or just, as supplier system of integrative data to others client applications. It can be used for intensive querying of multiple data sources and the creation of streamlined task-specific data sets on local PCs.

JBioWH is based on a MySQL relational database and provides four kind of access to the integrated data: a) direct access to the relational schema (SQL), b) programmatically access through the Java API (java persistence model and search classes), c) graphical access through the Desktop Client and d) html access through the webservices (JSON and XML).

The system has a modular design that can be easily modified accordingly to the biological context of the problem. Therefore, **JBioWH** can be tailored for use in specific circumstances, including the handling of massive queries for high-throughput analyses or CPU intensive

DOI:10.15774/PPKE.ITK.2014.011

calculations. At present, **JBioWH** contain parsers for retrieving data from 24 public databases (e.g. NCBI, KEGG, etc) [223].

Finally, the **JBioWH** framework has been used by several Bioinformatics projects associated to different OMICS disciplines like genomics, proteomics and drug design.

Taxoner is a pipeline of programs designed to perform taxonomic binning or metagenomics analysis of NGS data. Its main advantage over the equivalent programs is related to the correct identification of unknown strains. Therefore, this approach can be used for the detection of hazardous pathogens that requires strain level identification. Multiple datasets were analyzed by **Taxoner** showing its advantages over the rest of available programs. When applied to metagenomic datasets, **Taxoner** can provide a functional summary of the genes mapped and can provide strain level identification as shown [224]. It is much faster and at times more accurate than BLAST-based evaluation schemes as those used by the MEGAN program meaning that it can be run on desktop or laptop computers. The Taxoner source code is freely available and a demo server was published for demonstration porpoises.

Finally, a workflow for DNA sequencing comparison and DNA markers identification, based on *in-house* developed programs, that include **JBioWH** and **Taxoner,** was presented. DNA markers are unique nucleotide sequences allowing the detection of certain organisms and to distinguish those organisms from all other species, using *in silico* or experimental technologies. Markers can be used as the basis for diagnostic assays to detect microbes in environmental or clinical samples.

The workflow developed was used to study sequences similarities among the complete genomes of the *Burkholderia* genus. As a result, taxonomic levels were assigned to unique DNA segments of the members of this genus. This study shows the variability in the sequences among the taxonomic groups. There are special cases like *Burkholderia mallei* species which does not contain a single unique segment in their strains and on the other hand, *Burkholderia rhizoxinica HKI 454* has 97.78% of unique sequences. Additionally, we can say that the 32.72% of the sequences is similar for the whole genus.

Lastly, a marker database was generated from the NCBI Complete Genomes using this workflow. 5190 sequences were included, they generated 127,628,722 reads after the windows

DOI:10.15774/PPKE.ITK.2014.011

tiling. The study showed that the number of bp assigned by **Taxoner** does not change with the variation of the Taxoner's score. Just the distributions of the DNA segments among the taxonomic levels change accordingly with the percent of identity between the reads and the Taxoner database. The workflow description, programs and flat files of the marker database are freely available. Also, a web site was published for demonstration purposes.

DOI:10.15774/PPKE.ITK.2014.011

DOI:10.15774/PPKE.ITK.2014.011

5 References

1. Kuzniar A, Lin K, He Y, Nijveen H, Pongor S, Leunissen JAM: **ProGMap: an integrated annotation resource for protein orthology.** *Nucleic acids research* 2009, **37**:W428-W434.
2. Longtin R: **An Integrated Approach : Systems Biology Seeks Order in Complexity**. *Journal of the National Cancer Institute* 2005, **97**:6-8.
3. Medina MÁ: **Systems biology for molecular life sciences and its impact in biomedicine.** *Cellular and molecular life sciences : CMLS* 2013, **70**:1035-1053.
4. Sauer U, Heinemann M, Zamboni N: **Genetics. Getting closer to the whole picture.** *Science (New York, NY)* 2007, **316**:550-551.
5. Joyce AR, Palsson BØ: **The model organism as a system: integrating 'omics' data sets.** *Nature reviews Molecular cell biology* 2006, **7**:198-210.
6. Chuang H-Y, Hofree M, Ideker T: **A decade of systems biology.** *Annual review of cell and developmental biology* 2010, **26**:721-744.
7. Li H, Homer N: **A survey of sequence alignment algorithms for next-generation sequencing**. *Briefings in bioinformatics* 2010, **11**(5):473-483.
8. Metzker ML: **Sequencing technologies - the next generation**. *Nature reviews Genetics* 2010, **11**(1):31-46.
9. Malhis N, Butterfield YS, Ester M, Jones SJ: **Slider--maximum use of probability information for alignment of short sequence reads and SNP detection**. *Bioinformatics* 2009, **25**(1):6-13.
10. Malhis N, Jones SJ: **High quality SNP calling using Illumina data at shallow coverage**. *Bioinformatics* 2010, **26**(8):1029-1035.
11. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ: **Basic local alignment search tool**. *Journal of molecular biology* 1990, **215**:403-410.
12. Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ: **Gapped BLAST and PSI-BLAST: a new generation of protein database search programs**. *Nucleic acids research* 1997, **25**(17):3389-3402.
13. Smith TF, Waterman MS: **Identification of common molecular subsequences.** *Journal of molecular biology* 1981, **147**:195-197.
14. Gotoh O: **An improved algorithm for matching biological sequences**. *Journal of molecular biology* 1982, **162**(3):705-708.
15. Li M, Ma B, Kisman D, Tromp J: **Patternhunter II: highly sensitive and fast homology search**. *Journal of bioinformatics and computational biology* 2004, **2**(3):417-439.
16. Ma B, Tromp J, Li M: **PatternHunter: faster and more sensitive homology search**. *Bioinformatics* 2002, **18**(3):440-445.
17. Wu TD, Nacu S: **Fast and SNP-tolerant detection of complex variants and splicing in short reads**. *Bioinformatics* 2010, **26**(7):873-881.
18. Homer N, Merriman B, Nelson SF: **BFAST: an alignment tool for large scale genome resequencing**. *PloS one* 2009, **4**(11):e7767.
19. Li R, Li Y, Kristiansen K, Wang J: **SOAP: short oligonucleotide alignment program**. *Bioinformatics* 2008, **24**(5):713-714.

20. Jiang H, Wong WH: **SeqMap: mapping massive amount of oligonucleotides to the genome**. *Bioinformatics* 2008, **24**(20):2395-2396.
21. Li H, Ruan J, Durbin R: **Mapping short DNA sequencing reads and calling variants using mapping quality scores**. *Genome research* 2008, **18**(11):1851-1858.
22. Rumble SM, Lacroute P, Dalca AV, Fiume M, Sidow A, Brudno M: **SHRiMP: accurate mapping of short color-space reads**. *PLoS computational biology* 2009, **5**(5):e1000386.
23. Weese D, Holtgrewe M, Reinert K: **RazerS 3: faster, fully sensitive read mapping**. *Bioinformatics* 2012, **28**(20):2592-2599.
24. Weese D, Emde AK, Rausch T, Doring A, Reinert K: **RazerS--fast read mapping with sensitivity control**. *Genome research* 2009, **19**(9):1646-1654.
25. Burkhardt S, Kärkkäinen J: **Better Filtering with Gapped q-Grams**. *Fundamenta Informaticae* 2003, **56**(1):51-70.
26. Jokinen P, Ukkonen E: **Two algorithms for approxmate string matching in static texts**. In: *Mathematical Foundations of Computer Science 1991.* Edited by Tarlecki A, vol. 520: Springer Berlin Heidelberg; 1991: 240-248.
27. Cao X, Li S, Tung AH: **Indexing DNA Sequences Using q-Grams**. In: *Database Systems for Advanced Applications.* Edited by Zhou L, Ooi B, Meng X, vol. 3453: Springer Berlin Heidelberg; 2005: 4-16.
28. Farrar M: **Striped Smith-Waterman speeds database searches six times over other SIMD implementations**. *Bioinformatics* 2007, **23**(2):156-161.
29. Slater GS, Birney E: **Automated generation of heuristics for biological sequence comparison**. *BMC bioinformatics* 2005, **6**:31.
30. Eppstein D, Galil Z, Giancarlo R, Italiano GF: **Sparse dynamic programming**. In: *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms; San Francisco, California, USA*. 320238: Society for Industrial and Applied Mathematics 1990: 513-522.
31. Myers E: **AnO(ND) difference algorithm and its variations**. *Algorithmica* 1986, **1**(1-4):251-266.
32. Abouelhoda MI, Kurtz S, Ohlebusch E: **Replacing suffix trees with enhanced suffix arrays**. *Journal of Discrete Algorithms* 2004, **2**(1):53-86.
33. Ferragina P, Manzini G: **Opportunistic data structures with applications**. In: *Foundations of Computer Science, 2000 Proceedings 41st Annual Symposium on: 2000 2000*. 390-398.
34. Burrows M, Wheeler DJ: **A block-sorting lossless data compression algorithm**. In: *Technical report 124.* Digital Equipment Corporation, Palo Alto CA; 1994.
35. Huffman DA: **A Method for the Construction of Minimum-Redundancy Codes**. *Proceedings of the IRE* 1952:1098–1102.
36. Li H, Durbin R: **Fast and accurate short read alignment with Burrows-Wheeler transform**. *Bioinformatics* 2009, **25**(14):1754-1760.
37. Langmead B, Trapnell C, Pop M, Salzberg SL: **Ultrafast and memory-efficient alignment of short DNA sequences to the human genome**. *Genome biology* 2009, **10**(3):R25.
38. Miller JR, Koren S, Sutton G: **Assembly algorithms for next-generation sequencing data**. *Genomics* 2010, **95**(6):315-327.
39. Staden R: **A strategy of DNA sequencing employing computer programs**. *Nucleic acids research* 1979, **6**(7):2601-2610.

DOI:10.15774/PPKE.ITK.2014.011

40. Myers EW, Sutton GG, Delcher AL, Dew IM, Fasulo DP, Flanigan MJ, Kravitz SA, Mobarry CM, Reinert KH, Remington KA *et al*: **A whole-genome assembly of Drosophila**. *Science* 2000, **287**(5461):2196-2204.
41. Jaffe DB, Butler J, Gnerre S, Mauceli E, Lindblad-Toh K, Mesirov JP, Zody MC, Lander ES: **Whole-genome sequence assembly for mammalian genomes: Arachne 2**. *Genome research* 2003, **13**(1):91-96.
42. Batzoglou S, Jaffe DB, Stanley K, Butler J, Gnerre S, Mauceli E, Berger B, Mesirov JP, Lander ES: **ARACHNE: a whole-genome shotgun assembler**. *Genome research* 2002, **12**(1):177-189.
43. Huang X, Yang SP: **Generating a genome assembly with PCAP**. *Current protocols in bioinformatics / editoral board, Andreas D Baxevanis [et al]* 2005, **Chapter 11**:Unit11 13.
44. Jorde L, Little P, Dunn M, Subramaniam S: **Encyclopedia of Genetics, Genomics, Proteomics and Bioinformatics**; 2005.
45. Margulies M, Egholm M, Altman WE, Attiya S, Bader JS, Bemben LA, Berka J, Braverman MS, Chen YJ, Chen Z *et al*: **Genome sequencing in microfabricated high-density picolitre reactors**. *Nature* 2005, **437**(7057):376-380.
46. Hernandez D, Francois P, Farinelli L, Osteras M, Schrenzel J: **De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer**. *Genome research* 2008, **18**(5):802-809.
47. Pop M: **Genome assembly reborn: recent computational challenges**. *Briefings in bioinformatics* 2009, **10**(4):354-366.
48. Pevzner PA, Tang H: **Fragment assembly with double-barreled data**. *Bioinformatics* 2001, **17 Suppl 1**:S225-233.
49. Zhang Y, Waterman MS: **An Eulerian path approach to local multiple alignment for DNA sequences**. *Proceedings of the National Academy of Sciences of the United States of America* 2005, **102**(5):1285-1290.
50. Pevzner PA, Tang H, Tesler G: **De novo repeat classification and fragment assembly**. *Genome research* 2004, **14**(9):1786-1796.
51. Zerbino DR, Birney E: **Velvet: Algorithms for de novo short read assembly using de Bruijn graphs**. *Genome research* 2008, **18**(5):821-829.
52. Zerbino DR, McEwen GK, Margulies EH, Birney E: **Pebble and rock band: heuristic resolution of repeats and scaffolding in the velvet short-read de novo assembler**. *PloS one* 2009, **4**(12):e8407.
53. Simpson JT, Wong K, Jackman SD, Schein JE, Jones SJ, Birol I: **ABySS: a parallel assembler for short read sequence data**. *Genome research* 2009, **19**(6):1117-1123.
54. Maccallum I, Przybylski D, Gnerre S, Burton J, Shlyakhter I, Gnirke A, Malek J, McKernan K, Ranade S, Shea TP *et al*: **ALLPATHS 2: small genomes assembled accurately and with high continuity from short paired reads**. *Genome biology* 2009, **10**(10):R103.
55. Li R, Zhu H, Ruan J, Qian W, Fang X, Shi Z, Li Y, Li S, Shan G, Kristiansen K *et al*: **De novo assembly of human genomes with massively parallel short read sequencing**. *Genome research* 2010, **20**(2):265-272.
56. Ellegren H: **Genome sequencing and population genomics in non-model organisms.** *Trends in ecology & evolution* 2014, **29**:51-63.

DOI:10.15774/PPKE.ITK.2014.011

57. Hwang D, Rust AG, Ramsey S, Smith JJ, Leslie DM, Weston AD, de Atauri P, Aitchison JD, Hood L, Siegel AF *et al*: **A data integration methodology for systems biology.** *Proceedings of the National Academy of Sciences of the United States of America* 2005, **102**:17296-17301.
58. Hwang D, Smith JJ, Leslie DM, Weston AD, Rust AG, Ramsey S, de Atauri P, Siegel AF, Bolouri H, Aitchison JD *et al*: **A data integration methodology for systems biology: experimental verification.** *Proceedings of the National Academy of Sciences of the United States of America* 2005, **102**:17302-17307.
59. Toomula N, Kumar A, Kumar D S, Bheemidi VS: **Biological Databases- Integration of Life Science Data**. *Journal of Computer Science & Systems Biology* 2012, **04**:87-92.
60. Heath AP, Kavraki LE: **Computational challenges in systems biology**. *Computer Science Review* 2009, **3**:1-17.
61. von Mering C, Bork P: **Teamed up for transcription.** In: *Nature.* vol. 417; 2002: 797-798.
62. Date CJ: **An Introduction to Database Systems (8th Edition)**. 2003:1024.
63. Fernández-Suárez XM, Rigden DJ, Galperin MY: **The 2014 Nucleic Acids Research Database Issue and an updated NAR online Molecular Biology Database Collection.** *Nucleic acids research* 2013:1-6.
64. Wren JD, Bateman A: **Databases, data tombs and dust in the wind.** *Bioinformatics (Oxford, England)* 2008, **24**:2127-2128.
65. Fleischmann RD, Adams MD, White O, Clayton RA, Kirkness EF, Kerlavage AR, Bult CJ, Tomb JF, Dougherty BA, Merrick JM *et al*: **Whole-genome random sequencing and assembly of Haemophilus influenzae Rd**. *Science* 1995, **269**:496-512.
66. Coordinators NR: **Database resources of the National Center for Biotechnology Information.** *Nucleic acids research* 2012, **41**:8-20.
67. Hall N: **Advanced sequencing technologies and their wider impact in microbiology.** *The Journal of experimental biology* 2007, **210**:1518-1525.
68. Benson Da, Cavanaugh M, Clark K, Karsch-Mizrachi I, Lipman DJ, Ostell J, Sayers EW: **GenBank.** *Nucleic acids research* 2013, **41**:D36-42.
69. Kosuge T, Mashima J, Kodama Y, Fujisawa T, Kaminuma E, Ogasawara O, Okubo K, Takagi T, Nakamura Y: **DDBJ progress report: a new submission system for leading to a correct annotation.** *Nucleic acids research* 2013:gkt1066-.
70. Pakseresht N, Alako B, Amid C, Cerdeño-Tárraga A, Cleland I, Gibson R, Goodgame N, Gur T, Jang M, Kay S *et al*: **Assembly information services in the European Nucleotide Archive.** *Nucleic acids research* 2013, **42**:D38-D43.
71. Maglott D, Ostell J, Pruitt KD, Tatusova T: **Entrez Gene: gene-centered information at NCBI.** *Nucleic acids research* 2011, **39**:D52-D57.
72. Sayers EW, Barrett T, Benson DA, Bolton E, Bryant SH, Canese K, Chetvernin V, Church DM, DiCuccio M, Federhen S *et al*: **Database resources of the National Center for Biotechnology Information.** *Nucleic acids research* 2011, **39**:D38-D51.
73. Kanehisa M, Goto S, Sato Y, Kawashima M, Furumichi M, Tanabe M: **Data, information, knowledge and principle: back to metabolism in KEGG.** *Nucleic acids research* 2013:1-7.
74. Pagani I, Liolios K, Jansson J, Chen I-Ma, Smirnova T, Nosrat B, Markowitz VM, Kyrpides NC: **The Genomes OnLine Database (GOLD) v.4: status of genomic and**

DOI:10.15774/PPKE.ITK.2014.011

metagenomic projects and their associated metadata. *Nucleic acids research* 2012, **40**:D571-579.

75. Flicek P, Amode MR, Barrell D, Beal K, Billis K, Brent S, Carvalho-Silva D, Clapham P, Coates G, Fitzgerald S *et al*: **Ensembl 2014.** *Nucleic acids research* 2014, **42**:D749-755.
76. Field D, Amaral-Zettler L, Cochrane G, Cole JR, Dawyndt P, Garrity GM, Gilbert J, Glöckner FO, Hirschman L, Karsch-Mizrachi I *et al*: **The Genomic Standards Consortium.** *PLoS biology* 2011, **9**:e1001088.
77. Cole JR, Wang Q, Fish Ja, Chai B, McGarrell DM, Sun Y, Brown CT, Porras-Alfaro A, Kuske CR, Tiedje JM: **Ribosomal Database Project: data and tools for high throughput rRNA analysis.** *Nucleic acids research* 2014, **42**:D633-642.
78. Kozomara A, Griffiths-Jones S: **miRBase: annotating high confidence microRNAs using deep sequencing data.** *Nucleic acids research* 2014, **42**:D68-73.
79. Bartel DP, Lee R, Feinbaum R: **MicroRNAs : Genomics , Biogenesis , Mechanism , and Function Genomics : The miRNA Genes**. *Cell* 2004, **116**:281-297.
80. Li Y, Qiu C, Tu J, Geng B, Yang J, Jiang T, Cui Q: **HMDD v2.0: a database for experimentally supported human microRNA and disease associations.** *Nucleic acids research* 2014, **42**:D1070-1074.
81. Paraskevopoulou MD, Georgakilas G, Kostoulas N, Reczko M, Maragkakis M, Dalamagas TM, Hatzigeorgiou AG: **DIANA-LncBase: experimentally verified and computationally predicted microRNA targets on long non-coding RNAs.** *Nucleic acids research* 2013, **41**:D239-245.
82. Nagaswamy U, Larios-Sanz M, Hury J, Collins S, Zhang Z, Zhao Q, Fox GE: **NCIR: a database of non-canonical interactions in known RNA structures.** *Nucleic acids research* 2002, **30**:395-397.
83. Handelsman J, Rondon MR, Brady SF, Clardy J, Goodman RM: **Molecular biological access to the chemistry of unknown soil microbes: a new frontier for natural products**. *Chemistry & biology* 1998, **5**(10):R245-249.
84. Prakash T, Taylor TD: **Functional assignment of metagenomic data: challenges and applications**. *Briefings in bioinformatics* 2012, **13**(6):711-727.
85. Petrosino JF, Highlander S, Luna RA, Gibbs RA, Versalovic J: **Metagenomic pyrosequencing and microbial identification**. *Clinical chemistry* 2009, **55**(5):856-866.
86. Riesenfeld CS, Schloss PD, Handelsman J: **Metagenomics: genomic analysis of microbial communities**. *Annual review of genetics* 2004, **38**:525-552.
87. Wooley JC, Godzik A, Friedberg I: **A primer on metagenomics**. *PLoS computational biology* 2010, **6**(2):e1000667.
88. Kunin V, Copeland A, Lapidus A, Mavromatis K, Hugenholtz P: **A bioinformatician's guide to metagenomics**. *Microbiology and molecular biology reviews : MMBR* 2008, **72**(4):557-578, Table of Contents.
89. Williamson SJ, Rusch DB, Yooseph S, Halpern AL, Heidelberg KB, Glass JI, Andrews-Pfannkoch C, Fadrosh D, Miller CS, Sutton G *et al*: **The Sorcerer II Global Ocean Sampling Expedition: metagenomic characterization of viruses within aquatic microbial samples**. *PloS one* 2008, **3**(1):e1456.
90. Mavromatis K, Ivanova N, Barry K, Shapiro H, Goltsman E, McHardy AC, Rigoutsos I, Salamov A, Korzeniewski F, Land M *et al*: **Use of simulated data sets to evaluate the fidelity of metagenomic processing methods**. *Nat Meth* 2007, **4**(6):495-500.

91. Wommack KE, Bhavsar J, Ravel J: **Metagenomics: read length matters**. *Applied and environmental microbiology* 2008, **74**(5):1453-1463.
92. Nakamura S, Yang CS, Sakon N, Ueda M, Tougan T, Yamashita A, Goto N, Takahashi K, Yasunaga T, Ikuta K *et al*: **Direct metagenomic detection of viral pathogens in nasal and fecal specimens using an unbiased high-throughput sequencing approach**. *PloS one* 2009, **4**(1):e4219.
93. Breitbart M, Hewson I, Felts B, Mahaffy JM, Nulton J, Salamon P, Rohwer F: **Metagenomic analyses of an uncultured viral community from human feces**. *Journal of bacteriology* 2003, **185**(20):6220-6223.
94. Carlton RM, Noordman WH, Biswas B, de Meester ED, Loessner MJ: **Bacteriophage P100 for control of Listeria monocytogenes in foods: genome sequence, bioinformatic analyses, oral toxicity study, and application**. *Regulatory toxicology and pharmacology : RTP* 2005, **43**(3):301-312.
95. Mohammed MH, Ghosh TS, Singh NK, Mande SS: **SPHINX--an algorithm for taxonomic binning of metagenomic sequences**. *Bioinformatics* 2011, **27**(1):22-30.
96. Droge J, McHardy AC: **Taxonomic binning of metagenome samples generated by next-generation sequencing technologies**. *Briefings in bioinformatics* 2012, **13**(6):646-655.
97. Huson DH, Richter DC, Mitra S, Auch AF, Schuster SC: **Methods for comparative metagenomics**. *BMC Bioinformatics* 2009, **10 Suppl 1**:S12.
98. Segata N, Waldron L, Ballarini A, Narasimhan V, Jousson O, Huttenhower C: **Metagenomic microbial community profiling using unique clade-specific marker genes**. *Nature methods* 2012, **9**(8):811-814.
99. Koslicki D, Foucart S, Rosen G: **WGSQuikr: fast whole-genome shotgun metagenomic classification**. *PloS one* 2014, **9**(3):e91784.
100. Donoho DL: **For most large underdetermined systems of linear equations the minimal** *Communications on Pure and Applied Mathematics* 2006, **59**(6):797-829.
101. Langmead B, Salzberg SL: **Fast gapped-read alignment with Bowtie 2**. *Nat Methods* 2012, **9**(4):357-359.
102. Li H, Durbin R: **Fast and accurate long-read alignment with Burrows-Wheeler transform**. *Bioinformatics* 2010, **26**(5):589-595.
103. Hach F, Hormozdiari F, Alkan C, Birol I, Eichler EE, Sahinalp SC: **mrsFAST: a cache-oblivious algorithm for short-read mapping**. *Nat Methods* 2010, **7**(8):576-577.
104. Huson DH, Auch AF, Qi J, Schuster SC: **MEGAN analysis of metagenomic data**. *Genome Res* 2007, **17**(3):377-386.
105. Schloss PD, Westcott SL, Ryabin T, Hall JR, Hartmann M, Hollister EB, Lesniewski RA, Oakley BB, Parks DH, Robinson CJ *et al*: **Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities**. *Appl Environ Microbiol* 2009, **75**(23):7537-7541.
106. Monzoorul Haque M, Ghosh TS, Komanduri D, Mande SS: **SOrt-ITEMS: Sequence orthology based approach for improved taxonomic estimation of metagenomic sequences**. *Bioinformatics* 2009, **25**(14):1722-1730.
107. Haynes PA, Gygi SP, Figeys D, Aebersold R: **Proteome analysis: biological assay or data archive?** *Electrophoresis* 1998, **19**:1862-1871.
108. Miao Q, Zhang C-C, Kast J: **Chemical proteomics and its impact on the drug discovery process.** *Expert review of proteomics* 2012, **9**:281-291.

109. Aebersold R, Mann M: **Mass spectrometry-based proteomics**. *Nature* 2003, **422**:198-207.
110. Webb-Robertson BJ, Cannon WR: **Current trends in computational inference from mass spectrometry-based proteomics**. *Briefings in bioinformatics* 2007, **8**(5):304-317.
111. Consortium TU: **Activities at the Universal Protein Resource (UniProt).** *Nucleic acids research* 2014, **42**:D191-198.
112. Pruitt KD, Maglott DR: **RefSeq and LocusLink: NCBI gene-centered resources.** *Nucleic acids research* 2001, **29**:137-140.
113. Rose PW, Bi C, Bluhm WF, Christie CH, Dimitropoulos D, Dutta S, Green RK, Goodsell DS, Prlic A, Quesada M *et al*: **The RCSB Protein Data Bank: new resources for research and education.** *Nucleic acids research* 2013, **41**:D475-482.
114. Berman H, Henrick K, Nakamura H, Markley JL: **The worldwide Protein Data Bank (wwPDB): ensuring a single, uniform archive of PDB data.** *Nucleic acids research* 2007, **35**:D301-D303.
115. Suzek BE, Huang H, McGarvey P, Mazumder R, Wu CH: **UniRef: comprehensive and non-redundant UniProt reference clusters.** *Bioinformatics* 2007, **23**:1282-1288.
116. Finn RD, Bateman A, Clements J, Coggill P, Eberhardt RY, Eddy SR, Heger A, Hetherington K, Holm L, Mistry J *et al*: **Pfam: the protein families database.** *Nucleic acids research* 2014, **42**:D222-230.
117. Fitch WM: **Distinguishing homologous from analogous proteins.** *Systematic zoology* 1970, **19**:99-113.
118. Tatusov RL: **A Genomic Perspective on Protein Families**. *Science* 1997, **278**:631-637.
119. Jensen LJ, Julien P, Kuhn M, von Mering C, Muller J, Doerks T, Bork P: **eggNOG: automated construction and annotation of orthologous groups of genes.** *Nucleic acids research* 2008, **36**:D250-D254.
120. Powell S, Forslund K, Szklarczyk D, Trachana K, Roth A, Huerta-Cepas J, Gabaldón T, Rattei T, Creevey C, Kuhn M *et al*: **eggNOG v4.0: nested orthology inference across 3686 organisms.** *Nucleic acids research* 2014, **42**:D231-239.
121. Hatzimanikatis V, Li C, Ionita Ja, Henry CS, Jankowski MD, Broadbelt LJ: **Exploring the diversity of complex metabolic networks.** *Bioinformatics (Oxford, England)* 2005, **21**:1603-1609.
122. Griffin JL, Bollard ME: **Metabonomics: its potential as a tool in toxicology for safety assessment and data integration**. *Curr Drug Metab* 2004, **5**:389-398.
123. Yang GX, Li X, Snyder M: **Investigating metabolite–protein interactions: An overview of available techniques**. *Methods* 2012, **57**:459-466.
124. Chatr-Aryamontri A, Breitkreutz B-J, Heinicke S, Boucher L, Winter A, Stark C, Nixon J, Ramage L, Kolas N, O'Donnell L *et al*: **The BioGRID interaction database: 2013 update.** *Nucleic acids research* 2013, **41**:D816-823.
125. Stark C, Breitkreutz B-j, Reguly T, Boucher L, Breitkreutz A, Tyers M: **BioGRID a general repository for interaction datasets**. *Nucleic acids research* 2006, **34**:535-539.
126. Kerrien S, Aranda B, Breuza L, Bridge A, Broackes-Carter F, Chen C, Duesbury M, Dumousseau M, Feuermann M, Hinz U *et al*: **The IntAct molecular interaction database in 2012.** *Nucleic acids research* 2012, **40**:D841-846.
127. Orchard S, Ammari M, Aranda B, Breuza L, Briganti L, Broackes-Carter F, Campbell NH, Chavali G, Chen C, Del-Toro N *et al*: **The MIntAct project--IntAct as a common**

DOI:10.15774/PPKE.ITK.2014.011

curation platform for 11 molecular interaction databases. *Nucleic acids research* 2013:1-6.

128. Chatr-aryamontri A, Ceol A, Palazzi LM, Nardelli G, Schneider MV, Castagnoli L, Cesareni G: **MINT: the Molecular INTeraction database**. *Nucleic acids research* 2007, **35**:D572-D574.
129. Licata L, Briganti L, Peluso D, Perfetto L, Iannuccelli M, Galeota E, Sacco F, Palma A, Nardozza AP, Santonico E *et al*: **MINT, the molecular interaction database: 2012 update.** *Nucleic acids research* 2012, **40**:D857-861.
130. Salwinski L, Miller CS, Smith AJ, Pettit FK, Bowie JU, Eisenberg D: **The Database of Interacting Proteins: 2004 update**. *Nucleic acids research* 2004, **32**:D449-D451.
131. Croft D, Mundo AF, Haw R, Milacic M, Weiser J, Wu G, Caudy M, Garapati P, Gillespie M, Kamdar MR *et al*: **The Reactome pathway knowledgebase.** *Nucleic acids research* 2013:1-6.
132. Karp PD, Ouzounis CA, Moore-Kochlacs C, Goldovsky L, Tsoka S, Darzentas N, Kunin V, Kaipa P, Ahre D, Ahrén D *et al*: **Expansion of the BioCyc collection of pathway genome databases to 160 genomes**. *Nucleic acids research* 2005, **33**:6083-6089.
133. Krieger CJ, Zhang P, Mueller LA, Wang A, Paley S, Arnaud M, Pick J, Rhee SY, Karp PD: **MetaCyc a multiorganism database of metabolic pathways and enzymes**. *Nucleic acids research* 2004, **32**:438-442.
134. Federhen S: **The NCBI Taxonomy database.** *Nucleic acids research* 2012, **40**:D136-143.
135. Blake Ja, Dolan M, Drabkin H, Hill DP, Li N, Sitnikov D, Bridges S, Burgess S, Buza T, McCarthy F *et al*: **Gene Ontology annotations and resources.** *Nucleic acids research* 2013, **41**:D530-535.
136. Hull R: **Managing Semantic Heterogeneity in Databases: A Theoretical Perspective**. *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems - PODS '97* 1997, **34**:51-61.
137. Halevy AY, Halevy AY: **Answering queries using views: A survey**. *The VLDB Journal* 2001, **10**:270 - 294.
138. Ghisalberti G, Masseroli M, Tettamanti L: **Quality controls in integrative approaches to detect errors and inconsistencies in biological databases.** *Journal of integrative bioinformatics* 2010, **7**.
139. Thiele H, Glandorf J, Hufnagel P: **Bioinformatics strategies in life sciences: from data processing and data warehousing to biological knowledge extraction.** *Journal of integrative bioinformatics* 2010, **7**:141.
140. Shafer P, Isganitis T, Yona G: **Hubs of knowledge: using the functional link structure in Biozon to mine for biologically significant entities.** *BMC bioinformatics* 2006, **7**:71.
141. Altman RB: **Building successful biological databases.** *Briefings in bioinformatics* 2004, **5**:4-5.
142. Karp PD: **A strategy for database interoperation.** *Journal of computational biology : a journal of computational molecular cell biology* 1995, **2**:573-586.
143. Chen Y-PP, Chen Q: **Analyzing Inconsistency Toward Enhancing Integration of Biological Molecular Databases**. *Proceedings of the 4th Asia-Pacific Bioinformatics Conference* 2005:197-206.
144. Philippi S: **Data and knowledge integration in the life sciences**. *Briefings in bioinformatics* 2008, **9**:451.

DOI:10.15774/PPKE.ITK.2014.011

145. Sansone S-A, Rocca-Serra P, Field D, Maguire E, Taylor C, Hofmann O, Fang H, Neumann S, Tong W, Amaral-Zettler L *et al*: **Toward interoperable bioscience data.** *Nature genetics* 2012, **44**:121-126.

146. Burge S, Attwood TK, Bateman A, Berardini TZ, Cherry M, O'Donovan C, Xenarios I, Gaudet P: **Biocurators and Biocuration: surveying the 21st century challenges.** *Database* 2012, **2012**:bar059.

147. Stein LD: **Integrating biological databases.** *Nature reviews Genetics* 2003, **4**:337-345.

148. Wong L: **Technologies for integrating biological data**. *Briefings in bioinformatics* 2002, **3**:389-404.

149. Myers CL, Troyanskaya OG: **Context-sensitive data integration and prediction of biological networks.** *Bioinformatics (Oxford, England)* 2007, **23**:2322-2330.

150. Davidson SB, Overton C, Buneman P: **Challenges in integrating biological data sources**. *Journal of Computational Biology* 1995, **2**:557-572.

151. Sujansky W: **Heterogeneous database integration in biomedicine.** *Journal of biomedical informatics* 2001, **34**:285-298.

152. Etzold T, Argos P: **SRS–an indexing and retrieval tool for flat file data libraries**. *Bioinformatics* 1993, **9**:49.

153. Bizer C, Berlin FU, Heath T, Berners-Lee T: **Linked Data - The Story So Far**. *International Journal on Semantic Web and Information Systems* 2009, **5**:1-22.

154. Haas LM, Schwarz PM, Kodali P, Kotlar E, Rice JE, Swope WC: **DiscoveryLink: A system for integrated access to life sciences data sources**. *IBM Systems Journal* 2001, **40**(2):489-511.

155. Wang K, Tarczy-Hornoch P, Shaker R, Mork P, Brinkley JF: **BioMediator data integration: beyond genomics to neuroscience data.** *AMIA Annual Symposium proceedings / AMIA Symposium AMIA Symposium* 2005:779-783.

156. Wilkinson MD, Links M: **BioMOBY: an open source biological web services proposal.** *Briefings in bioinformatics* 2002, **3**:331-341.

157. Lee TJ, Pouliot Y, Wagner V, Gupta P, Stringer-Calvert DWJ, Tenenbaum JD, Karp PD: **BioWarehouse: a bioinformatics database warehouse toolkit.** *BMC bioinformatics* 2006, **7**:170.

158. Birkland A, Yona G: **BIOZON: a system for unification, management and analysis of heterogeneous biological data**. *BMC bioinformatics* 2006, **7**:70.

159. Shah SP, Huang Y, Xu T, Yuen MMS, Ling J, Ouellette BFF: **Atlas - a data warehouse for integrative bioinformatics.** *BMC bioinformatics* 2005, **6**:34.

160. Kasprzyk A, Keefe D, Smedley D, London D, Spooner W, Melsopp C, Hammond M, Rocca-Serra P, Cox T, Birney E: **EnsMart: a generic system for fast and flexible access to biological data.** *Genome research* 2004, **14**:160-169.

161. Hagstrom R, Overbeek R, Price M, Micheals G, Taylor R: **Overview of the integrated genomic data system (IGD)**. In: *English: 1992*. Argonne National Laboratory: 1-28.

162. Töpel T, Kormeier B, Klassen A, Hofestädt R: **BioDWH: a data warehouse kit for life science data integration.** *Journal of integrative bioinformatics* 2008, **5**.

163. Birkland A, Yona G: **BIOZON: a system for unification, management and analysis of heterogeneous biological data.** *BMC bioinformatics* 2006, **7**:70.

164. Jensen LJ, Kuhn M, Stark M, Chaffron S, Creevey C, Muller J, Doerks T, Julien P, Roth A, Simonovic M *et al*: **STRING 8--a global view on proteins and their functional interactions in 630 organisms.** *Nucleic acids research* 2009, **37**:D412-D416.

DOI:10.15774/PPKE.ITK.2014.011

165. Knox C, Law V, Jewison T, Liu P, Ly S, Frolkis A, Pon A, Banco K, Mak C, Neveu V *et al*: **DrugBank 3.0: a comprehensive resource for 'omics' research on drugs.** *Nucleic acids research* 2011, **39**:D1035-D1041.
166. Tatusov RL, Fedorova ND, Jackson JD, Jacobs AR, Kiryutin B, Koonin EV, Krylov DM, Mazumder R, Mekhedov SL, Nikolskaya AN *et al*: **The COG database: an updated version includes eukaryotes**. *BMC bioinformatics* 2003, **4**:41.
167. Lachmann A, Ma'ayan A: **Lists2Networks: integrated analysis of gene/protein lists.** *BMC bioinformatics* 2010, **11**:87.
168. Li W, Jaroszewski L, Godzik A: **Tolerating some redundancy significantly speeds up clustering of large protein databases.** *Bioinformatics (Oxford, England)* 2002, **18**:77-82.
169. Reese MG, Moore B, Batchelor C, Salas F, Cunningham F, Marth GT, Stein L, Flicek P, Yandell M, Eilbeck K: **A standard variation file format for human genome sequences.** *Genome biology* 2010, **11**:R88.
170. **Extensible Markup Language (XML)**. 2014.
171. Achard F, Vaysseix G, Barillot E: **XML bioinformatics and data integration**. *Bioinformatics* 2001, **17**:115-125.
172. **Proteomics Standards Inititative** [http://www.psidev.info/]
173. **The Google Cloud Platform** [https://cloud.google.com/]
174. Sterling TL: **Beowulf cluster computing with Linux**. Cambridge, Mass: MIT Press; 2002.
175. **The MySQL DBMS** [http://www.mysql.com/]
176. **The MySQL Workbench Tool** [http://www.mysql.com/products/workbench/]
177. **Oracle Java SE 7** [http://www.oracle.com/technetwork/java/javase/downloads/index.html]
178. **The Netbeans IDE** [https://netbeans.org/]
179. **Apache Maven Project** [http://maven.apache.org/]
180. **The EclipseLink Project** [https://www.eclipse.org/eclipselink/]
181. **Red Hat JBoss Middleware** [http://www-beta.jboss.org]
182. **JGraph** [http://www.jgraph.com/]
183. **Mojarra JavaServer Faces** [https://javaserverfaces.java.net/]
184. **Primefaces** [http://primefaces.org/]
185. Northrup CJ: **Programming with UNIX Threads**. New York: John Wiley & Sons; 1996.
186. **The MPICH library** [http://www.mpich.org]
187. Foster I: **Designing and building parallel programs : concepts and tools for parallel software engineering**. Reading, Mass.: Addison-Wesley; 1995.
188. Bayer RaM, E. : **Organization and Maintenance of Large Ordered Indexes**. *Acta Informatica* 1972, **1**(3):173-189.
189. **Google Code Platform** [https://code.google.com/]
190. **Apache Subversion System** [http://subversion.apache.org/]
191. **Google Cloud Platform** [https://cloud.google.com/]
192. Hamosh A, Scott AF, Amberger JS, Bocchini Ca, McKusick Va: **Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders.** *Nucleic acids research* 2005, **33**:D514-517.

DOI:10.15774/PPKE.ITK.2014.011

193. Coordinators NR: **Database resources of the National Center for Biotechnology Information.** *Nucleic acids research* 2014, **42**:D7-D17.
194. Schmitt T, Messina DN, Schreiber F, Sonnhammer EL: **Letter to the editor: SeqXML and OrthoXML: standards for sequence and orthology information**. *Briefings in bioinformatics* 2011, **12**(5):485-488.
195. Schneider A, Dessimoz C, Gonnet GH: **OMA Browser--exploring orthologous relations across 352 complete genomes**. *Bioinformatics* 2007, **23**(16):2180-2182.
196. Altenhoff AM, Schneider A, Gonnet GH, Dessimoz C: **OMA 2011: orthology inference among 1000 complete genomes**. *Nucleic acids research* 2011, **39**(Database issue):D289-294.
197. Gallardo D: **Java design patterns**. In: *Development*. 1-22.
198. Fenyo D: **The Biopolymer Markup Language**. *Bioinformatics* 1999, **15**(4):339-340.
199. Lipman DJ, Pearson WR: **Rapid and sensitive protein similarity searches**. *Science* 1985, **227**(4693):1435-1441.
200. **The Apache Tomcat Server**. 2014.
201. Kasprzyk A: **BioMart: driving a paradigm change in biological data management.** *Database* 2011, **2011**:bar049.
202. Durinck S, Moreau Y, Kasprzyk A, Davis S, Moor BD, Brazma A, Huber W: **BioMart and Bioconductor a powerful link between biological databases and microarray data analysis**. *Bioinformatics* 2005, **21**:3439-3440.
203. Mazola Reyes Y, Chinea Santiago G, Guirola Cruz O, Vera Alvarez R, Huerta Galindo V, Fleitas Salazar N, Musacchio Lasa A: **Chemical compounds having antiviral activity against dengue virus and other flaviviruses**. In.: WO/2009/106019; 2009.
204. Rodriguez Fernandez RE, Vera Alvarez R, de la Nuez Veulens A, Mazola Reyes Y, Perea Rodriguez SE, Acevedo Castro BE, Musacchio Lasa A, Ubieta Gomez R: **Antineoplastic compounds and pharmaceutical compositions thereof**. In.: WO/2006/119713; 2006.
205. Perez-Riverol Y, Vera R, Mazola Y, Musacchio A: **A parallel systematic-Monte Carlo algorithm for exploring conformational space.** *Current topics in medicinal chemistry* 2012, **12**:1790-1796.
206. **Matrix Science Mascot** [http://www.matrixscience.com]
207. Sanchez A, Perez-Riverol Y, González LJ, Noda J, Betancourt L, Ramos Y, Gil J, Vera R, Padrón G, Besada V: **Evaluation of Phenylthiocarbamoyl-Derivatized Peptides by Electrospray Ionization Mass Spectrometry: Selective Isolation and Analysis of Modified Multiply Charged Peptides for Liquid Chromatography-Tandem Mass Spectrometry Experiments.** *Analytical chemistry* 2010, **xxx**:552-559.
208. Perez-Riverol Y, Sánchez A, Ramos Y, Schmidt A, Müller M, Betancourt L, González LJ, Vera R, Padron G, Besada V: **In silico analysis of accurate proteomics, complemented by selective isolation of peptides.** *Journal of proteomics* 2011, **74**:2071-2082.
209. Dogsa I, Choudhary KS, Marsetic Z, Hudaiberdiev S, Vera R, Pongor S, Mandic-Mulec I: **ComQXPA Quorum Sensing Systems May Not Be Unique to Bacillus subtilis: A Census in Prokaryotic Genomes**. *PloS one* 2014, **9**(5):e96122.
210. Powell S, Forslund K, Szklarczyk D, Trachana K, Roth A, Huerta-Cepas J, Gabaldon T, Rattei T, Creevey C, Kuhn M *et al*: **eggNOG v4.0: nested orthology inference across 3686 organisms**. *Nucleic Acids Res* 2014, **42**(Database issue):D231-239.

DOI:10.15774/PPKE.ITK.2014.011

211. Vera R, Perez-Riverol Y, Perez S, Ligeti B, Kertesz-Farkas A, Pongor S: **JBioWH: an open-source Java framework for bioinformatics data integration**. *Database (Oxford)* 2013, **2013**:bat051.
212. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R: **The sequence alignment/map format and SAMtools**. *Bioinformatics* 2009, **25**(16):2078-2079.
213. Benson DA, Clark K, Karsch-Mizrachi I, Lipman DJ, Ostell J, Sayers EW: **GenBank**. *Nucleic Acids Res* 2014, **42**(Database issue):D32-37.
214. Bayer R, McCreight E: **Organization and Maintenance of Large Ordered Indexes**. *Acta Informatica* 1972, **1**(3):173-189.
215. Jensen LJ, Julien P, Kuhn M, von Mering C, Muller J, Doerks T, Bork P: **eggNOG: automated construction and annotation of orthologous groups of genes**. *Nucleic Acids Res* 2008, **36**(Database issue):D250-254.
216. **The AngularJS library** [https://angularjs.org/]
217. **The Node.JS platform** [http://nodejs.org/]
218. Shiryev SA, Papadopoulos JS, Schaffer AA, Agarwala R: **Improved BLAST searches using longer words for protein seeding**. *Bioinformatics* 2007, **23**(21):2949-2951.
219. Zhang Z, Schwartz S, Wagner L, Miller W: **A greedy algorithm for aligning DNA sequences**. *J Comput Biol* 2000, **7**(1-2):203-214.
220. Tu Q, He Z, Zhou J: **Strain/species identification in metagenomes using genome-specific markers**. *Nucleic acids research* 2014, **42**(8):e67.
221. Iqbal SS, Mayo MW, Bruno JG, Bronk BV, Batt CA, Chambers JP: **A review of molecular recognition technologies for detection of biological threat agents**. *Biosensors & bioelectronics* 2000, **15**(11-12):549-578.
222. Baker GC, Smith JJ, Cowan DA: **Review and re-analysis of domain-specific 16S primers**. *Journal of microbiological methods* 2003, **55**(3):541-555.
223. Vera R, Perez-Riverol Y, Perez S, Ligeti B, Kertesz-Farkas A, Pongor S, Kertész-Farkas A: **JBioWH: an open-source Java framework for bioinformatics data integration.** *Database : the journal of biological databases and curation* 2013, **2013**:bat051.
224. Pongor LS, Vera R, Ligeti B: **Fast and sensitive alignment of microbial whole genome sequencing reads to large sequence datasets on a desktop PC: application to metagenomic datasets and pathogen identification**. *PloS one* 2014, **submitted**.